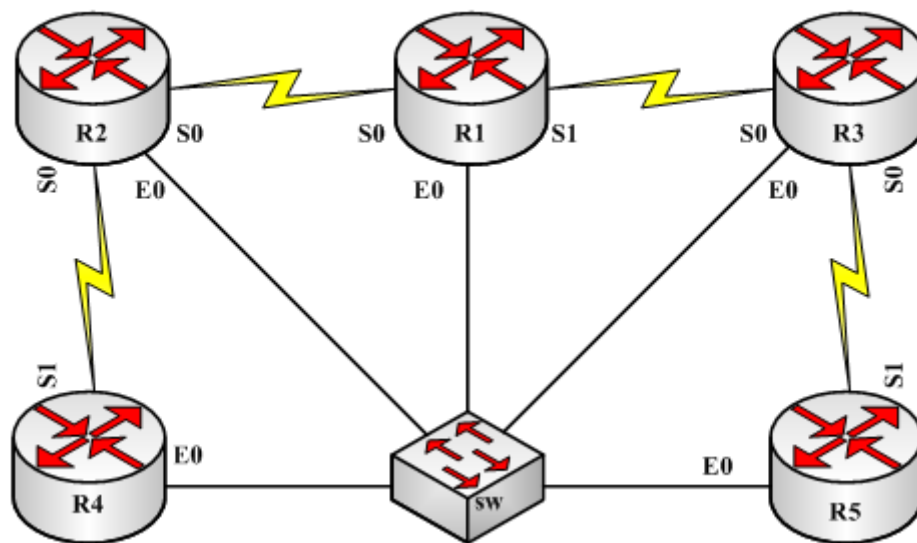
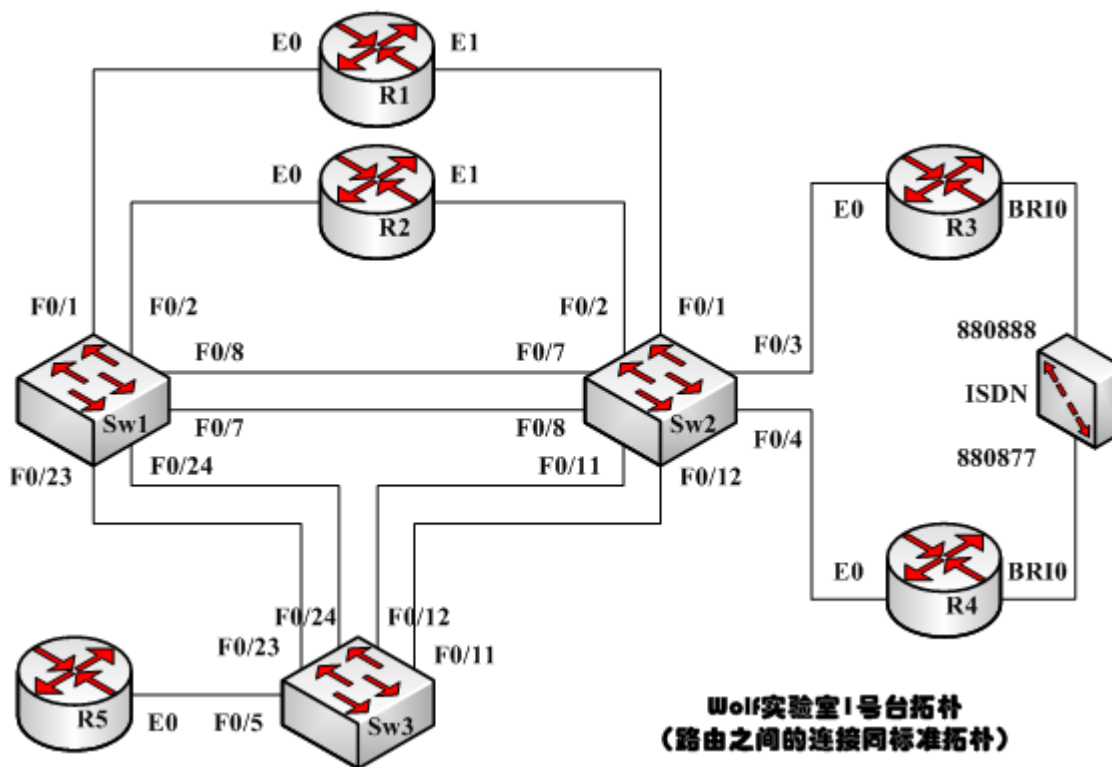


My Note

WOLF实验室标准拓扑:



WOLF实验室一号台拓扑:



网关: 192.168.2.2

老马（马骞）的MSN: mark_ipv6@hotmail.com

Cisco在线文档: www.cisco.com/univerd

Cisco认证: www.net130.com

Wolf-NA笔记

640-801考试IOS命令大全

? 给出一个帮助屏幕

0.0.0.0 255.255.255.255 通配符命令; 作用与any命令相同

access-class 将标准的IP访问列表应用到VTY线路

access-list 创建一个过滤网络的测试列表

any 指定任何主机或任何网络; 作用与0.0.0.0 255.255.255.255命令相同

Backspace 删除一个字符

Bandwidth 设置一个串行接口上的带宽

Banner 为登录到本路由器上的用户创建一个标志区

cdp enable 打开一个特定接口的CDP

cdp holdtime 修改CDP分组的保持时间

cdp run 打开路由器上的CDP

cdp timer 修改CDP更新定时器

clear counters 清除某一接口上的统计信息

clear line 清除通过Telnet连接到路由器的连接

clear mac-address-table 清除该交换机动态创建的过滤表

clock rate 提供在串行DCE接口上的时钟

config memory 复制startup-config到running-config

config network 复制保存在TFTP主机上的配置到running-config

config terminal 进入全局配置模式并修改running-config

config-register 告诉路由器如何启动以及如何修改配置寄存器的设置

copy flash tftp 将文件从闪存复制到TFTP主机

copy run start copy running-config startup-config 的快捷方式, 将配置复制到NVRAM中

copy run tftp 将running-config文件复制到TFTP主机

Copy tftp flash 将文件从TFTP主机复制到闪存

Copy tftp run 将配置从TFTP主机复制为running-config文件

Ctrl+A 移动光标到本行的开始位置

Ctrl+D 删除一个字符

Ctrl+E 移动光标到本行的末尾

Ctrl+F 光标向前移动一个字符

Ctrl+R 重新显示一行

Ctrl+Shift+6,then X 当telnet到多个路由器时返回到原路由器

Ctrl+U 删除一行

Ctrl+W 删除一个字

CTRL+Z 结束配置模式并返回EXEC(执行状态)

debug dialer 显示呼叫建立和结束的过程

debug frame-relay lmi 显示在路由器和帧中继交换机之间的lmi交换信息
debug ip igrp events 提供在网络中运行的IGRP路由选择信息的概要
debug ip igrp transactions 显示来自相邻路由器要求更新请求消息和由路由器发到相邻路由器的广播消息
debug ip rip 发送控制台消息显示有关在路由器接口上收发RIP数据包的信息
debug ipx 显示通过路由器的RIP和SAP信息
debug isdn q921 显示第二层进程
debug isdn q931 显示第三层进程
delete nvram 删除1900交换机-NVRAM的内容
delete vip 删除交换机的VTP配置
description 在接口上设置---个描述
dialer idle-timeout number 告诉BRI线路如果没有发现触发DDR的流量什么时候断开
dialer list number protocol 为DDR链路指定触发DDR的流量
protocol permit/deny
dialer load-threshold number 设置描述什么时候在ISDN链路上启闭第二个BRI的参数
inbound/outbound/either
Dialer map protocol address 代替拨号串用于作ISDN网络中提供更好的安全性
name hostname number
dialer string 设置用于拨叫BRI接口的电话号码
disable 从特权模式返回用户模式
disconnect 从原路由器断开同远端路由器的连接
duplex 设置一个接口的双工
enable 进入特权模式
enable password 设置不加密的启用口令
enable password level 1 设置用户模式口令
enable password level 15 设置启用模式口令
enable secret 设置加密的启用秘密口令。如果设置则取代启用口令
encapsulation 在接口上设置帧类型
encapsulation frame-relay 修改帧中继串行链路上的封装类型
encapsulation frame-relay 将封装类型设置为因特网工程任务组 (IETF, Internet Engineering Task Force)类型。连接Cisco路由器和非Cisco路由器
encapsulation hdlc 恢复串行路由器的默认封装HDLC
encapsulation isl 2 为VLAN 2设置ISL路由
encapsulation ppp 将串行链路上的封装修改为PPP
erase startup 删除startup-config
erase startup-config 删除路由器上的NVRAM的内容
Esc+B 向后移动一个字
Esc+F 向前移动一个字
exec-timeout 为控制台连接设置以秒或分钟计的超时
exit 断开远程路由器的Telnet连接
frame-relay interface-dlci 在串行链路或子接口上配置PVC地址
frame-relay lmi-type 在串行链路上配置LMI类型
frame-relay map protocol 创建用于帧中继网络的静态映射
Host 指定一个主机地址
Hostname 设置一台路由器或交换机的名字
int e0.10 创建一个子接口
int f0/0.1 创建一个子接口

interface e0/5 配置Ethernet接口5
interface ethernet 0/1 配置接口e0/1
interface f0/26 配置Fast Ethernet接口26
interface fastethernet 进入Fast Ethernet端口的接口配置模式
interface fastethernet 0/0.1 创建一个子接口
interface fastethernet0/26 配置接口f0/26
interface s0.16 multipoint 在串行链路上创建用于帧中继网络的多点子接口
interface s0.16 point-to-point 在串行链路上创建用于帧中继网络的点对点子接口
interface serial 5 进入接口serial 5的配置模式，也可以使用show命令
ip access-group 将IP访问列表应用到一个接口
ip address 设置一个接口或交换机IP地址
ip classless 一个全局配置命令，用于告诉路由器当目的网络没有出现在路由表中时通过默认路由转发数据包
ip default-gateway 设置该交换机的默认网关
ip domain-lookup 打开DNS查找功能（默认时打开）
ip domain-name 将域名添加到DNS查找名单中
ip host 在路由器上创建主机表
ip name-server 最多设置6个DNS服务器的IP地址
Ip route 在路由器上创建静态和默认路由
ipx access-group 将IPX访问列表应用到一个接口
ipx input-sap-filter 将输入型IPX SAP过滤器应用到一个接口
ipx network 为接口分配IPX网络号
ipx output-sap-filter 将输出型IPX SAP过滤器应用到一个接口
ipx ping 用于测试互联网络上IPX包的因特网探测器
ipx routing 打开IPX路由
isdn spid1 为ISDN交换机设置标识第一个DSO的号码
isdn spid2 为ISDN交换机设置标识第二个DSO的号码
isdn switch-type 设置路由器与之通信的ISDN交换类型。可以往接口模式和全局配置模式下设
K 用在1900交换机启动时并将该交换机置于CLI模式
Line 进入配置模式以修改和设置用户模式口令
line aux 进入辅助接口配置模式
line console 0 进入控制台配置模式
line vty 进入VTY(Telnet)接口配置模式
logging synchronous 阻止控制台信息覆盖命令行上的输入
logout 退出控制台会话
mac-address-table permanent 在过滤数据库中生成一个永久MAC地址
mac-address-table restricted 在MAC过滤数据库中设置一个有限制的地址，只允许所配置的接口与有限制的地址通信
media-type 在接口上设置硬件介质体类型
network 告诉路由选择协议要发通告的网络
no cdp run 关闭单个接口上的CDP
no inverse-arp 完全关闭路由器上的CDP
no inverse 关闭帧中继中的动态IARP。必须已配置了静态映射
no ip domain-lookup 关闭DNS查找功能
no ip host 从主机表删除一个主机名
No IP route 删除静态或默认路由

no shutdown 打开一个接口
o/r Ox2142 修改2501以便启动时不使用NVRAM的内容
ping 测试一个远程设备的IP连通性
port secure max-mac-count 只允许配置的设备量连接并在一个接口上工作
ppp authentication chap 告诉PPP使用CHAP认证方式
ppp authentication pap 告诉PPP使用PAP认证方式
router igrp as 在路由器上打开IP IGRP路由选择
router rip 使用户进入路由器rip配置模式
secondary 在同一个物理接口上添加辅助IPX网络
Service password-encryption 加密用户模式和启用口令
show access-list 显示路由器上配置的所有访问列表
show access-list 110 只显示访问列表110
show cdp 显示CDP定时器和保持时间周期
show cdp entry * 同show cdpneighbor detail命令一样，但不能用于1900交换机
show cdp interface 显示启用了CDP的特定接口
show cdp neighbor 显示直连的相邻设备及其详细信息
show cdp neighbor detail 显示IP地址和IOS版本和类型，并且包括show cdp neighbor命令显示的所有信息
show cdp traffic 显示设备发送和接收的CDP分组数以及任何出错信息
Show controllers s 0 显示接口的DTE或DCE状态
show dialer 显示拨号串到达的次数、B信道的空闲超时时间值、呼叫长度以及接口所连接的路由器的名称
show flash 显示闪存中的文件
show frame-relay lmi 在串行接口上设置LMI类型
show frame-relay map 显示静态的和动态的网络层到PVC的映射
show frame-relay pvc 显示路由器上已配置的PVC和DLCI号
show history 默认时显示最近输入的10个命令
show hosts 显示主机表中的内容
show int f0/26 显示f0/26的统计
show interface e0/1 显示接口e0/1的统计
show interface S0 显示接口serial上的统计信息
show ip 显示该交换机的IP配置
show ip access-list 只显示IP访问列表
show ip interface 显示哪些接口应用了IP访问列表
show ip interface 显示在路由器上配置的路由选择协议及与每个路由选择协议相关的定时器
show ip route 显示IP路由表
show ipx access-list 显示路由器上配置的IPX访问列表
trunk on 将一个端口设为永久中继模式 "
username name password 为了Cisco路由器的身份验证创建用户名和口令
variance 控制最佳度量和最坏可接受度量之间的负载均衡
vlan 2 name Sales 创建一个名为Sales的VLAN2
vlan-membership static 2 给端口分配一个静态VLAN
vtp client 将该交换机设为一个VTP客户
vtp domain 设置为该VTP配置的域名
vtp password 在该VTP域上设置一个口令
vtp pruning enable 使该交换机成为一台修剪交换机

vtp server 将该交换机设为一个VTP服务器
show ipx interface 显示一个特定接口上发送和接收的RIP和SAP信息
show ipx servers 显示接口的IPX地址
show ipx route 显示IPX路由表
show ipx traffic 显示Cisco路由器的SAP表
show ipx traffic 显示Cisco路由器上发送和接收的RIP和SAP信息
show isdn active 显示呼叫的号码和呼叫是否在进行中
show isdn status 显示SPID是否有效、是否已连接以及和提供商交换机的通信情况
show mac-address-table 显示该交换机动态创建的过滤表
show protocols 显示在每个接口上配置的被动路由协议和网络地址
show run showrunning-config的缩写，显示当前在该路由器上运行的配置
show sessions 显示通过Telnet到远程设备的连接
show start 命令show startup-config的快捷方式。显示保存在NVRAM中的备份配置
show terminal 显示配置的历史记录
show trunk A 显示端口26的中继状态
show trunk B 显示端口27的中继状态
show version 给出该交换机的IOS信息以及正常运行时间和基本Ethernet地址
show vlan 显示所有已配置的VLAN
show vlan-membership 显示所有端口的VLAN分配
show vtp 显示一台交换机的VTP配置
shutdown 设置接口为管理性关闭模式
Tab 为操作者完成命令的完整输入
telnet 连接、查看并在远程设备上运行程序
terminal history size 改变历史记录的大小由默认的10改为256
trace 测试远程设备的连通性并显示通过互联网络找到该远程设备的路径
traffic-share balanced 告诉IGRP路由选择协议要反比于度量值分享链路
traffic-share min 告诉IGRP路由选择协议要使用只有最小开销的路由
trunk auto 将该端口设为自动中继模式

0.0—一些要点

0.0—一些要点

考试范围：

TK640-801 (Cisco Certified Network Associate 640-801 ICND Course Notes)。

信息单位：

1Byte=8Bit。

网卡NIC (Network Interface Card)：

Bps=byte per second;

网络介质Media: 传输电磁波用, 有铜缆、光缆等;

协议Protocol: 计算机之间相互交换信息的语言;

网络操作系统NOS (Network ogenated System): Unix/Linux/Windows系列等;

IOS: 是Cisco建立在Unix基础的操作模式, 基于CLI (Command Line Interface) 字符交互模式。

常见网络:

局域网 (LAN): 网速快, 规模小 (10-10K米范围);

城域网 (MAN): 网速适中, 规模适中, 没有什么优点, 淘汰中;

广域网 (WAN): 网速慢, 规模大 (无地域限制, 通过ISP遍布全球)。

网络形式:

直连网络 (Bus Topology): 以太网常用;

环形网络 (Ring Topology): 令牌环网络常用;

星形网络 (Star Topology): 也叫Hub&Spoke网络, 最常用的;

扩展星形网络 (Extended Star Topology): 星形网络的扩展;

饱和网络 (Mesh Topology): 所用路由间都有直连, 可靠性好但是造价最高, 高端用户的必然选择。

引起LAN拥塞的可能的原因是:

太多的主机存在于1个广播域(broadcast domain);

广播风暴;

多播;

带宽过低。

连接设备 (Connectivity Devices):

主要有Model、Hub、Bridge、Switch、Router; 他们的作用是把大的网络分成几个小点的网络称之为“网络分段” (Network Segment)简称网段。

路由器 (Router) 和交换机 (Switch):

路由器 (Router) /交换机 (Switch) 都运行IOS;

路由器 (Router) /交换机 (Switch) 就是计算机;

路由器可以进行数据包的路由 (L3), 按照数据包的L3包头进行转发, 实现不同网络的数据路由;

路由器可以进行数据帧的路由 (L2), 按照数据包的L2帧头进行转发, 实现不同网络的数据交换。

在网络中使用路由器 (router) 的优点:

它们默认是不会转发广播的;

它们可以基于协议层L3(Network layer)来对网络进行过滤。

在网络中使用交换机 (switch) 的优点:

能提高LAN的性能, 提供给用户更多的带宽。

网桥 (bridge) :

Bridges在某种意义上等同Switch, 它们都可以分割冲突域 () : 1个端口即为1个冲突域 (它们仍然处在一个广播域, 分割广播域的任务一般由router完成); 但是Bridge一般只有2~4个端口而Switch可以拥有多达上百个端口所以渐渐被淘汰;

OSI的产生:

早期各个网络厂商拥有私有网络, 不便于同其他厂商的网络进行通讯; 于是在20世纪70年代末期, ISO组织创建了OSI (Open System Interconnection) 参考模型;

OSI参考模型最大的特点是分层, 但是它仍然只是个参考模型而非物理模型;

OSI参考模型分层化的优点: 允许多厂家共同发展网络标准化组件; 允许不同类型的网络硬件和软件相互通信; 防止其中某层的变化影响到其他层牵制整个模型。

OSI的参考模型:

L7-L5被统称为应用层 (Application Layers), 详分三层:

L7应用层 (Application): 提供用户接口也就是完成应用程序的使用;

L6表示层 (Presentation): 按照应用层的要求进行压缩、加密、翻译等操作;

L5会话层 (Session): 应用程序间的终端连接和会话建立和控制, 分隔不同应用程序的数据;

L5-L1被统称为数据传输层 (Data Flow Layers), 详分四层:

L4传输层 (Transport): 数据间的传输End-To-End connections, 还负责在传输前的数据纠错;

L3网络层 (Network): 网络间的协商, 用于路由器间的路由选择;

L2链路层 (Data Link): 路由协议的建立 (根据MAC地址), 实行错误检测但是不负责更正;

L1物理层 (PhySical): 设备的物理链路连接。

OSI的封装:

L7应用层 (Application): 在数据 (Data) 前加L7 HDR (APDU);

L6表示层 (Presentation): 再在前加L6 HDR (PPDU);

L5会话层 (Session): 再在前加L5 HDR (SPDU);

L4传输层 (Transport): 再在前加L4 HDR (Segment);

L3网络层 (Network): 再在前加L3 HDR (Packet);

L2链路层 (Data Link): 再在前加L2 HDR (Frame); 后加信息终止符FCS;

L1物理层 (PhySical): 转变成Bit流。

L5会话层 (Session Layer) :

负责建立、管理和终止会话, 也就是设备和节点(nodes)之间的会话控制; 一般有3种模式: Simplex (单工: 只能由某端发送数据给另一端)、Half Duplex

Duplex（半双工：某端发给另一端时另一端无法发送数据）和Full Duplex（双工：可以两端同时发送数据）。

TCP/IP的模型：

L7-L4被统称为协议层（Protocol），详分两层：

应用层（Application）：OSI的L7应用层+L6表示层+L5会话层；

传输层（transport）；OSI的L4传输层；

L3-L1被统称为网络（Network），详分两层：

Internet服务层（Internet）：OSI的L3网络层；

网络协议层（Network Access）：OSI的L2链路层+L1物理层；

结论：OSI模型是按照数据的变化分层，TCP/IP按照网络的结构分层。

0.1—NA路由①

0.1—NA路由①

Cisco设备的端口：

在Cisco的路由器上都有一个带外网管口（Console/AUX）；

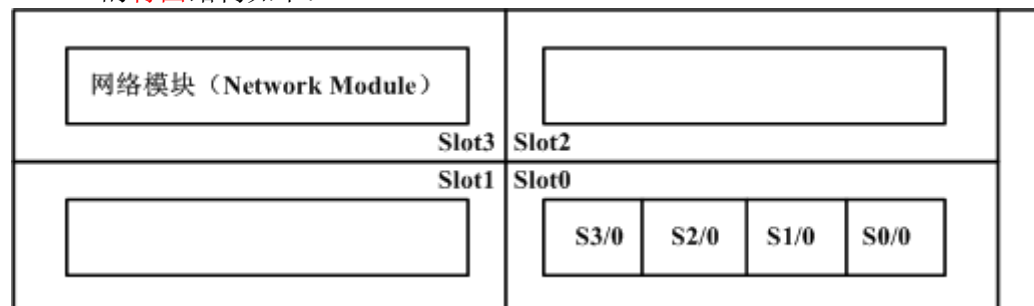
Con口主要用于本地的con线进行本地网管；

AUX口主要与Modem连接通过固话网PSTN连接远端AUX进行远程网管；

带内网管接口：Ether口（AUI）/Serail口（SERAIL）/ISDN口（BRI），通过虚拟接口（VTY）实现网管，每个路由器默认有5个TELnet进程（VTY 0 4）。

模块化接口的排列顺序规则：

从0算起；从下到上；从右而左；0/1(模块号/接口号)；2u（1u=英寸）的Router的背面结构如下：



一些快捷键：

?：调用帮助；

tab：补全命令；

CTRL+A：移动到句首；

CTRL+E：移动到句末；

CTRL+F：向句末移动光标，以字节为单位；（同方向键→）

ESC→F：向句末移动光标，以字段为单位，不可连接；

CTRL+B：向句首移动光标，以字节为单位；（同方向键←）

CTRL+B: 向句首移动光标, 以字节为单位; (同方向键←)
ESC→B: 向句首移动光标, 以字段为单位, 不可连接;
Backspace: 向前删除
CTRL+D: 向后删除;
CTRL+U: 删除整句命令;
CTRL+P: 调出上一个命令并向上翻页; ; (同方向键↑)
CTRL+N: 调出上一个命令并向下翻页。; (同方向键↓)
CTRL+Z: 从较高级别模式快速切换至特权模式;
CTRL+SHIFT+6: 停止当前进行的进程;

CISCO设备的文件系统:

RAM: 动态内存, 承载配置文件: running-config ;
ROM: 只读存储器 (BootStarp/Mini IOS) ;
NVRAM: 非易失性的存储器, 32-128K, 承载启动文件: startup-config ;
FLASHROM: 相当于硬盘, 4M-256M, 承载重要的系统文件: IOS ;
外部文件服务器: TFTPsever/TFTPd32等等 (TFTP—UDp69, TFP—TCP21) 。

路由器的相关概念:

路由器 (Router n. 名词): 核心词Computer电脑, 路由器就是一个用于数据包转发的电脑: 硬件上路由器有多个网络接口; 软件上路由器运行一套专用的基于UNIX的操作系统IOS; 路由器的功能是互联不同的物理类型的网络和不同的网段;

路由 (Route n. 名词): 核心词Path网段, 是路由器把信息转发给特定网络目标的路径, 对于一个路由器来说去往同一网络可能有很多条不同的路径;

路由 (Route v. 动词): 核心词Forwarding转发, 过程为路由器从入接口收到信息后剥掉L2帧头FRAME (**源地址和目标地址是前一个发此信息帧的路由器和接收此信息帧的本路由器**), 然后查看L3层包头PACKET (**源地址和目标地址是发此信息包的根源路由器和要接收此信息包的最终路由器**), 接着查询自己的路由表找到到达目标地址的出接口和要经过的下一个路由器 (**也就是下一跳**), 并按照两者之间的封装方式重新封装L2帧头FRAME (**源地址和目标地址是发送此信息帧的本路由器和下一个要接收此帧的路由器也就是下一跳**) 并从出接口发出此帧; 不断封装→解封装→封装→解封装……; 跳数 (Hops) 因此影响到路由的开销;

路由协议 (Routing Protocol adj.): 核心词Language语言, 是用于为IP数据包寻路和维护路由表的协议; 路由器与路由器之间用于交互信息的语言也叫做内部网关协议 (IGP) ;

被路由协议 (Routed Protocol adj.): 核心词Payload网络的净荷; 是用于为数据包指路的协议。

LAB1: Cisco设备的基本操作 (带外网管):

STEP1: 接通链路:

将电源接好, 将con线 (交叉线) 的RJ-45接头插入路由/交换机的con口,

LAB1: Cisco设备的基本操作（带外网管）：

STEP1: 接通链路：

将电源接好，将con线（交叉线）的RJ-45接头插入路由/交换机的con口，DB-9插入笔记本的DB-9异步串口（RS-232）；

STEP2: 启动终端：

五个参数：每秒数据—9600，数据位—8，校检—无，停止位—1，流控—无；

STEP3: 开机：

是否载入配置文件：No；然后默认进入用户模式R>，只能查看简单的信息不可以操作；

STEP4: 进入特权模式R#：

Enable：可以查看高级信息了；

STEP5: 进入配置模式R(config)#：

Configure terminal：可以修改配置了；可以用？查看可用的命令，用TAB自动补全命令；通过配置命令可以进入其他模式如接口模式R(config-if)#和协商模式R(config-router)#等等；

STEP6: 常用开机命令：

R>enable（进入特权模式）；

R#configure terminal（进入配置模式）；

R(c)#no ip domain-lookup（关闭域名查询）；

R(c)#line console 0（配置con口）；

R(c-i)#no exec-timeout（设置为永不超时，与命令exec-t 0 0效果相同，默认为exec-t 10 0即10分钟无操作即超时断开连接，工作一般设置为3 0）；

R(c-i)#logging synchronous（关闭显示同步）；

R(c)#host ~~xx~~（设置路由器姓名）；

可以用CTRL+Z快速返回特权模式；

STEP7: 在特权模式下查看和设置：

查看/设置时间：R#clock show/set 11:38:20（时间） 5（日） aug（月）2006（年）；

查看版本和CPU/Flashram/NVram等软件硬件固件的信息R#show version；

查看运行状态R#show runing-config；

STEP7: 时间戳（计时方式）：

在配置模式下R(c)#service timestamps debug uptime/datatime；R(c)#service timestamps log uptime/datatime：以启动时间/当前日期计时；

STEP8: 配置接口（同一条链路上的接口要在同一网段）：

R(c)#interface Serial 0（进入接口）；

R(c-i)#description connect to sh（贴个标签描述接口作用）；

R(c-i)#ip address 12.0.0.1（IP地址） 255.255.255.0（子网掩码）（配置IP）；

R(c-i)#no shutdown/shutdown（打开/关闭接口）；

LAB2:通过Telnet远程管理网络设备（带内网管）：

STEP1: 以太网线（1236）和交叉线（1326）的制作：

然后使用con线（交叉线）连接Com口到Ether口的链路，并配置IP地址；

STEP2: 接着确认网管IP地址（要与接口在同一网段）：

C:\>ip config ;

C:\>ping IP;

C:\>arp -a ;

R#show interfaces ethernet 0 ;

STEP3: 对路由器发动Telnet管理：

C:\>telnet IP 路由器拒绝了：没有VTY密码，如下设置：

R(c)#line vty 0 4（共五个VTY厕所蹲位）；

R(c-i)#password *** ;

R(c-i)#login（可以不加系统会自动进入，如果改成no login就会不需要密码即可以TELnet）；

STEP3: 配置特权模式的密码：

R(c)#enable password/secret ***（配置明文/密文密码，设置密文会覆盖已有的明文密码）；

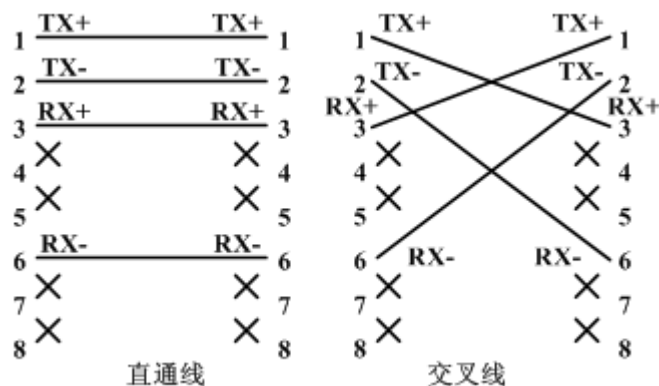
STEP4: 然后再TELnet：

C:\>telnet IP 然后输入密码后成功……不要更改自己进入Telnet设备使用的接口的参数；

R#exit 退出TELnet；

STEP5: 在被TELnet的设备强行断开TELnet：

首先查看TELnet的用户：R(c)#show user/who；然后清线：R(c)#clear line *；



LAB3: Cisco路由的文件处理：

LAB3: Cisco路由的文件处理:

STEP1: 确定链路的通达;

R#ping IP ;

STEP2: 运行TFTP服务器 (采用TFTPd32) ;

首先运行TFTP服务器 (双击图标不用介绍吧) ; 然后查看文件夹: R#dir nvram (查看NVRam的配置文件: startup-config) ; R#dir flash: (查看Flash的IOS文件: 16423684) ;

STEP3: 配置文件的管理:

R#write/copy running-config nvram:startup-config (备份或者说存盘配置文件) ;

R#copy nvram:startup-config running-config (读取启动文件文件) ;

R#copy nvram:startup-config tftp: (将NVRAM的启动文件备份到TFTP) ;

R#copy running-config tftp: (将RAM的配置文件备份到TFTP) ;

R#copy tftp: nvram:startup-config (调用TFTP到NVRAM的启动文件) ;

R#copy tftp: running-config (调用TFTP到RAM的配置文件) ;

STEP4: IOS的升级:

R#copy flash: tftp: (备份IOS) ;

R#copy tftp: flash: (升级IOS) ;

注意: 确认升级成功才可以重启!

LAB4: Cisco路由的密码恢复:

STEP1: 确定链路的通达;

使用con线 (只能通过con线恢复密码) 连接好console口, 确认笔记本能够正常显示路由器的信息;

密码保存在配置文件中, 确认路由的配置寄存器开机模式为默认模式调用: 0x2102 (sh ru查看, 0x2142为开机时不调用启动文件) ; 不是时用命令R(c) #config-register 0x2102恢复;

STEP2: 重启路由器:

重启路由器, 在重启开始的3-8秒内按住CTRL+BREAK直到进入只读监控的ROM MONITOR模式 (相当于windows的安全模式, 特点时显示 >, 前面没有路由器名字) 后松手;

STEP3: 修改寄存器模式为0X2142:

2500系列中:

>o/r 0x2142 ;

>i (重启路由器) ;

2600系列中:

>confreg 0x2142 ;

>i (重启路由器) ;

STEP4: 在路由器中调用启动文件到内存中:

再进入后发现不载入启动文件开机也就是不需要密码即可以进入, 用R#copy nvram:startup-config running-config读取启动文件文件;

STEP4: 在路由器中调用启动文件到内存中:

再进入后发现不载入启动文件开机也就是不需要密码即可以进入, 用R#copy

nvram:startup-config running-config读取启动文件文件;

STEP5: 删除原密码并设置新密码:

R(c)#no enable password/secret *** ;

R(c)#enable password/secret *** ;

STEP6: 恢复配置寄存器的设置为0x2102:

R(c)#config-register 0x2102 ;

STEP7: 实验室请恢复配置寄存器的设置为0x2142:

R(c)#config-register 0x2142 ;

LAB5: Cisco路由的广域网连接:

STEP1: 连接链路:

使用同步串行链路V.35模拟, 这种连接方式也叫Back-to-Back连接; 注意V.35的公头DTE/母头DCE;

STEP2: 开机配置:

R>enable (进入特权模式);

R#configure terminal (进入配置模式);

R(c)#no ip domain-lookup (关闭域名查询);

R(c)#line console 0 (配置con口);

R(c-i)#no exec-timeout (防止超时);

R(c-i)#logging synchronous (关闭显示同步);

R(c)#host R1 (设置路由器姓名);

STEP3: 配置接口:

首先配置R1:

R1(c)#default interface Serial 0 (恢复默认设置);

R1(c)#interface Serial 0 (打开接口);

R1(c-i)#ip address ~~12.0.0.1~~ (IP地址) 255.255.255.0 (子网掩码);

R1(c-i)#no shutdown (打开接口);

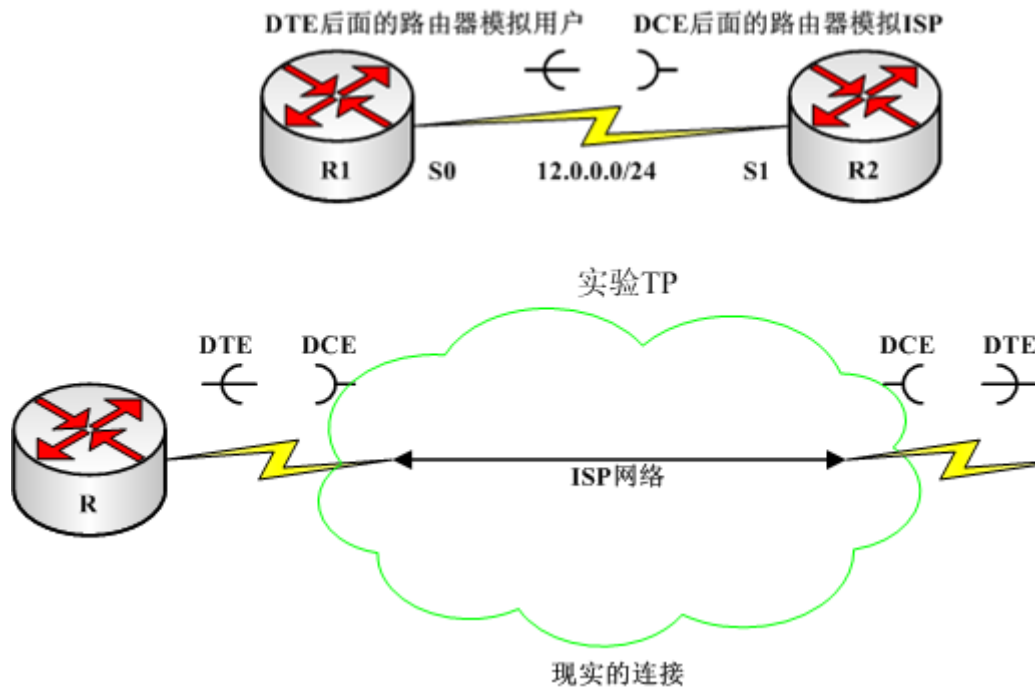
R2上在S1同样配置;

接口的查看: 注意sh ru in s0查看配置信息和sh in s0查看工作状态的差别;

STEP4: 配置同步时钟:

发现ping不通, 在sh in s0下显示L2 is Down; 同步串行链路的DCE端要配同步时钟! 配置命令: R1(c-i)#clock rate 2000000 (2000000最常用, 同步时钟范围一般为1200-4000000, 有些好的路由器可以达到640000或者更多);

同步时钟直接决定了同步串行链路的物理最大速率, 现实网络中ISP由此控制带宽!



0.1—NA路由②

0.1—NA路由②

CISCO常见的命令语法:

R(c)#`ip route network {mask} address/interface ;`
 参数 {} 可选项 _/_为选择

子网划分的原则:

主机位全0表示为网路号;

主机位全1表示为广播地址;

子网掩码的规则: 网路长度为: $8 \times n + m$, 则子网掩码为: 255.255 (n段).x.0(3-n段), x的取值: $n=1 \rightarrow 128$, $n=2 \rightarrow 192$, $n=3 \rightarrow 224$, $n=4 \rightarrow 240$, $n=5 \rightarrow 248$, $n=6 \rightarrow 252$, $n=7 \rightarrow 254$ 。

路由的简单分类:

直链路由: 就是到路由器接口所在的子网的路由, 条件是接口状态必须是ACTIVE (L1/L2全UP); 非直链路由: 从别的路由器通过路由协议学到的路由;

静态路由: 由网络管理员手工输入的路由, 可以有效降低路由器的CPU/RAM资源利用率, 但是手工输入工作量大而且无法适应网络的变化;

动态路由: 路由器根据网络拓扑和网络流量自动学到的路由, 可以降低网管的工作量并且能够自动调整路由表的条目和内容以适应网络的变化, 但是需要消

管的工作量并且能够自动调整路由表的条目和内容以适应网络的变化，但是需要消耗一定的CPU/RAM资源。

路由协议的分类：

距离矢量协议DV（Distance Vector）：RIP、IGRP；

链路状态协议LS（Link State）：OSPF、ISIS；

混合协议DV/LS协议：EIGRP。

主流的路由协议：

RIPv1/v2：用于小型网络（v1已淘汰）；

IGRP/EIGRP：用于中型网络（IGRP以淘汰），Cisco私有协议；EIGRP是唯一的DV/LS协议，拥有最快的收敛速度，支持多协议；

OSPF：用于大型网络，业界开放型标准；

ISIS：用于大型网络，业界开放型标准，扩展性好，同时支持IP和CLNS网络（只是支持而已但都不是很专业）。

IGP中的选路：

如果全网运行同一种路由协议，那么路由器会选择Metric（路由的开销）最小的；各种IGP的Metric标准：Rip→Hops；IGRP/EIGRP→Metric；OSPF→Cost；

如果全网运行不同的路由协议，那么路由器会选择AD（协议的管理距离）最小的；各种IGP的AD标准：Rip→120；ISIS→115；OSPF→110；IGRP→100；EIGRP→90；静态路由→1；直链路由→0。

主流的被路由协议：

IP/IPX：正在使用；

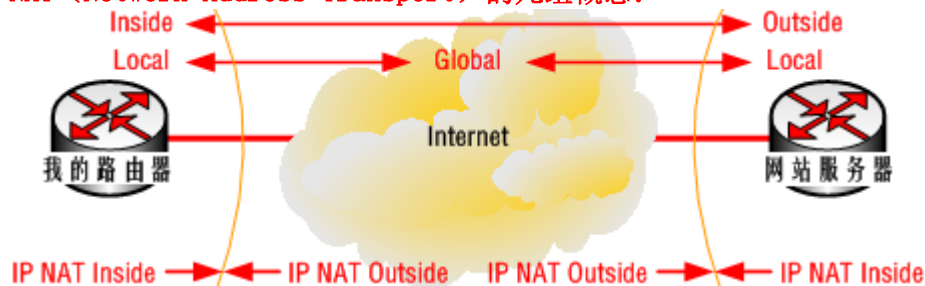
AT/CLNS/IBM/DEC：已淘汰或正在淘汰中。

访问控制列表ACL（Access-List）：

标准访问列表：编号1-99，只能控制源地址；

扩展访问列表：编号100-199，可以控制源/目标地址、端口、协议号等各种参数。

NAT（Network Address Transport）的几组概念：



LAB1: IP子网的划分:**STEP1: 网路要求:**

将195. 15. 3. 0划分为能容纳100/25/5个主机的子网, 而且划分两条点对点链路的地址:

STEP2: 确定主机位:

将所需要的网络自大而小的排列出来: 100/25/5/2/2 (最后二个为连接网段 Serial Link); 然后根据网络所拥有的IP数目确定每个子网的主机位: 如果2的N次方大于 (不含等于) 该网段的IP数目+2, 那么主机位就等于N。得: 7/5/3/2/2;

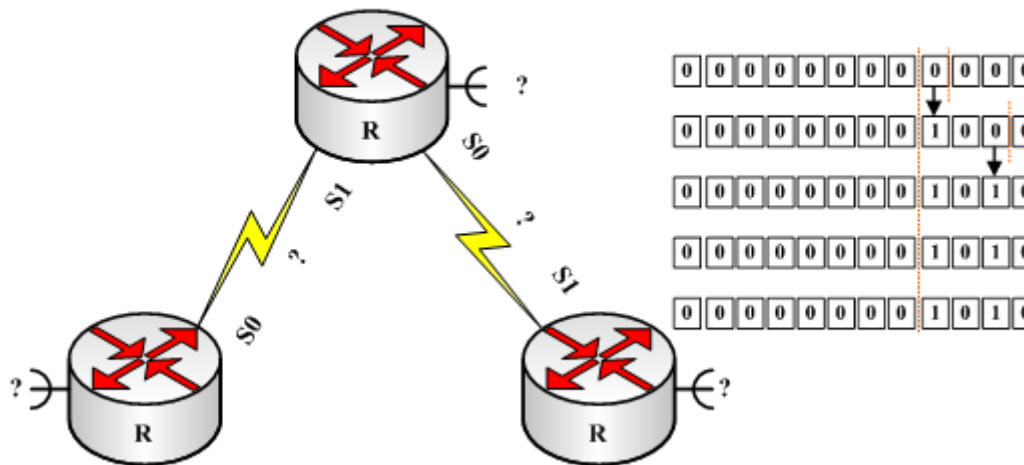
STEP3: 根据主机位决定网络位:

用32减去主机位剩下的数值就是网络位, 得: 25/27/29/30/30;

STEP4: 确定详细的IP地址:

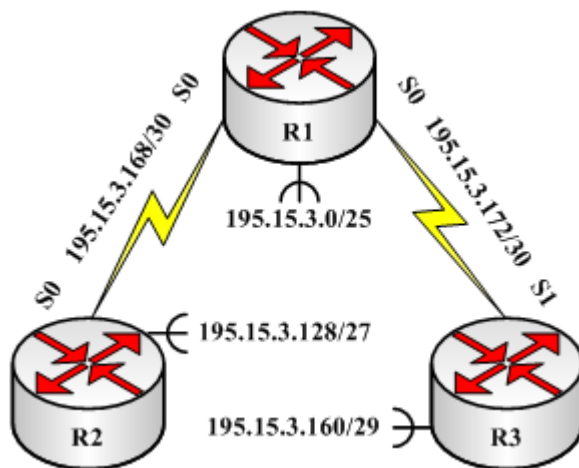
在2进制用网络位数值掩盖IP的前面相应位数, 然后后面的为IP位, 选取每个子网的第一个IP为网络号 (IP位全部都为0的默认为网络号), 最后一个为广播号 (IP位全部都为1的默认为广播号), 之间的为有效IP, 得:

网络号	有效IP	广播号
195. 15. 3. 0/25	195. 15. 3. 1/23--195. 15. 3. 126/25	
195. 15. 3. 127/25		
195. 15. 3. 128/27	195. 15. 3. 129/27--195. 15. 3. 158/27	
195. 15. 3. 159/27		
195. 15. 3. 160/29	195. 15. 3. 161/29--195. 15. 3. 166/26	
195. 15. 3. 167/29		
195. 15. 3. 168/30	195. 15. 3. 169/30--195. 15. 3. 170/30	
195. 15. 3. 171/30		
195. 15. 3. 172/30	195. 15. 3. 173/30--195. 15. 3. 174/30	
195. 15. 3. 175/30		



LAB2: 配置静态路由:**STEP1: 配置链路并且配置接口:**

用上个实验的IP设置, 网络段用LOOPback模拟, DCE要配时钟;

STEP2: 配置静态路由:配置各Route到所有不相邻网段的路由: R(c)#`ip route network {mask} address/interface` ;**STEP3: 检验 (ping ip) :****LAB3: 动态路由RIP (Request IP Protocol) :****STEP1: 配置链路并且配置接口:**用上个实验的设置, 删除静态路由 (`c#no ip routing` → `ip routing`) ;**STEP2: 配置RIP:**运行RIP: R(c)#`router rip`→`network` 需要宣告的直链网段 ;**STEP3: RIP的版本控制:**用R#`debug ip rip`查看RIP的运行, 发现v1的RIP封装的帧不携带子网掩码 (这也是RIPv1淘汰的原因) 所以无法适应现在的网路需求, 应该升级: R(c-r) `#version 2` ;**STEP3: 水平分隔:**

水平分隔 (Split Horizon): 路由发送包含整个路由表的更新信息浪费了带宽并不必要, 因此只发送路由矢量方向的路由, 也就是只延续收到的路由; 与路由矢量方向相反的路由是逆向路由 (Reverse Route), 默认被水平分隔阻挡; 水平分隔不仅节约了带宽, 而且避免了网络回路也就是把从邻居路由器学到的路由回发给邻居路由器; 分成简单水平分隔 (发送更新时接口不能发送本接口得到的跟新信息) 和毒性逆转水平分隔 (发送更新时通过指定跳数的inf无穷大来指定向该接口发送此更新信息的网络不可达);

STEP4: 查看RIP的路由:

STEP4: 查看RIP的路由:

```
show ip route rip ;  
show ip rip neighbour ;  
show ip rip topology ;  
路由表上RIP路由的标示为R;
```

STEP5: 观察动态路由协议的自动选路:

用E0连接R2/R3并配置接口宣告进RIP, 在R3扩展pingR2 (ping 目标ip source 源ip repeat n个包) 时sh掉R1/R3间的S接口来查看RIP的收敛;

LAB4: 动态路由EIGRP(Enhanced Interior Gateway Routing

Protocol) :

STEP1: 配置链路并且配置接口:

用上个实验的设置, 删除RIP (c-r#no router rip) ;

STEP2:配置EIGRP:

运行EIGRP, 同一个EIGRP网络的自治领域号AS (autonomous System) 必需相同: R(c)#router eigrp AS号→network 需要宣告的直链网段 ;

STEP3: 查看EIGRP的路由:

```
show ip route eigrp ;  
show ip eigrp neighbour ;  
show ip eigrp topology ;  
路由表上EIGRP路由的标示为D (E被EGP老爷子占了) ;
```

STEP4: 观察动态路由协议的自动选路:

用E0连接R2/R3并配置接口宣告进EIGRP, 在R3扩展pingR2 (ping 目标ip source 源ip repeat n个包) 时sh掉R1/R3间的S接口来查看EIGRP的收敛;

LAB5: 动态路由OSPF (Open Shortest Path Frist) :

STEP1: 配置链路并且配置接口:

用上个实验的设置, 删除EIGRP (c-r#no router eigrp AS号) ;

STEP2:配置OSPF:

运行OSPF, 同一个OSPF区域的PRO-ID必需相同, 但是Router-id必需全网唯一: R(c)#router ospf Pro号→router-id x.x.x.x→network 需要宣告的直链网段 反掩码 area x;

STEP3: 查看OSPF的路由:

```
show ip route ospf ;  
show ip ospf neighbour ;  
show ip ospf database ;  
路由表上OSPF路由的标示为O;
```

STEP4: 观察动态路由协议的自动选路:

用E0连接R2/R3并配置接口宣告进OSPF, 在R3扩展pingR2 (ping 目标ip source 源ip repeat n个包) 时sh掉R1/R3间的S接口来查看OSPF的收敛;

反掩码的格式与掩码相同, 但是作用不是标示网段而是指定路由器寻路时搜索接口的模糊搜索的范围, 匹配规则是0表示要准确匹配的位数而1表示可以不同的位数。OSPF中NET命令的反掩码不表示接口的网络长度, 而是表示运行OSPF的接口范围!

LAB6—1: 使用访问列表ACL (Access-List) 实现数据包的过滤:

STEP1: 配置链路并且配置接口以及IGP:

用上个实验的设置, 在R2上增加环回路口 (`in lo 1 → ip add 195.15.3.200 255.255.255.255`);

STEP2: 定义ACL:

`R(c)#access-list 1 deny 195.15.3.202 → access-list 1 permit any`;

STEP3: 在接口调用ACL过滤数据包:

`R(c-i)#ip access-group 1 in`;

LAB6—2: 使用扩展访问列表ACL (Access-List) 实现数据包的过滤:

STEP1: 配置链路并且配置接口以及IGP:

用上个实验的设置, 在R2上增加环回路口 (`in lo 1 → ip add 195.15.3.200 255.255.255.255`);

STEP2: 定义ACL:

`R(c)#access-list 100 deny ip any 195.15.3.202 → access-list 100 permit any any`;

STEP3: 在接口调用ACL过滤数据包:

`R(c-i)#ip access-group 100 in`;

LAB7: 网络地址转换NAT (Network Address Transport

(Access-List)):

STEP1: 配置链路并且配置接口以及IGP:

用上个实验的设置, 删除ACL设置 (`c-i#no ip access-group 100 in`);

STEP2: 定义NAT的内、外口:

);

STEP2: 定义NAT的内、外口:

R(c-i)#ip nat outside/inside ;

STEP3: 定义需要进行NAT的接口:

R(c)#access-list 10 permit 195.15.3.0 0.0.0.255 → access-list 10 deny any (系统默认Deny any, 此句可省略);

STEP4: 进行NAT:

R(c-i)#ip nat inside source list 10 interface serial 0 (NAT外口) overload (端口复用);

STEP5: 大型企业会购买一段地址, 使用pool实现负载均衡:

在outside的路由建立环回路口模拟买到的IP并宣告进IGP: in lo 200 → ip add 200.0.0.1 255.255.255.0 → ip ospf network point-to-point;

然后把ip放进pool: ip nat pool xx 200.0.0.1 200.0.0.4 prefixlength 28 ;

最后调用pool: ip nat inside source list 10 pool xx overload ;

0.2—NA交换①

0.2—NA交换①

常用的交换设备:

交换机 (ASIC) 和网桥 (Bridge);

交换机的三种转发方式:

直通式 (Cut-Through): 一旦检测到MAC即转发, 速度快但是无法保证准确性;

存储转发 (Store and Forward): 将数据包完整接收并缓存于Cache后才转发, 速度稍慢但是准确性有保障;

即存即发 (Fragment Free): 交换机会检测帧的前64个字节如果正确就开始转发, 并且在转发时不断检测数据帧, 是折中方案。

交换机的三大功能:

MAC地址的学习 (MAC Address Learning) → 交换机有一张MAC地址表, 记录了交换机端口和与端口相连的主机的MAC地址 (也就是数据帧的源MAC地址和目标MAC地址) 的匹配 (Mapping) 关系;

数据帧的转发/过滤机制 (Forward/Filter Decision) → 交换机是根据数据帧的目标MAC地址进行转发的: 如果目标MAC地址是已知的单播 (Unicast) 地址则使用目标MAC匹配的接口转发; 如果目标MAC地址是未知的单播地址则向除了入口的所有接口广播, 找到后同上 (第一次Ping会丢一个包, 其实包送到了但是回包被交换机用来建立MAC表映射了, 所以路由器不能接收到回包, 因此路由器认为丢了一个包; 以后Ping就不显示丢包了因为MAC表已经建立); 如果目标MAC地址匹配的接口是数据帧的入口则丢弃该包; 如果目标MAC地址是组播/广播

地址匹配的接口是数据帧的入口则丢弃该包；如果目标MAC地址是组播/广播（Multicast/Broadcast）地址则向除了入口的所有接口广播；
避免环路（Loop Avoidance）→在冗余网路中避免网络回路（STP）。

交换的广播地址：

FFFF.FFFF.FFFF.FFFF（就是所有位数全是1，与Mask同）。

以太口的种类：

Ethernet：10MPS的接口；

FastEthernet：100MPS的接口；

GiFastEthernet：1000MPS的接口；

LAB1：观察交换机的MAC地址表（与ARP查询）：

STEP1：连接交换机：

插上电源用con线连接交换机和主机并使用终端软件；

STEP2：常用命令：

```
#show version ;
```

```
#show running-config ;
```

```
#show interface fastethernet 0/1 ;
```

```
#show interface status ;
```

STEP3：查看MAC地址表：

```
#show mac-address-table dynamic ;
```

STEP4：查询对方的地址：

```
C:\>ping..... →arp -a
```

LAB2：交换机的基本配置：

STEP1：接通链路：

将电源接好，将con线（交叉线）的RJ-45接头插入路由/交换机的con口，DB-9插入笔记本的DB-9异步串口（RS-232）；

STEP2：启动终端：

五个参数：每秒数据—9600，数据位—8，校检—无，停止位—1，流控—无；

STEP3：开机：

是否载入配置文件：No；然后默认进入用户模式SW>，只能查看简单的信息不可以操作；

STEP4：进入特权模式R#：

Enable：可以查看高级信息了；

STEP5：进入配置模式R(config)#：

Configure terminal: 可以修改配置了；可以用？查看可用的命令，用TAB自动补全命令；通过配置命令可以进入其他模式如接口模式S(config-if)#和协商模式S(config-router)#等等；

STEP6: 常用开机命令:

S>enable (进入特权模式)；
 S#configure terminal (进入配置模式)；
 S(c)#no ip domain-lookup (关闭域名查询)；
 S(c)#line console 0 (配置con口)；
 S(c-i)#no exec-timeout (设置为永不超时，与命令exec-t 0 0效果相同，默认为exec-t 10 0即10分钟无操作即超时断开连接，工作一般设置为3 0)；
 S(c-i)#logging synchronous (关闭显示同步)；
 S(c)#host SWX (设置交换机姓名)；
 可以用CTRL+Z快速返回特权模式；

STEP7: 配置密码对交换机进行网管:

如果需要对交换机进行远程控制则还需要配置网关: (c)#ip default-gateway 192.168.2.2；

为交换机配置网管IP: R(c)#interface vlan 1 → ip address 192.168.2.100 255.255.255.0 → no shutdown；

STEP8: 对交换机发动Telnet管理:

C:\>telnet IP 交换机拒绝了: 没有VTY密码, 如下设置:
 S(c)#line vty 0 4 (共五个VTY厕所蹲位)；
 S(c-i)#password ***；
 S(c-i)#login (可以不加系统会自动进入, 如果改成no login就会不需要密码即可以TELnet)；

STEP9: 配置特权模式的密码:

S(c)#enable password/secret *** (配置明文/密文密码, 设置密文会覆盖已有的明文密码)；

STEP10: 然后再TELnet:

C:\>telnet IP 然后输入密码后成功……不要更改自己进入Telnet设备使用的接口的参数；

S#exit 退出TELnet；

STEP11: 交换机的接口工作状态:

S(c)#interface fastethernet 0/1 (进入接口)；
 S(c-i)#description connect to sh (贴个标签描述接口作用)；
 S(c-i)#switch mode trunk/access (交换机将连接交换机/连接主机)；
 S(c-i)#speed auto/xxx (交换机的速率控制: 默认/指定为xxx)；
 S(c-i)#duplex auto/full/half (交换机的工作模式: 默认/双工/半双工)；
 S(c-i)#ip address 12.0.0.1 255.255.255.0 (配置IP)；
 S(c-i)#no shutdown/shutdown (打开/关闭接口)；

STEP12: 端口安全:

S(c-i)#switchport mode access (接口将要连接PC)；
 S(c-i)#switchport port-security (启动端口安全)；
 S(c-i)#switchport port-security mac-address 00e0.abcd.1234 (设置允许的MAC)；

S(c-i)#switchport port-security mac-address 00c0.abcd.1234(设置允许的MAC);

S(c-i)#switchport port-security violation protect/restrict/shutdown(在发现接口被非法TelNet后: 抑制接口/抑制接口并记载/关闭接口并记载);

可以在一个端口上绑定多个允许的MAC地址: S(c-i)#switchport port-security macimum 3 → switchport port-security mac-address 00c0.abcd.1234 →.....

这样工作量会很大, 可以使用黏性工具减小: switchport port-security macimum 100 → switchport port-security mac-address sticky ; 激活黏性工具后进入过该交换机的MAC地址都将定义为合法。

LAB3: Cisco交换机的文件处理:

STEP1: 确定链路的通达;

S#ping IP ;

STEP2: 运行TFTP服务器(采用TFTPd32);

首先运行TFTP服务器(双击图标不用介绍吧); 然后查看文件夹: S#dir nvram(查看Nvram的配置文件: startup-config); S#dir flash: (查看Flash的IOS文件: 16423684);

STEP3: 配置文件的管理:

S#write/copy running-config nvram:startup-config(备份或者说存盘配置文件);

S#copy nvram:startup-config running-config(读取启动文件文件);

S#copy nvram:startup-config tftp:(将NVRAM的启动文件备份到TFTP);

S#copy running-config tftp:(将RAM的配置文件备份到TFTP);

S#copy tftp: nvram:startup-config(调用TFTP到NVRAM的启动文件);

S#copy tftp: running-config(调用TFTP到RAM的配置文件);

STEP4: IOS的升级:

S#copy flash: tftp:(备份IOS);

S#copy tftp: flash:(升级IOS);

注意: 确认升级成功才可以重启!

LAB4: Cisco交换机的密码恢复:

STEP1: 确定链路的通达;

使用con线(只能通过con线恢复密码)连接好console口, 确认笔记本能够正常显示交换机的信息;

STEP2: 重启交换机:

拔掉电源线重启交换机, 在重启开始的3-8秒内按住交换机面板上的Mode键

够正常显示交换机的信息;

STEP2: 重启交换机:

拔掉电源线重启交换机, 在重启开始的3-8秒内按住交换机面板上的Mode键直到Fa0/0的指示灯灭掉后松手, 这时候进入只读监控的ROM MONITOR模式(相当于windows的安全模式, 特点是显示 switch:);

STEP3: 在交换机中调用启动文件到内存中:

输入: flash-init → boot-loader → loader-helper → dir flash: ;

然后把startup.config改名(改成交换机不认识的名字): rename

config.text xx.text ;

再开机后发现不载入启动文件也就是进入全默认状态→不需要密码即可以进入, 用S#copy nvram:xxx.text running-config 读取启动文件;

STEP4: 删除原密码并设置新密码:

S(c)#no enable password/secret xxx ;

S(c)#enable password/secret xxx ;

这时可以删除原启动文件也就是xxx.text: S#delete flash: xxx.text 。

0.2—NA交换②

0.2—NA交换②

虚拟局域网VLAN的核心目的:

将一个大的网络划分为小的网络, 也称为网络分片 (Segementation); 一个VLAN对应着一个广播域, 最好对应一个网络子网(为VLAN间的路由作准备)。

HUB: 所有端口都在一个冲突域, 一个广播域中;

Switch: 一个端口对应一个冲突域, 所有端口都在一个广播域中;

Router: 一个端口对应一个冲突域, 一个端口对应一个广播域;

VLAN的分类:

静态VLAN (Static VLAN): 由MAC表建立的VLAN→匹配信息存储在MAC表;

动态VLAN (Dynamic VLAN): 由服务器VMPS建立的VLAN→匹配信息存储在VMPS。

CDP (Connect Discover Protocol):

CDP是CISCO私有的协议, 只可以发现直链的设备!

Trunk:

Trunk可以在一条网络介质Segement (网线/光纤) 上同时传输多个VLAN的信息 (1次最多可以携带1005个VLAN的信息), 必须使用100Mbps以上的端口来进行点对点连接, Trunk使你的单独的1个端口同时成为数个VLAN的端口, 这样可以不需要L3设备;

如果在Switch之间使用了Trunk, 那么多个VLAN间的信息将从这个连接上通过; 如果没有使用Trunk而使用一般的连接, 那么将只有VLAN1的信息能通过这个

点对点连接，Trunk使你的单独的1个端口同时成为数个VLAN的端口，这样可以不需要L3设备；

如果在Switch之间使用了Trunk, 那么多个VLAN间的信息将从这个连接上通过；如果没有使用Trunk而使用一般的连接，那么将只有VLAN1的信息能通过这个连接传递：VLAN1默认作为管理VLAN。

Trunk的种类：

- 1、cisco的私有标准：ISL ；
- 2、开放性的业界标准：802.1Q 。

VTP (VLAN Trunk Protocol)：

在一个交换机（VTP-Server）设定好Vlan信息，让别的交换机（VTP-Client）都能够自动从这个交换机动态地学习到VLAN信息的协议；VTP用于构建富有效率、便于管理的Vlan网络；VTP信息只能在trunk链路上传输，无法完成将端口放入VLAN的操作；

VTP的主要目的是在一个交换性的环境中使所有的VLAN容易保持一致性：现在只需要在VTP-Server上增加、删除和重命名VLAN，然后这些修改信息会自动传播到整个VTP域里的所有交换机上；

VTP是Cisco私有的协议，是Cisco创建的，但是现在已经不再为Cisco所私有。

冗余网络（Full-Mesh）带来的核心问题和三个表现：

核心问题是网络环路；

三大表现：

多帧复制（Multiple Frame Copies）→未知MAC地址的数据帧在交换机间不断的复制转发复制转发……直到找到该目标地址构建MAC映射，这时该目标已经收到了N个同样的数据帧了；

MAC表地址跳动（MAC-Address Instability）→因为几个接口都能够连接同一个PC因此MAC表中该设备的MAC值匹配的接口不断在几个接口间跳动；

广播风暴（Broadcast Storms）→广播会永无休止的发送下去占用越来越多的资源直到资源耗尽网络堵塞交换机Down机。

生成树协议STP (Spanning-Tree Protocol) 的规则：

One root bridge per network →每个交换网络，都有一个根桥Root-Bridge（根交换机：Bridge-ID最小的交换机或优先级最小的）；

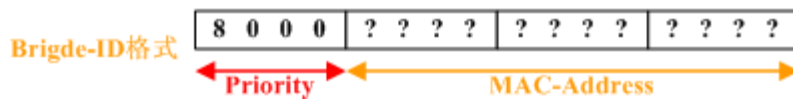
One root port per non-root bridge →每个非根桥，都有一个根端口（根端口：去往根桥开销最小的端口）；

One designated port per segment →每条网络介质，都必定有一个指定端口（指定端口，根端口都是转发数据包Forwarding状态的）；

Nondesignated ports are unused →指定端口，根端口以外的端口，都被禁用，不转发数据包。

Bridge-ID：

是8BIT的ID，由优先级+MAC地址组成，优先级一般默认为0x8000（32768）；



LAB1: VLAN的基本操作:

STEP1: 在R1/R2上关闭路由能力模拟PC:

在(c)#no ip routing后和路由器上一样配置IP地址;

STEP2: 查看VLAN的基本信息:

#show vlan brief → show interface status → show mac-address-table
dynamic ; 默认情况下所有的接口都在VLAN1;

STEP3: 创建VLAN:

(c)#vlan * → name *** (老旧版本中: #vlan database → vlan * name
*** → vlan * name *** → → apply → exit ; 区别就是要用vlan database
进入VLAN区而新版本直接可以进入);

STEP3: 将端口划分入VLAN:

(c-i)#switchport access vlan * ;

可以使用Range一次定义多个接口: in range fastethernet 0/1 - 5 , 0/9
(注意空格!);

LAB2: 跨交换机的同一VLAN通讯:

STEP1: 构建Trunk:

Trunk有两种:

802.1Q的Trunk: (c-i)#switchport trunk encapsulation dot1q
→ switchport mode trunk ;

ISL的Trunk: (c-i)#switchport trunk encapsulation isl → switchport
mode trunk ;

两个交换机要配置相同的Trunk;

STEP2: 察看Trunk链路的连接状态:

#sh in trunk ;

STEP3: 在PC上设置网关:

(c)#ip default-gateway 192.168.10.100 ; 设置了网关后就可以实现跨
vlan、跨交换机的通信。

LAB3: VTP:

(c)#vtp mode server/client (指定为VTP的服务器/客户端) ;

STEP2: 确定在一个相同VTP域名:

(c)#vtp domain CCNA ;

STEP3: 检查vtp:

#sh vtp status (注意: 所有vtp信息在sh run是看不到的) ;

STEP4: 观察VTP信息的同步过程:

在VTP-Server添加vlan观察版本号(Configuration Revision 2): 每进行一个VLAN操作时如增加、删除、改名等, 版本号就会加1。

LAB4: STP根桥的选定:

STEP1: 察看各个交换机的MAC地址:

察看MAC地址: #sh version →Base ethernet MAC

Address:00:0F:.....;

察看优先级: #sh spanning-tree →Bridge id priority 32768 (priority 32768 sys-id-ext 1);

还有sh cdp nei也能察看到MAC地址;

STEP2: 直接指定根桥:

根桥: sw1(config)#spanning-tree vlan 1 root primary; (默认PRI=24576/0x6000);

备份根桥: sw2(config)#spanning-tree vlan 1 root secondary; (默认PRI=28672/0x7000);

STEP3: 还可以更改优先级:

通过#spanning-tree vlan 1 priority 4096 。

0.3—NA远程

0.3—NA远程

远程网络按照L1分类:

租用专线 (Leased Line): 一般采用同步串行链路, 使用HDLC/PPP封装;

线路交换 (Circuit-Switched): 一般采用异步串行链路, 使用HDLC/PPP封装;

分组/包交换 (Packet-Switched): 一般采用同步串行链路, 使用Frame-Relay/ATM/X.25封装。

同步串行链路 (Serial Point-to-Point Link) 的封装:

HDLC: 不支持多协议, Cisco为了使它支持多协议在其帧格式上增加了一个私有位→虽然支持多协议了但是与其他品牌不兼容了! 所有的Cisco设备都默认采用HDLC; HDLC不支持认证, 无法保证链路的安全性;

默认采用HDLC；HDLC不支持认证，无法保证链路的安全性；

PPP：支持多协议，是**两层半协议**也就是拥有L3网络层功能的L2链路层协议，主要由NCP（IPCP+CDPCP）和LCP两部分组成；它拥有四大模块：认证（Authentication）/压缩（Compress）/捆绑（Multi-Link）/纠错（Error-Detection）/回拨（Call Back）。

PPP的认证：

∴PPP是L2协议∴PPP的认证是链路认证→一旦认证成功就接通链路在链路Down掉以前都不再认证了；PPP认证有两种：

明文密码认证PAP（Password Authentication Protocol）：二次握手（**被认证方发送账号密码→主认证方对比账号密码后根据结果作出回应**）；PAP的账号密码在网路上传播，不安全；

密文密码认证CHAP（Challenge Handshake Authentication Protocol）：三次握手（**主认证方发送乱码→双方同时使用MD5算法运算并且被认证方发送结果→主认证方对比运算结果后根据结果作出回应**）；CHAP的账号密码不会在网路上传播，安全性好，推荐使用。

帧中继（Frame-Relay）：

帧中继是以虚链路VC（Virtual Circuits）为基础的**纯L2协议**：必需要有事先建立好的虚链路才可以实现连通；虚链路VC主要有两类：永久性虚链路PVC（Permanent VC）和交换性虚链路SVC（Switch VC）。

帧中继的接口类型：

帧中继的接口有用户网路接口UNI（User-Network Interface）和NNI（Network-Network Interface）两大类，UNI又分成DCE和DTE（**帧中继的客户端永远是DTE端**）；PVC两端的接口还应配置DLCI（Data-Link Connection Identifier）号用于FR的寻路。



帧中继的信令（Signaling）模式：

帧中继的信令使用本地管理接口协议LMI（Local Management Interface），有三种模式：

Cisco兼容；

ANSI T1.617 Annex D；

ITU-T Q.933a Annex A；

12.0以后的IOS都拥有检测LMI格式的能力→可以不指定。

PVC的连接模式：

冗余连接（Full-Mesh）：每两个节点之间都有直接相连的连接，N个节点的

的Full-Mesh网路有 $N(N-1)/2$ 个连接；Full-Mesh的可靠性极好但是成本极高，是高端用户的必然选择；

星型网路（Hub&Spoke）：每两个节点（Spoke）都只直连主机（Hub），N个节点的Hub&Spoke网路有N-1个连接；Hub&Spoke成本低但是冗余性不好，而且存在因为水平分隔导致的一个接口有多条PVC时路由不能全面通达的问题（用帧中继子接口解决）。

PVC状态：

选举（active）、自己没配好（deleted）、对方没有配好（inactive）。

帧中继的映射表（Mapping Table）：

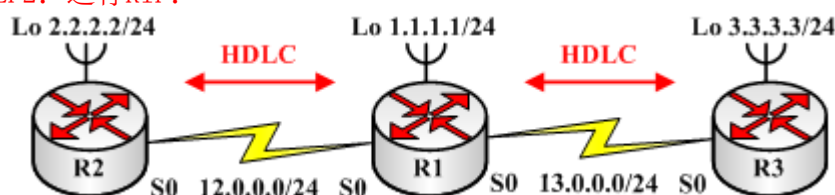
描述的是本机接口的DLCI号（FR的L2地址）与对端接口的IP（Route的L3地址）的匹配关系。

LAB1：使用HDLC封装点对点链路：

STEP1：在接口之间封装HDLC（默认的）：

构建拓扑，查看即可（sh in s 0）：L1 is up, L2 is up……Encapsulation HDLC；

STEP2：运行RIP：



LAB2：使用PPP封装点对点链路：

STEP1：在串行链路的接口之间封装PPP：

接上一个拓扑，在R1/R3之间的接口：(c-i)#encapsulation ppp；

可以打开Debug后SH/no SH接口来查看PPP的协商：#debug ppp

encapsulation : L1 State is up → LCP State is open → IPCP State is open → CDPCP State is open → L2 State is up；

再查看接口信息（sh in s 0）：L1 is up, L2 is up……Encapsulation PPP；

观察完后最好关闭Debug：#no debug ppp encapsulation；

STEP2：在R3和R1之间互Ping：

能通的！为什么？路由的工作原理……

路由 (Route v. 动词)：核心词Forwarding转发，过程为路由器从入接口收到信息后剥掉L2帧头FRAME（源地址和目标地址是前一个发此信息帧的路由器和接收此信息帧的本路由器），然后查看L3层包头PACKET（源地址和目标地址是发此信息包的根源路由器和要接收此信息包的最终路由器），接着查询自己的路由表找到到目标地址的出接口和下一个路由器（下一跳），并按照两者之间的封装方式重新封装L2帧头FRAME（源地址和目标地址是发送此信息帧的本路由器和下一个要接收此帧的路由器）并从出接口发出此帧；不断封装→解封装→封装→解封装→封装；跳数 (Hops) 因此影响到路由的开销。

LAB3: PPP的认证:

STEP1: 进行PAP认证:

首先确定R1/R3链路已经是PPP封装 (sh in s 0) : L1 is up, L2 is up.....Encapsulation PPP ;

然后在双方的路由器上为对方建立账号/密码: (c)#username R3N password R3P 和..... ;

接着在双方接口上选定认证方式为PAP: (c-i)#ppp authentication pap ;

最后在双方接口上输入账号/密码: (c-i)#ppp pap sent-username R1N password R1P 和..... ;

可以打开Debug后SH/no SH接口来查看PPP的协商: #debug ppp encapsulation ;

在R3和R1之间互Ping能通的，为什么；

STEP2: 进行CHAP认证:

这次在R1/R2上做，首先确定链路之间已经是PPP封装，不是就改 (c-i#en p) ;

然后同样是在双方的路由器上为对方建立账号/密码: (c)#username R2 password R1R2P 和.....，注意双方的密码必需一致哦，不然MD5运算的结果必定不同的，还有账号要用对方的路由器名字；

接着在双方接口上选定认证方式为PPP: (c-i)#ppp authentication chap ;

可以打开Debug后SH/no SH接口来查看PPP的协商: #debug ppp encapsulation 。

LAB4: 帧中继的基本配置:

STEP1: 配置帧中继交换机:

首先把一个路由器R2变成帧中继交换机: (c)#no ip routing →frame-relay switching ;

然后在接口封装帧中继: (c-i)#encapsulation frame-relay ;

接着指定接口类型，而且由于FR-sw总是充当DCE所以还需要配置时钟，当然

然接口也要打开: (c-i)#frame-relay intf-type dce →clock rate 2000000
→no shutdown;

STEP2: 在帧中继交换机上配置FR路由:

要在两边接口有去有回的配置: (c-i)#frame-relay route ~~入PVC~~
interface 出接口 出PVC ; 然后可以用#show frame-relay route查看;

STEP3: 配置用户端接口:

首先在接口封装帧中继: (c-i)#encapsulation frame-relay ;

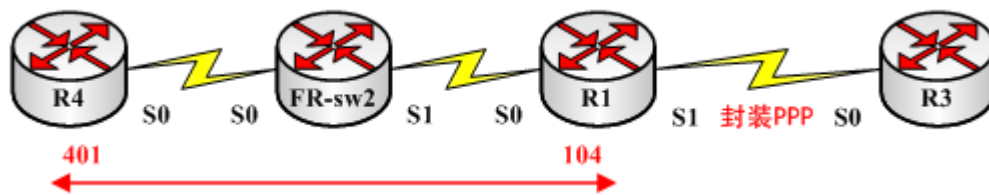
接着指定接口类型并打开: (c-i)#frame-relay intf-type dte (已默认, 可以不指定) →no shutdown;

嘿嘿, 路由的接口可别忘记配IP了……

查看命令: #show frame-relay pvc 和#show frame-relay map 。

STEP4: 测试链路:

设置环回路口运行IGP然后Ping: ! ! ! ! !



Wolf-NP笔记

CISCO私有的: EIGRP、CDP、ODR、IGRP、VTP、HSRP、PVST。

Routing

LV1(必须掌握)

- 4、FrameRelay
- 6、RIP
- 8、ospf
- 9、eigrp
- 10、bgp
- 11、route redistribution
- 12、acl
- 13、pbr
- 14、cdp
- 15、nat
- 16、hsrp

LV2:

- 设备IOS的升级 / 备份
- 设备的密码恢复
- ISDN (用于链路备份)
- IGRP (已经淘汰)
- NTP

1.0—路由的基础知识和静态路由

1.0—路由的基础知识和静态路由

网络的分层：

Access layer：终端用户的接入；
Distribution layer：实现基本的网络策略控制；
Core layer：高速、高效的传送数据包。

网络划分的种类：

可以按照网络的不同的功能部门进行划分；
可以按照网络所分布的地域范围进行划分。

网络拓扑：

Full Meshed / 全互连

任意两点（节点）之间，都有直接相连的连接。

Full Meshed连接，所需连接个数的公式：

公式： C_n^2 （上标） n （下标） $=n(n-1)/2$

∴：在核心网中，构建双冗余的Full Mesh是较为昂贵， $N*(N-1)$

∴：可以考虑构建双Hub&Spoke。 $2*(N-1)$

既可以保持冗余性，也可以降低建网成本。

一个规划良好的网络中，通常考虑到以下3点：

Scalability / 可扩展性

Predictability / 可预测性

Flexibility/灵活性

Benefits of Hierarchical Addressing/网络地址层次化的划分：

- 1：在路由器上，有更高的路由转发效率，减少路由表的路由条目。
- 2：提高地址的利用率。

VLSM: Variable-Length Subnet Mask(可变长子网掩码)

VLSM:

可以实现根据不同网络需求，拆分 / 划分不同大小的子网，

可以实现根据不同网络需求，拆分 / 划分不同大小的子网，

特别注意：

VLSM不能凭空额外的创造出更多的IP地址来。

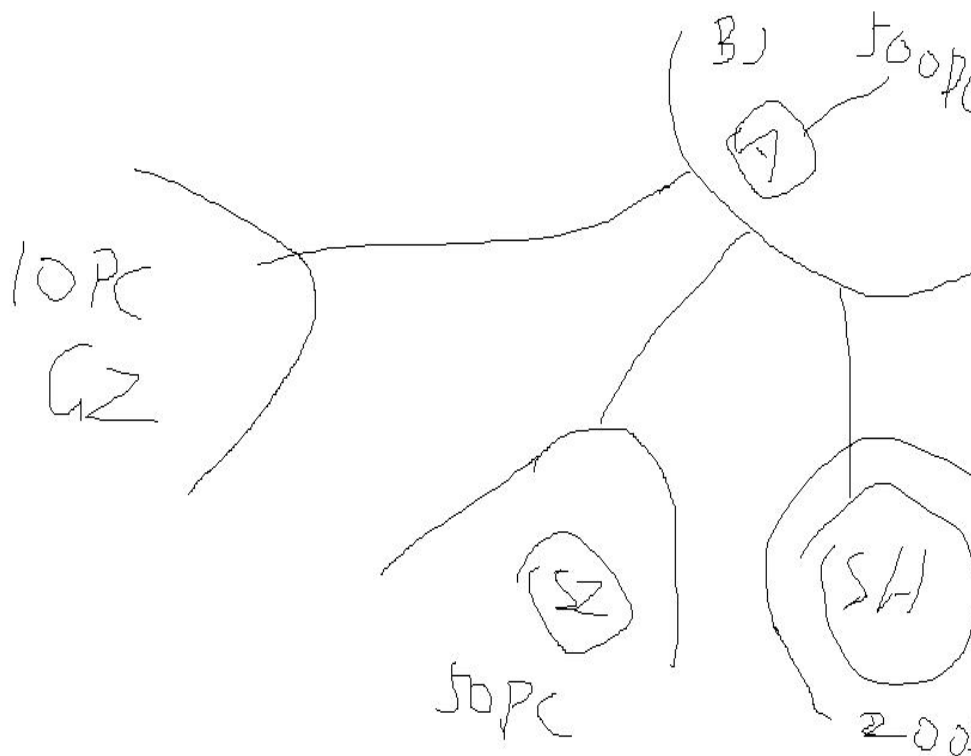
Step1:将所需要划分的网络，自大而小地排列出来。

195.15.20.0 / 22 （有效IP：1024-2=1022）主机位有10位 2的10次方
=1024

（32位-网络

位）

500 / 200 / 50 / 10 / 3 (Serial Link)



500IP: 9个主机位 / Host Bits

200IP: 8个主机位

50IP: 6个主机位

10IP: 4个主机位

Serial Link: 2个主机位

step3: 根据每个子网的主机位, 算出子网的网络位:

建议: 在选择网络位时, 先取0, 再取1.

500IP: \therefore 有23个网络位 ($32-9=23$)

200IP: \therefore 有24

50IP 有26

10 有28

2 有30

Step4: 计算每个子网的:

网络号, 广播地址, 首个有效地址, 最后一个的有效地址

原则:

如果主机位为全0, 那么这个地址为这个子网的网络号。

如果主机位为全1, 那么这个地址为这个子网的广播地址。

建议: 在选择新增的网络位时, 先取0, 再取1。

能够容纳500个主机的子网:

网络号: 195. 15. 20. 0 / 23

第一个有效IP: 195. 15. 20. 1 / 23

最后一个有效IP: 195. 15. 21. 254 / 23

广播地址: 195. 15. 21. 255 / 23

能够容纳200个主机的子网:

网络号: 195. 15. 22. 0 / 24

第一个有效IP: 195. 15. 22. 1 / 24

最后一个有效IP: 195. 15. 22. 254 / 24

子网的广播地址: 195. 15. 22. 255 / 24

能够容纳50个主机的子网:

网络号: 195. 15. 23. 0 / 26

第一有效IP: 195. 15. 23. 1 / 26

最后一个有效IP: 195. 15. 23. 62 / 26

子网的广播地址: 195. 15. 23. 63 / 26

能够容纳10个主机的子网:

网络号: 195. 15. 23. 64 / 28

第一个有效的IP: 195. 15. 23. 65 / 28

最后一个有效的IP: 195. 15. 23. 78 / 28

子网的广播地址: 195. 15. 23. 79 / 28

NO.1 Serial Link:

网络号: 195.15.23.80 / 30 有效IP: 81 / 82

NO.2 Serial Link:

网络号: 195.15.23.84 / 30 有效IP: 85 / 86

NO.3 Serial Link:

网络号: 195.15.23.88 / 30 有效IP: 89 / 90

Route Summarization/路由汇总:

~~~~~

路由汇总的总体思想:

在需要进行汇总的明细路由中,

寻找前部相同的位数, 以这一点为分界点, 将明细路由汇总到这一位, 形成了汇总路由。

Step1: 按照IP路由的4个字段, 寻找相同 / 不同点的分界。

Step2: 将第一个不同的字段, 展开为2进制数, 进一步寻找相同 / 不同点的分界点。

Step3: 将不同的位数, 置为全0

以相同的位数作为汇总路由的路由长度。

172.16.12.0 / 22

#### CIDR: (Classless Interdomain Routing) / 无类域间路由

可以突破网络的边界。

CIDR是可以突破网络主类边界, 实现网络汇总。

#### CIDR:

Block addresses can be summarized into single entries without

without regard to the classful boundary.

### Routing Principles/路由原理

静态路由:

router(config)#

**ip route** *prefix mask {address | interface} [distance]*

IOS语法:

正体字: 表示需要准确的IOS命令 (不变) (可以缩写)

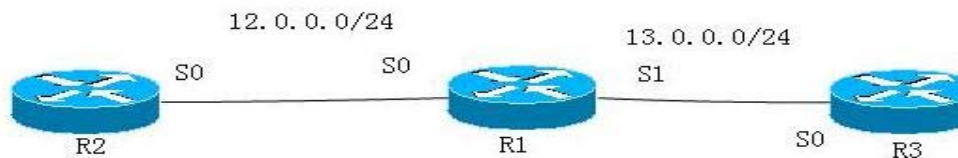
斜体字: 根据实际情况, 输入的参数 (变化)

{        }: 必选项

[        ]: 可选项

| :        OR, 或关系, 多个元素中, 选择一个

### LAB1: 静态路由的原理性实验:



### 理论部分:

NO.1: 关于直链路由:

如果物理接口的链路状态: L1 / L2都能够UP / UP

那么接口所在的网段, 就以直链的形式出现在路由表中。

R1#show ip interface brief (察看接口简要IP信息。)  
UP/UP

R1#show ip route

```
R1#show ip route connect
```

```
R2#debug ip packet （察看IP数据包）
```

**特别注意：**

路由器不是基于接口考虑数据包的路由，而是基于整个路由的路由表，进行数据包的路由。

**NO.2: IP路由的核心原理（路由对联）：（路由的单向性）**

上联：“去往目标网络的路径上的”所有路由器，都要有，去往目标网络的路由。

（保证数据包能够送到目标网络）

下联：：“在返回源网络的路径上的”所有路由器都要有，返回源网络的路由。

（保证数据包能够从目标网络返回源地址）

横批：有去有回，肯定能通！

**NO.3: 路由的下一跳的可达性：**

对于每条路由，都必须检查其路由的下一跳是否可达，  
如果下一跳不可达，那么这条路由会被路由器从路由表中删除。

```
R2#clear ip route * （reset/复位路由表）
```

**NO.4: 路由的单向性：**

IP路由是单向的。

IP数据包所经过的路径，有可能往返是重叠的，也有可能是不同路径的，取决于路由的指向。

**实验部分：**

**Step1: 查看接口的工作状态**

```
RI#show ip interface brief
```

**Step2: 查看直链路由**

```
ri#show ip route
```

c 直链路由

```
c 12.0.0.0 is directly connected, serial0
c 13.0.0.0 is directly connected, serial1
```

Step3: R2 ping R1 s0 ! ! ! ! !

Step4: R2 ping R1 s1  
(unroutable:原因是R2没有13.0.0.0这条路由)  
R2(config)#ip route 13.0.0.0 255.255.255.0 12.0.0.1 ! ! ! ! !

Step5:R2 ping R3 s0  
原因是: R3没有去住12.0.0.0网段的路由, 所以无法回包  
R3(config)#ip route 12.0.0.0 255.255.255.0 13.0.0.1 ! ! ! ! !

Step6:R2 ping R3 s1  
(unroutable:原因是R2没有35.0.0.0这条路由)  
R2(config)#ip route 35.0.0.0 255.255.255.0 12.0.0.1  
(unroutable:原因是R1没有去往目标网段的路由)  
R1(config)#ip route 35.0.0.0 255.255.255.0 13.0.0.3 ! ! ! ! !

step7:R2 ping R5 s0  
原因是R5没有返回12.0.0.0网段的路由。  
R5(config)#ip route 12.0.0.0 255.255.255.0 35.0.0.3  
尽管R5没有去住中间网段13.0.0.0的路由, 但只要满足“路由对联”, 就可以成功通信。

Step8:R5 ping R2 s0 ! ! ! ! !

Step9:R5 ping R1 s1  
(unroutable:原因是R5没有13.0.0.0网段的路由)  
R5(config)#ip route 13.0.0.0 255.255.255.0 35.0.0.3 ! ! ! ! !

Step10:更改路由的下一跳  
R3(config)#ip route 35.0.0.0 255.255.255.0 13.0.0.3

路由的递归查询:  
S 35.0.0.0 [1/0] Via 13.0.0.3  
S 13.0.0.0 [1/0] Via 12.0.0.1  
C 12.0.0.0 is directly connected, serial0

R2#ping 35.0.0.5 ! ! ! ! !  
一旦删除13.0.0.0的路由, 因为35.0.0.0网段的路由的下一跳变得不可达  
所以35.0.0.0网段的路由条目, 将在路由表中消失。  
R2#clear ip route \* (reset/复位路由表)

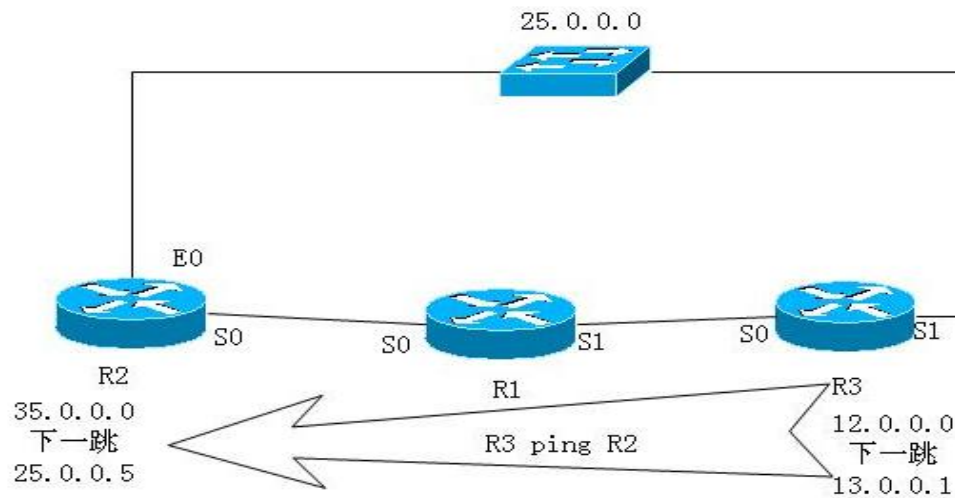
Step11:路由的单向性

达

所以35.0.0.0网段的路由条目，将在路由表中消失。

R2#clear ip route \* (reset/复位路由表)

### Step11:路由的单向性



指定源IP:25.0.0.2, 目标IP:13.0.0.3

Q1: 能否通达?

### LAB2:静态路由的两种下一跳的写法:

2-1: 对于点对点链路(串行链路 / 点对点接口), (不适用MA网络)  
以自己的接口为下一跳的出口。

R2 (config) #ip route 13.0.0.0 255.0.0.0 serial 0  
己方的出接口

2-2: 对于MA: multi-access(多路访问网络 / 以太网)

不建议使用以自己的出接口为下一跳的路由,

而应该使用准确的下一跳转发地址:

R2 (config) #ip route 13.0.0.0 255.0.0.0 12.0.0.1 (适合于所有的链路)

### LAB3:默认路由:

R2(config)#ip route 0.0.0.0 0.0.0.0 12.0.0.1

r2(config)#ip route 0.0.0.0 0.0.0.0 serial 0  
(任何路由 / Any route)

Any route:指没有明细路由的任何未知路由

对于R1没有明细路由的任何未知路由, 都从此接口 / 此下一跳发送出去

对于R1没有明细路由的任何未知路由，都从此接口 / 此下一跳发送出去

## 1.1—动态路由协议基础知识

# 1.1—动态路由协议基础知识

### 动态路由协议 (Dynamic Routing)：

高自动化，减少工作量；现代IP网络中主要采用动态路由协议。

### 距离矢量协议DV (Distance-Vetor) 和链路状态协议LS (Link-State)：

DV协议的路由以矢量（**跳数+下一跳=距离+方向**）标记的“路标”形式存在；每个路由器在信息上依赖于邻接路由器，而邻接路由器又依赖它的邻接路由器，依次类推，DV协议“依照传闻选择路线”；主流的距离矢量协议有：RIP、IGRP；

LS协议的路由以“公路线路图”的形式存在；也叫最短路径优先协议和分布式数据库协议，LS协议“依照地图选择路线”；主流的链路状态协议有：OSPF、ISIS；。

DV协议和LS协议最大的区别在于拓朴表数据库的交换处理上：DV协议每个路由器都只向邻居转发自己处理过的拓朴表（**留下有用的删除不需要的然后跳数+1，最多16跳**）；LS协议在收到拓朴表后备份并转发（**并不经过处理**），然后在处理时也不会删除不需要的，而是首先找到自己和目标在“地图”上的位置并选取到达各路由器最优的路径写入路由表其余的保留在拓朴表→收敛速度因此快但资源要求较高；

唯一的混合协议DV/LS：EIGRP。

### DV协议的通用属性：

定期更新 (Priodic Updates)：DV协议每经过特定时间就发送更新信息，周期为10秒 (AppleTalk的RTMP) 到90秒 (Cisco的IGRP)；更新信息发送间隔过短会造成拥塞，而过长又会失去意义；

邻居 (Neighbour)：即共享相同数据链路的一组路由器；DV协议在信息上是依赖于邻居的逐跳更新方式；

**广播更新 (Broadcast Updates)**：向广播地址 (IP网为 255.255.255.255) 发送更新信息；相同路由协议的邻接路由器收包后回应，不同路由协议的邻接路由器丢弃；更新信息包含整个路由表，邻居会搜集自己需要的信息 (跳数+1)，丢弃不需要的；网络矢量算法只给出了网络上的路标也就是方向和直线距离，但是没有给出沿路径行走的细节，就像叉路口的路标一样，它很容易受到意外或故意的破坏；广播更新有一个失效计时器，也就是一定时间 (deadtime) 内hello包无回应即删除该邻居；

**水平分隔 (Split Horizon)**：路由器向外发送包含了整个路由表的更新信息不仅浪费了带宽，还有可能造成Full Mesh网络的网络回路也就是不

更新信息不仅浪费了带宽，还有可能造成Full Mesh网络的网络回路也就是不断把从邻居路由器学到的路由回发给邻居路由器（你收到并发出后经过Full Mesh回路往往会再发给邻居，邻居有更新了按照RIP的协议规则也会不断发给你，于是很可能一直循环下去浪费资源造成拥塞甚至导致段网），这并不必要，因此规定只发送路由矢量方向的路由也就是只延续收到的路由，而与路由矢量方向相反的路由是逆向路由（Reverse Route）默认被水平分隔阻挡；分成简单水平分隔（发送更新时接口不能发送从本接口得到的跟新信息）和毒性逆转水平分隔（发送更新时通过指定跳数的inf无穷大来指定向该接口发送此更新信息的网络不可达）；

跳数的无穷大（ $\infty$ ）：Full Mesh网络的环路下会不断循环更新某路由使跳数直到无穷大而使路由不可达（默认跳数16的网络不可达），解决方法是设定最大跳数15；但收敛速度大大降低！其他解决方法是触发更新和抑制计时器；触发更新（Triggered Update）：路由在发生变化时立刻发布更新而不到计时器超时；抑制计时器（Holddown Timer）：路由跳数变化时立刻抑制（不收发有关其的更新信息）等时间结束后再查看，这是折中的方法，虽然有效解决了跳数无穷大问题但是抑制时间过短会造成拥塞，而过长又会失去意义，不建议低端路由器使用；

异步更新（Asynchronous Update）：MP子接口中避免碰撞。

### LS协议的通用属性：

对网络发生的变化能够快速响应；当网络发生变化时发送触发式更新（Triggered Update）；不断发送间隔时间为30分钟的周期性更新（链路状态刷新）；

洪泛（Flooding）：LS协议只在网络拓扑发生变化后产生路由更新：当链路状态变化后检测到变化的设备创建LSA（Link State Advertisement），然后通过组播地址传送给所有的邻居；每个邻居在收到LSA后都会拷贝一份更新自己的链路状态数据库LSDB（Link State DataBase），并且接着再把LSA转发给其他的邻居；这种LSA的洪泛保证了所有的路由设备在更新路由表之前更新它的LSDB；

SPF树（Shortest-Path-First tree）：LS协议在一个特定的区域（Area）内从邻居处收集网络信息并构建LSDB；一旦路径信息被集齐以后每个路由器便会根据LSDB生成SPF树；然后路由器通过使用Dijkstra算法计算到达各个目标网络的最佳路径并从SPF树里面选出来放进路由表里；

信息的跟踪：LS协议依靠信息跟踪建立拓扑而不是像DV协议那样依据传闻；运行了LS协议的路由器跟踪以下信息：各自的邻居、同一个区域的路由器、到达目标网络的最佳路径。

### 管理距离（AD）和协议号：

路由条目的比较步骤：先按照最长匹配选择最长的；长度相同则选AD最小的；AD相同才比较Metric。

各种IGP的AD标准：Rip→120；ISIS→115；SPF→110；IGRP→100；EIGRP→90；静态路由→1；直链路由→0。

| Route Source                                                     | Default Distance Values |
|------------------------------------------------------------------|-------------------------|
| Connected interface                                              | 0                       |
| Static route*                                                    | 1                       |
| Enhanced Interior Gateway Routing Protocol (EIGRP) summary route | 5                       |
| External Border Gateway Protocol (BGP)                           | 20                      |
| Internal EIGRP                                                   | 90                      |
| IGRP                                                             | 100                     |
| OSPF                                                             | 110                     |
| Intermediate System-to-Intermediate System (IS-IS)               | 115                     |
| Routing Information Protocol (RIP)                               | 120                     |
| Exterior Gateway Protocol (EGP)                                  | 140                     |
| On Demand Routing (ODR)                                          | 160                     |
| External EIGRP                                                   | 170                     |
| Internal BGP                                                     | 200                     |
| Unknown**                                                        | 255                     |

| Routing Protocol | Protocol No. | Port No. | Update Reliability   |
|------------------|--------------|----------|----------------------|
| IGRP             | 9            |          | Best effort delivery |
| EIGRP            | 88           |          | 1-to-1 window        |
| OSPF             | 89           |          | 1-to-1 window        |
| RIP              |              | UDP 520  | Best effort delivery |
| BGP              |              | TCP 179  | Uses TCP windowing   |

主流IGP的概述:

|       | 类别(Class)                                                | 更新方式(Update)                                     | 认证(Authentication)                        |
|-------|----------------------------------------------------------|--------------------------------------------------|-------------------------------------------|
| RIPv1 | 有类(classful)                                             | 定期广播(255.255.255.255)<br>可配置成单播；带触发更新            | 无                                         |
| RIPv2 | 无类(classless)                                            | 定期组播(224.0.0.9)<br>可配置成单播；带触发更新                  | 简单认证(明文)<br>MD5 认证                        |
| IGRP  | 有类(classful)                                             | 同 RIPv1                                          | 无                                         |
| EIGRP | 无类(classless)                                            | 非周期的、部分有边界的“可靠组播”(224.0.0.5)                     | 仅 MD5 认证                                  |
| OSPF  | 无类(classless)                                            | 组播，DRouters(224.0.0.5)<br>AllDRouters(224.0.0.6) | 简单认证(明文)<br>MD5 认证                        |
| ISIS  | 无类(classless)                                            | 定期组播<br>触发更新                                     | 明文认证<br>增强明文认证<br>MD5 认证                  |
|       | 度量(Metric)                                               | 管理距离(Distance)                                   | 负载均衡(Load Balance)                        |
| RIPv1 | Hop count(1~16)                                          | 120                                              | 等价(Default=4,Max=16)                      |
| RIPv2 | Hop count(1~16)                                          | 120                                              | 等价/非等价                                    |
| IGRP  | $BW_{min} + DLY_{sum}$<br>Default=100 hop<br>Max=255 hop | 100                                              | 等价(Default=4,Max=255)<br>非等价(参数 variance) |
| EIGRP | $IGRP_{metric} \times 256$                               | 90(内部) 170(外部) 5(汇总)                             | 同上                                        |
| OSPF  | $10^8/BW(1 \sim 65535)$                                  | 110                                              | 同上                                        |
| ISIS  | Default=10(0~63)<br>Max=1024                             | 115                                              | 等价(Max=6)                                 |
|       | 进程(Process)                                              | 域(Domain)                                        | 协议(Protocol)                              |
| RIPv1 | 仅单个                                                      | 无                                                | UDP 520                                   |
| RIPv2 | 仅单个                                                      | 无                                                | UDP 520                                   |
| IGRP  | 可多个                                                      | 进程域(Process Domain)                              | IP 协议号 9                                  |
| EIGRP | 可多个(1~65536)                                             | 同上                                               | IP 协议号 88                                 |
| OSPF  | 可多个                                                      | 同上                                               | IP 协议号 89                                 |
| ISIS  | Max=3                                                    | 路由选择域(Routing Domain)                            | CLNS                                      |
|       | 汇总(Summary)                                              | 可变子网掩码(VLSM)                                     | 无类别域间路由选择(CIDR)                           |
| RIPv1 | 不可关闭自动汇总<br>无手工汇总                                        | √                                                | ×                                         |
| RIPv2 | 可以关闭自动汇总<br>只能在网络边界手工汇总                                  | √                                                | ×                                         |
| IGRP  | 不可关闭自动汇总<br>无手工汇总                                        | ×                                                | ×                                         |
| EIGRP | 可关闭自动汇总<br>可在任何地方手工汇总                                    | √                                                | √                                         |
| OSPF  | 无自动汇总<br>可在任何地方手工汇总                                      | √                                                | √                                         |
| ISIS  | 无自动汇总<br>可在任何地方手工汇总                                      | √                                                | √                                         |

## 1.2—动态路由协议RIP

## 1.3—动态路由协议RIP①

**Dynamic Routing Protocol:**动态路由协议

现代IP网络中，主要的动态路由协议：

**AD/管理距离：**

- 1: DV/距离向量协议: RIP(120)/IGRP(100)
- 2: LS/链路状态协议: OSPF(110) /IS-IS(115)
- 3: DV-LS/混合协议:EIGRP(90)
- 4: ODR(160)

**LAB1: ODR(On-Demand Routing)**

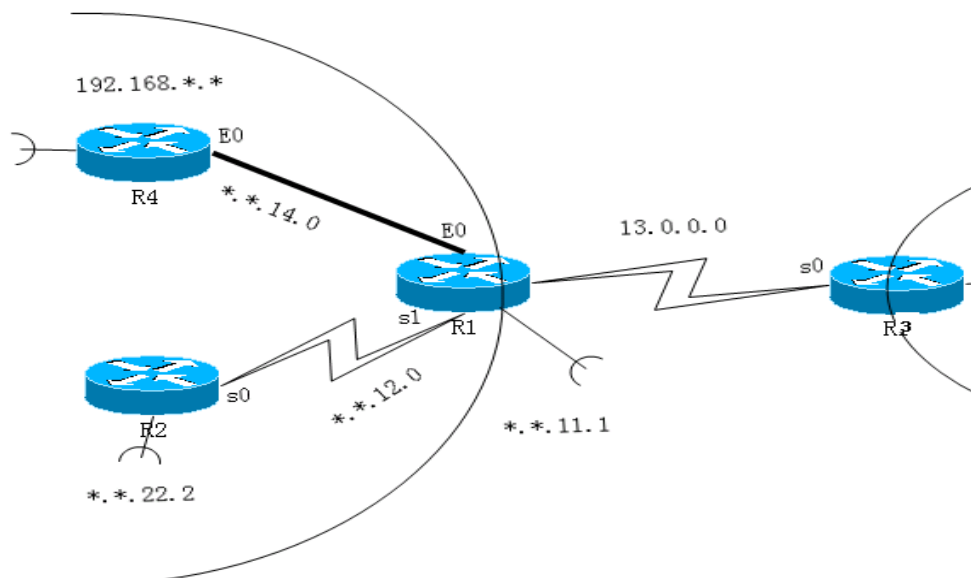
ODR是CISCO私有的协议，基于CDP协议的运行。  
(CDP只能发现直接相连的Cisco设备)

ODR适用于：  
在全网都是CISCO设备的网络环境中，

而且网络拓扑是HUB&SPOKE架构的简单网络中，  
(所有的分支点都是直接与Hub相连)

ODR协议的目的：使网络配置达到最简化的操作

CDP的Hello周期是60S, Hold-time: 3\*60=180.



**Step0:**按图配置网络

**Step1:**确认设备的连接(察看CDP的运行状态)

1-1:

show cdp interface

(察看正在运行CDP的接口, 默认所有接口都运行CDP)

1-2:

R5#show cdp neighbors

| Device ID | Local Intrfce | Holdtme | capability | Platform | Port     |
|-----------|---------------|---------|------------|----------|----------|
| R3        | ser 0         | 120~179 | R          | 2500     | ser 1 对方 |

接口

1-3:控制运行CDP的接口范围:

R3(config)# (NO) cdp run (在全局上, 启动 / 关闭CDP)

R1(config)#in serial 1

R1(config-if)# (NO) cdp enable (对特定接口, 启动 / 关闭CDP)

**注意:**

1:在ISP的角度考虑, ISP的PE (ISP连接用户的设备) 是不会与用户的CE (用户连接ISP的设备) 运行CDP的.

2:注意以太网交换机的对CDP的影响. (如果两个路由器上中间有个CISCO的交换机, 那么这台交换机将会阻止ODR的运行)

**step2:**在中心路由(HUB)R1上, 启动ODR:

2-1:

```
R1(config)#router odr
```

2-2: 在中心点R1上, 查看每个分支点的路由:

```
R1#show ip route
```

```
o 192.168.22.0/24 [160/1] via 192.168.12.2, serial 0
```

但是没有192.168.44.0 / 24

2-3:在分支点上, 查看中心点发来的ODR默认路由:

```
R2#show ip route
```

```
o* 0.0.0.0/0 via 192.168.12.1
```

但是在R4上没有默认路由。

考虑问题1: 为何R2/R4的路由不对称?

**step3:**

3-1: 分支点之间的内网用户互通:

分支点之音的用户, 依靠ODR生成的默认路由, 实现互通。

3-2: 内网用户与外网用户的互通:

3-2-1: R1(config)#ip route 0.0.0.0 0.0.0.0 13.0.0.3

3-2-2: R3(config)#ip route 192.168.0.0 255.255.0.0 13.0.0.1

### ODR路由的局限性:

1:全网设备必须都是CISCO设备.

2:分支点的路由器, 必须跟中心点直接相连.

(注意以太网交换机对CDP的影响)

### Classful Routing &Classless Routing

落后的淘汰的      先进的, 正在运行的

~~~~~

Classful Routing/有类路由协议:

IGRP /RIP V1

1. 在发送路由更新信息时, 不携带子网掩码, 无法描述路由条目的路由长度.

2. 在主类的网络边界上, 自动发生路由汇总,

汇总到主类网络的默认的路由长度(自动汇总是无法关闭的)
(不支持VLSM, 只能汇总为A/B/C类).

3. 由于上述原因, 有类路由协议会产生“不连续子网”的路由通达性问题.

Classless Routing/无类路由协议

~~~~~

RIP V2 的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后, 再通告出去.
3. 把收到的明细路由放进路由表中, 但会对明细路由进行汇总后再通告出去!

EIGRP的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后, 再通告出去
3. 把收到的明细路由放进路由表中, 并且把收到的明细路由通告出去!

RIPv2/EIGRP/OSPF/IS-IS/BGPv4

1. 在发送路由更新信息时, 已经携带子网掩码.
  2. 支持VLSM, 路由的手工/自动汇总, (可以关闭自动汇总)
  3. 在部分先进的路由协议中, 支持CIDR(超网)
- ip classless (在IOS为12.0以后的版本中, 默认启动无类路由。)

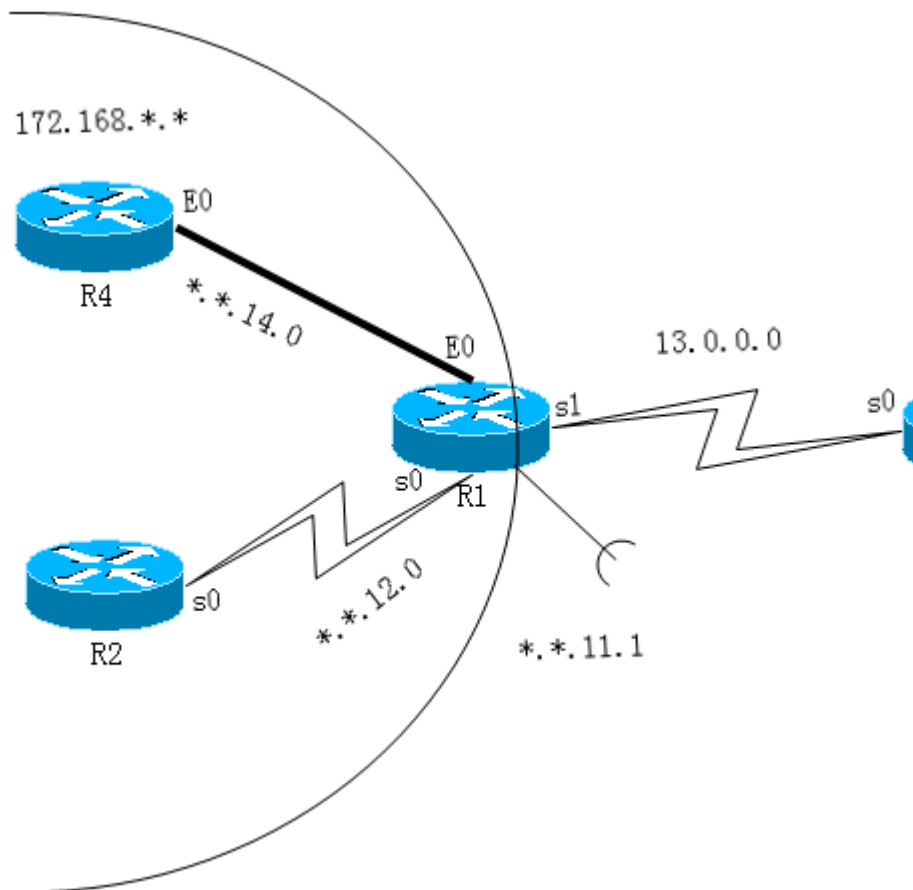
在网络边界:

DV协议默认会执行自动汇总, (RIP/IGRP/EIGRP)

LS协议默认不执行自动汇总. (OSPF/ISIS)

### LAB2:RIP的基本配置:

~~~~~



```
R3#sh run | b rip
R1(config)#router rip
R1(config-router)#network 13.0.0.0
R1(config-router)#network 172.168.0.0
```

(宣告与本路由直接相连的网络, 只需要宣告其主类网络)

debug ip rip (察看RIP的路由信息的收发)

show ip route rip (察看从RIP学到的路由信息)

undebug all (关闭所有的Debug)

LAB3:RIP的版本控制:默认情况运行v1

show ip protocols(察看RIP的版本控制)

在未指定版本时, RIP的默认版本是:发1, 收1/2

指定V1:发1收1

指定V2:发2收2.

在全局配置, 对所有接口生效:

```
R1(config)#router rip
R1(config-router)#version 1
R1(config-router)#version 2
```

对特定接口, 指定RIP版本, (接口的优先级一般比全局要高)

```
interface serial 0
 ip rip send version 1/2
 ip rip receive version 1/2
```

V1与V2不兼容~!

V1:不携带子网掩码, 不携带下一跳信息, 向广播地址发送路由更新.

V2:携带子网掩码, 携带下一跳信息, 向组播地址(224. 0. 0. 9)发送路由更新.

LAB4:RIP的自动 / 手工汇总

RIPv1/v2, 在**网络边界**都会发生自动汇总, 汇总到其**主类网络的默认路由长度**。

A: 5. 0. 0. 0 / 8

B:128. 5. 0. 0/16

C:192. 5. 5. 0/24

R1#在网络13. 0. 0. 0和172. 168. 0. 0的边界上, 进行自动汇总。

```
RIP:sending v2 update to 224.0.0.9 via serial 1 (13.0.0.1)
172.168.0.0/16
```

但RIPv2可以关闭自动汇总

```
R1#
router rip
no auto-summary
```

R3上, 就有了明细路由。

手工汇总:

在需要进行路由汇总的**出接口**:

```
R3(config-if-S1)#IP summary-address rip 172.168.0.0 255.255.192.0
```

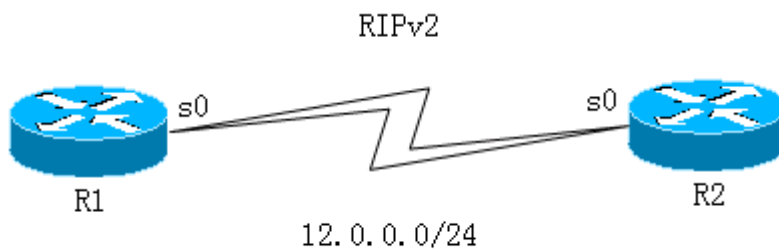
```
R3(config-if-S1)#IP summary-address rip 172.168.0.0 255.255.192.0
R 172.168.0.0/18
```

在运行RIP的接口中进行手工汇总,把汇总后的结果从接口通告出去

```
R1(config-if)#ip summary-address eigrp 90 172.16.0.0 255.255.248.0
在运行EIGRP的接口中进行手工汇总,把汇总后的结果从接口通告出去
```

```
R5#
R 175.100.0.0/18
```

LAB5:RIP的单播更新: (Unicast-Update)/被动接口
(适用于物理链路上,无法支持广播/组播流量的情况)



Step1:被动接口:

让需要进行单播更新的接口, **不再发送** 广播/组播的路由更新信息.

1-1:需要PASS的接口很少时:

```
R2(config-router)#passive-interface serial 0
(让特定一个接口不再发送广播/组播的路由更新.)
```

1-2:需要pass的接口**很多**时:

```
R1(config-router)#passive-interface default
(让所有接口都不发送广播/组播的路由更新.=(passive all interface))
```

```
R1(config-router)#no passive-interface serial 0
(单独让一个接口可以发送广播/组播更新)
```

Step2:让已经Pass的接口,发送单播**的路由更新.**

```
R1(config-router)#neighbor 12.0.0.2
R2(config-router)#neighbor 12.0.0.1(对方的IP)
```

```
R1(config-router)#neighbor 12.0.0.2  
R2(config-router)#neighbor 12.0.0.1(对方的IP)
```

debug ip rip (察看RIP路由的收发情况)

LAB6:RIP authentication/RIP 认证: (保证RIP路由网络, 路由信息交换的安全性)

~~~~~

**Step1:**在全局模式, 配置KEY-CHAIN:

```
R1/23#  
key chain ccnp (自定义的名称)  
key 1 (若有多个值 / 1、2、3, 默认发最小的那个)  
key-string cisco (密码)
```

**step2:**在接口中, 调用key chain:

```
R1/R2(config-if)#ip rip authentication key-chain CCNP
```

**Step3:**在接口中, 选择认证类型: (明文/密文) 无默认, 必须选择  
认证方式两边必须一样。

```
R1/2 (config-if)#ip rip authentication mode text (明文)
```

```
R1/2 (config-if)#ip rip authentication mode md5 (密文)
```

### 关于水平分割 (Split horizon)

1、对于一个路由器, 路由信息从一个接口进入后, 不能再从这个接口发出。

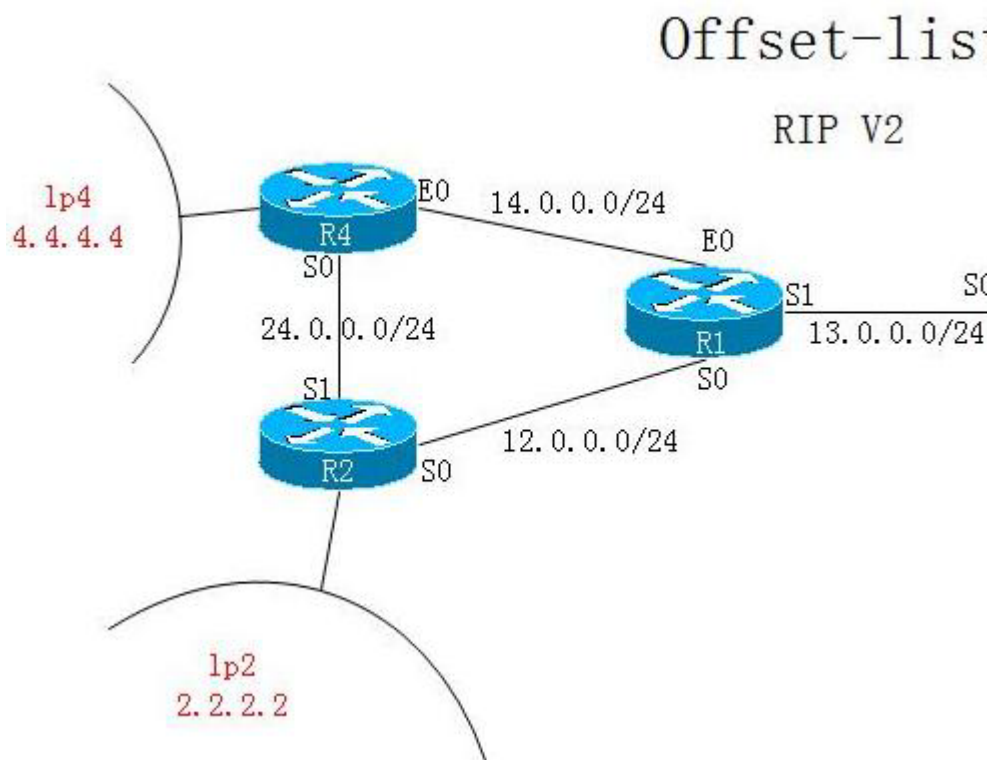
(但可以从别的接口发出)

2、适用于哪些协议:

只适用于DV协议, 不适用于LS协议。

### LAB7: Offset-list / 偏置列表, 实现RIP路由的控制:

~~~~~



step0: 观察RIP的等价的负载均衡:

R4:
R 12.0.0.0/8 [120/1] Via

step1:

通过offset-list, 实现RIP中的不等价网络拓扑中的, 等价负载均衡:
(R4观察2.2.2.0/24) RIP不支持非等价负载均衡

出方向的控制(R2当前是汇总的):

Step2: 通过ACL定义需要控制的路由:

R2(config):#access-list 2 permit 2.0.0.0

Step3: 在R2的S1口做路由偏置(+1):

因为对R4来说, 到达R2有两条路, 其中一条是一跳, 另一条是两跳. 如果将原本是一跳的路由人为的+1, 使得它的跳数变成二, 那么R4到R2就能实现负载均衡.

R2(config)#router rip

R2(config-router)#offset-list 2 out 1 serial 1

(出口的偏置, 只影响下游路由器, 不影响本机)

```
R4#  
R 2.0.0.0/8 [120/2]VIA 14.0.0.1, e0  
[120/2]VIA 24.0.0.2, Serial0
```

入方向的控制:(no auto-summary)

Step4:通过ACL定义需要控制的路由:

```
R4(config)#access-list 20 permit 2.2.2.0 0.255.255.255
```

step5:

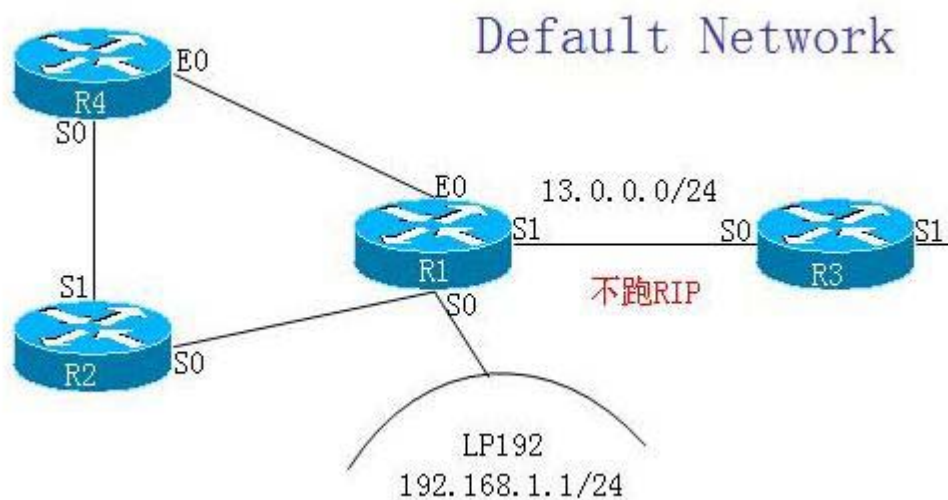
```
R4(config-router)#offset-list 20 in 1 serial 0
```

(入口的偏置, 本机和下游路由器都受影响)

```
R 2.2.2.0/24 [120/2]via 14.0.0.1, e0  
[120/2]via 24.0.0.2, s0
```

LAB8:default-network:

~~~~~



**Step0:**

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial 13.0.0.3
```

**step1:**

```
R1(config)#interface loopback 192
R1(config-if)#ip add 192.168.1.1 255.255.255.0
```

**step2:**

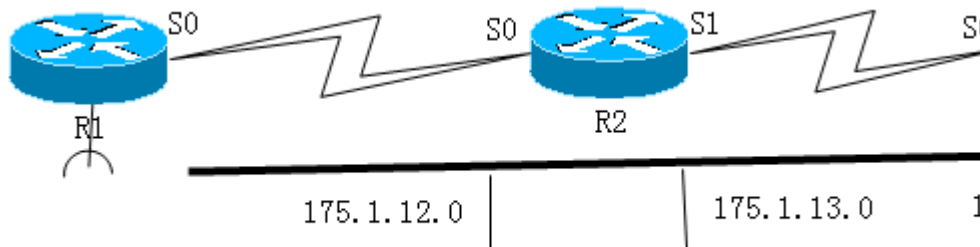
```
R1(config)#router rip
R1(config-router)#network 192.168.1.0
```

**Step3:指定默认网络**

```
R1(config)#IP default-network 192.168.1.0
```

```
R* 0.0.0.0/0
R 192.168.1.0
```

**LAB10:不连续子网 (Discontinued Network) 一个主类被某些设备分割开了**



**解决方案一:**

RIPv1:第二地址的虚拟管道,

要求一:与不连续子网同在一个主类网络(172.10.\*.\*)

要求二:与不连续子网的网络长度相同(/26)

**Step1:配置虚拟管道 (Second 地址)**

```
R3(config-if)#ip address 175.1.13.3 255.255.255.0 secondary
R1 R2 R3都要配置第二地址, R1要配置两个第二地址
```

**Step2:在中间的路由器上,宣告第二地址所在网段:**

```
R1(config)#router rip
R1(config-router)#network 175.1.0.0
```

解决方案二:

RIPv2:不需要第二地址:  
v2默认自动汇总

Step1:

```
Router rip  
Version 2
```

Step2:在全网的RIP路由器上, 自动关闭汇总

### 1.3—动态路由协议EIGRP

## 1.3—动态路由协议EIGRP

EIGRP (Enhanced IGRP)

EIGRP的特点:

IGRP/EIGRP都是CISCO的私有协议.

1:是唯一的一种LS/DV的混合协议.

2:Rapid convergence

EIGRP拥有目前最快的网络路由收敛性. (依靠后备路由器FS)

3. 配置简单, 能够支持中型到大型网络.

4:Incremental updates

增量/触发更新.

5:EIGRP可以支持等价/不等价的负载均衡,

默认是支持等价负载均衡,

可以通过调整Variance, 来实现不等价的负载均衡.

EIGRP到达同一个目标网络, 可以同时有4条路径 (默认), 最多6条.

6:EIGRP默认使用组播(224.0.0.10)进行路由更新.

也可以支持单播更新.

(IGRP:广播:225.255.255.255)

7:EIGRP可以支持VLSM,  
支持汇总:默认有自动汇总/手工汇总.  
EIGRP可以汇总到超网, CIDR

8:EIGRP可以支持多种网络协议:  
IP/IPX/AT (AppleTalk)

### EIGRP Packets

- 1:Hello:用于建立/维护EIGRP邻居关系.
- 2:Update:更新包:发送路由更新信息.
- 3:Query:查询包:当路由器丢失了原有的路由后, 会向邻居发送“查询请求”.
- 4:Reply:当被查询路由器, 收到“查询请求”后, 将自己知道的路由信息回应给发起查询路由器.
5. Ack:用于对EIGRP的可靠传输报文的进行确认.  
(其中2、3、4都是可靠传输报文, 收到后要发送ACK进行确认, 但ack自己本身不用确认)

### EIGRP的Hello包:

- 1:EIGRP路由器, 向224. 0. 0. 10, 发送Hello包, 同时也监听这个组播地址。
- 2:Hello包中, 包含了EIGRP的K值, 两个路由器的K值必需匹配, 如果不匹配无法建立邻居。  
(K值的默认值: K1=K3=1, K2=K4=K5=0)
3. Hello包中, 包含了AS号, 两个路由器的AS号必需相同, 如果不相同, 无法建立邻居。  
(Autonomous-System/自治系统。)
- 4:两个建立EIGRP邻居关系的路由器的直链接口, 其IP地址必需在同一个IP子网, 否则无法建立邻居。

5. 即使EIGRP的Hello/hold计时器不完全匹配,只要自己Hello的间隔不超过对方的Hold间隔,邻居是可以建立的。(但默认情况下,建议不要修改这两个计时器。)

#### EIGRP Timer (计时器):

Hello Timer:

在大于T1链路,点对点链路上,默认5秒发送一次Hello包  
在小于/等于T1的多点链路上,默认60秒发送一次Hello包.

Hold Timer:

Hold timer默认是Hello Timer的3倍,  
如果Hold Timer所定义的时间内,收不到对方的Hello包,邻居关系就会Reset,即邻居会DOWN下去。

在本路由器上设置hello time和hold time是告诉邻居的!!也就是说在本路由器上设置的hello time和hold time是影响邻居的!!

只要在本路由器上设置的hello time 小于hold time,那么邻居还是可以建立起来的,而不管邻居的hello time是多少!!

#### EIGRP的可靠传输报文:

Update/Query/Reply, 收到此包后,需要发送ACK进行确认。

#### EIGRP的非可靠传输报文:

Hello/Ack, 收到此包后,不需要进行确认。

#### EIGRP Retransmission Policy/重传机制:

稳定的网络:

如果Hold Timer所定义的时间内,收不到对方的HELLO包,邻居关系就会Reset

不稳定的网络:

EIGRP对于可靠传输报文,有必需的确认机制,  
如果没有收到对方的确认,那么可靠传输包就会发生重传,极限是16次,  
如果达到极限,都没有收到对方的确认,那么就RTO, Reset。

如果没有收到对方的确认, 那么可靠传输包就会发生重传, 极限是16次, 如果达到极限, 都没有收到对方的确认, 那么就RTO, Reset。  
(RTO:Retransmission TimeOut.)

因为EIGRP的是windows size of one (stop-and-wait mechanism)  
所以如果在对一组邻居, 进行组播的路由更新时,  
有个别路由器响应特别慢,  
可能导致整个EIGRP网络的收敛效率低下.

解决方案是:

对正常的大部分路由器做组播更新,  
对特别慢的路由器, 单独进行单播更新.

### EIGRP的3张表:

#### NO. 1:EIGRP的邻居表:

本路由器的接口, 所直接相连的EIGRP邻居的信息.

#### NO. 2:EIGRP的拓扑表: (详细的拓扑表)

本路由器, 从自己的邻居那里, 得到去往特定目标网络的(一切)可能的路径, 都承载/存在于拓扑表中.

#### NO. 3:从EIGRP形成的路由表:

是EIGRP路由器, 从拓扑表中, 择优将 " 去往特定目标网络开销最小的 " 路由, 放入了路由表.

EIGRP和IGRP在AS号相同的情况下, 可以实现自动重分布.

EIGRP的Metric值是IGRP的256倍.

### EIGRP的DUAL算法: (EIGRP Diffusing Update Algorithm)

FD/AD:

Feasible Distance vs. Advertised Distance

AD是通告距离

FD可行性距离

Successor: (后继路由器)

当前的, 去往特定目标网络的, 转发路由器  
(正在工作运行的轮胎)

## 成为Successor的条件:

通过某个路由器, 去往**特定目标网络**的**开销/cost/Metric/FD**, 最小的路由器, 那么这个路由器就是**Successor**. (只有相邻的路由器才可能成为**Successor**)

通过Successor到达目标网络的这条路径,就能进入路由表,成为去往这个目标网络的路由。

如果一个路由器, 去往特定目标网络有**多个Successor**, 那么此路由器就是在进行**等价负载均衡**.

**Feasible Successor:** (可行性后继/后备轮胎)

成为FS的条件:

后备轮胎的AD, 必需小于通过“当前使用/正在运行的”轮胎的FD.

|           |         |                  |
|-----------|---------|------------------|
| <u>FS</u> | <u></u> | <u>successor</u> |
|-----------|---------|------------------|

AD of Second Best Route < FD of Best Route  $\Rightarrow$  Feasible Successor  
(除了Best之外, 其余都是Second Best, 不是最优就是次优的)

```
sh ip eigrp 90
```

SRTT:平均往返时间:发送更新与回应ack之间时间的平均值,计算方法是CISCO内部保密的

RT0: 重传超时 (判断过多长时间为超时--等待时间) --根据SRTT得出

seq num:sequence number of the last update, query, or reply

packet... (连接时序号加1代表对方收到并可可靠传输)

Q:0是正常的（等待重传的包数）

```
10.2.2.0/24 -> r2 -> 10.1.12.0 -> r1 -> 10.1.13.0 -> r3 -> 10.3.3.0
20.2.2.0
```

### EIGRP手工汇总特点:

- 1: 手工汇总是无类的
- 2: 可以汇总非直连的路由 (EIGRP学习过来的路由也可以)

LAB: 等价负载均衡 (FD一样)

step 1:

LAB: 等价负载均衡 (FD一样)

step 1:

r2(config)#access-list 1 permit 3.3.3.0 用ACL匹配需要做偏移的路由

step 2:

sh ip eigrp topology

offset list 只能增加不能减小, 所以要先看哪个接口的FD小。两个FD相减为48600

r2(config-router)#offset-list 1 in 48600 serial 1 (如果不指定端口则所有3.3.3.0进来的接口都加48600) (用偏移列表增加度量值)

sh ip route/sh ip eigrp topology 可以看到到3.3.3.0负载均衡

其实用偏移列表更改的并非真正的metric值, 而是改变了带宽或者延迟, 至于到底改变了什么? 要下课后查。

LAB2: 非等价负载均衡

r2(config-router)#variance 2

注意点:

1: variance的数值必须大于FS' FD/S' FD

2: 必须存在FS, 否则variance调128都没有用。

查看负载均衡多少条?

sh ip protocol ---max path 4

LAB3: r1-r2相连(s口)

eigrp密文认证配置方法:

step1:

r1(config)#key chain AAA (建立钥匙串, 名字本地有意义)

step2:

r1(config-keychain)#key 1 (这个两边要一样)

r1(config-keychain-key)#key-string CCNP (定义key 1的密钥是CCNP)

然后退出

step3:

r1(config)#int s 0

r1(config-if)#ip authentication key-chain eigrp 90 AAA (调用钥匙串AAA)

step 4:

r1(config-if)#ip authentication mode eigrp 90 md5 (开启认证)

查看命令

r1#debug eigrp packet

\*Mar 1 06:17:18.626: EIGRP: Sending HELLO on Serial0

\*Mar 1 06:17:18.626: AS 90, Flags 0x0, Seq 0/0 idbQ 0/0 iadbQ

un/rely 0/0

\*Mar 1 06:17:18.830: EIGRP: Serial0: ignored packet from

```

iadbQ un/rely 0/0
*Mar  1 06:17:18.830: EIGRP: Serial0: ignored packet from
12.1.1.2, opcode = 5 (missing authentication)
r1#
*Mar  1 06:17:23.130: EIGRP: Serial0: ignored packet from
12.1.1.2, opcode = 5 (missing authentication)
*Mar  1 06:17:23.594: EIGRP: Sending HELLO on Serial0
*Mar  1 06:17:23.594:   AS 90, Flags 0x0, Seq 0/0 idbQ 0/0 iadbQ
un/rely 0/0

```

如果是通过ETHERNET连，那么连接的路由器所有都要认证否则全部都连不了邻居，可以说认证是对链路而不是对应路由器的。

STUB:

为了防止查询环路，通常在最末端设备做stub（减少查询）。

stub的邻居不会向stub路由器本身作查询

```
r1(config)#router eigrp 90
```

```
r1(config-router)#eigrp stub
```

```
r2#sh ip eigrp nei detail
```

IP-EIGRP neighbors for process 90

| H   | Address | Interface | Hold Uptime | SRTT |
|-----|---------|-----------|-------------|------|
| RT0 | Q       | Seq Type  | (sec)       | (ms) |

Cnt Num

|      |          |     |             |     |
|------|----------|-----|-------------|-----|
| 0    | 12.1.1.1 | Se0 | 11 00:01:17 | 600 |
| 3600 | 0        | 3   |             |     |

Version 12.2/1.2, Retrans: 0, Retries: 0

Stub Peer Advertising ( CONNECTED SUMMARY ) Routes

Suppressing queries

eigrp stub 默认有加connected（直连）与summary（汇总）路由信息给邻居。

LAB2: EIGRP默认路由

方法一:

step:

```
r1(config)#ip route 0.0.0.0 0.0.0.0 12.1.1.2
```

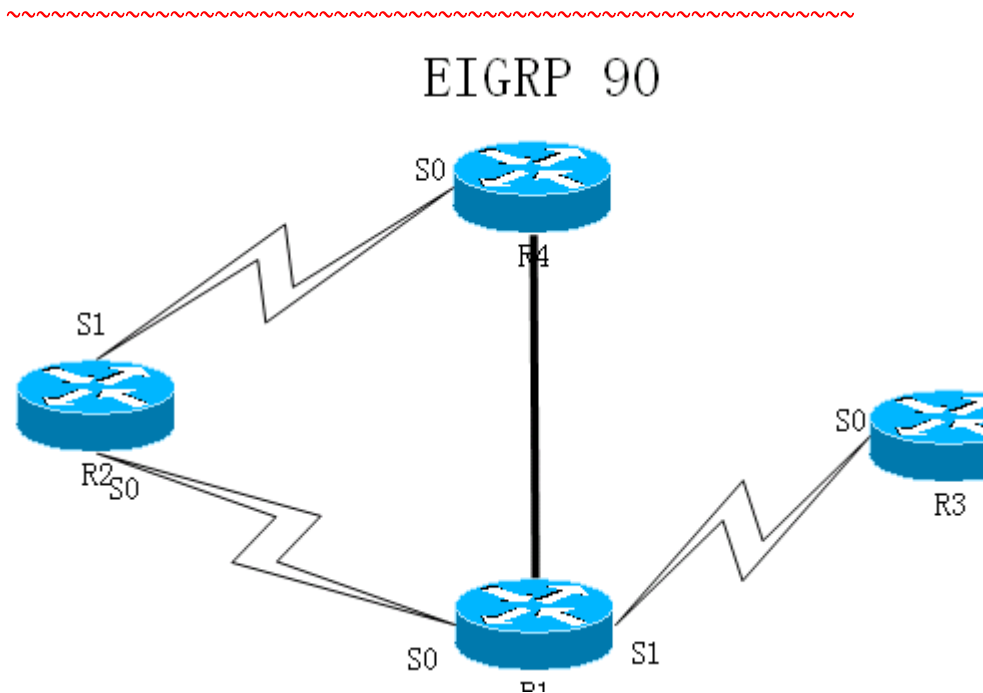
```
r1(config)#router eigrp 90
```

```
redistribute static
```

方法二:

```
r1(config)#ip default-network 12.0.0.0 (主类)
```

**LAB1:EIGRP的基本配置:**



#### Step1:

R3#show running-config | b eigrp

```
router eigrp 90
network 12.0.0.0 (每个Router都只宣告与本机直接相连的网络)
network 13.0.0.0
network 14.0.0.0
auto-summary (系统默认的, 路由在穿越主类网络边界时, 会发生自动汇总, 可以关闭)
```

#### Step2:

R1#show ip eigrp interfaces (察看当前正在运行EIGRP的接口)  
R1#show ip eigrp neighbors (察看EIGRP的邻居表)  
show ip eigrp topology detail-links (详细的拓扑表)  
show ip eigrp topology (拓扑表)  
show ip route eigrp (从EIGRP学到的路由表)

#### LAB2:EIGRP的路由的Metric值的准确计算:

~~~~~

Step1:确定路由的入口:

(也就是:访问该目标网络的**数据包的出口**)

R2上, 观察13.0.0.0 / 24

Step2:收集各个**路由的入口**的BW/DL信息:

R2#show interfaces serial 1

Step2:收集各个路由的入口的BW/DL信息:

R2#show interfaces serial 1

BW 1544 Kbit , DLY 20000 usec,

(这两个参数是用来控制三层协议的选路,而clock rate是反应链路上真实的速度,这两个参数可以在接口上更改,但仅仅是影响路由协议的选路)

Step3:

Metric={ (10,000,000/BW)+(DL/10)}*256 $10^7 / 1544 = 6476$

BW:从目标网络,流向本EIGRP路由器的,所有的路由入口中,带宽最小的带宽,单位是:Kbps, 多个路由入口中的带宽最小值.

DL:从目标网络,流向本EIGRP路由器的,所有的路由入口的延迟之和,单位是:us(微秒), 多个路由入口的延迟之和.

1.3—动态路由协议EIGRP②

LAB3:Wildcard Mask in EIGRP

(通过反掩码,控制运行EIGRP的接口的范围)

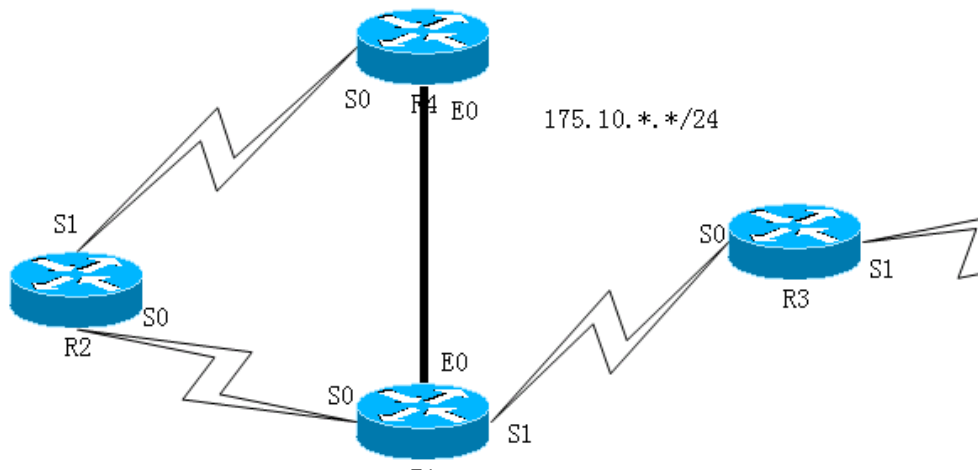
作用:控制有哪些接口在运行EIGRP)

Wildcard Mask/反掩码的匹配原则:

0:表示准确匹配

1:表示忽略不计

EIGRP 90



show ip eigrp interface 察看有哪些接口在运行EIGRP

当EIGRP passive接口时,不会向外路由,也不会接收邻居发过来的路由.

由.

(在所有路由器上, 关闭自动汇总)

```
network 175.10.0.0 0.0.255.255
```

Step0:路由器的所有接口都运行EIGRP

```
network 0.0.0.0 255.255.255.255
```

(network 0.0.0.0)

Step1:要求以155.*.*.*的所有接口都运行EIGRP:

```
network 155.9.8.7 0.255.255.255
```

(network 155.0.0.0 0.255.255.255)

Step2:要求以155.2.*.*的所有接口都运行EIGRP:

```
network 155.2.0.0 0.0.255.255
```

(反掩码不表示网络长度!!)

Step3:要求以155.2.4.*的所有接口都在运行EIGRP:

```
155.2.4.0 0.0.0.255
```

Step4:

对于A类接口, 不写反掩码时, 其默认的反掩码是0.255.255.255

对于B类接口, 不写反掩码时, 其默认的反掩码是0.0.255.255

对于C类接口, 不写反掩码时, 其默认的反掩码是0.0.0.255

Step5:要求以155.2.4.33的这个特定接口运行EIGRP:

```
R2(config-router)#network 155.2.4.33 0.0.0.0
```

在R3上收到的路由是155.2.4.32 / 27的路由, 即为这个IP的网络号。

结论:

EIGRP的反掩码

不控制EIGRP的接口在路由条目中, 表现出来的路由长度.

EIGRP/OSPF都是通过这种反掩码的方式, 只控制运行协议的接口范围.

LAB4:EIGRP Equal-Cost Load Balancing /等价负载均衡:

R2观察175.10.14.0 / 24

```
R2#show ip protocols
```

```
P 175.10.14.0 / 24 , 2 successors, FD is 2195456
  via 175.10.24.4 (2195456/281600), serial 1
  via 175.10.12.1 (2195456/281600), serial 0
```

```
EIGRP maximum metric variance 1
```

```
show ip eigrp topology detail-links
```

```
show ip eigrp topology
```

```
show ip route eigrp
```

LAB5:EIGRP Unequal-cost Load Balancing

/使用variance, 实现不等价负载均衡: 有FS时

R2观察175.10.13.0/24

```
P 175.10.13.0/24, 1 successors, FD is 2681856
  via 175.10.12.1 (2681856 / 2169856) , serial 0
  via 175.10.24.4 (2707456 / 2195456) , serial 1
  (R4是: R2去往175.10.13.0/24的FS)
```

```
R4(config)#router eigrp 90
```

```
R4(config-router)#variance 2 (同1调整为2)
```

只要比 $2.68M \times 2 = 5.36M$, 小的FS的路径, 都可以进行负载均衡。

调整后 clear ip eigrp neighbor (强制让EIGRP邻居复位)

意味着:

去往175.10.13.0/24这目标路由, 的开销小于 $2682856 \times 2 = 5.36M$ 的路由, 都可以列入不等价负载均衡的范围。

在此两条不等价的路由中, 其数据包的转发比例是反比关系。

提醒:

如果要使用variance, 实现EIGRP的不等价负载均衡, 只能在Successor和FS实现。

LAB6:无法使用variance, 实现不等价负载均衡的情况: (无FS时)

如果不满足EIGRP的DUAL算法中的FS条件, 不管variance是多大, 都不可能实现负载均衡:

R2观察175.10.34.0 / 24

R2#show ip eigrp topology detail-links

```
P 175.10.34.0/24, 1 successors, FD is 2195456,
  via 175.10.24.4 (2195456/281600), serial1
  via 175.10.12.1 (2707456/2195456), serial0
```

R1无法成为R2, 去往网络175.10.34.0/24的FS

LAB7:无法使用variance, 对EIGRP协议的进行人为路由操纵:

方法1: Offset-list (偏置列表)

step1:在R2上, 察看:

R2#show ip eigrp topolog detail-links
查看两条链路的FD分别是: 2195456 / 2707456

step2:通过ACL, 定义需要进行偏置控制的路由条目:

R2(config)#access-list 34 permit 175.10.34.0

step3:

R2(config)#router eigrp 90

R2(config-router)#offset-list 34 in 512000

serial 1

(acl定义的路由) (入方向) (增加的偏置值) (入口)

step4:在R2上观察175.10.34.0 / 24的路由, 实现了等价负载均衡:

show ip eigrp topology

show ip route

方法2:在R4的E0口增加延迟DL, 等于另外一条路由的总延迟:

或R2 S1

R4(config)#in e0

R4(config-if)#delay 2100 (单位: 10微秒) (包括原来的DELAY)

r4#show interface e0

r2(config)#in s1

r2(config-if)#delay 4000 (单位: tens of microsecond)

microsecond:微秒

millisecond:毫秒, 千分之一秒

方法3:

调整DL所对应的K3=0, 使EIGRP在计算路由的METRIC时, 不使用DL这参数。

默认的K值: K1=K3=1 K2=K4=K5=0

在同一个AS中的所有路由器上, 调整相同的K值

```
R1/2/3/4(config-router)#metric weights 0      1  0  0  0  0
                                     TOS      K1
```

方法4:

调整带宽

interface serial 0

bandwidth 2048 (只影响路由协议的选路)

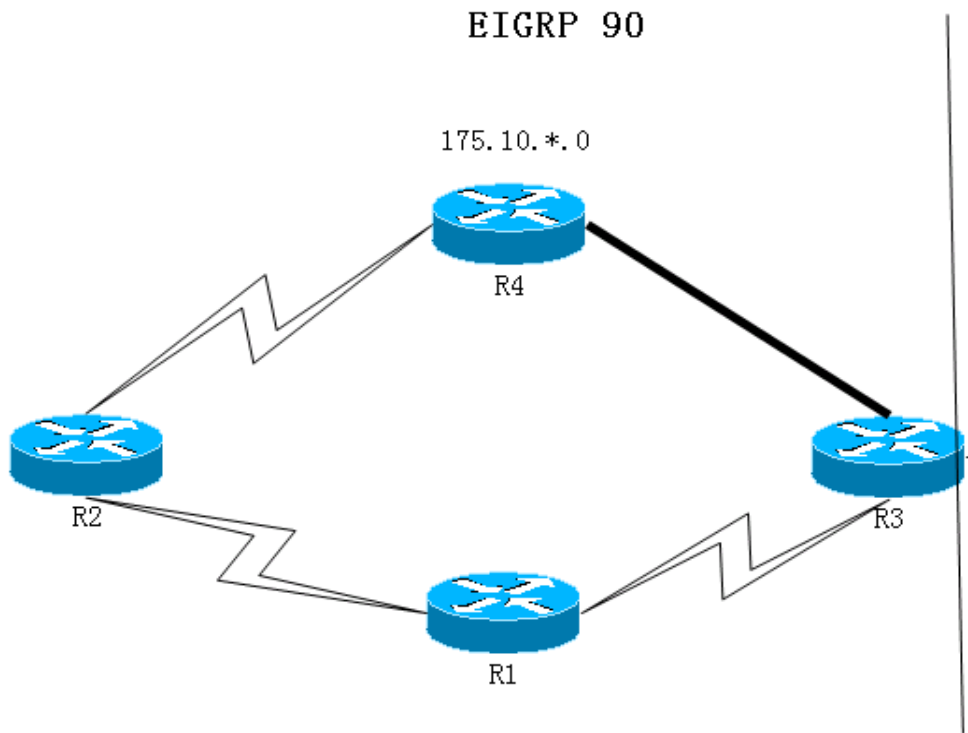
clock rate 2000000 (只影响到链路的真实物理速率)

interface ethernet

speed 10/100/1000 (两者都影响)

LAB4:Automatic Summarization:

~~~~~



EIGRP在主类网络的边界/(On major network boundaries), 会发生自动汇总, 是默认产生的.

自动汇总会将该子网的路由, 汇总到该主类的默认网络长度:

A>>>>/8

B>>>>/16

C>>>>/24

**step1:**

只在R5上关闭EIGRP的自动汇总:

只要在某条路由的起源路由器上关闭了自动汇总, 则不管该条路由以后穿过多少个网络边界都不会发生自动汇总。

在R1/2/3/4, 收到的都是明细路由: 5.5.5.0 / 24

在R3上关闭自动汇总:

在R5上收到了明细路由

**step2:**

175.10.0.0 / 16与176.35.0.0 / 16是两个不同的主类网络, 所以在R3上, 是网络边界

在R3 R5上配不相同的子网掩码, 观察EIGRP邻居是否抖动。

step3:

175.10.0.0 / 16与175.35.0.0 / 16是两个不同的主类网络，所以在R3上，是网络边界

step4:

175.10.0.0 / 16与175.10.35.0 / 16是同一个主类网络，所以在R3上，不是网络边界

### LAB5:Summarization: Manual(手工汇总,是基于接口的)

step0:

```
R3(config)#router eigrp 100
```

```
R3(config-router)#no auto-summary
```

Step1:在汇总路由的出接口中,手工汇总:

```
R3(config-if-S1)#ip summary-address eigrp 90 175.10.0.0
```

```
255.255.192.0 (5)
```

(EIGPR对自动汇总和手工汇

总的路由,管理距离都是5)

Step2:在生成汇总的EIGRP路由器上,既有明细路由,也有汇总:

在生成汇总路由的R3上

```
R3#show ip route 175.10.0.0 255.255.248.0  
distance 5
```

察看某条路由的详细信息.

```
R3#show ip route
```

```
D 175.10.0.0/18 (AD:5) is a summary, Null0
```

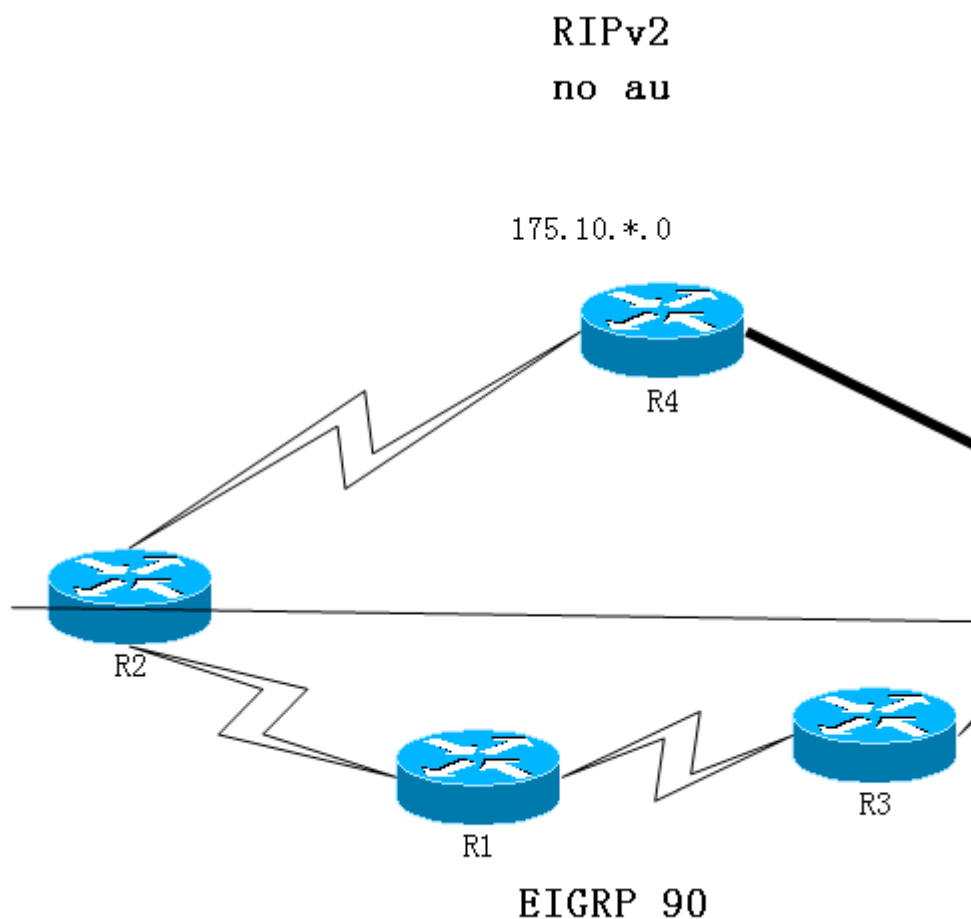
```
D 175.10.12.0/24 (AD:90), Ethernet0
```

在EIGRP中,无论是自动汇总还是手工汇总其AD都是: 5

在计算汇总路由的METIRC值时,在产生该路由的路由器上选择DELAY最小的接口。

LAB:Prefix-Lenth/AD/Metric的比较

~~~~~



观察R2对目标网络5.0.0.0 / 8 或5.5.5.0 / 24的选路。

Step1:Longest Prefix-Length Match (最长匹配)

在配置RIP时，要PASSIVE R5的S0和R2的S0口 LP接口两个协议都要跑

show ip route:

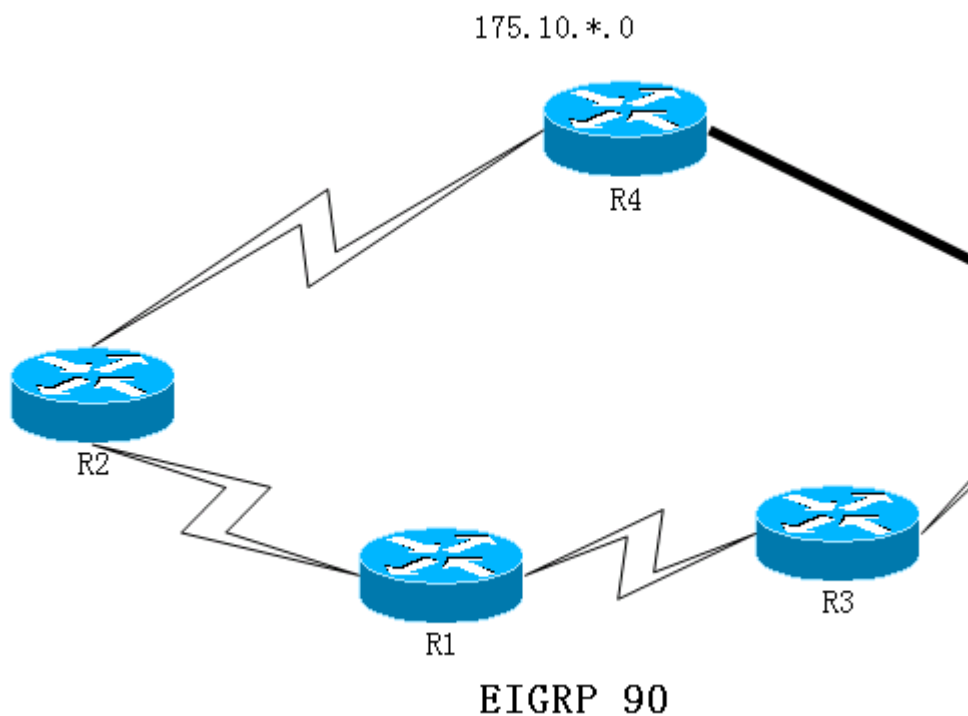
R 5.5.5.0/24 [120/2] via 175.10.24.4, serial1

Step2:当路由长度相同时，才比较AD。

把所在的EIGRP路由器都 NO Auto-Summary

D 5.5.5.0/24 [120/2] via 175.10.12.1, serial0

Step3:当AD相同时，才比较Metric.



在全网的路由器上运行EIGRP 90 (no auto-summary)

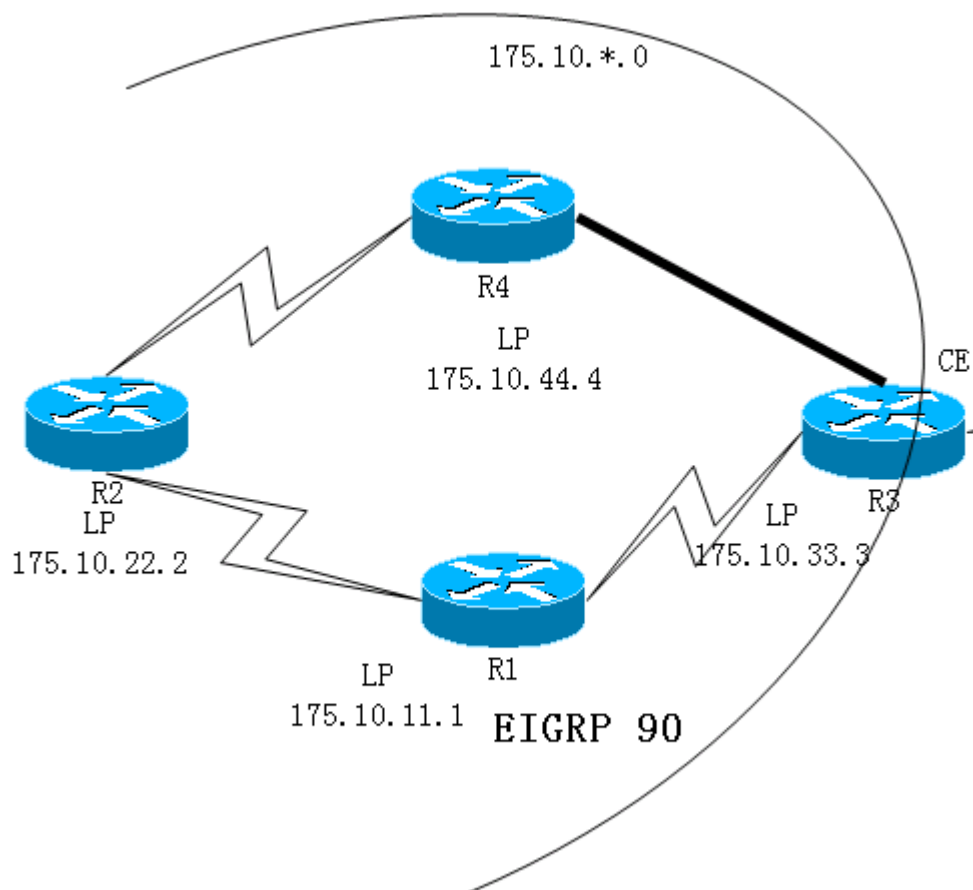
```
D 5.5.5.0/24 [90/2323456] via 175.10.24.4, serial1
```

但汇总路由一定比明细路由“短”，
按照“最长配置原则”，一定不会发到空接口，
而是按照特定明细路由所指示的接口，发送出去。

路由条目的比较步骤：

- 1: 首先按照“最长配置原则”，优先选择路由长度最长的路由。
- 2: 假如，有多条长度相同的路由，才按照AD最小进行比较。
- 3: 如果，连AD也相同，才比较每条路由的Metric值。

LAB6:Default-network for EIGRP(在用户连接ISP的边缘路由器上/R3)



在每个路由器上都建立一个LOOPBACK接口来模拟自己内部的网络。
`ip add 175.10.xx.x`

Step0:网络背景:

通常ISP与用户之间是不运行IGP的:

0-1:

在ISP R3上做去往用户的静态路由:

```
R3(config)#ip route 0.0.0.0 0.0.0.0 35.0.0.5
```

0-2:

```
R5(config)#ip route 175.10.0.0 255.255.0.0 serial 0
```

4-1:EIGRP产生默认路由的方法A:

Step1:重分布静态路由到EIGRP进程中:

```
R3#
```

```
router eigrp 90
```

```
redistribut static metric 1544 1000 255 1 1500
```

带宽 延迟 可靠性 负载率

Step2:内部的EIGRP路由器上察看由EIGRP生成的默认路由:

R1/2/4#

```
D*EX 0.0.0.0/0 [170/.....] via R3
```

Step3:

内网的EIGRP路由器上察看同EIGRP生成的默认路由:

R1/2/4#

```
ping 5.5.5.5 ! ! ! ! !
```

4-2:EIGRP产生默认路由的方法B:

Step1:

```
R3(config)#ip route 0.0.0.0 0.0.0.0 35.0.0.5
```

Step2:创建一个私网地址, 提供默认网络:

R3#

```
interface loopback192
```

```
ip address 192.168.1.1 255.255.255.0
```

(这个地址建议使用主类网络, 不建议使用子网地址)

Step3:将私网宣告到EIGRP中:

R3#

```
router eigrp 90
```

```
network 192.168.1.0
```

Step4:指定这个网络是默认网络, 它可以在EIGRP域中, 以EIGRP默认路由的形式存在和传播:

```
R3(config)#ip default-network 192.168.1.0
```

Step5:R1/2/4上察看路由:

```
D* 192.168.1.0/24 [90/
```

LAB9:EIGRP Route Authentication(路由协议的, 链路级别的, 链路认证)

Step1:定义Key-chain

R1/R3#

key chain CCNP (名称, 随便取, 都区分大小写)

key 1

key-string CISCO

Step2:

R1/R3#

interface s0/s1

ip authentication mode eigrp 90 md5 (EIGRP只支持MD5的加密认证)

ip authentication key-chain eigrp 90 CCNP

如果EIGRP的认证失效, 连EIGRP邻居都无法建立.

空格也是密码的一部分

LAB10:Adjusting the EIGRP Metric Weights(调整EIGRP的K值)

R3#show ip protocols

EIGRP metric weight k1=1, k2=0, k3=1, k4=0, k5=0

Router(config-router)#metric weights tos k1 k2 k3 k4 k5

在全AS的EIGRP路由器, 都必须有相同的K值, 建议不要轻易更改:

R5/R3(config-router)#metric weights 0 1 2 3 4 5

实际上, K值是EIGRP协议衡量:

BW/DL/Reliability/Loading/MTU, 这5个衡量路径优劣的不同参数的权重默

认只考虑BW/DL,

∴K1=K3=1 K2=K4=K5=0

1.4—动态路由协议OSPF①

∴K1=K3=1 K2=K4=K5=0

1.4—动态路由协议OSPF①

1.4—动态路由协议OSPF①

```
r2#sh ip ospf border-routers 查看ABR
```

修改OSPF接口优先级

```
r1(config)#int e 0
```

```
r1(config-if)#ip ospf priority 100
```

修改OSPF默认LOOPBACK口地址32位

```
r1(config)#int lo 0
```

```
r1(config-if)#ip ospf network point-to-point
```

DR选举原则：

1：在选举期间（默认40秒），优先级高的成为DR，次高的成为BDR；

2：在选举期间，如果优先级一样，router-id高的成为DR，次高的成为BDR；

3：在选举期外，不存在抢占性；

4：DR失效以后，BDR升级成为DR，重新选举BDR；

5：clear ip ospf process（重启OSPF进程）可以重选。

查看LSDB：

```
sh ip ospf database 包括所有LSA类型
```

查看LSA 1型（类型一描述邻居）

```
sh ip ospf database (router) 查看LSA类型1（Router Link States）
```

类型二描述DR，所以要多路访问才会出现：

```
sh ip ospf database (network)
```

多了Net link states 类型二（包括link id，DR是谁，接口，掩码多少，存在多少邻居）

通过上两种LSA信息可以描述这个AREA的拓扑。

类型三描述跨区域信息（ABR）

summary net link 不同区域不能描述对方拓扑。

LSA 5型
external link states

stub区域（所有该区域的路由器都要配）

r3/5:

```
router ospf 110
```

```
area 35 stub
```

过滤5类的LSA（外部路由）

生成3类的LSA默认路由

total stub除了过滤域外路由（如引入的eigrp路由）外，还过滤OSPF域间路由LSA 3。

同样生成3类默认路由

该区域所有路由器都打上：

```
router ospf 110
```

```
area 35 stub
```

该区域的ABR上打

```
area 35 stub no-summary
```

LSA 4型：

因为LSA 5型的公告路由器（ASBR）是不会改变的，所以需要4型来为除了AREA0 以外的区域寻路。

LSA类型4的作用，描述ASBR所在位置

LSA类型4由区域的ABR产生。

NSSA区域

Not-so-stubby areas:

可以过滤area0发来的外部路由，但可以引进外部路由到其他area。

过滤5类LSA，但是可以引入外部路由

产生LSA 7（由区域概念 nssa本区域可以看到0 N2，别的区域看不到。）

NSSA区域的ABR同时也是ASBR，负责把7类LSA转换成5类LSA。

OSPF (Open Shortest Path First)

理论：

OSPF三张表（OSPF AD:110）

~~~~~

1. Neighbor table（列出了所有和本路由器直接相连的OSPF邻居）

2. Topology table (LSDB链路状态数据库)

列举了所有从自己的邻居那得到的LSA, (Flooding/泛洪),  
在同一个OSPF区域中的路由器, 都有完全一致的OSPF Database。  
一个OSPF区域, 就对应着一个OSPF Database。

3. Routing table: (从OSPF这个路由协议, 学到的路由。)

在OSPF的数据库中, 通过SPF算法, 计算得到了路由。  
也称为: Forwarding Database

OSPF网络的层次化设计, 区域划分, 及划分的目的:

ospf two-level hierarchy:

Transit area (backbone or area 0)

Regular areas (nonbackbone areas)

**划分的目的:**

1. 提高路由效率:

缩减部分路由器的OSPF的路由条目。

对某些特定的LSA, 可以在区域边界 (ABR/ASBR) 上, 实现汇总/控制/过滤。

(通过OSPF的汇总路由/默认路由实现OSPF区域之间的全网互通)

2. 提高网络稳定性:

当某个区域内的一条OSPF路由出现抖动时, 可以有效控制受影响的波及面。

(对于大型的路由协议来说, 稳定是很重要的一个因素。)

3. OSPF VS. IS-IS的区域可扩展性的对比: 两种协议的算法都是基于SPF算法

OSPF: 以Area0为BackBone。 (比较好)

IS-IS: 以Level2的链路为BackBone, 以链路为区域分界。 (很好)

OSPF Routing updates and topology information are only passed  
between FULL adjacent routers.

物理网络链路类型分类: (L2概念)

1. P2P (HDLC/PPP Serial/ Point2Point Sub-if)

一定要求是Full状态。

的。

(点对点链路)

(没有DR / BDR的选举)

的, 代表是E1, 两台路由器直连)

(WAN / 广域网)

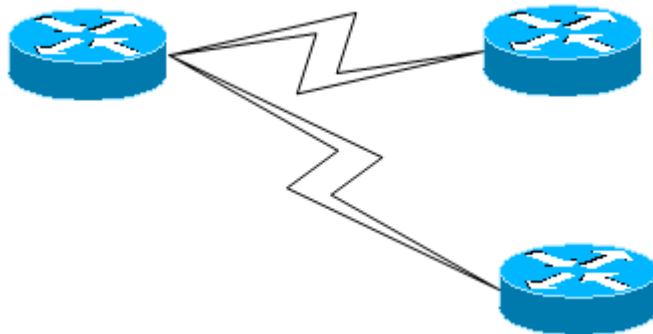
(WAN / 广域网)



## 2. BMA: Broadcast Multi-Access

(Ethernet/TR/FDDI) (代表是局域网) (有DR/BDR的选举的)

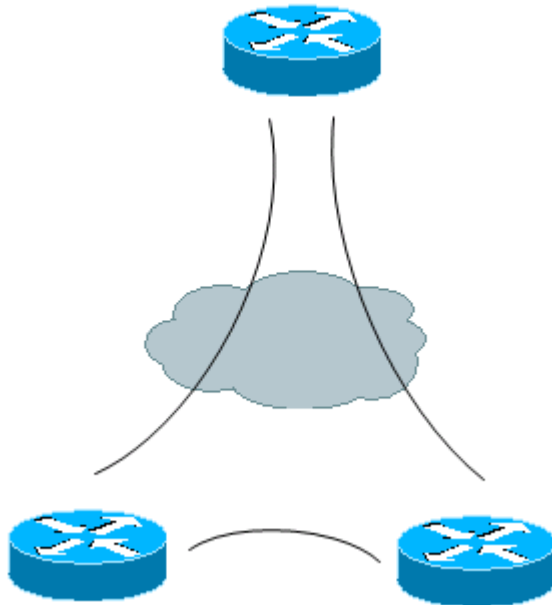
默认可以传输广播流量的，多路访问网络

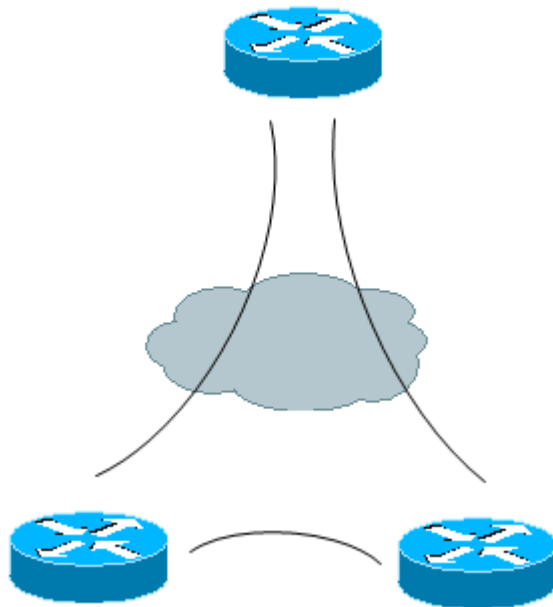


## 3. NBMA: Non-Broadcast Multi-Access

(FR/ATM/X.25) (代表是广域网，如帧中继) (有DR/BDR的选举的)

默认不传输广播流量的，多路访问网络





### OSPF的SPF算法:

1. 每一个OSPF区域，就对应着一个独立的OSPF Database (LSDB)  
意味着同在一个OSPF区域中的，所有路由器，都有相同的一个LSDB
2. 每一个OSPF路由器，都生成了以自己为根的，一棵SPF树。
3. 从本路由器出发，到特定目标网络的**整体开销最小**的那个路径，成为**最佳**路径。
4. 那么这条**最佳路径**，就成为OSPF这个协议，**提交给路由表的**，到达这个目标网络的**路由**。

### LSA的传播更新规律（OSPF是LS协议，无需遵循水平分割，DV协议才遵循水平分割。）

**Step1:**如果本路由器从来没有收到过此LSA，那么路由器就将其加入LSDB，并且转发/泛洪此LSA，同时继续SPF计算，得出达到此目标的最佳路由。

**Step2:**如果本路由器，曾经受到过描述同一个网络的LSA: (seq:100)

- 2-1. 如果新收到的LSA序号与自己已有的**相同** (seq:100)，则**丢弃**此LSA。
- 2-2. 如果新收到的LSA序号比自己已有的**更新** (seq:101)，则同Step1，去计算最佳路由。
- 2-3. 如果新收到的LSA的序号，比自己的**更旧** (seq:99)，就将自己较新的

去计算最佳路由。

2-3. 如果新收到的LSA的序号，比自己的更旧(seq:99)，就将自己较新的LSA，发送给源。

### OSPF的5种数据包

1. Hello（建立和维护邻居（Neighbor）关系，路由器发送Hello包的缺省时间间隔是10秒）
2. DBD(Database Description)
3. LSR(LinkStatus Request)
4. LSU(LinkStatus Update) (LSA是包含在LSU中的)
5. LSAck

### OSPF的Protocol ID:89 (EIGRP:88)

在OSPF的Hello包中，影响建立邻居关系的4个关键因素：

Hello/dead interval

Area-ID（链路所在的Area ID）

Authentication password（OSPF认证的密码）

Stub area flag（NSSA标示位）

这四个因素必须匹配才能建立邻居，否则无法建成OSPF邻居

在BMA网络和点对点网络上，默认的Hello Interval值是10秒，Dead Interval值是40秒。在NBMA网络上，默认的Hello Interval值是30秒，Dead Interval值是120秒。

修改Hello Interval和Dead Interval的值：（在接口上修改）

R1(config-if)#ip ospf hello-interval time(time的取值为1-65535秒)

R1(config-if)#ip ospf dead-interval time(time的取值为1-65535秒)

### OSPF邻接关系的建立过程：

1. Down（路由器A从运行OSPF的接口以组播地址224.0.0.4发送Hello数据包）
2. Init（所有收到从路由器A发送来的Hello数据包的路由器，都把路由器A添加到自己的邻居Neighbor列表中）
3. Two Way（所有收到路由器A的Hello包的路由器都向其发送一个单点传送的回复Hello包，其中包含有它们的信息。路由器A收到这些信息后，检查这些数据包，把哪些Hello包的邻居域中有自己ID的路由器也加入

送的回复Hello包，其中包含有它们的信息。路由器A收到这些信息后，检查这些数据包，把哪些Hello包的邻居域中有自己ID的路由器也加入自己的邻居列表中。在这个过程中同时选举出DR和BDR）

4. Exstart（DR和BDR与其他的路由器建立相邻关系（Adjacency）。
5. Exchange（由DR向其他路由器发送数据库描述数据包（DBD, Database Description）。DBD有序号，由DR决定DBD的序号）
6. Loading（发送链路状态请求包的过程）
7. Full（路由器及哪个新的链路状态条目添加到它们的链路状态数据库中。当所有的LSR都得到答复时，相邻的路由器就被认为达到了同步并处于“Full”状态了。路由器必须在达到Full状态后才能正常转发数据。此时区域内的每个链路应该都有相同的数据链路状态数据库。）

### OSPF数据包的发送地址

DR/BDR notifies LSU on 224.0.0.5（映射到二层MAC地址：010005e00000005）  
DR-Other notifies LSU to OSPF DR on 224.0.0.6（映射到二层MAC地址：010005e00000006）  
DR负责宣告整个网络的路由更新，BDR或DR-Other只能先把路由更新先发给DR，然后再由DR发给BDR和DR-Other

每次收到LSU，路由器在重新计算路由表之前等待一段时间，默认是5秒。每个LSA都有一个老化（Aging）计时器，到期时由产生该LSA的路由器再发送一个有关该网络的LSU以证实该链路仍然是活跃的，这个Aging时间默认是1800秒。

DR和BDR是在交换Hello数据包的过程中选举出来的，然后其他路由器都与DR和BDR建立相邻关系。每台DR-other路由器都只与DR和BDR建立相邻关系（Adjacency），交换链路状态信息。

### LAB1. OSPF的Router-ID（要求全网唯一）

一旦启动OSPF，立刻确定Router-ID

通过此命令察看Router-ID：

```
R1#show ip ospf
```

在OSPF路由器上，确定Router\_ID的3个优先级别：

**Step1:**（建议使用router-id命令来确定Router-ID）

通过router-id命令，修改Router-ID，其优先级别最高，也是建议的。（先

(先建立一个LOOPBACK口作为R-ID用)

```
R1(config)#router ospf 110  
R1(config-router)#router-id 100.0.0.1
```

**Step2:**假如没有通过router-id命令指定router-id,  
那么路由器会自动的将自己的环回口的IP, 作为router-id.

如果存在多个环回口,  
那么路由器会自动的选择一个IP地址最大的那个环回口IP作为自己的Router-ID。

**Step3:**

如果路由器上, 连一个环回口都没有,  
那么路由器会自动从当前是Active (激活状态下: UP/UP) 的物理接口中,  
选择IP地址最大的那个接口的IP作为自己的Router-ID。这是很不稳定的,  
不建议的方法。

LAB2. 通过反掩码控制有哪些接口, 在运行OSPF (在OSPF/EIGRP中, network命令中携带的反掩码不表示接口网络长度, 而表示运行路由协议的接口的范围, 即有哪些接口在运行OSPF)

反掩码/通配符:wild card bits

反掩码原则:

0:表示准确匹配

1:表示忽略不计

LAB3.

show ip ospf neighbor (detail) (查看路由器的OSPF邻居表, 当前有哪些OSPF的邻居, DR/BDR/DR-other状态)

show ip ospf interface (查看有哪些接口在运行OSPF, 本路由器是DR, 或者BDR, 还是DR-other, 还有优先级)

show ip ospf database

show ip route ospf

show ip route ospf

LAB4. DR/BDR的选举：（只发生在多路访问网络/Multi-Access Network，BMA和NBMA）

1. 在点对点链路，是没有DR/BDR的选举

2. 在BMA网络中：

2-1. OSPF首先通过优先级，控制DR/BDR的选举：

优先级越大，越可能成为DR。OSPF路由器的优先级，默认是1。

如果需要进行DR的人为控制，应该建议，通过OSPF的接口优先级进行控制。

修改特定接口的优先级

```
R1(config)#int s0
```

```
R1(config-if)#ip ospf priority 10
```

```
R1#clear ip ospf process（清OSPF进程）
```

特别注明：OSPF的优先级是针对某个特定的MA接口而言的，不是针对整个路由器的。

2-2. 如果OSPF路由器的优先级，全部都是默认值1，路由器默认通过Router-ID, 选举DR/BDR，Router-ID最大的成为DR，次大的成为BDR。其余的统统都是DR-other。

3. 在Hub&Spoke的NBMA网络中，中心点（HUB）应该成为DR。

结论：

1. 同一个路由器的不同MA接口，可能在不同的MA网络中，充当不同的DR/BDR/DR-other。

2. 在一个MA网络中：

DR/BDR与所有的邻居都是Full状态，DR-Other与DR/BDR是Full的，但与别的DR-Other是2way状态。

特别注意：

只有Full状态才能交换路由信息。

关于MA (Multi-Access) 网络的DR/BDR的选举：

Step1. 根据OSPF路由器的OSPF接口的优先级选举DR/BDR：

每个接口默认的优先级都是：1。

其中优先级最大的成为DR，次大的成为BDR，其它的都是DR-Other。

如果有路由器的Pri:0，放弃DR/BDR的选举，成为DR-Other。

（OSPF Priority:0~255）

Step2. 如果接口的优先级相同，将使用Router-ID来进行DR/BDR的选举：

其中Router-ID最大的成为DR，次大的成为BDR，其它的都是DR-Other。

在选出DR/BDR后，如果有新的优先级更高的路由器加入，那么新加入的路由器并不会成为DR/BDR，需要在下次选举中才能生效。

最长匹配，AD比较，Metric比较

先比较最长匹配如（10.2.2.0/24，10.0.0.0/8，将选10.2.2.0/24，而不管路由协议的AD），然后比较AD，最后比较Metric。

## 1.4—动态路由协议OSPF②

# 1.4—动态路由协议OSPF②

### LAB2. 通过反掩码控制有哪些接口，在运行OSPF

~~~~~

step1:启动OSPF，并宣告网络：

```
R1(config)#router ospf 110
```

```
R1(config-router)#network 192.16.1.1 0.0.0.0 area 0
```

（表示特定一个接口，在运行OSPF协议）

```
R3(config-router)#network 0.0.0.0 255.255.255.255 area 0
```

（表示路由器上的所有接口，都运行OSPF协议）

反掩码/通配符:wild card bits

反掩码的匹配原则：

0:表示准确匹配

1:表示忽略不计

结论：

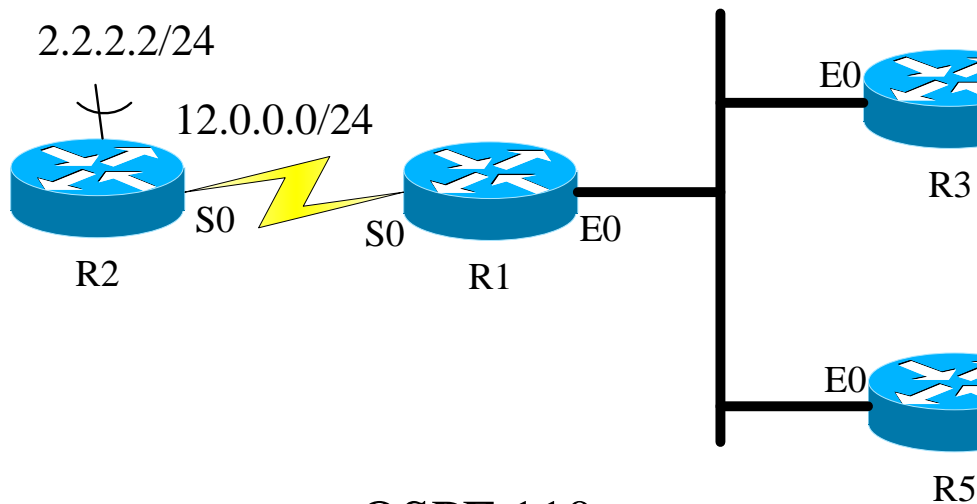
network命令中携带的反掩码，

不表示这个接口所在的网络长度

而表示运行路由协议的接口范围（有哪些接口在运行EIGRP/OSPF）

LAB3:OSPF必需查看的4个SHOW

~~~~~



## OSPF 110

show ip ospf interface

(查看有哪些接口在运行OSPF, 本路由器是DR, 或者BDR, 还是DR-other, 还有优先级)

show ip ospf neighbor

(查看路由器的OSPF邻居表, 当前有哪些OSPF的邻居, DR/BDR/DR-other状态)

show ip ospf database

(察看路由器的LSDB:)

show ip route ospf

(察看从OSPF学到的路由)

### LAB4. DR/BDR的选举: (前提: 只发生在多路访问网络/Multi-Access Network, BMA和NBMA)

~~~~~

察看OSPF路由器的DR/BDR的状态

show ip ospf interface ethernet 0

Router ID 100.0.0.1,

state DR?BDR/DR-other, Priority 1

1. 在点对点链路, 是没有DR/BDR的选举

2. 在MA网络中:

2-1: OSPF首先通过接口优先级, 控制DR/BDR的选举: (优先级越大, 越可能成为DR。)

OSPF路由器的接口优先级, 默认是1。

OSPF路由器的接口优先级，默认是1。
如果需要进行DR的人为控制，
应该建议，通过OSPF的接口优先级进行控制。

```
R1(config)#int e0 (修改特定接口的优先级)
R1(config-if)#ip ospf priority 10
(OSPF Priority:0~255)
```

```
R1#clear ip ospf process (清OSPF进程)
```

特别注明：OSPF的优先级是针对某个特定的MA接口而言的，不是针对整个路由器的。

2-2:OSPF的接口优先级相同的情况：

如果OSPF路由器的优先级，全部都是默认值1，路由器默认通过Router-ID，选举DR/BDR，Router-ID最大的成为DR，次大的成为BDR。其余的统统都是DR-other。

2-3:OSPF的接口优先级如果为0, 表示该路由器放弃DR选举

2-4:：在Hub&Spoke的NBMA网络中，中心点（HUB）应该成为DR，无BDR。

2-5:OSPF的DR/BDR的选举，**无抢占性**。

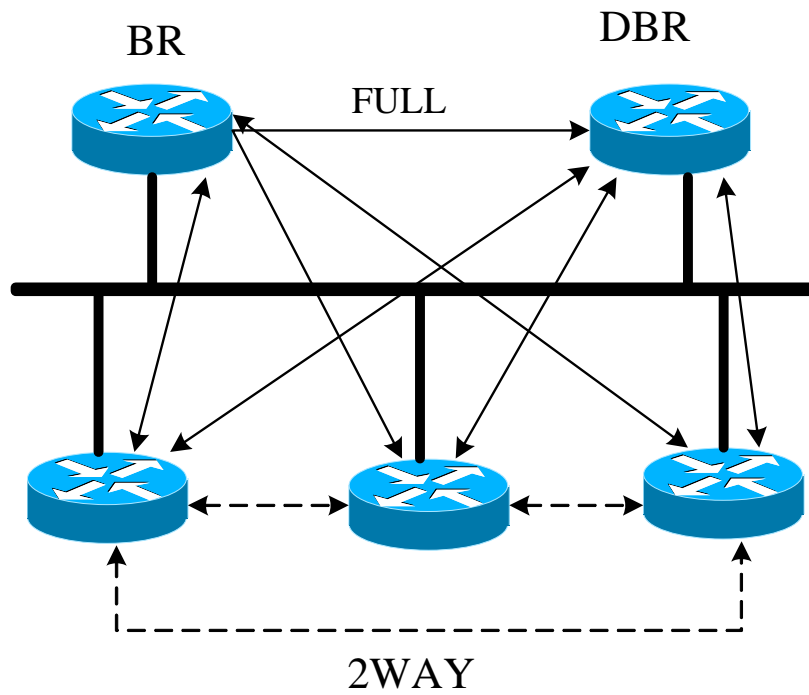
结论：

1. 同一个路由器的不同MA接口，可能在不同的MA网络中，充当不同的DR/BDR/DR-other。

DR/BDR只是针对接口而言，而不是针对整个路由器。

2. 在一个MA网络中：

DR/BDR与所有的邻居都是Full状态，DR-Other与DR/BDR是Full的，但与别的DR-Other是2way状态。



特别注意：

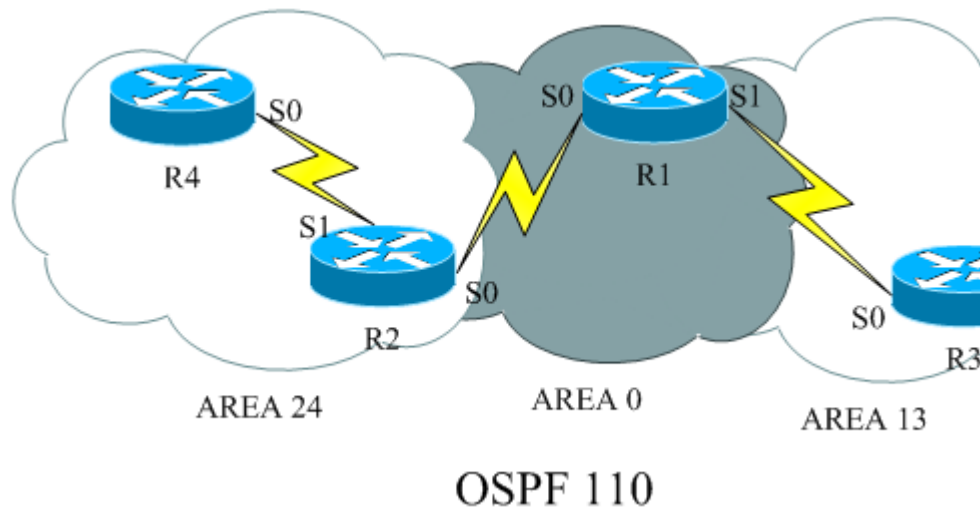
只有Full状态才能交换路由信息。

在选出DR/BDR后，如果有新的优先级更高的路由器加入，那么新加入的路由器并不会成为DR/BDR，需要在下次选举中才能生效。

Advance OSPF Controlling

LAB5: OSPF的基本实验配置

~~~~~



**Step1:**

```
int loopback1
ip add 195.100.0.x 255.255.255.0
```

**Step2:** 启动OSPF, 明确指定Router-ID

```
router ospf 110
router-id 195.100.0.x
```

**Step3:** 宣告OSPF的链路在哪个区域。(宣告网络) (激活接口, 运行OSPF)

```
router ospf 110
network *.*.*.* *.*.*.*
```

建议宣告每一个Router-ID的网络, 便于网管Telnet。

如Router-ID是195.100.0.1, 则宣告网络network 195.100.0.0 0.0.0.255 area 0。

**Step4:** OSPF的三类路由

```
R4#show ip route
```

0 100.0.0.0/16 (域内路由)

0 IA 12.0.0.0/24 (域间路由: IA-OSPF inter area)

0 IA 13.0.0.0/24 (域间路由)

0 E2 5.0.0.0 / 8 (域外路由:E-OSPF external)

0 E2 35.0.0.0 / 24 (域外路由)

在R3这个同时运行EIGRP/OSPF的路由器上, 将EIGRP路由重分布到OSPF:

```
R3(config)#router ospf 110
```

```
R3(config)#router ospf 110
R3(config-router)#redistribute eigrp 90 subnets
```

### LAB6:OSPF的Metric计算

影响OSPF Metric计算的3种操纵方法  
在路由协议计算Metric/Cost值时，只计算路由的入口

OSPF计算Cost的公式：

$Cost = 100,000,000 / BW$ （路由入口的带宽，其单位是bps）（ $10^8 / BW$ ，BW的单位是bps）

OSPF 的路由的Metric值：每个接口Metric值的累加。

**Step1:**收集每种接口的带宽：

```
Show interfaces serial 1 (BW 1544 Kbit)
```

```
show interfaces ethernet 0 (BW 10,000 Kbit)
```

```
show interfaces fast-ethernet 0 (BW 100,000 Kbit)
```

show interfaces loopback 5 (BW 8,000,000 Kbit, 环回口的OSPF Cost固定为1，而不管参考带宽的值是多少)

R1访问35.0.0.0网络的Cost是 $10^8 / BW + 10^8 / BW$ （BW是bps，即1544Kbps=1,544,000）

**Step2:**把每个接口的OSPF COST值累加，就可以得到OSPF路由的Metric。

### LAB7：影响OSPF Metric计算的3种操纵方法：（操纵OSPF路由）

**方法1：**直接修改接口Cost值

```
R4(config)#int E0
```

```
R4(config-if)#ip ospf cost 128
```

（用这个命令修改后，就不再用 $10^8 / BW$ 这个公式计算Cost了）

**方法2:**修改接口的带宽BW

```
R3(config)#int s0
```

```
R3(config-if)#bandwidth 2048
```

（单位是Kbps，2.048Mbps/E1）

**结论:**

bandwidth: 影响路由协议的选路

clock rate: 影响物理链路的真实速率

**方法3: 修改参考带宽 (分子)**

工程/题目需求:

考虑到为了的网络发展, 要求将来再10Gbps链路上, 其OSPF Cost为: 10,  
所以我们要把分子更改为100G, 即 $10^{11}$ 。(1G的链路, 其COST为100)

**在全网路由器上:**

```
R1(config)#router ospf 110
```

```
R1(config-router)#auto-cost reference-bandwidth 100000 (单位是  
Mbps)
```

要求: 一旦准备更改OSPF的参考带宽, 就必须在所有的OSPF路由器 (包含不同区域) 上一起更改。

"ip ospf cost"设置的值要优先于 "auto-cost reference-bandwidth"命令计算出来的值。

**特别注意:**

1. 路由入口的定义。

2. 同步串行的同步时钟速率: clock rate 2400, 只会影响链路的真实物理速率 (带宽), 不会影响OSPF的Cost的计算。

但接口种配置的 "Bandwidth 2048", 只影响OSPF的Cost的计算, 影响OSPF选路, 不影响真实物理速率

3. 对于以太网, 其速率就等于其接口的带宽。

改接口速率:

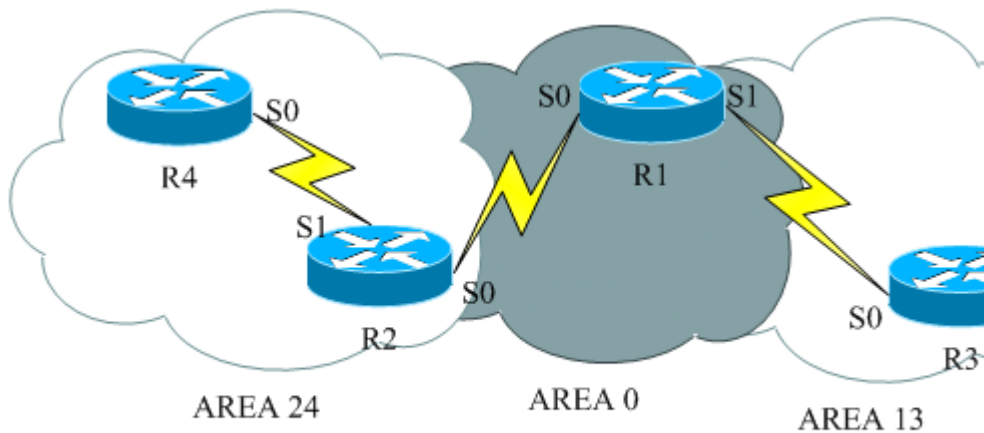
```
int fastethernet 0/1  
speed 10(100)
```

## 1.4—动态路由协议OSPF③

## 1.4—动态路由协议OSPF③

## 1.4—动态路由协议OSPF③

### OSPF的路由汇总



OSPF 110

用默认的网络地址

因为

无论在何种路由协议的路由汇总中：

生成的汇总路由包含范围过大，则很可能形成路由黑洞。

生成的汇总路由包含范围过小，则很可能丢失部分明细路由。

所以

默认情况下，在工程/题目没有指定汇总的长度的时候，应该进行最精准的汇总。

RIP和EIGRP的路由汇总是设置在接口上的，它们是DV协议。

链路状态路由协议的路由汇总需要在路由进程中设置，链路状态协议没有自动汇总的特性。

1:OSPF的域间汇总，发生在连接不同OSPF区域的ABR上。

0 IA (IA:inter-area)

ABR:(互联了2个，或者2个以上的OSPF区域的路由器。)

2:OSPF的域外汇总，发生在OSPF与别的路由协议相连的ASBR上。

0 E1 (OSPF external type 1)

0 E2 (OSPF external type 2)

ASBR: (在OSPF区域的边界上，互联了OSPF和别的其它路由协议的路由器。)

### LAB3. OSPF的域间路由的汇总

Step1:按图配置网络拓扑

Step2: 在Area24中的路由器R4上, 模拟4条/26的路由:

```
interface loopback1
ip add 175.14.14.1 255.255.255.192
ip ospf network point-to-point
```

```
175.14.14.1/26
175.14.14.65/26
175.14.14.129/26
175.14.14.193/26
```

全网的OSPF路由器上, 都可以察看到4条 / 26的明细路由

Step3: 在明细路由所在的区域Area 24的ABR上, 进行OSPF域间路由汇总:

```
R2(config)#router ospf 110
R2(config-router)#area 24 range 175.14.14.0 255.255.255.0
    原明细路由所在区域 (在ABR上做)
```

Step4:

R1/R3, 只有汇总路由

```
0 IA 175.14.14.0/24
```

R2, 即有明细路由, 也有汇总路由

R4, 没有汇总, 只有明细。R4代表整个Area24中的所有路由器。

路由汇总黑洞 (null0), 只会出现在做路由汇总的路由器上。

在OSPF协议下向区域内产生一条默认路由的语法:

```
R1 (config-router)#default-information originate [always]
```

使用Default-information originate命令产生缺省路由的前提是, 使用该命令的路由器必须存在一条默认路由。

如果不使用参数(always), 那么路由器上必须存在一条0/0的默认路由, 它把默认路由通过到整个区域, 否则该命令不起作用。但使用参数(always)时, 无论路由器上是否存在0/0的默认路由, 使用该命令的路由器总会向区域内注入一条默认路由。

### LAB4: OSPF的域外路由的汇总。

**LAB4: OSPF的域外路由的汇总。**

```
R3/R5上运行EIGRP,
R5上,
interface Loopback1
 ip address 175.15.13.129 255.255.255.240
!
interface Loopback2
 ip address 175.15.14.161 255.255.255.240
!
interface Loopback3
 ip address 175.14.15.177 255.255.255.240
```

**Step3:**在R3上察看EIGRP的明细路由

```
R3#sh ip route eigrp
D      175.14.15.176/28 [90/2297856] via 35.0.0.5, 00:03:04,
Serial1
D      175.15.14.160 [90/2297856] via 35.0.0.5, 00:03:04, Serial1
D      175.15.13.128 [90/2297856] via 35.0.0.5, 00:03:04, Serial1
```

**Step4:**

在R3（ASBR）上将域外的EIGRP路由，重分布（Redistribute）到OSPF中。

```
R3(config)#router ospf 110
R3(config-router) redistribute eigrp 90 subnets
```

**Step5:** 在OSPF域内的路由器，察看到域外的明细路由：

```
R1/2/3#
O E2    175.14.15.176/28 [110/20] via 24.0.0.2, 00:03:57, Serial0
O E2    175.15.14.160 [110/20] via 24.0.0.2, 00:03:58, Serial0
O E2    175.15.13.128 [110/20] via 24.0.0.2, 00:03:58, Serial0
```

**Step6:**

在R3（ASBR）上将域外的路由，做OSPF的域外汇总：

```
R3(config)#router ospf 110
R3(config-router)#summary-address 175.15.12.0 255.255.252.0（路由
长度，即/22）
```

（在ASBR上做）

**Step7:**在OSPF域内的路由器，察看到域外的汇总路由：

```
0 E2    175.15.12.0
```

E1 - OSPF external type 1,

（在路由传播的路径上，OSPF的Cost会随着路径的远进叠加链路的开销，其

**Step7:**在OSPF域内的路由器，察看到域外的汇总路由：

```
0 E2    175.15.12.0
```

E1 - OSPF external type 1,

(在路由传播的路径上，OSPF的Cost会随着路径的远进叠加链路的开销，其Cost会发生改变)

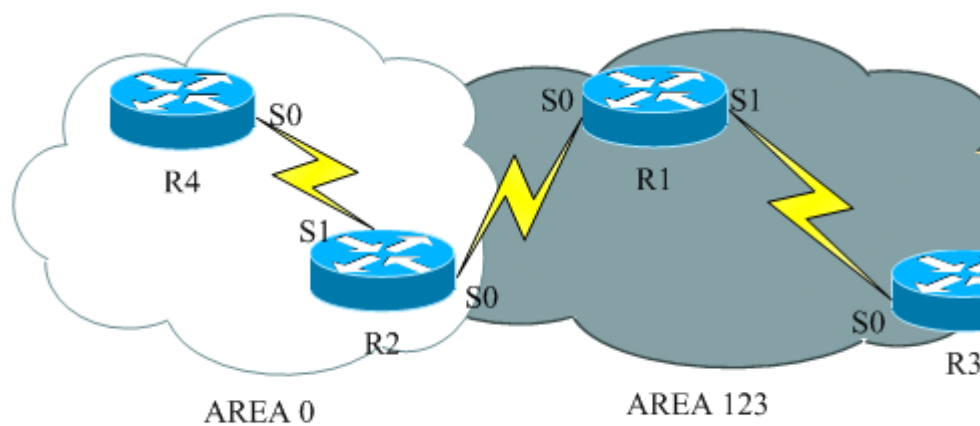
查看OSPF路由小技巧：

```
show ip route | include 0 (大写 显示OSPF域内路由)
```

```
show ip route | include 0 E2 (显示OSPF域间路由)
```

```
show ip route | include 0 IA (显示OSPF域外路由)
```

### LAB5. OSPF Virtual Links(虚链路)



## OSPF 110

**Step1:** 在Area0和非连续区域（Area35）的ABR上（R2/R3）

在virtual link穿越的区域内（area 123）的ABR上，互相指向对方的Router-ID

R2的Router-ID是192.100.0.2

R3的Router-ID是192.100.0.3

```
R2(config-router)#area 123 virtual-link 192.100.0.3
```

```
R3(config-router)#area 123 virtual-link 192.100.0.2
```

**Step2:** 检查OSPF的virtual link的连接状态：

```
process 110,
```

```
nbr 192.100.0.2 on ospf_vll from loading to full, loading done
```

```
R2#show ip ospf virtual-links
virtual link ospf_vl0 to router 192.100.0.2 is up Adjacency State
Full(检查Full, 不能看UP)
```

**Step3:** 在骨干区域的路由器, 就可以接受到非连续区域的路由。全网路由正常。(Full route/全路由)

**Step4:** 每个路由器的数据库的个数:

R4:1个 (Area 0)

R5:1个 (Area 35)

R1:1个 (Area 123)

R2:2个 (Area 0, Area 123)

R3:3个 (Area0, Area 35, Area 123)

**Step5:** OSPF virtual link的应用要点:

5-0. By default, all areas must connect to area 0

5-1. 在大型网络工程中, 由于历史原因, 导致网络扩展不佳, 迫于网络扩容的原因, 被迫新建OSPF区域, 使用OSPF虚链路。

5-2. 在大型网络中, 处于网络冗余考虑, 选择合适的ABR做OSPF-VL, 避免因为个别物理链路的的中断, 导致整个OSPF区域的全网中断。

5-3. 导致OSPF的Area0区域出现双BackBone

## 1.4—动态路由协议OSPF④

# 1.4—动态路由协议OSPF④

**多区域的OSPF:**

**划分多区域的主要目的:**

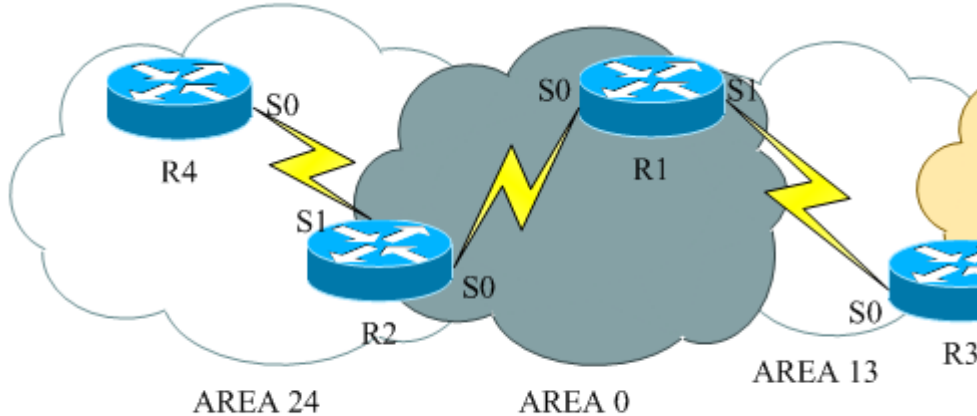
1、减少每个区域中的路由条目, 进而减少每个路由器的内存中的路由, 及其内存消耗, 提高转发效率。

2、因为每一个OSPF区域对应在一个OSPF LSDB, 配合在ABR/ASBR上做路由汇总, 划分多区域可以防止某个区域中的路由抖动, 影响到别的区域。(减少爱波及的范围)

OSPF的Area 0是Backbone区域, 必需配置有。

OSPF的其它区域都是普通区域, 可选配置。

### OSPF路由器的分类:



## OSPF 110

1、Internal Route: (R4)

路由器的所有接口 / 网段都在同一个OSPF区域中;

2: Backbone Route: R1 / R2:

至少有一个接口, 运行在Area0的路由器;

3、ABR (Area Border Route) / 区域边界路由器:

承担两个或多个OSPF区域的互联的路由器

4、ASBR / 自治系统的边界路由器: (R3)

至少有一个接口不是在运行OSPF, 而是运行别的路由协议, 它可将非OSPF路由传入OSPF的区域。

### OSPF的LSA (Link-State Advertisement) 种类:

#### LSA1 (Route LSA) :

跟本路由器相邻的直链路由 (sh ip os database查看)

由Originating router生成的, 通告。

只在**本区域**传递, 而不穿越ABR到别区域去。

描述了链路状态 (link-ID) 和是否ABR/ASBR; LSA1所描述的link-ID在不同的链路类型如下: P2P的是邻居的Route-ID; TransitNetwork的是DR的接口地址; StubNetwork的是IP网络号; 虚链路VL的是邻居的Route-ID;

#### LSA2 (Network LSA) :

show ip ospf database network

描述根本路由器的BMA和NBMA网络路由

### show ip ospf database network

描述根本路由器的BMA和NBMA网络路由

由DR生成，通告。

只在本区域内传递，而不穿越ABR到别区域去。

只在本区域传递并由DR宣告；LSA1所描述的link-ID是DR进行宣告的那个接口的IP地址；

### LSA3 (Summary LSA) :

show ip ospf database summary

指的是OSPF的域间路由。(O IA)

原LSA1所描述的路由，会由原区域的ABR将其转换为LSA3，

LSA3可以传播到整个OSPF的所有区域（特殊区域除外）

LSA3每穿越一个ABR，其ADV Router会发生改变。

### LSA4 (ASBR-summary LSA) :

sh ip ospf database asbr-summary

LSA4所承载的内容是：ASBR的Router ID

是由ASBR所在的那个区域的ABR产生并通告的

LSA4可以传播到整个OSPF的所有区域（特殊区域例外）

LSA4每穿越一个区域其Advertising Router都会发生改变（是下一个区域的ABR）

LSA4, 其实就是R1将AREA 13中的，描述R3的R-ID的LSA1，转换成LSA4，在整个OSPF域中进行泛洪传播。

Type-5 External Link States(不分区域的)

| Link ID  | ADV Router |
|----------|------------|
| 5.5.5.0  | 110.0.0.3  |
| 35.0.0.0 | 110.0.0.3  |

Area13域内的路由器：

area 13-----

LSA1:

Net Link States (Area 13)

| Link ID   | ADV Router |
|-----------|------------|
| 110.0.0.3 | 110.0.0.3  |

AREA 0-----

LSA4:

Summary ASB Link States (Area 0)

| Link ID   | ADV Router |
|-----------|------------|
| 110.0.0.3 | 110.0.0.1  |

Router Link States (Area 0)

| Link ID | ADV Router |
|---------|------------|
|---------|------------|

110.0.0.1            110.0.0.1

area 24-----

Summary ASB Link States (Area 24)

Link ID            ADV Router

110.0.0.3            110.0.0.2 (LSA的Adv router, 每穿过一个ABR/区域, 都会发生改变)

Router Link States (Area 24)

Link ID            ADV Router

110.0.0.2            110.0.0.2

### LSA5 (External LSA) :

show ip ospf database external

描述OSPF区域以外的路由

由ASBR产生并通告

LSA5可以传播到整个OSPF所有区域（特殊区域例外）

关键问题:

The advertising ASBR Router ID is unchanged throughout the AS.

LSA5需要LSA4协同工作, 让访问的网络路由器寻找到ASBR的所在位置;

### LSA 7 :

show ip os da nssa-ex

nssa就是允许存在NSBR, 允许引入外部路由的Stub区域。

外部路由, 在进入NSSA区域后, 以LSA7存在于NSSA区域内, (LSA 7只在NSSA区域内传播)

然后在NSSA区域与普通区域交界的ABR:R1上,

将LSA7转换成LSA5, 向别的区域泛洪 / 传播

NSSA是标准的RFC定义。

### LSA6 (ADV Route LSA/Multi Broad LSA) :

使用在OSPF多播应用程序里;

### LSA7 (NSSA External LSA) :

使用在Not-So-Stubby area (NSSA) 里 ;

### LSA8 (DataBase Summary) :

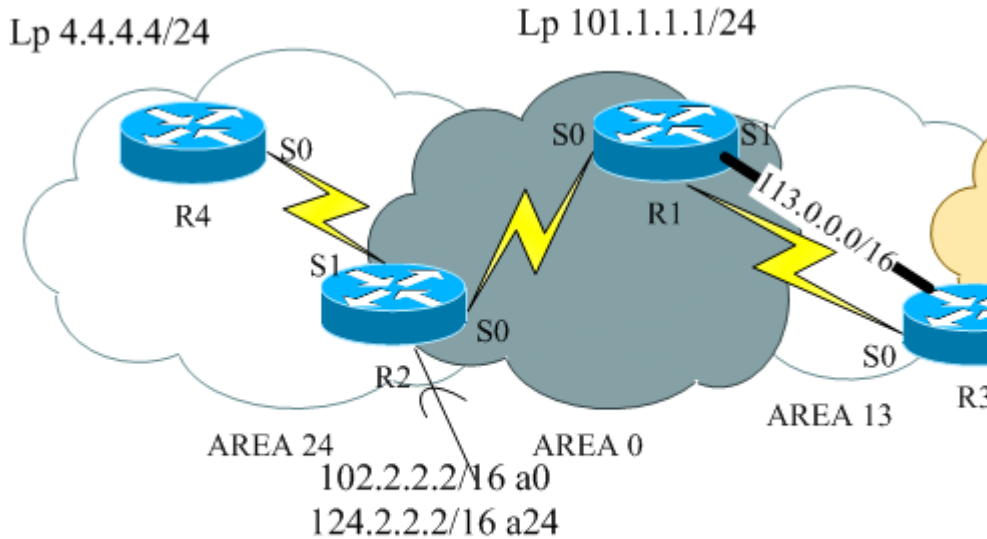
汇总的数据库, 用来连接OSPF和BGP的特殊LSA;

汇总的数据库，用来连接OSPF和BGP的特殊LSA；

LSA9/10/11 (Opaque Link/Area/AS LSA)

：用于今后OSPF的升级等。

LAB1：观察LSA1/LSA2/LSA3/LSA4/LSA/LSA5



## OSPF 110

Step1: 按图配置OSPF网络

router-id 110.0.0.X(路由器上没有配置这个地址，工程中建议配置)

LSA1: show ip ospf database router

Step2: 在R3/R5之间构建EIGRP网络

2-1: 在R3上，将EIGRP路由，重分布到OSPF中。

R3#

```
router ospf 110
```

```
redistribute eigrp 90 subnets
```

2-2: 察看LSA5:

```
show ip ospf database external
```

观察到:

LSA5, 描述的是OSPF的域外路由，这种LSA是不属于任何一个OSPF区域的，它可以泛洪到OSPF的所有区域。（没有某个AREA的标识符）

不管LSA5泛洪到哪里（哪个区域 / 数据库），其ADV Router都是ASBR的R-ID，一直不会改变。

5.5.5.0 110.0.0.3  
35.0.0.0 110.0.0.3

而且R4无法通过IP路由（IGP：RIP/EIGRP/OSPF），获得到达110.0.0.3的路由。

这就是LSA5需要LSA4提供协作的原因。

LSA4的作用：

LSA4协助LSA5，让“内部的”OSPF路由器，在访问AS以外的网络时，寻找到ASBR的所在位置。

“内部的”：除了ASBR所在区域(Area 13)，以外的所有OSPF区域(Area0/24)的路由器

一些Route designator:

- 0 : 代表OSPF area内(intra-area)路由，为router LSA;
- 0 IA : 在一个AS里的area之间(inter-area)的路由，为summary LSA;
- 0 E1/E2 : AS外路由，为external LSA。

## 1.4—动态路由协议OSPF⑤

### 1.4—动态路由协议OSPF⑤

OSPF的特殊区域（Stub/total Stub区域，无法引入外部路由）：

第一种级别：减少（Area 2 4 中的路由，去掉域外的，保留域间 / 域内）

Stub区域的路由器，只有域内 / 域间的路由，没有域外路由。

Area24                      Area24/0/13                      EIGRP

阻止了LSA5/4进入Stub区域

第二种级别:total stub (totally stubby area) :

Total stub区的路由器，只有区域内的路由，没有域间和域外路由。

阻止了LSA5/LSA4/LSA3进入total stub区。

Stub区域的ABR，也会自动向Stub区域的路由器，发送默认路由。

Stub区域的ABR，也会自动向Stub区域的路由器，发送默认路由。

**OSPF的特殊区域（NSSA/total NSSA）：（Area 13）**  
**（NSSA=Stub区域功能+能够引入外部路由的功能）**  
~~~~~

第一种级别，NSSA：

标准NSSA区域路由器，只有域内/域间的路由，

但没有其他区域的ASBR（R2）所引入的域外路由，（1.0.0.0，
100.0.0.0）

但本NSSA区域中的ASBR（R3），却可以引入域外路由（EIGRP）

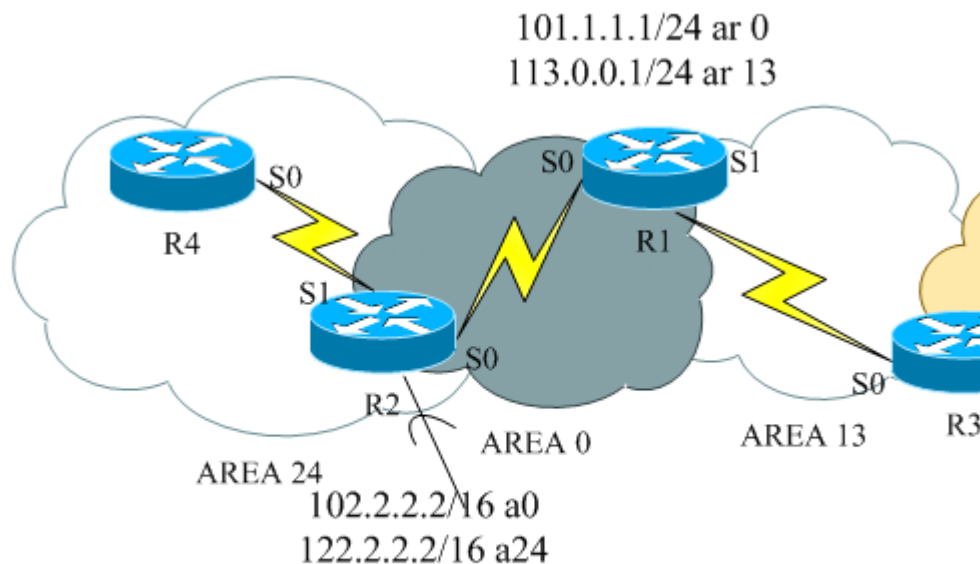
（与本区域直接相连的其他协议）；假设A13为NSSA区域，R2上建立LP口模拟域外路由运行EIGRP，那么R2就成为ASBR，其结论：R3是A13（NSSA）区域中的ASBR，只能引入与本区域（R3/5）相连的域外路由，但不能引入R2上LP路由；

第二种级别，total NSSA：只有域内路由，没有域间和其他区域的ASBR所引入的域外路由，定义后会自动向NSSA区域生成默认路由D* IA
0.0.0.0/0（LSA3类）。

Stub & Total Stub		NSSA & Total NSSA	
没有 ASBR 不能引入域外路由 ABR 会自动为 Stub 区域的路由器，产生默认路由		有 ASBR 可以引入域外路由 ABR 是否为 NSSA 区域的路由器取决于 NSSA 类型	
无 LSA4/5 有 LSA3 RFC 定义	区域: Stub 命令: 所有路由器: Area 1 stub	区域: NSSA 命令: 所有路由器: Area 1 <u>nssa</u>	NSSA 区域 要通告的默认路由由 ABR 产生
无 LSA4/5 无 LSA3 Cisco 私有	区域: Total Stub 命令: 非 ABR: Area 1 stub ABR: Area 1 stub no-sum	区域: Total NSSA 命令: 非 ABR: Area 1 <u>nssa</u> ABR: Area 1 <u>nssa</u> no-sum	NSSA 区域 由 ABR 产生
1、Stub 区域不能引入域外路由。		1、同 ASBR 引入域外路由， 2、LSA7 仅存在于 NSSA 区域 3、ABR 将 7 转换为 5，传递	

LAB1: STUB区域

~~~~~



## OSPF 110

在area24的所有路由器上（包括此区域ABR）：

```
router ospf 110
```

```
area 24 stub
```

注意：在stub区的所有路由器上都必须要做，否则会DOWN掉；

注意观察：

增加了：

在Area24中的所在路由器上，（ABR / R2除外），  
都自动获得了一条到达Stub ABR 的默认路由

```
0*IA 0.0.0.0/0 [110/65] via R2, LSA3
```

减少了：（LSA4/LSA5）

也接收不到ASBR引入的域外路由（EIGRP）

而且，

Stub区域本身也无法引入外部路由。

Stub区域应该只有一个ABR，不能有多，否则可能导致次优路径的产生。  
不能有虚链路穿越它。

LAB2: total STUB区域:

### LAB2: total STUB区域:

~~~~~

```
Stub ABR: (R2)
router ospf 110
area 24 stub no-summary
```

```
Stub区域内的所在内部路由器: (R4)
router ospf 110
area 24 stub
```

ABR所产生的默认路由, 与标准的Stub情况完全一样
0*IA 0.0.0.0/0 [110/65] via R2, LSA3

Total Stub区域, 实现了让OSPF区域路由的更进一步的简化目的。
把域间路由也简化掉了

此时R4的路由表为:

Gateway of last resort is 24.0.0.2 to network 0.0.0.0

```
24.0.0.0/24 is subnetted, 1 subnets
C      24.0.0.0 is directly connected, Serial0
122.0.0.0/32 is subnetted, 1 subnets
O      122.2.2.2 [110/65] via 24.0.0.2, 00:18:09, Serial0
0*IA 0.0.0.0/0 [110/65] via 24.0.0.2, 00:18:09, Serial0
```

此时R4的database为:

OSPF Router with ID (110.0.0.4) (Process ID 110)

Router Link States (Area 24)

Link ID	ADV Router	Age	Seq#	Checksum
Link count				
110.0.0.2	110.0.0.2	1151	0x80000015	0x0098E1 3
110.0.0.4	110.0.0.4	1151	0x80000018	0x009C69 2

Summary Net Link States (Area 24)

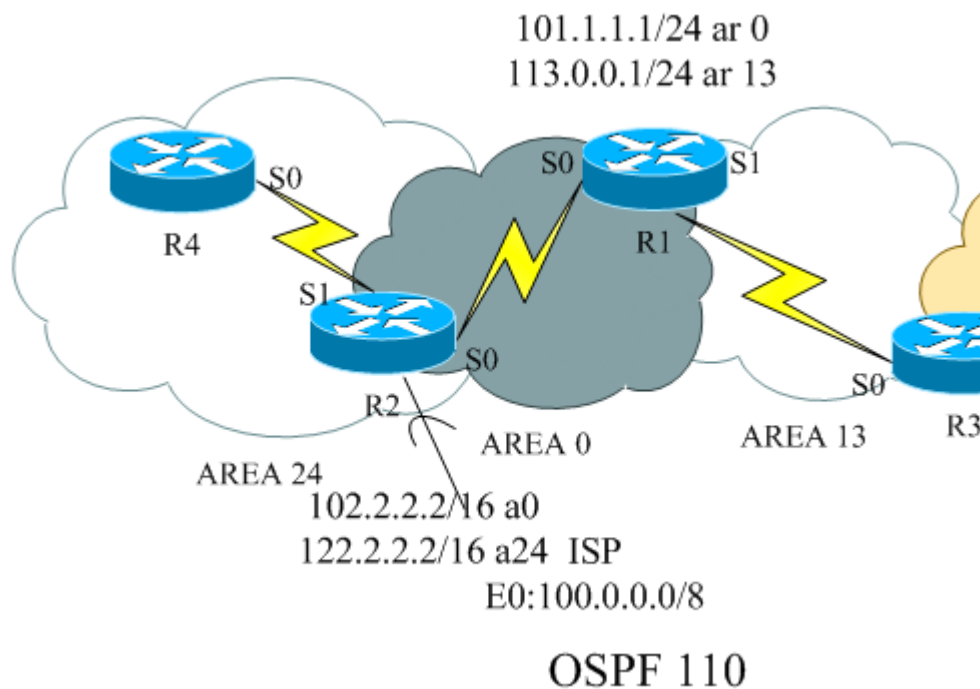
Link ID	ADV Router	Age	Seq#	Checksum
0.0.0.0	110.0.0.2	473	0x80000002	0x00C10B

LAB3: NSSA区域:

~~~~~

LAB3: NSSA区域:

~~~~~



```
interface Ethernet0
ip address 100.0.0.2 255.0.0.0

ip route 100.0.0.0 255.0.0.0 ethernet 0

router ospf 110
 redistribute static subnets
 redistribute connected subnets
```

STEP1: 在NSSA的所有路由器上，包括NSSA ABR（R1）上：

```
router ospf 110
area 13 nssa
```

观察Area13内部的路由变化
只有域内&域间，没有域外，也没有默认路由！

Step2:标准NSSA区域的ABR，是不会自动为NSSA区域内的路由器，自动生成默认路由的

NSSA的ABR上：

```
R1(config-router)#area 13 nssa default-information-originate
```

LAB4: total NSSA区域:

(NSSA=Stub区域功能+能够引入外部路由的功能)

~~~~~

Step1:

R1# (在NSSA的ABR上)

```
router ospf 110
```

```
area 13 nssa no-summary
```

NSSA的区域内部路由器，无需任何改动

因为凡是Total的Stub/NSSA区域的ABR，都会为本区域中的路由器，自动产生默认路由（LSA3）

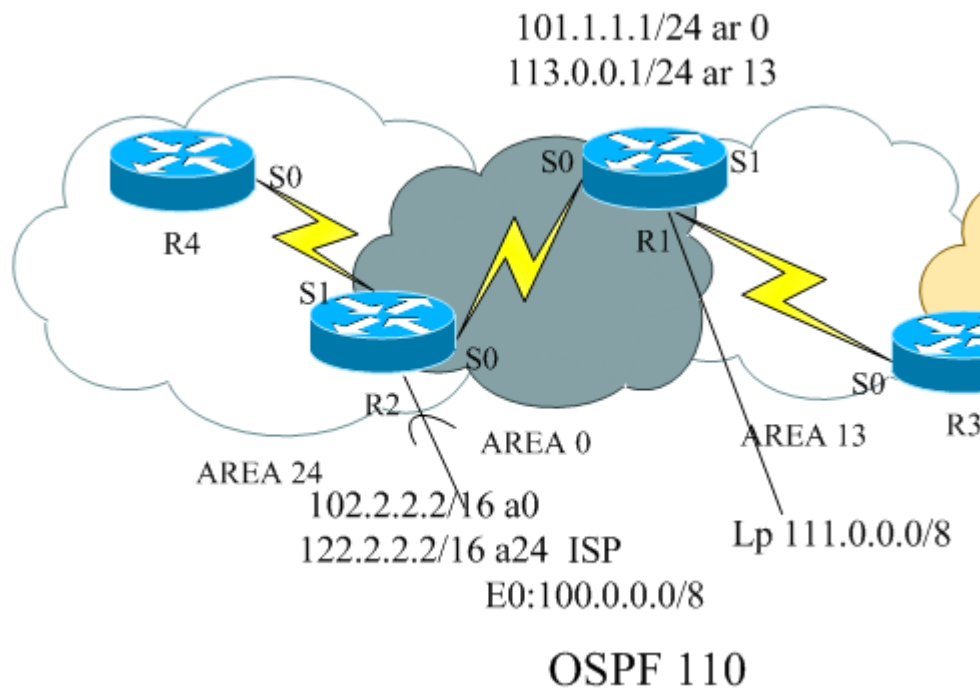
所以无需添加default-information-originate

NSSA的区域内部路由器，收到的默认路由是：

0\*IA 0.0.0.0/0 (LSA3) (NSSA的ABR发送的)

实现了NSSA区域的最简化，同时还可以引入EIGRP的外部路由

Step2:



对于NSSA区域:

如果不希望NSSA区域内的路由器, 接收到同是ABR&ASBR (R1) 引入的LSA7的外部路由

```
R1(config-router)#area 13 nssa no-redistribution no-summary
(只能在Total NSSA中做)
```

OSPF 3 种默认路由的使用场景以及配置操作

1: Stub/Total Stub/total NSSA的ABR自动产生的默认路由  
0\*IA 0.0.0.0/0 (LSA3)

2: 在标准的NSSA区域的ABR上,  
通过default-information-originate参数产生的默认路由  
0\*N2 0.0.0.0/0 (LSA7)

3: 在OSPF普通区域, 连接到ISP的边缘路由器上, 向整个OSPF域发送默认路由:

R2#

```
ip route 0.0.0.0 0.0.0.0 ethernet 0 (指向ISP)
```

```
router ospf 110
default-information originate
```

0\*E2 0.0.0.0/0 (LSA5)

## 1.4—动态路由协议OSPF⑥

### OSPF Network Type/网络类型

(Run Mode / 运行模式)

物理链路的类型:

1、Point to Point大类:

主要是Serial链路 (HDLC/PPP), P2P子接口

没有DR/BDR的选举

2、Multi-Access/多路访问大类:

2—1: 默认支持广播流量传输的MA网络: (BMA)

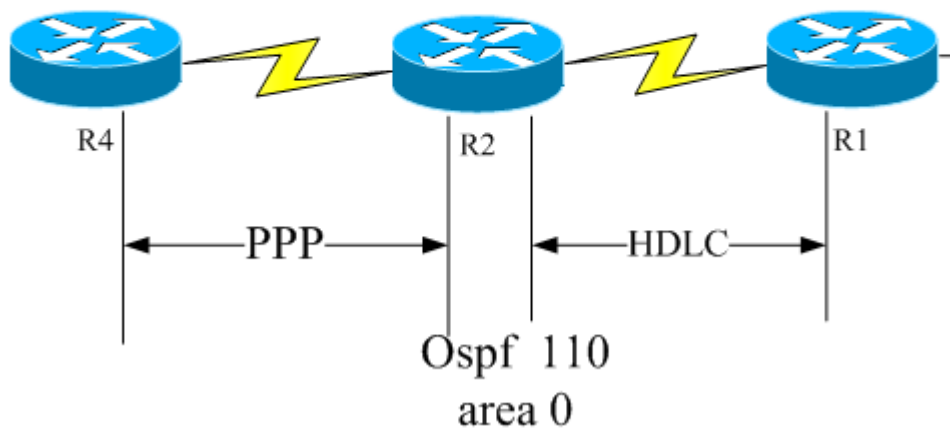
典型: Ether-net / TR/FDDI (多见于局域网)

2—2: 默认不支持广播流量传输的MA网络: (NBMA)

典型: Frame-Relay / X.25 / ATM(多见于广域网)

LAB1:OSPF的 3 种运行模式: (Loopback, P2P, Broadcast)

LP 4.4.4.4/24



**Step1:**将R2/4之间的同步串行链路，更改为PPP链路：

```
int s0  
encapsulation PPP
```

**Step2:**使用最少的命令构建OSPF网络

**Step3:**查看OSPF运行模式 1：**Loopback**

show ip ospf interface (Loopback 4) (查看OSPF接口的运行模式)

Network Type:LOOOPBACK

环回中：默认都是以 3 2 位主机路由的形式，存在于路由表中。

```
R4(config)#interface loopbakc 4
```

R4(config-if)#ip ospf network point-to-point (以接口的准确长度，显示于路由表中)

**Step4:**查看OSPF运行模式 2：**P2P**

只是是“Point to Point大类”的链路，OSPF都默认以P2P的模式来运行。

Network Type:POINT\_TO\_POINT

无DR / BDR, Hello:10, Dead:40

**Step5:**查看OSPF运行模式 3：**Broadcast**

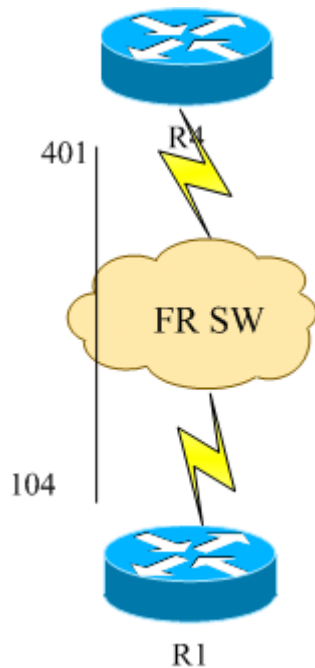
在“默认支持广播流量传输的MA网络”的(BMA)网络中，OSPF都默认以Broadcast的模式来运行。

Network Type:BROADCAST

在DR/BDR的选举，Hello:10, Dead:40

## **LAB2: 3口的FR-SW**

~~~~~



Step1: 在R2上关闭路由功能，启用FrameRelay交换功能。

```
no ip routing
frame-relay switching
```

Step2: FR-SW的接口基本配置

```
interface serial 1/0
clock rate 2000000
encapsulation frame-relay
frame-relay lmi-type cisco
frame-relay intf-type dce
```

Step3: 配置FR的路由：（PVC401/PVC104）

在FR-SW的角度观察：

从S1口，以PVC401进入交换机的数据，将以PVC104，从S0离开交换机

在PVC的入口配置

```
FR-SW2(config-if-S1)#frame-relay route 401 interface serial
0 104
```

入方向的PVC 出

口 出方向的PVC

从S0口，以PVC104进入交换机的数据，将以PVC401，从S1离开交换机

```
FR-SW2(config-if-s0)#frame-relay route 104 interface serial 1 401
```

两端口的FR-SW配置完成。

Step4:

在FR交换机上检查FR路由:

show frame-relay route (status必须是active才是正确的。)

两端口的FR SW配置完成

Step5: 开始配置FR-SW的第3个口, 构建R2-R3之间的Tunnel.

FR-SW3(config-if-e0)#ip ad 23.0.0.3 255.255.255.0

FR-SW2(config-if-e0)#ip ad 23.0.0.2 255.255.255.0

FR-SW3(config)#

interface tunnel 1

tunnel source 23.0.0.3

tunnel destination 23.0.0.2

FR-SW2(config)#

interface tunnel 1

tunnel source 23.0.0.2

tunnel destination 23.0.0.3

Step6: 在FR-SW3的S1口, 也要有接口的配置 (同Step2)

Step7: PVC405/PVC504

FR-SW2(config-if-s1)#Frame-relay route 405 interface tunnel 1 1000

FR-SW3(config-if-s1)#Frame-relay route 504 interface tunnel 1 1000

PVC105/PVC501

FR-SW3(config-if-s1)#Frame-relay route 501 interface tunnel 1 1001

FR-SW2(config-if-s0)#Frame-relay route 105 interface tunnel 1 1001

3口FR SW配置完成

FR-SW的基本检查:

1: show frame-relay route (电信运营商 / ISP的检查)

2: 在FR的用户端R1/R4/R5上, show frame-relay pvc, PVC Status必须是Active (L2的检查)

3: 在FR的用户端配置IP地址, 进行ping的L3测试。

4: show frame-relay map

(观察FR的自动反向ARP是否能够成功建立, L3的IP地址, 与L2的DLCI的映

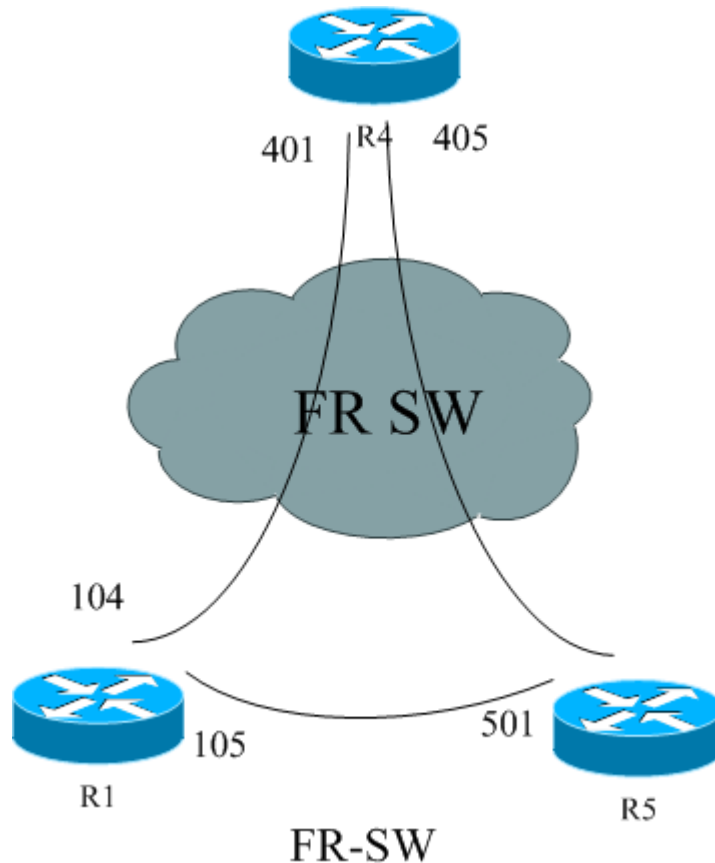
是Active (L2的检查)

3: 在FR的用户端配置IP地址, 进行ping的L3测试。

4: show frame-relay map

(观察FR的自动反向ARP是否能够成功建立, L3的IP地址, 与L2的DLCI的映射)

LAB3:在Full-Mesh的PVC环境中, 通过Broadcast模式, 构建OSPF网络



Full Mesh: 在N节点的网络中, 每个节点都有到别的所有节点的, 直接相连的连接/PVC.

在FR网络里面, 动态映射dynamic默认允许广播数据通过, 手动/静态映射(static)需要添加broadcast来允许广播数据通过。

Step0: 在三个路由器上配置IP地址

Step1: 确认L2 FR支持广播数据流通过

show frame-relay map

broadcast, (此FR PVC连接, 能够支持广播流量通过)

```
show frame-relay map  
broadcast, (此FR PVC连接, 能够支持广播流量通过)
```

在FR的自动反向ARP, 所形成的映射, 默认是可以支持广播流量通过的。

```
show ip ospf interface ethernet 0  
Network Type:BROADCAST (这只是OSPF的一种OSPF运行的模式)
```

Step2: 确定运行模式

因为FR的主接口, 点对多点子接口, 其默认的运行模式是NBMA
点对点子接口, 其默认运行模式是P2P

```
show ip ospf interface Serial0  
Network Type:NON_BROADCAST
```

- ∴OSPF不主支发送组播的Hello包
- ∴无法建立邻居

将3个点对多点子接口的运行模式改为: Broadcast

```
R1/4/5(config)#  
in s0  
ip ospf network broadcast
```

注意: Serial接口如果封装了FR, 其OSPF的运行模式, 默认是NBMA
(NON_BROADCAST), 默认情况下, 不发送OSPF的组播包的, 所以无法自动建立邻居。

如果Serial接口封装的是HDLC/PPP, 其OSPF的运行模式是: P2P。

Step3: 邻居问题:

```
show ip ospf neighbor 察看OSPF邻居  
L2允许广播流量通过  
L3运行模式为Broadcast, 所以可以建立邻居
```

Step4: DR问题

确认DR/BDR的正确选择。

因为PVC是Full mesh的, 任何一个点都是可达的, 并且, FR携带了Broadcast能力 (允许广播数据包通过PVC)

所以每一个OSPF路由器, 都能收到所有别的路由器的Hello包,
所以他们都能直接协商出正确的DR/BDR来。
DR/BDR不存在问题

Step5: 下一跳问题:

在Full-Mesh的FR中, 已经 (自动 / 手工) 做好到每一个节点的映射,

所以FR中的每一个节点都是可达的，
show frame-relay map / ping
所以不存在下一跳问题

实验结论：

OSPF的路由没有问题。

如果L2 FR不支持广播的结果：
没邻居，没一切！

1.4—OSPF运行模式run mode⑦

OSPF的运行模式

帧中继的子接口选用原则：

1、在一个封装FR的物理接口上，可以同时承载多条PVC。

为了网络的可扩展性，建议不论在考试环境还是在工程环境中，都应该优先考虑使用子接口

2、应该创建几个子接口：在一个物理接口中，对应着几个网络，就应该建几个子接口。

一个IP子网对应着一个子接口。

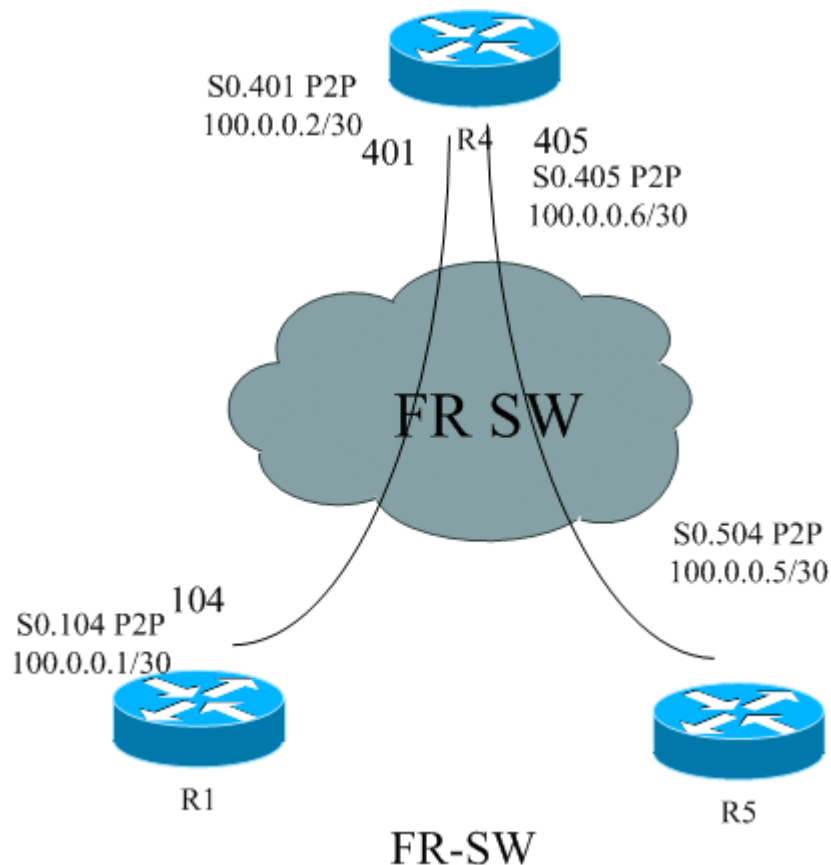
3、（每个子接口分别应该是什么类型）（点对点/多点）

在一个子接口中：如果对应着一个点，那么子接口类型应该是P2P。默认情况下，其OSPF Running Mode是Point-to-Point。OSPF对待这种子接口就像对待点对点串行链路一样。

如果对应着多个点，那么子接口类型应该是Multi Point。默认情况下，其OSPF的Running Mode是NBMA。OSPF对待这种子接口就像对待FR的主接口/物理接口一样。

↵		不能自动建成邻居↵	能够自动建成邻
OSPF RUN MODE ↵		L2 不需要支持广播流量通过↵	L2 的 FR 能支持
有 DR/BDR 的选举↵	手工解决 下一跳↵	NBMA ↵ 手工指定 DR ↵	BMA ↵ 自动 DR ↵
无 DR/BDR 的选举↵	自动下一 跳↵	P2MP(NON Broadcast) ↵	P2MP(Broadcas

LAB1:在Hub&Spoke网络中，通过点对点接口，使用Point to Point模式，
构建OSPF网络：
~~~~~  
~~~~~



Step1:配置FR主接口的基本配置:

```
R1/4/5#  
encapsulation frame-relay (封装FR)  
no frame-relay inverse-arp (关闭FR的自动反向ARP)  
no sh
```

Step2:按图配置点对点接口:

(特别注意: 每个子接口对应一条PVC, 同时, 每条PVC对应一个 / 30 IP 子网)

```
pvc 401/104  
R1#  
interface serial 0.104  
ip add 100.0.0.1 255.255.255.252  
frame-relay interface-dlci 104
```

```
R4#  
interface serial 0.401  
ip add 100.0.0.2 255.255.255.252  
frame-relay interface-dlci 401
```

测试：（L2的FR子接口）

```
show frame-relay map
```

R4 Ping所有4个子接口，都OK，测试成功

Step3:运行OSPF协议：

Step4:

问题一：OSPF邻居问题

4-1: 察看子接口的L3的OSPF运行模式：

```
show ip ospf interface serial 0.104
      Network Type POINT_TO_POINT
```

因为Serial 0.104这个FR子接口是点对点子接口，
所以OSPF默认将这种子接口的运行模式置为POINT_TO_POINT，
说明此子接口已经向外发送组播包（224.0.0.5）

4-2: 察看子接口的L2的FR特性：

```
show frame-relay map
serial 0.104:point to point dlci,dlci 101
      broadcast
```

因为FR的点对点子接口，默认就携带了让广播 / 组播包通过的能力，
所以L3的OSPF所发出的Hello包，可以成功到达PVC的对端
所以能够成功建立邻居。

Step5:

问题2：DR问题

在OSPF的P2P运行模式中，根本没有DR/BDR，DR-Other。

Step6:

问题3：下一跳问题：

```
show ip route ospf
```

```
0 100.0.0.4 [110/128] via 100.0.0.2
0 4.4.4.4 [110/65] via 100.0.0.2
0 5.5.5.5 [110/129] via 100.0.0.2
```

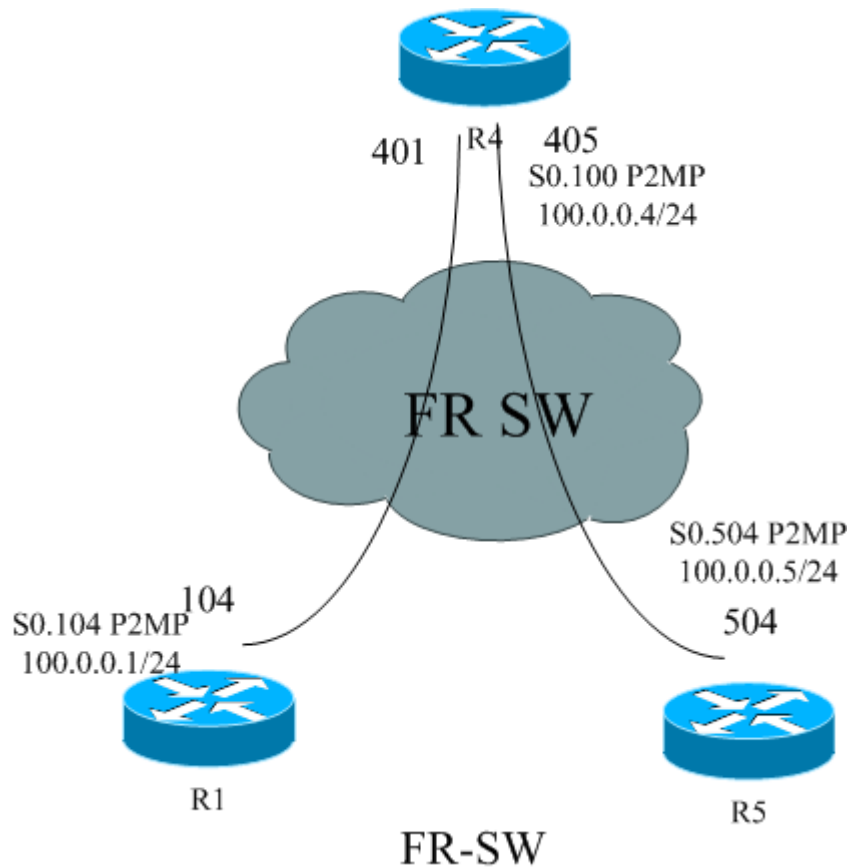
上一个LAB，中心点以外的节点，是分别在**不同的IP网段**。

以下的三个LAB，中心点以外的节点，都在**相同的IP网段**。

LAB2:在Hub & Spoke的PVC环境中，FR无法支持广播流量，使用NBMA模式，
构建OSPF网络。

~~~~~

~~~~~



R4#

```
interface serial 0.100 multipoint
ip address 100.0.0.4 255.255.255.0
frame-relay map ip 100.0.0.1 401 (后面无Broadcast关键字，即不允许广播流量通过)
frame-relay map ip 100.0.0.5 405
```

R1/R5也做无Broadcast的PVC映射。

Step2:确定OSPF的运行模式:

R1/R4/R5#

```
show ip ospf interface serial 0.100
Network Type: NON_BROADCAST
```

R1/R4/R5#

```
ip ospf network non-broadcast
```

Step3:OSPF邻居问题:

无法自动建立邻居。

解决方案:

通过单播更新，手工解决OSPF邻居问题。

在中心点路由器（HUB）上配置：

```
router ospf 110
neighbor 100.0.0.1
neighbor 100.0.0.5
```

关于单播更新：

RIP： 要PASS接口， neighbor对方，双方路由器都需要配置

EIGRP： 不能PASS接口， neighbor对方，双方路由器都需要配置。

OSPF： 不能PASS接口， neighbor对方，单个路由器需要配置。

Step4:OSPF DR 问题

单播更新可以建立邻居，但DR/BDR信息混乱，OSPF路由不正常

解决方案：

在Hub & Spoke网络中：

始终让中心点HUB，保持是DR，其余所有路由器都是DR-Other（无BDR）

通过OSPF的接口优先级，控制DR选举：

HUB：

```
in s0.100
ip ospf priority 10
```

SPOKE：

```
in s0.100
ip ospf priority 0
```

DR正常了，全网路由器都有正常OSPF路由了，

在中心点访问分支点，都是没有问题。

分支点，可以访问中心点，但不能访问别的分支点，

原因是路由的下一跳不可达。

Step4:下一跳问题：（不是L3的路由问题，而是L2的数据包封装问题）

解决方案：（手工指定路由下一跳的映射）

R1#frame-relay map ip 100.0.0.5 104

R5#frame-relay map ip 100.0.0.1 504

实验结论：

OSPF的路由没有问题

LAB4:在Hub&spoke的PVC环境中，FR不可以支持广播，使用P2MP Non-Broadcast模式，构建OSPF网络

~~~~~  
~~~~~  
Step1:L2 FR不可以支持广播

Step2:选定OSPF运行模式为：P2MP Non-Broadcast
R1/4/5#
in serial 0.100
ip ospf network point-to-multipoint non-broadcast

Step3:邻居问题：（单播解决）

Step4:DR问题：
因为在OSPF P2MP运行模式中，根本没有DR/BDR的概念。
所以无DR问题，不需要指定优先级。

Step5:下一跳问题：（自动下一跳）
任何一个分支点收到的所在路由的下一跳都是中心点
所以没有下一跳问题
所以在分支点之间，无需进行相互映射。

P2MP运行模式特有的主机路由：
所在P2MP路由器，都获得了该MA网络中的所有节点32位主机路由。

实验结论：
OSPF的路由没有问题

LAB5:在Hub&spoke的PVC环境中，FR可以支持广播，使用P2MP Broadcast模式，构建OSPF网络

~~~~~  
~~~~~  
Step1:L2 FR可以支持广播：
R1:frmae-relay map ip 100.0.0.4 104 broadcast

Step2:选定OSPF运行模式为P2MP（Broadcast）
R1/4/5#
in s0.100 m
ip ospf network point-to-multipoint

Step3:邻居问题
因为P2MP Bro中，OSPF会主支发送组播HELLO，而FR又允许组播通过
所以自动建成邻居，不存在邻居问题，无需单播更新。

Step4:DR问题：自动DR

Step5:下一跳问题：自动下一跳

实验结论：

OSPF的路由没有问题。

1.4—动态路由协议OSPF⑧

OSPF认证（保证寻路协议级别的网络安全）

~~~~~

按照参与认证的成员，进行分类：

1：链路认证（参与认证的成员：一条链路两端的路由器）  
（只能确保两路由设备之间的链路是安全的）  
链路：两个路由器之间的物理链路。

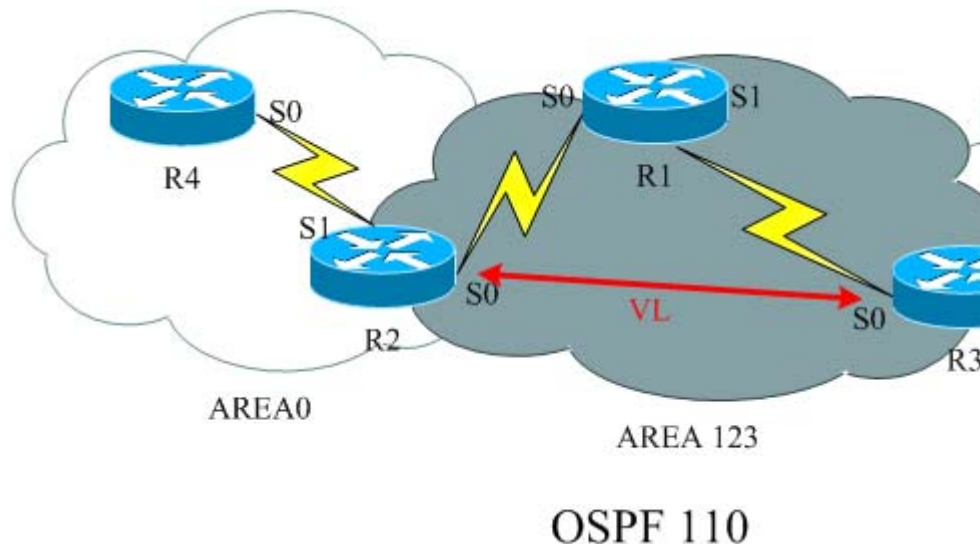
2：区域认证（参与认证的成员：这个区域中的所有路由器）  
（能够确保整个OSPF区域的所有路由器的安全性）

3：虚链路认证（Virtual Link）（参与认证的成员：OSPF VL两端ABR路由器）  
（能保证跨越多个路由器的，Virtual Link的连接安全性）

按照认证的加密方式，进行分类：

1. 明文认证
2. 密文认证

OSPF 认证的基本步骤：



**Step1:** 在接口配置密码

**Step2:** 在OSPF进程（区域认证）启用认证。

或：在接口（链路认证）启用认证。

#### 1-1: 链路的明文认证

```
(int S 1)ip ospf authentication-key cisco1(配置认证密码)
```

```
(int S 1)ip ospf authentication (启动明文认证)
```

#### 1-2: 链路的密文认证

```
(int S 1)ip ospf message-digest-key 1 md5 cisco1(配置认证密码)
```

```
(int S 1)ip ospf authentication message-digest (启动链路的密文认证)
```

**service password-encryption** (在show running-config中以密文的方式显示密码)

#### 2-1: 区域的明文认证（Area0举例）

对于区域认证：

凡是Area0的路由器，都必需参与认证，如果不参与认证/认证不成功，将无法交换OSPF路由信息。

```
(int s 0)ip ospf authentication-key cisco2
```

```
(router ospf 110)area 0 authentication
```

如果两台路由都没有在接口上打上密码,而且启用了区域或链路的认证,则也可以正常地建立起邻居。

也可以正常地建立起邻居。

存在问题：

Area123中的OSPF VL，在R3无需获知Area0的区域认证密码的情况下，仅配置“area 0 authentication”就可以成功连接进入Area0

Area0中的路由器，没有Area35的路由

原因是：R1-R3之间的VL，出现中断，

R3相当于是Area0的路由器，但是没有参加Area0的区域认证。

解决方法：在建立VL的R3上，只要宣告：

R3#

router ospf 110

area 0 authentication

即使R3没有配置密码，也可以成功与BackBone进行认证。

又带来了VL的安全问题。

## 2-2：区域的密文认证

```
(int s 0)ip ospf message-digest-key 3 md5 cisco2
```

```
(router ospf 110)area 0 authentication message-digest
```

(明文认证对应明文的密码!!!密文认证对应密文的密码!!!)

3:key ID 两端的key ID不一样,即使密码一样也会认证失败!!所以, key ID和密码要两端都一样!!

存在问题：

同：区域的明文认证

解决VL安全性的办法：单独进行基于VL的认证。

在ABR（VL的两端）R2/R3上进行

## 3-1:VL的明文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication
```

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication-key cisco3
```

对方的ROUTER-ID

## 3-2:VL的密文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication message-digest
```

authentication message-digest

```
(router ospf 110)area 123 virtual-link 100.0.0.3 message-digest-  
key 1 md5 cisco3
```

**查看验证类型：**

R1#

```
00:18:30:      OSPF:  Rcv pkt from 192.168.10.65, FastEthernet0/1  
: Mismatch Authentication type. Input packet specified type 1, we  
use type 0
```

type0:不验证，默认状态

type1:明文验证

type2:密文验证

IS-IS

**特征：**

- 1: 协议操作起来，比OSPF要简单
- 2: 扩展性比OSPF要好，易于扩展
- 3: 对IPv6有很好的支持
- 4: 能够同时支持IP网络，和CLNS网络（OSI网络）（集成的IS-IS）
- 5: 十分稳定，使用与OSPF相同的算法：SPF算法。  
（稳定 / 收敛较快 / 对CPU / 内存 / 链路利用率，都有很好的效率）
- 6: IS-IS是链路状态路由协议（LS），使用LSP来描述路由信息。  
（相当于OSPF的LSA）

7:支持两种级别的路由：

Level0: 路由器（IS）与终端系统（ES）之间；

Level1:The route of local area (system ID): 域内路由；

Level2:The route between areas (Area ID): 域间路由；

Level3: 两个IS-IS间。

8: IS-IS的3种路由器

8-1: L1 Router: 路由器只有本区域的路由, 相当于OSPF的非骨干路由器或内部路由器:

8-2: L2 Router: 拥有区域间路由, 相当于OSPF骨干路由器:

8-3: L1&L2 Router: 既有域间路由也有域内路由, 相当与OSPF的ABR骨干路由器:

9: IS-IS目前存在的一些缺陷:

9-1: ISIS的度量值取值范围较小:

使用6Bits描述一个接口优劣, (0-63), Cisco默认10, 而不分接口实际带宽使用10Bits描述一条路由的优劣(0-1023);

9-2: 目前, ISIS只支持以下四种变量, 用于描述路由的优劣:

Default, Delay, expense, error; 但目前Cisco只支持Default这一种!

10: OSPF VS ISIS

相同点:

都使用相同的SPF算法:

导致路由的计算, 更新, 传播, 决策, 收敛都非常类似。

不同点:

OSPF:

有一个BackBone区域, 所有的普通区域都必需连接到Backbone区,

每一条链路都完整的属于一个区域,

对应的连接每个区域的ABR, 是分属于不同的区域;

OSPF通过LSA进行路由的描述和传播, 支持多达几十种LSA

OSPF可以通过带宽的反比来描述接口的优劣。

有较多的厂商对OSPF提供很好、全面的支持;

IS-IS:

凡是L2链路的路由和链路都是ISIS的骨干区,

每个路由都属于同一个区域,

ISIS通过链路来区分区域而不是通过路由区分区域;

ISIS通过L1/L2的LSP来描述和传播路由,

ISIS不论是哪种接口, 其默认Metric是10,

而能够全面支持ISIS的厂商不多。

11: NSAP (Network Service Access Point):

是OSI/CLNS网络中的地址 (CLNP协议), 相当与IP地址。

Area-ID: 1-13字节可变长;

Area-ID: 1-13字节可变长;

System-ID: 6字节, 定长;

NSEL: 此网络节点, 所能提供的服务类型的标识位;

如果NSEL等于0, 表示此节点是一个路由器; 此地址可简写为NET地址。

可以称为网络实体标识地址NET (Network Entity Title) ;

## 12: 在CLNS网络中:

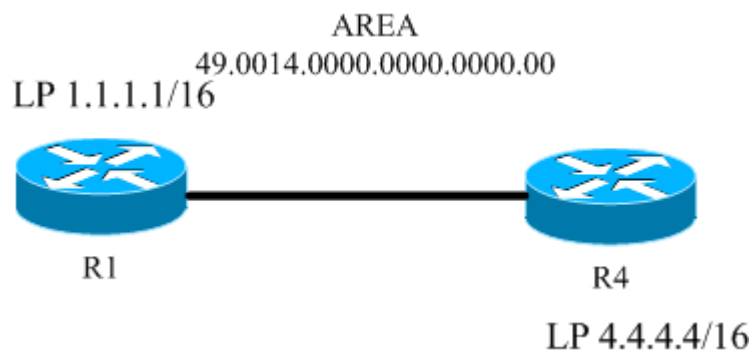
每个网络节点 / 设备, 都只有一个NSAP地址,

而不像IP网络中, 每个接口都有一个IP地址

## 13: 在CLNS网络中:

区域号如果为49, 表示私有网络。

LAB1. 使用IS-IS协议构建集成的IP网络:



Step1. 规划IS-IS区域, 规划每个路由器的NET (network entity title) / 网络实体标识。

NET: -----|-----|----- (16进制)  
Area-ID, System-ID, NSEL  
1-13byte, 6byte, 1byte (路由器固定是00)

```
router isis
net 49.0014.0000.0000.0002.00
    area id |system id |NSEL
```

Step2. 配置IP地址:

Step3. 在接口中激活IS-IS的运行, 为IP 网络进行路由 (让IS-IS为这个接口所在的网段进行路由)

```
int s0
```

```
ip router isis
```

Step4. 察看IS-IS的邻居建立:

```
show clns
```

```
R4#show clns
```

Global CLNS Information:

2 Interfaces Enabled for CLNS

NET: 49.0014.0000.0000.0004.00

Configuration Timer: 60, Default Holding Timer: 300, Packet  
Lifetime 64

ERPDU's requested on locally generated packets

Running IS-IS in IP-only mode (CLNS forwarding not allowed)

```
show clns protocol
```

```
show clns neighbors
```

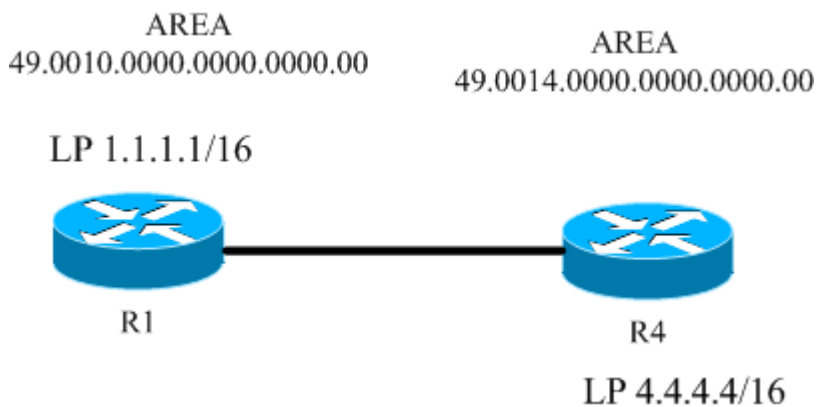
```
show clns interface
```

```
show clns route
```

```
show clns database
```

```
show clns topology
```

```
show ip route
```



Step5. 控制接口的IS-IS的Level类型（默认是L1L2）

```
int s0
```

```
isis circuit-type level-2/Level-1
```

## 1.5—动态路由协议ISIS

IS-IS

特征:

- 1: 协议操作起来, 比OSPF要简单
- 2: 扩展性比OSPF要好, 易于扩展
- 3: 对IPv6有很好的支持

- 4: 能够同时支持IP网络, 和CLNS网络 (OSI网络) (集成的IS-IS)
- 5: 十分稳定, 使用与OSPF相同的算法: SPF算法。  
(稳定 / 收敛较快 / 对CPU / 内存 / 链路利用率, 都有很好的效率)
- 6: IS-IS是链路状态路由协议 (LS), 使用LSP来描述路由信息。  
(相当于OSPF的LSA)

7: 支持两种级别的路由:

Level0: 路由器 (IS) 与终端系统 (ES) 之间;

Level1: The route of local area (system ID): 域内路由;

Level2: The route between areas (Area ID): 域间路由;

Level3: 两个IS-IS间。

8: IS-IS的3种路由器

8-1: L1 Router: 路由器只有本区域的路由, 相当于OSPF的非骨干路由器或内部路由器:

8-2: L2 Router: 拥有区域间路由, 相当于OSPF骨干路由器:

8-3: L1&L2 Router: 既有域间路由也有域内路由, 相当与OSPF的ABR骨干路由器:

9: IS-IS目前存在的一些缺陷:

9-1: ISIS的度量值取值范围较小:

使用6Bits描述一个接口优劣, (0-63), Cisco默认10, 而不分接口实际带宽使用10Bits描述一条路由的优劣 (0-1023);

9-2: 目前, ISIS只支持以下四种变量, 用于描述路由的优劣:

Default, Delay, expense, error; 但目前Cisco只支持Default这一种!

10: OSPF VS ISIS

相同点:

都使用相同的SPF算法:

导致路由的计算, 更新, 传播, 决策, 收敛都非常类似。

不同点:

OSPF:

有一个BackBone区域, 所有的普通区域都必需连接到Backbone区,

每一条链路都完整的属于一个区域,

对应的连接每个区域的ABR, 是分属于不同的区域;

OSPF通过LSA进行路由的描述和传播, 支持多达几十种LSA

OSPF可以通过带宽的反比来描述接口的优劣。

有较多的厂商对OSPF提供很好、全面的支持;

IS-IS:

凡是L2链路的路由和链路都是ISIS的骨干区,

每个路由都属于同一个区域,

每个路由都属于同一个区域，  
ISIS通过链路来区分区域而不是通过路由区分区域；  
ISIS通过L1/L2的LSP来描述和传播路由，  
ISIS不论是哪种接口，其默认Metric是10，  
而能够全面支持ISIS的厂商不多。

**11: NSAP (Network Service Access Point) :**  
是OSI/CLNS网络中的地址（CLNP协议），相当与IP地址。

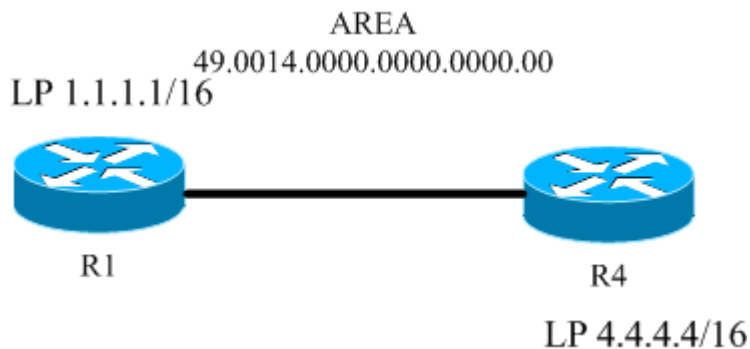
Area-ID: 1-13字节可变长；  
System-ID: 6字节，定长；  
NSEL: 此网络节点，所能提供的服务类型的标识位；

如果NSEL等于0，表示此节点是一个路由器；此地址可简写为NET地址。  
可以称为网络实体标识地址NET (Network Entity Title) ；

**12: 在CLNS网络中:**  
每个网络节点 / 设备，都只有一个NSAP地址，  
而不像IP网络中，每个接口都有一个IP地址

**13: 在CLNS网络中:**  
区域号如果为49, 表示私有网络。

LAB1. 使用IS-IS协议构建集成的IP网络:



Step1. 规划IS-IS区域，规划每个路由器的NET (network entity title) / 网络实体标识。

NET: -----|-----|----- (16进制)  
Area-ID, System-ID, NSEL

1-13byte, 6byte, 1byte(路由器固定是00)

```
router isis
net 49.0014.0000.0000.0002.00
  area id |system id |NSEL
```

Step2. 配置IP地址:

Step3. 在接口中激活IS-IS的运行, 为IP 网络进行路由 (让IS-IS为这个接口所在的网段进行路由)

```
int s0
ip router isis
```

Step4. 察看IS-IS的邻居建立:

**show clns**

R4#show clns

Global CLNS Information:

2 Interfaces Enabled for CLNS

NET: 49.0014.0000.0000.0004.00

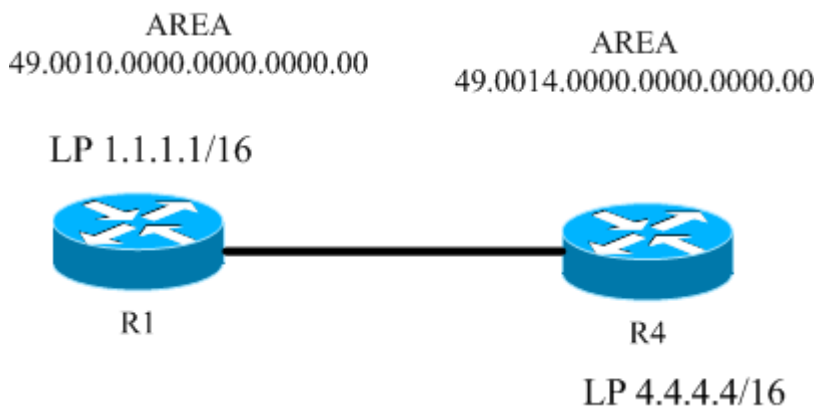
Configuration Timer: 60, Default Holding Timer: 300, Packet

Lifetime 64

ERPDU's requested on locally generated packets

Running IS-IS in IP-only mode (CLNS forwarding not allowed)

```
show clns protocol
show clns neighbors
show clns interface
show clns route
show clns database
show clns topology
show ip route
```



Step5. 控制接口的IS-IS的Level类型 (默认是L1L2)

```
int s0
```

isis circuit-type level-2/Level-1

## 1.6一路由的控制①

## 1.6一路由的控制①

注意：充分分布的本质：只会重分布本路由器相关路由协议的路由条目。

### Redistributing/重分布、重分发

将A的路由，重分布到B。

使A路由协议中的路由，以外部路由的形式，进入B路由协议。

router b

redistribute a

将EIGRP重分布 到OSPF

router ospf 110

redistribute digrp 90 subnets

**default-metric/默认的度量值（种子度）：**

A协议中的路由，进入B路由协议后，在B协议中表现出来的，默认的Metric值（度量值）

default-metric在不同的路由协议中的默认取值：

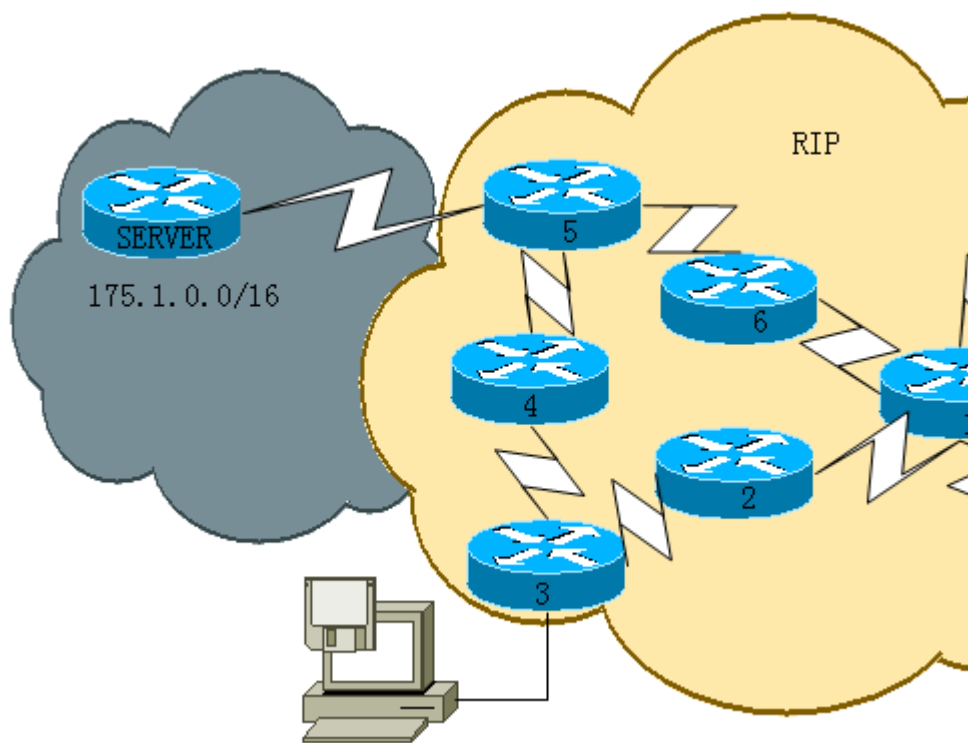
如果有外部路由（不管此路由源自哪种协议），进入以下的路由协议，默认在这些协议中表现的 Metric是：

1: DV协议（RIP/IGRP/EIGRP），其默认Metric是无穷大(不可达不可用)  
所以例如rip的话要指定metric为小于15的数。

2: OSPF : 20

3: IS-IS : 0

4: BGP: 默认是IGP的原始的METRIC。通常会人为的修改。



在两种路由协议进行重分布时：

要特别注意路由的控制/过滤，避免路由环路的出现。

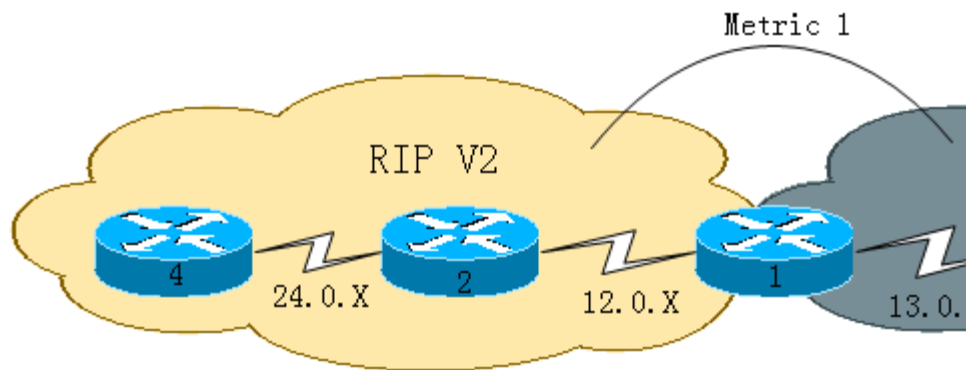
如果控制不慎，可能出现路由环路/错误的路由信息/不一致的收敛时间

一般不建议使用双向出口的重分布

三种建议解决办法：

- 1) 在一个方向上做重分布，另外一个方向做静态、默认路由。
- 2) 在一个方向上做重分布，反方向做带过滤的重分布。
- 3) 在一个方向上做重分布，反方向做带修改AD的重分布。

**LAB1：将外部路由重分布到RIP中：**

**Step1: 按图构建拓扑并运行IGP:**

R1/R2/R4运行RIP v2 (no auto summ

R1/R3/R5运行EIGRP 90(no auto summ

没有重分布前没有对方路由，两区域ROUTER不能互

**Step2: 将EIGRP的路由重分布进RIP:**

在两个协议的边缘路由器上，做重分布:

```
r1#router rip
```

```
redistribute eigrp 90 ;
```

**STEP3: 在重分布中，要携带Metric参数:**

然后观察（在R2上sh ip ro）发现看不到重分布后的路由，

因为DV协议若不指定Metric则默认为无穷大导致网络不可达，路由没能成功进入RIP;

所以DV协议的重分布要在边缘路由器上定义Metric参数:

```
r1#router rip
```

```
redistribute eigrp 90 metric 1 ;
```

然后sh ip rip da 和debug ip r,

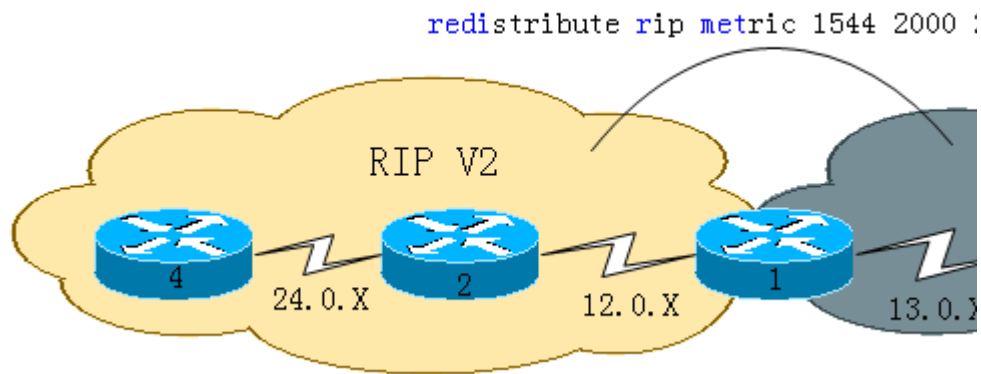
```
r1/r2#show ip rou rip
```

```
R 35.0.0.0 [120/1]
```

```
r4#
```

```
R 35.0.0.0 [120/2]
```

**LAB2: 将外部路由重分布到EIGRP中:**



**Step1: 按图构建拓扑并运行IGP:**

上一个实验的拓扑;

**Step2: 将RIP的路由重分布进EIGRP:**

在两个协议的边缘路由器上, 做重分布:

```
r1#router eigrp 90
```

```
redistribute rip ;
```

**STEP3: 在重分布中, 要携带Metric参数:**

然后观察 (在R3上sh ip ro) 发现看不到重分布后的路由, 原因同上;

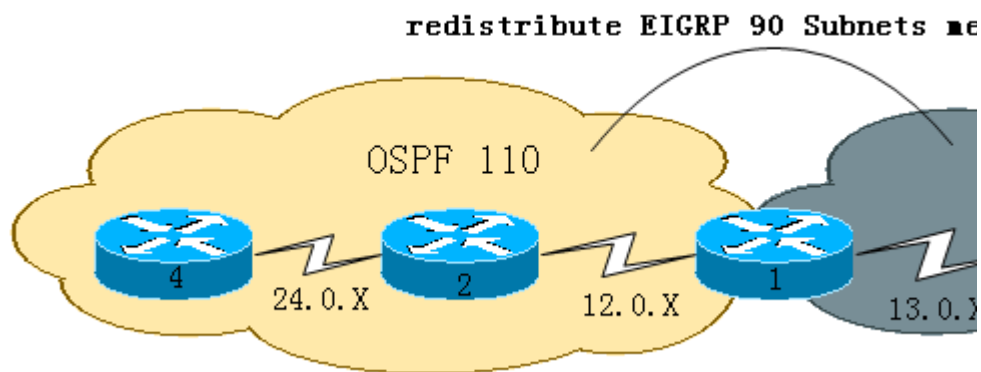
DV协议的重分布要在两个协议的边缘路由器上定义Metric参数:

```
r1#router eigrp 90
```

```
redistribute rip metric 1544 2000 255 1 1500 ;
```

DUAL算法: 2681856 。

**LAB3: 将外部路由重分布到OSPF中:**



**Step1: 按图构建拓扑并运行IGP:**

R1/R2/R4运行OSPF,

R1/R3/R5运行EIGRP90;

**Step2: 将EIGRP的路由重分布进OSPF:**

在两个协议的边缘路由器上, 做重分布:

```
R1(config)#router ospf 110
```

```
redistribute EIGRp 90
```

如果没有使用"Subnet"参数,将只有有类的路由重分布到OSPF中:

O E2 5.0.0.0/8

注意: 只能让主类路由重分布到OSPF, 子网的路由将不能被重分布!

**STEP3: 如果能让全部路由都能够重分布:**

然后观察(在R4上sh ip ro)发现看不到重分布后的路由, 原因同上;

DV协议的重分布要在两个协议的边缘路由器上定义Metric参数:

使用"Subnet"参数:

R1(config-router)#redistribute EIGRp 90 subnets

不论是子网路由,还是主类的路由,都将被重分布到OSPF:

O E2 13.0.0.0/24

O E2 35.0.0.0/24

O E2 5.0.0.0/8

**STEP4: 进入OSPF后的外部路由的类型: E1/E2:**

默认是E2型, 其OSPF的cost/Metric值, 不会随着路径远近的变化而变化。

(无法反映路径的远近)

R2# O E2 14.0.0.0/8 [110/20]

R4# O E2 14.0.0.0/8 [110/20]

2型: 不会随路由远近而变化(默认)

1型: 会随路由远近而叠加变化

R1(config-router)#redistribute EIGRp 90 subnets metric-type 1

OSPF E1, 会随着路径的远近, 其Cost会累加:

R2# O E1 14.0.0.0/8 [110/84]

R4# O E1 14.0.0.0/8 [110/148]

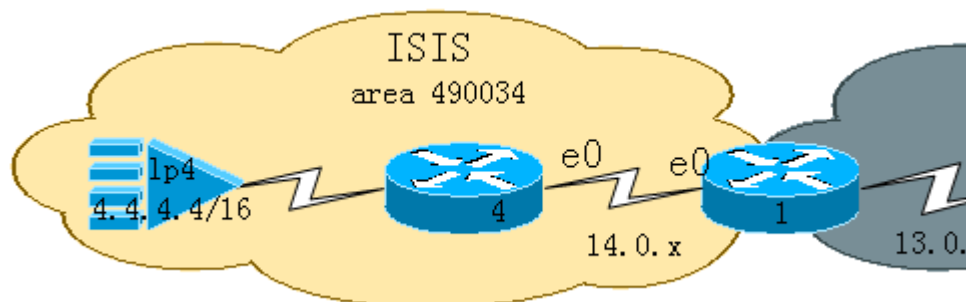
**Step5: 更改默认进入OSPF的Metric值:**

Metric值: 默认20

R1(config-router)#

redistribute EIGRp 90 subnets metric-type 1 metric 100

**LAB4: 将外部路由重分布到ISIS中:**



**step0: 按图构建拓扑并运行IGP:**

**step0:按图构建拓朴并运行IGP:**

R1/R3/R5运行EIGRP,

R1/R4运行ISIS (要在接口ip ro is激活); R4上增加环回路口;

**Step1: 运行ISIS, 定义NET地址,在接口中激活ISIS, 为IP进行路由**

```
r4/1#router isis
```

```
net 49.0034.0000.0000.0004/1.00
```

```
r4/1#int e0
```

```
ip router isis
```

**Step2: 将EIGRP的路由重分布进ISIS:**

在两个协议的边缘路由器上, 做重分布

```
R1 (config) #router isis
```

```
redistribute eigrp 90
```

**Step3: 将ISIS的路由重分布回EIGRP:**

```
R1 (config) #router eigrp
```

```
redistribute isis metric 1544 2000 255 1 1500
```

然后观察 (在R5上sh ip ro) 发现没有到14.0.0.0/24网段的路由, 因为ISIS不宣告直链路由;

解决方法: 重分布直链路由:

**LAB5: 重分布直链路由**

```
r1#router a(ospf/rip/eigrp)
```

```
redistribute connectd (metric *)DV协议就要加
```

```
redistribute connectd metric 1 - rip
```

```
redistribute connectd metric 1544 2000 255 1 1500 - EIGRP
```

```
redistribute connectd subnets - ospf
```

**LAB6: 重分布静态/默认路由**

不做把ISIS重分布到EIGRP, 而做静态或默认路由

**step1:**

```
r1#ip route 0.0.0.0 0.0.0.0 14.0.0.4
```

or

```
ip route 35.0.0.5 255.255.255.0 13.0.0.3
```

**step2:**

```
router eigrp 90
```

```
redistribute static metric 1500 2000 255 1 1500
```

**1.6一路由的控制②**

## 1.6一路由的控制②

路由过滤/控制的常用工具:

### 1、访问控制列表ACL (Access-List):

其开发本意不是用于路由的控制，而是用于数据包的过滤；  
在路由过滤中此方法比较繁琐落后，实际应用中并不推荐；

### 2、前缀列表 (Prefix-List):

本身就是为控制路由而设计的专业工具，先进的，推荐的；

### 3、Route-Map:

高级综合操作工具，可以实现多种功能，可以应用于路由控制。

路由过滤的基本方法:

分布列表 (Distribute-List):

- 1.通过Distribute-List可以调用Access-List实现路由过滤
- 2.通过Distribute-List可以调用Prefix-List实现路由过滤

按路由过滤级别分类

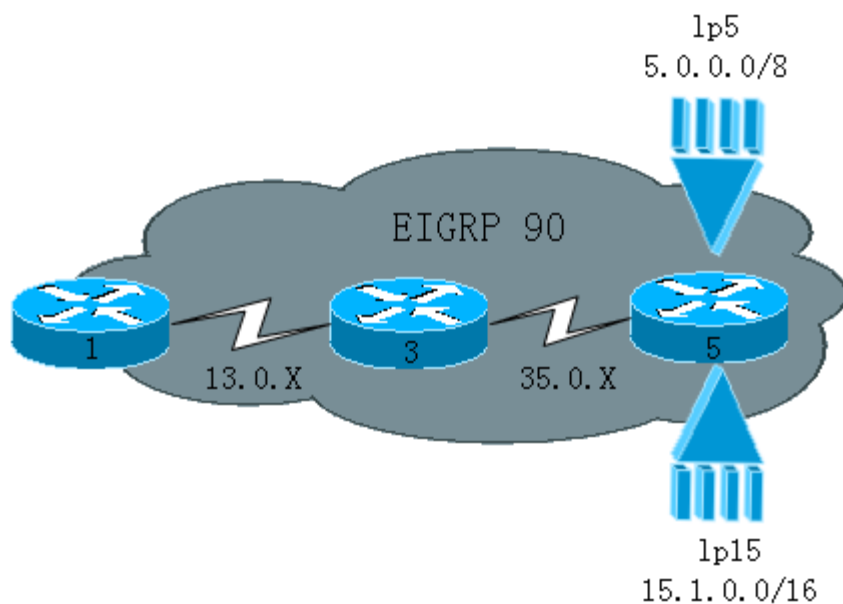
#### 1.接口级别过滤

实现基于接口级别的过滤（只可在DV协议中实现，LS协议因为泛洪无法实现）；

#### 2.协议级别的控制

是在路由协议的重分布中实现的重分布时的路由过滤。

## LAB7: 对用户数据包的过滤



在R3上，对从本机S1进入路由器的  
访问目标网络是13.0.0.0/24的数据包，进行过滤/DENY

**step1.通过ACL列表，定义需要进行过滤的数据包**

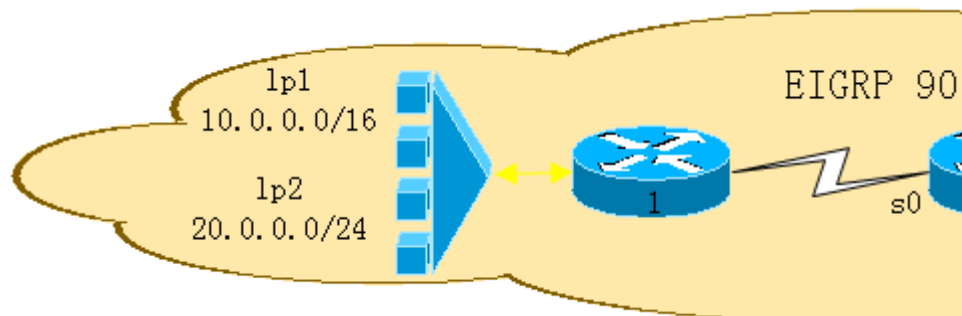
```
r3#Access-list 3 deny 15.1.0.0 0.0.255.255
Access-list 3 permit 5.0.0.0 0.255.255.255(可省略)
Access-list 3 permit any
```

**step2.**

```
r3#int s1
ip access-group 3 in
```

## LAB8: 在DV协议 (RIP/EIGRP) 中，实现基于接口的路由过滤:

8-1: Distribute-list调用ACL，比较落后:



**step1:通过ACL，定义所需要过滤的路由:**

```
r3#access-list 11 deny 10.0.0.0(这里的deny跟数据包过滤是一样的意
```

意思,都是拒绝的意思)

access-list 11 permit any(这里的permit跟数据包过滤是一样的意思,都是允许的意思)

**step2:在DV协议路由进程中,使用distribute-list 调用ACL 11**

R3(config)#router eigrp 90

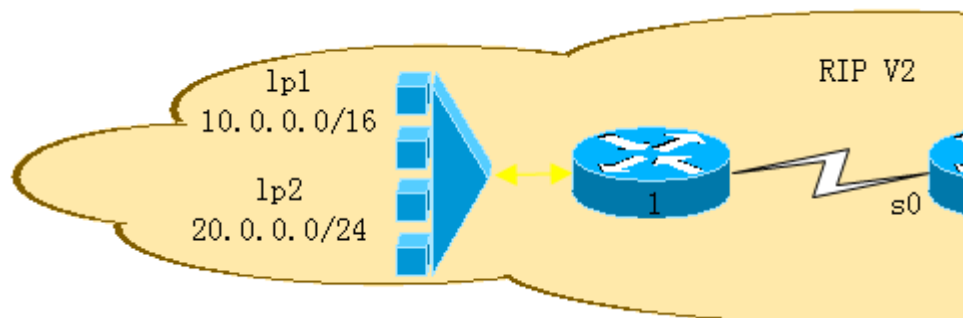
R3(config-router)#distribute-list 11 out serial 1

IP access list number in Filter incoming routing updates

out Filter outgoing routing updates

clear ip route \*

## 8-2: Distribute-list调用前缀列表, 比较先进



(用RIP举例, 也可用于EIGRP) ACL的缺陷, 无法定义路由的长度

**Step1: 使用prefix-list定义所需要过滤的路由: (SEQ5为首, 预留空间方便编辑)**

R3(config)#ip prefix-list R-5 seq 5 deny 20.0.0.0/24

R3(config)#ip prefix-list R-5 seq 10 permit 0.0.0.0/0 le 32 (any)

**Step2:在EIGRP进程中,使用distribute-list,调用Prefix-list,进行基于接口级别的过滤:**

R3(config)#router rip

distribute-list prefix R-5 in serial 1(调用了整个R-10的访问列表,往下匹配)

distribute-list(可以调用访问列表,也可以调用前缀列表!!)

**out方向的控制:**

只在路由出口方向进行控制,影响路由下游方向的路由器,而不影响本路由器.

**In方向的控制:**

在路由器的入口方向进行控制,直接影响到本路由器.

### In方向的控制:

在路由器的入口方向进行控制,直接影响到本路由器.

## LAB09:前缀列表的详细描述

prefix-list的语法

15.5.0.0/A ge B le C  
(make sure: len < ge-value <= le-value)  
          A      B      C

记得清路由表clear ip route

例子1

如果A/B/C

15.5.0.0/16 ge 20 le 30

表示: 15.5.\*.\* (前16位相同)

而且满足路由长度是大于等于/20, 小于等于/30的所有路由

例子2

如果只有A/B, 没有C

15.5.0.0/16 ge 20 (le 32) 默认的

表示: 15.5.\*.\* 前16位相同, 而且满足路由长度是大于等于/20, 小于等于/32的所有路由

例子3

如果只有A/C

15.5.0.0/16 (ge 16) le 30

表示: 15.5.\*.\* 前16位相同, 而且满足路由长度大于等于16, 小于等于30的所有路由

例子4

如果只有A

15.5.0.0/16 (ge 16)(le 16)

表示: 15.5.\*.\* 相同, 满足路由长度是16位的路由

例子5

0.0.0.0/0 le 32=any

例子6

0.0.0.0/0 = 默认路由0.0.0.0

## LAB10:在路由协议之间的重分布中,实现路由重分布时的基于协议的路由过滤:

在OSPF和RIP之间做双向重分布:

R1(config)#router ospf 110

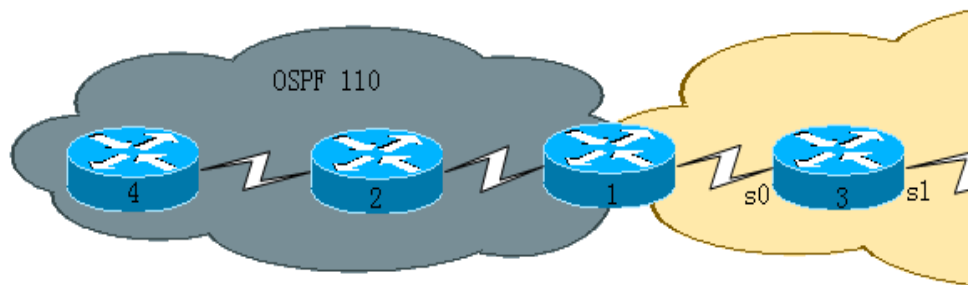
R1(config-router)#redistribute rip subnets

```
R1(config-router)#redistribute rip subnets
```

```
R1(config)#router rip
```

```
R1(config-router)#redistribute ospf 110 metric 1
```

### 10-1:通过ACL,实现路由控制:



#### Step1:通过ACL,定义所需要控制的路由:

```
R1(config)#access-list 5 deny 15.5.16.0
```

```
R1(config)#access-list 5 permit any
```

#### Step2:在OSPF进程中:

```
R1(config)#router ospf 110
```

```
R1(config)#distribute-list 5 out  
rip (理解成rip out)
```

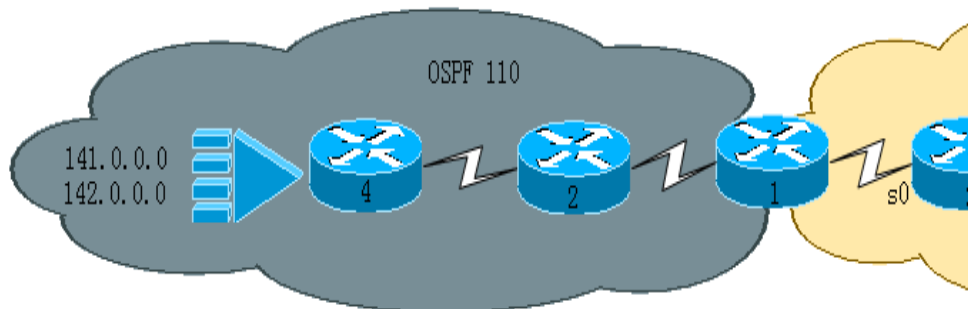
Filter networks in routing updates      IP access list number  
in      Filter incoming routing updates

outgoing routing updates

out      Filter

意思是: 禁止把rip中的15.5.16.0路由重分布到OSPF里面,也就是说除了R1之外,其它的ospf的路由器都没有15.5.16.0的路由!!!!

### 10-2:通过prefix-list ,实现路由控制:



#### Step1:通过Prefix-list,定义所需要控制的路由:

```
R1(config)#ip prefix-list xxx seq 5 deny 12.0.0.0/24
```

```
ip prefix-list xxx seq 10 deny 141.0.0.0/24
```

```
ip prefix-list xxx seq 15 permit 0.0.0.0/0 le 32
```

### Step2:

R1(config)#router rip

R1(config-router)#distribute-list prefix xxx out ospf 110

调用一个空的,不存在的prefix-list,相当是permit any:

R1(config-router)#distribute-list prefix xxx out ospf 110

0 - 0 小心字符相同

1 - 1

prefix-list 命名,都是大小写敏感的:

R1(config-router)#distribute-list prefix XXX out ospf 110

## route-map

route-map的特征:

1.route-map 由一组statements/声明,所组成,  
每句声明中,可能包含了Match, set语句.

每句statements有个编号,10/20/30.....

路由器就按照statements的编号,按从小到大(自上而下)顺序执行.

2.

match commands specify criteria to be matched

一旦匹配/符合特定某些条件 (match语句)

set commands modify matching routes.

就立刻执行由set所定义的参数/操作,并且离开route-map.(而不再往下执行)

3:

the first match found for a route is applied.

once there is a match, leave the route map.

路由器在向下检查匹配条件时,不断地与每个statements中的匹配条件进行比较,

如果不匹配,则向下继续检查下一个statements;

如果匹配,则按照set所定义的参数,进行操作,然后立刻离开route-map

(即使下面还有statements没有进行匹配检查,也不往下检查了)

4:route-map 的逻辑关系:

in the same line uses a logical OR(水平方向:或关系)

vertical match uses a logical AND (垂直方向:与关系)

5: 在route-map中:

如果没有match,表示: match any (上面statements剩余的any)

如果没有set,表示: set nothing !! (保留其原始的默认值)

## LAB11:通过route-map,实现路由协议之间的路由过滤/控制:

~~~~~  
~~~~~

Step1: 通过前缀列表定义所需控制的路由: (通过ACL也可以,但比较落后)

```
R1(config)#ip prefix-list R-15 permit 15.0.0.0/8
R1(config)#ip prefix-list R-25 permit 25.5.0.0/16
R1(config)#ip prefix-list R-35 permit 35.0.0.0/24
```

注意: 这里的permit是表示: "匹配",而不是"允许".

Step2: 创建用于重分布的路由控制的route-map:

```
R1(config)#route-map RP-SPF permit 10 (permit: 允许)
R1(config)#match ip address prefix-list R-15 R-25
R1(config-route-map)#set metric 100
R1(config-route-map)#set metric-type type-1
```

```
R1(config)#route-map RP-SPF deny 20 (deny 不允许重分布)
R1(config-router-map)#match ip address prefix-list R-35 没有
Set: Set nothing!
```

```
R1(config)#router-map RP-SPF permit 30
```

step3: 在重分布的协议进程中,调用route-map RP-SPF:

```
R1(config)#router ospf 110
R1(config-rouiter)#redistribute rip route-map RP-SPF subnets
```

Step 4 : 如果没有最后的这句空的route-map,将有什么效果?

```
R1(config)#route-map RP-SPF permit 30
```

剩余的部分路由,都将被Deny.

Step5: 在调用时出错:

```
router ospf 110
redistribute rip route-map RP-SP subnets
```

空的route-map,意味着deny any.

### 1.6—路由的控制③

R3(config-if)#ip policy route-map PBR

#### Step4:

R3#debug ip policy

如果匹配的statements中是permit,意味着进行策略路由

如果匹配的statements中是Deny,意味着不进行策略(要进行正常路由)  
不会丢掉数据包

如果连一个statements都匹配不上,意味着不进行策略路由(正常路由)

show route-map

### 1.7—BGP①

## 1.7—

### BGP①

#### IGP:

包括RIP/EIGRP/OSPF/ISIS/ODR等动态路由协议  
运行在同一个AS中,  
通过Cost/Metric来判断路由的优劣(越小越好);

#### AS:自治系统(小)

An AS is a collection of networks under a single technical administration.(独立的一个技术/管理域)

#### IGP主要任务

正确的描述路由信息和尽快的将数据包送到目的地:  
IGP强调收敛速度。

### BGP(v4)

## BGP(v4)

### 边界网关协议 (Border Gateway Protocol):

运行在不同的AS之间,用于对BGP路由进行控制/策略,

BGP的主要任务:网管的人为意图的集中体现,强调策略控制,不强调收敛.

BGP是通过BGP的属性/Attributes,判断多条路径的优劣,从中进行择优选取.

BGP可以通过网管定义的策略/Policies,实现数据或路由的控制/操纵.

### 自治系统AS:(autonomous Systems)(大):

独立的技术/管理域,通常是一个大型公司,或者组织,或者国家.

这是区别于IGRP/EIGRP的AS(小)的概念;

### BGP本身就是一种策略路由/PBR(Policy-Based Routing)

实现网管的人为意图的集中体现.

### BGP是一种AS BYAS的高级距离向量/DV协议.

BGP认为,每经过一个AS是一跳

RIP认为,每经过一个router是一跳

### 应该使用BGP的情况:

1: ISP:

当允许AS 1 的数据包穿越AS2,

但是不允许AS 1的用户访问AS2内部的时候:

2: Multihome/多宿主:

对于一个用户的AS,如果他同时连接到多个AS/ISP

3: PBR:

当需要对BGP路由/数据,进行人为控制/操纵的时候.

### 不该使用BGP的情况:

1. 与ISP只有单连接,没有同时连接到多个ISP

2. 如果网络硬件设备的档次不够(内存/CPU).

3: 对BGP路由操纵理解有限,无法预计BGP的后果.

4. 链路带宽不足.

### BGP特征:

1: BGP是高级DV协议.

2: BGP工作在TCP/IP协议栈的4层: TCP179端口.

3: BGP是触发/增量更新的协议

4: BGP通过周期性的发送Keepalive信息,保证TCP连接的可靠性.

5: BGP使用多种"BGP属性/Attribut",来衡量路径的优劣.

6: BGP是为巨型网络设计的,意味着这种路由协议,可能带来海量的路由和数据.

### BGP协议的三张表:

1: BGP的邻居表(BGP neighbor table)

BGP的邻居可以直接相连,也可以凌空建立.

2: BGP表(BGP forwarding table/database)

2-1: 一个BGP路由器,可能从它的多个BGP Neighbor那里,学到到达同一个目标网络的多条可能路径.

2-2: 但是,BGP这种路由协议,默认情况下,是不进行负载均衡的.也就是说,到达一个目标网络,只允许有一条可用路径.

2-3: BGP路由器通过"BGP属性",从中多条候选路由中,优选出一条,它自己认为最好/最优/Best的路径,作为到达这个目标网络的"最佳路由",这条路由也会被称之为"优化"了.("优化"的路由,会标打上">")

2-4: 只有"优化"了的路由,才能作为BGP选送出的,最佳的,种子选手,去参加,到达这个目标网络(10.0.0.0/8)的"AD竞争/竞选."

3: IP路由表:

上面BGP提交的"优化/最佳"路由,在经过不同寻路协议之间的"AD的竞争/竞选"之后,如果获胜,才能成功的送进路由表.

### BGP的4种信息包类型:

1: Open

2: Keepalive

3: Update

4: Notification

Peers=neighbors

Any two routers

that have formed a TCP connection,  
to exchange BGP routing information,  
are called BGP peers or neighbors.

## 1:EBGP Neighbor

两个BGP Neighbor分别属于两个不同的AS.

EBGP Neighbor通常是直接相连的. (也会有特例哦, CCSP才会遇到这种非直连的情况)

## 2:IBGP Neighbor

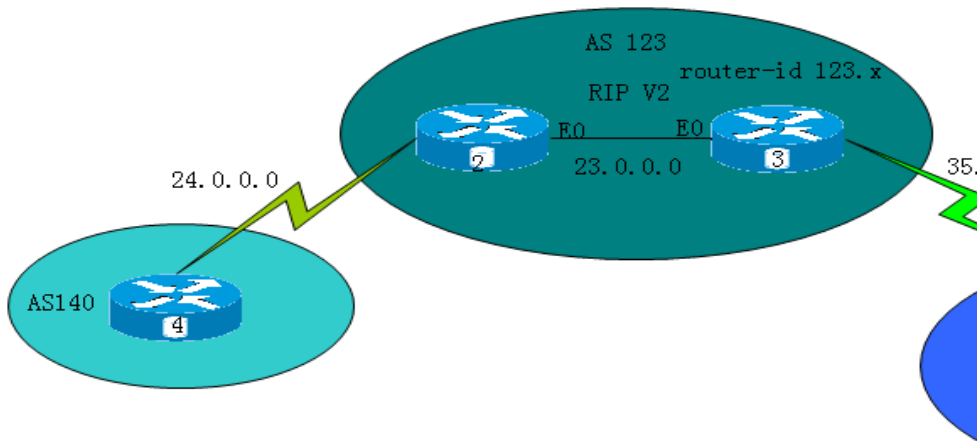
两个BGP Neighbor同时属于一个相同的AS.

IBGP Neighbor可以是直连的,也可以是非直连的.

## BGP的操作步骤

- 1: 确认L1/L2连路通 (通过Ping, 进行链路测试)
- 2: 确认L3的IGP的ROUTE通过 (Show ip route rip/eigrp/ospf)
- 3: 启动BGP, 建立BGP邻居 (TCP179) (show ip bgp summary)
- 4: 宣告BGP路由 (network 1.0.0.0 mask 255.0.0.0)
- 5: 判断BGP优化的基础条件 (同步, 下一跳)

## LAB1: BGP的基本配置:



### Step1: 确认L1/L2的连通性.

可以通过Ping, 进行链路测试.

### Step2: 确保L3的IP路由的通达 (是通过IGP实现):

(一般都是在同一个AS中完成)

(两个AS之间是不运行IGP的)

在AS123中的所有路由器(R2/R3), 运行IGP: RIP V2

R2/R3#

router rip

version 2

network 23.0.0.0

no auto-summary

测试L3的IGP网络(ping)  
show ip route rip

**step3:启动BGP,并且立刻指定全局唯一的bgp router-id.**

R2(config)#router bgp 123  
R2(config-router)#bgp router-id 123.0.0.2

**Step4:构建BGP Neighbor**

R5-R3 EBGP

R5(config-router)#neighbor 35.0.0.3 remote-as 123  
R3(config-router)#neighbor 35.0.0.5 remote-as 150

R3-R2 IBGP:

R3(config-router)#neighbor 23.0.0.2 remote-as 123  
R2(config-router)#neighbor 23.0.0.3 remote-as 123

R2-R4 EBGP:

R2(config-router)#neighbor 24.0.0.4 remote-as 140  
R4(config-router)#neighbor 24.0.0.2 remote-as 123

**Step5:观察BGP邻居建立的过程,及其经历的5个状态:**

**R4#debug ip bgp** 观察BGP邻居建立的过程,及其经历的5个状态:

1-1: Idle:router is searching

1-2: connect: Completed three-way TCP 179 handshake.

1-3: Open sent: Open message sent

1-4: Open confirm:router received agreement

1-5: Established: Peering is established;BGP Routing begins!!

**R5#show tcp brief**(察看TCP连接的摘要信息)

| Local Address  | Foreign Address | (state) |
|----------------|-----------------|---------|
| state/PfxRcd   |                 |         |
| 35.0.0.5.11001 | 35.0.0.3.179    | ESTAB   |

收到的路由条目

**R5#show ip bgp summary**(察看BGP邻居的简要信息)

BGP router identifier 150.0.0.5,local AS number 150

| Neighbor | V | AS       | ***** |
|----------|---|----------|-------|
| State    |   | / PfxRcd |       |

```

35.0.0.3      4      123      * * *      (如果邻居已经
建立好了,这里必需空白) / 0(收到的路由条目)
18.0.0.8      4      100      * * *
/ 3

```

0: 没有从这个邻居收到BGP路由,但是和这个邻居的BGP邻居关系已经建好了.

3: 从这个邻居收到三条BGP路由,同时和这个邻居的BGP邻居关系已经建好了.

### Step6:通过 network 命令,宣告BGP路由

BGP Network配置:

```
R5(config)#router bgp 150
```

```
R5(config-router)#network 105.5.0.0 mask 255.255.0.0
                                (接口所在的网络号) (这个网络的准确路由长度)
```

#### network命令的含义:

在IGP中:

1: 激活接口,在这个接口中运行此IGP路由协议.

(在IGP中,路由协议是向运行的接口发送路由更新)

2: 这个路由器的IGP路由协议,正在为此接口所在的网段进行路由.

在BGP中:

1. 这个路由器的BGP路由协议,正在为此BGP网段进行路由.

(在BGP中,BGP路由协议是向BGP Neighbor发送路由更新.

所以,没有了"激活接口"的含义)

在BGP路由器上,检查自己的BGP路由,是否已经成功地宣告到BGP进程中:

```
R5#show ip bgp (察看BGP表,BGP数据库)
```

```

      Network          Next Hop(是指更新源)
* > 105.5.0.0/16      0.0.0.0(源自本路由器)

```

```
* valid
```

```
> best
```

0.0.0.0 表示下一跳为0

```
R3#sh ip bgp
```

```

      Network          Next Hop          Metric LocPrf
Weight Path
*   i104.4.4.0/24      24.0.0.4 (源自AS140)          0      100

```

```

0    140 i
*> 105.5.0.0/16    35.0.0.5 (源自AS150)    0
0    150 i

```

**BGP 属性:**

Metric: MED

LocPrf : local preference

Weight

Path: AS Path

Origin codes: i (表示使用Network命令, 宣告到BGP中的)

R2#sh ip bgp

| Network         | Next Hop | Metric | LocPrf | Weight | Path |
|-----------------|----------|--------|--------|--------|------|
| *> 104.4.4.0/24 | 24.0.0.4 |        | 0      |        | 0    |
| 140 i           |          |        |        |        |      |
| * i105.5.0.0/16 | 35.0.0.5 |        | 0      | 100    | 0    |
| 150 i           |          |        |        |        |      |

**step7:EBGP邻居的路由优化问题:**

BGP路由优化必要条件 (注意是必要而不是充分)

1: 下一跳问题:

- ∴ BGP路由的下一跳是其直连路由,
- ∴ 不存在下一跳不可达的问题.

(整个AS140中的100个路由器, 察看到这条BGP路由的下一跳都是: 24.0.0.2)

(整个AS123中的100个路由器, 察看到这条BGP路由的下一跳都是: 35.0.0.5)

其下

一跳保持为相邻的AS的边缘节点 (35.0.0.5)

2: 同步问题:

对于EBGP邻居,

- ∴ 无需遵循同步规则,
- ∴ 没有同步问题!!

结论: 对于EBGP, 只要是从EBGP邻居那里传来的路由, 在不考虑其它BGP属性的情况下, 是肯定可以优化的.

注意: 只有已经优化了的BGP路由, 才能进入IP路由表。

**Step8:IBGP路由的优化**

BGP路由器,把自己学到的BGP路由,转发给别的BGP邻居的必要条件: (R3)

每个BGP路由器,对于特定的某条BGP路由,  
必须是自己已经优化的路由,才具备转发给别的BGP邻居的能力.

注意:

这是必要条件,不是充分条件!!!

这意味着:即使自己已经优化,但此路由器,可能转发,也可能不转发.

察看R3向R2发送了哪些BGP路由条目:

```
R3#show ip bgp neighbors 23.0.0.2 advertised-routes
```

IBGP路由的必要优化条件: (以下是必要条件,而不是充分条件)

1: 下一跳可达

2: 路由的同步

**8-1:** 察看R2上,在R3没有作任何更改前,当前的BGP路由:

注意观察:

1: 本路由器R2,是否能够到达这条BGP路由的下一跳?

(在R2上,观察从R3这个IBGP邻居获得的BGP路由)

```
R2#show ip bgp
```

| Network         | Next Hop  |
|-----------------|-----------|
| * i105.5.0.0/16 | 35.0.0.5  |
|                 | 当前下一跳它不可达 |

解决方案:

```
R3(config)#router bgp 123
```

```
R3(config-router)#neighbor 23.0.0.2 next-hop-self
```

```
R3#clear ip bgp *soft out (软清除, 避免收敛速度过慢)
```

```
R2#show ip bgp
```

| Network          | Next Hop | Metric | LocPrf | Weight | Path              |
|------------------|----------|--------|--------|--------|-------------------|
| * >i105.5.0.0/16 | 23.0.0.3 | 0      | 100    | 0      | 150               |
|                  |          |        |        |        | 当前的下一跳是可达的,所以是最优的 |

**8-2:** 同步问题(在新版的IOS里面,默认是关闭同步的)

(本质: R2是否能够通过IGP,得到这条105.1.0.0/16路由)

所谓同步,是指:

如果R2能够通过IGP(RIP),学到这条路由(105.5.0.0/16),那么就同步.

(也就是: 满足同步要求). (无论此路由是否本AS或者其他AS)

如果R2不能够通过IGP,学到这条路由(105.5.0.0/16),那就不满足同步,(也就是:不满足同步要求).

BGP路由器,对于从IBGP邻居学到的路由,默认要求同步.

但是,

∴实际上R2是不可能通过IGP,学到另外的一个AS中的路由.

∴只能关闭同步规则,也就是不遵循同步规则.

解决方案:关闭同步规则:

R2#

router bgp 123

no synchronization

### Step9:检测各路由器能否通达

#### EBGP

R3#show ip bgp

|    | Network      | Next Hop |
|----|--------------|----------|
| *> | 105.5.0.0/16 | 35.0.0.5 |

R3#show ip route

B 105.5.0.0 [20/0] via 35.0.0.5,

#### IBGP:

R2#show ip bgp

|     | Network      | Next Hop |
|-----|--------------|----------|
| *>i | 105.5.0.0/16 | 23.0.0.3 |

R2#show ip route

B 105.5.0.0 [200/0] via 23.0.0.3

### 1.7—BGP②

## 1.7—BGP②

### BGP的更新源 (BGP Neighbor Update Source Address) :

原则1:

在默认情况下,

在默认情况下,  
BGP路由器以自己路由表中,到达对方BGP邻居的地址的那条路由所指示的  
出接口(物理接口) 的地址,作为自己的BGP更新源(源地址)

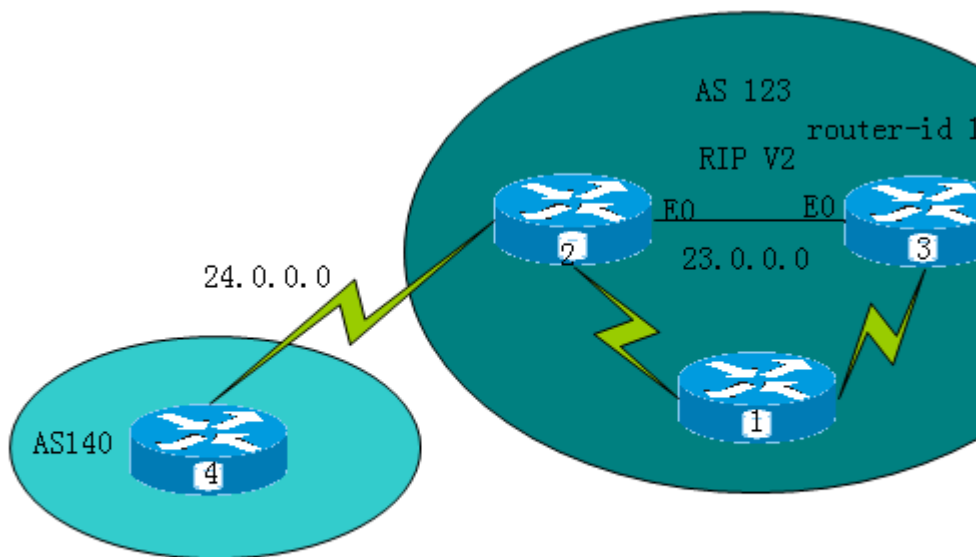
原则2:

当BGP路由器,收到邻居发来的BGP信息时,会检查其源地址,  
然后和自己宣告的Neighbor的目标地址进行比较,  
如果一致,这个BGP Session才可建立起来.

### BGP路由黑洞的解决方案:

- 1: 选择性重分布 (Redistribute Selected BGP Route into IGP) ;
- 2: 冗余的IBGP (Full-mesh IBGP) ;
- 3: 路由反射器 (Part-mesh IBGP+Reflector) ;
- 4: 联邦 (Confederation) ;
- 5: (MPLS) 。

### LAB2: 验证通过物理接口,构建IBGP邻居的不稳定性:



Step1:确认L1/L2通达

Step2:确认L3的IGP通达(RIP),在AS123内R1/2/3

Step3:通过物理接口,在R2/R3之间构建IBGP邻居

```
show run | begin router bgp
R2#neighbor 23.0.0.3 remote-as 123
R3#neighbor 23.0.0.2 remote-as 123
```

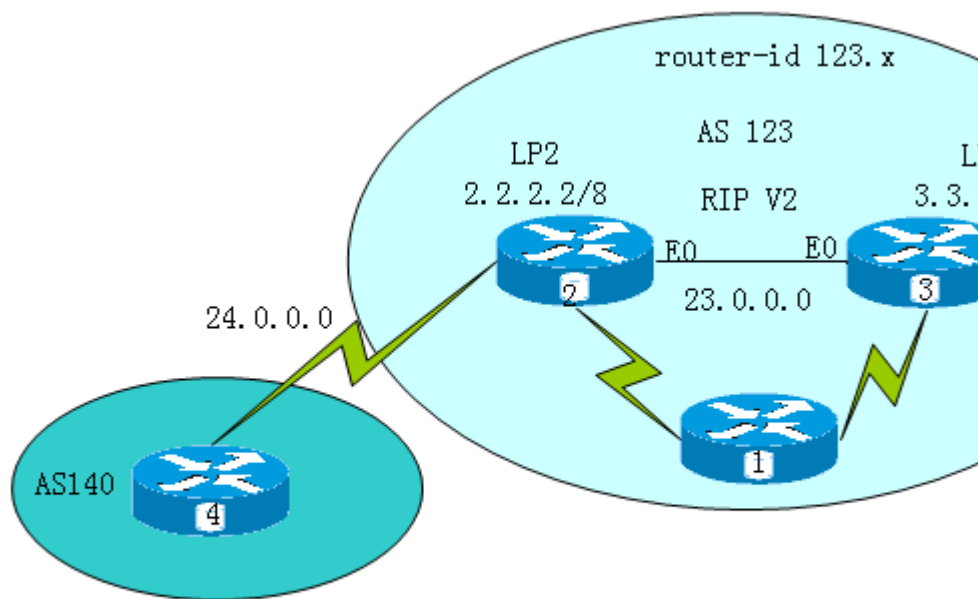
结论:

在IBGP中,如果使用物理接口构建邻居,是很不稳定的.  
很可能因为某条物理链路的抖动,导致IBGP邻居的Flapping/抖动:

建议:

使用环回口/Loopback接口,构建IBGP邻居.

## LAB3:以Loopback接口作为BGP更新源,构建稳定的IBGP Session



**Step1:**为每个IBGP路由器,构建一个环回口:

**Step2:**把此Loopback接口,宣告到IGP(RIP)中.

```
R2/R3#
router rip
network 2.0.0.0
or
network 3.0.0.0
```

**step3:**在R2/R3上,删除原来的,通过物理接口构建的邻居.记得建立

### router-id

```
R2#router bgp 123
  no nei 23.0.0.3
r3#no nei 23.0.0.2
```

### Step4:通过环回口构建IBGP邻居:

4-1:

以对方的环回口,作为IBGP的目标地址:

```
R2#neighbor 3.3.3.3 remote-as 123
R3#neighbor 2.2.2.2 remote-as 123
```

4-2:

更改了IBGP邻居后需要把下一跳也做相应更改

```
R2(config-router)#nei 3.3.3.3 next-hop-self
R3(config-router)#nei 2.2.2.2 next-hop-self
```

注意:删除原物理接口所做的IBGP邻居时,相应的下一跳将自动删除.

4-3:

以自己的环回口,作为IBGP连接的源地址:

```
R2#neighbor 3.3.3.3 update-source loopback 2
R3#neighbor 2.2.2.2 update-source loopback 3
```

### step5:

任意切断本AS123 中的物理链路,

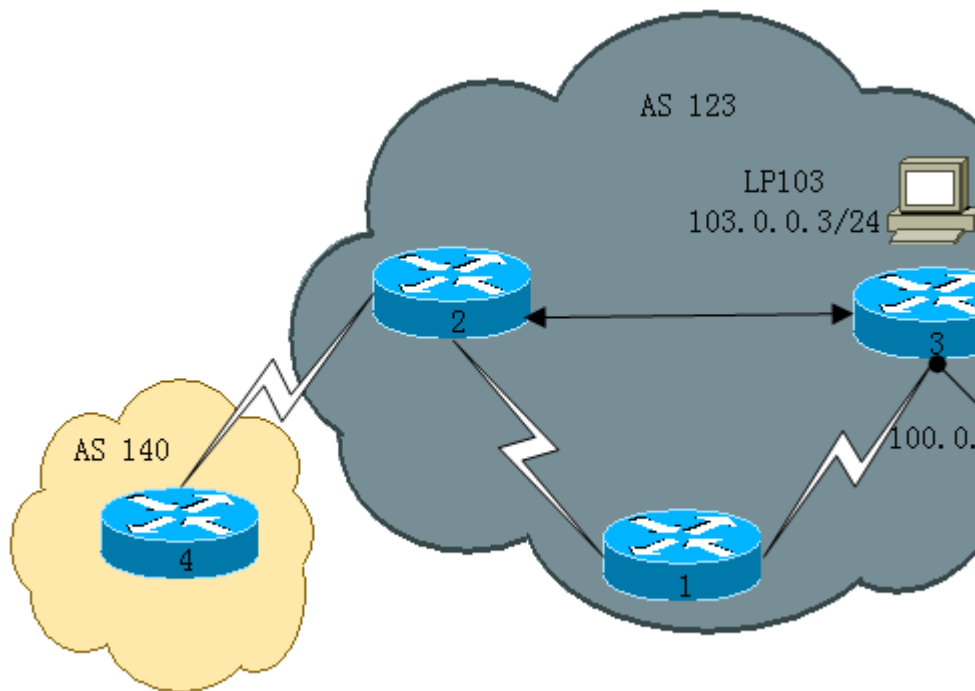
只要两个IBGP路由器R2/R3之间,还有最后一条能够到达,对方的环回口的路由,IBGP邻居都不会中断.

建议:

凡是构建IBGP,默认都使用环回口做更新新源,以构建稳定的IBGP.

**LAB4:在两AS间,存在多条冗余链路的网络环境中:**

**以LoopBack接口作为EBGP更新源,构建稳定的EBGP Session.**



**Step1:**在两AS之间回口,构建多条冗余链路.

如果两AS间,没有多条冗余链路,就使用物理接口构建EBGPP即可.

**Step2:**为两AS间的EBGP路由器,构建环回口

**Step3:**在各自路由器上,指定到达对方环回口静态路由:

- ∴有两条冗余链路,
- ∴要有两条到达对方环口的静态路由

```
R5: (config)#  
ip route 3.3.3.3 255.255.255.255 35.0.0.3  
ip route 3.3.3.3 255.255.255.255 100.0.0.1
```

```
r3#: (config)#  
ip route 5.5.5.5 255.255.255.255 35.0.0.5  
ip route 5.5.5.5 255.255.255.255 100.0.0.2
```

测试:

```
R3#ping 5.5.5.5 source 3.3.3.3 !!!!!!!!!!!!!!!
```

**Step4:**建立邻居EBGP邻居:

```
R3#  
router bgp 123  
bgp router-id 123.0.0.3
```

```
neighbor 5.5.5.5 remote-as 150
```

```
R5#  
router bgp 150  
bgp router-id 150.0.0.5  
neighbor 3.3.3.3 remote-as 123
```

**step5:**告知对方,自己的更新源:

```
r3(config-router)#neighbor 5.5.5.5 update-source loopback 3
```

```
r5(config-router)#neighbor 3.3.3.3 update-source loopback 5
```

**Step6:更改EBGP的TTL值(Time to Live)**

∴ EBGP TTL值默认是1,  
∴ EBGP的TTL值最少要设为2  
而实际上EBGP多跳这个命令,在不指定其取值时,会自动默认指定为255

```
r3(config-router)#neighbor 5.5.5.5 EBGP-multihop(255)
```

```
r5(config-router)#neighbor 3.3.3.3 EBGP-multihop 2
```

**TTL:time to live**是L3的IP包头中的一个特定字段,IP包每经过一个路由设备,其TTL会自动减1. 如果TTL减到为0,即使路由器有去往目标的路由,也不会继续转发这个IP包.

**Step7:测试**

在R3上添加103.0.0.3/24  
在R5上添加105.5.5.5/24

```
r3(config)#router bgp 123  
r3(config-router)#net 103.0.0.0 mask 255.255.255.0
```

```
r5(config)#router bgp 123  
r5(config-router)#net 105.5.5.0 mask 255.255.255.0
```

```
R3#ping 105.5.5.5 source 103.0.0.3 repeat 1000000 size 15000
```

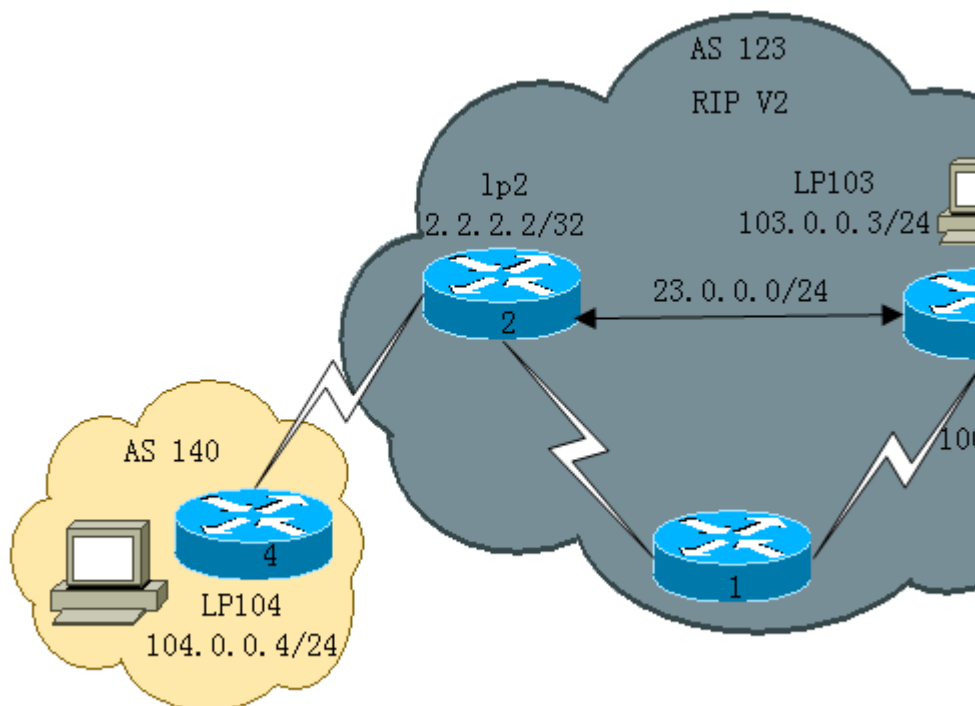
结论:

在一般情况下,EBGP的邻居关系,是不需要使用环回口构建邻居的.  
默认都直接使用物理接口,  
在只有单链路的时候,都是使用物理接口构建邻居.

只有在两AS之间,存在多条冗余链路的时候,才需要考虑使用环回口构建

只有在两AS之间,存在多条冗余链路的时候,才需要考虑使用环回口构建EBGP邻居,以确保其EBGP的稳定性.

## LAB5: 观察BGP黑洞的形成



### Step1:按图配置BGP网络

注意:R1不运行BGP.

R2#

```
router rip
ver 2
network 2.0.0.0
net 23.0.0.0
net 12.0.0.0
```

R3#

```
router rip
ver 2
network 3.0.0.0
net 23.0.0.0
net 13.0.0.0
```

r2/3#

```
router bgp 123
bgp router-id 123.0.0.2/3
```

R2#neighbor 3.3.3.3 remote-as 123

R3#neighbor 2.2.2.2 remote-as 123

```
R3#neighbor 2.2.2.2 remote-as 123
```

```
R2(config-router)#nei 3.3.3.3 next-hop-self
```

```
R3(config-router)#nei 2.2.2.2 next-hop-self
```

```
R2#neighbor 3.3.3.3 update-source loopback 2
```

```
R3#neighbor 2.2.2.2 update-source loopback 3
```

### Step2:将AS140/AS150中的BGP路由,宣告到BGP网络中:

```
R4#(config-if)#
```

```
router bgp 140
```

```
network 104.0.0.0 mask 255.255.255.0
```

记得R4与R2建邻居

```
R5(config-if)#
```

```
router bgp 150
```

```
network 105.5.5.0 mask 255.255.255.0
```

### Step3:过程AS123中的BGP路由用户问题:

3-1: 同步

```
R2#
```

```
router bgp 123
```

```
no synchronization
```

### Step4:这时候全网络互通

```
R4#ping 105.5.5.5 source 104.0.0.4 repeat 1000000 size 15000
```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

### Step5:这时候切断23.0.0.0.

```
R4#ping 105.5.5.5 source 104.0.0.4 repeat 1000000 size 15000
```

.....

R1因为没有运行BGP,

∴不会有导致BGP的路由,成为BGP路由的黑洞.

### BGP路由黑洞的解决方案:

解决BGP路由黑洞,可供选择的解决方案/Solution:

1: Redistribute Selected BGP Route into IGP

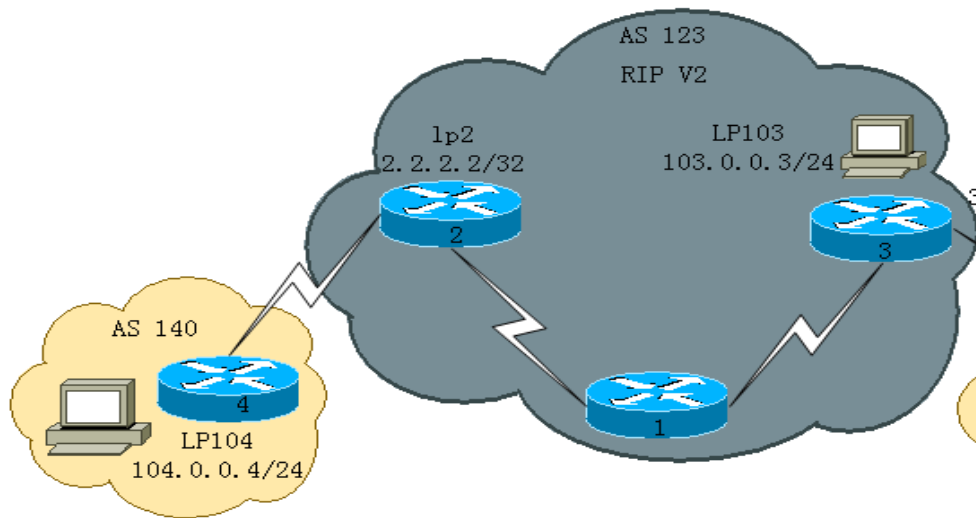
2: Full-mesh IBGP

3: Part-mesh IBGP+Reflector

4: Confederation

5: MPLS

## LAB6: part-mesh IBGP,Redistribute Selected BGP Route into IGP,with sync (同步/synchronization)



Step1: 定义需要重分布到IGP中的,BGP的路由:

```
R3(config)#ip prefix-list B-105 permit 105.1.0.0/16
R2(config)#ip prefix-list B-104 permit 104.1.0.0/24
```

Step2: 通过Route-map,控制重分布到IGP的范围,

```
route-map R3-BGP-RP permit 10
match ip address prefix-list B-105
set metric 1
```

R2#

```
route-map R2-BGP-PR permit 10
match ip address prefix-list B-104
set metric 1
```

小提醒:

不要配置:"route-map R3-BGP\_RP permit 20"  
一旦配置,意味着所有一切BGP路由都进入IGP!

Step3: 按照route-map所定义的条件,将BGP路由注入RIP:

```
R3#
router rip
redistribute bgp 123 route-map R3-BGP-RP
```

Step4: 在R2上观察,105.5.0.0/16,  
R2: 同时从RIP和BGP,都能学到路由,但因为AD竞争原因,  
从RIP所获得的路由,成功进入路由表,  
而从BGP所获得的路由,不能进入路由表.

```
R2#show ip route
R    105.5.0.0[120/2]
```

```
R2#show ip bgp
r>i105.5.0.0/16          3.3.3.3
```

Step5: 在R2观察,如果R2,此时启动了BGP的"同步",  
是否还能优化?

结果:可以优化~~~!!!

因为:R2此时通过RIP,学到105.5.0.0/16,  
结果:可以优化!!!!

Step6: 在R1观察两条路由:  
R 104.4.4.0[120/1] via 12.0.0.2  
R 105.5.0.0[120/1] via 13.0.0.3

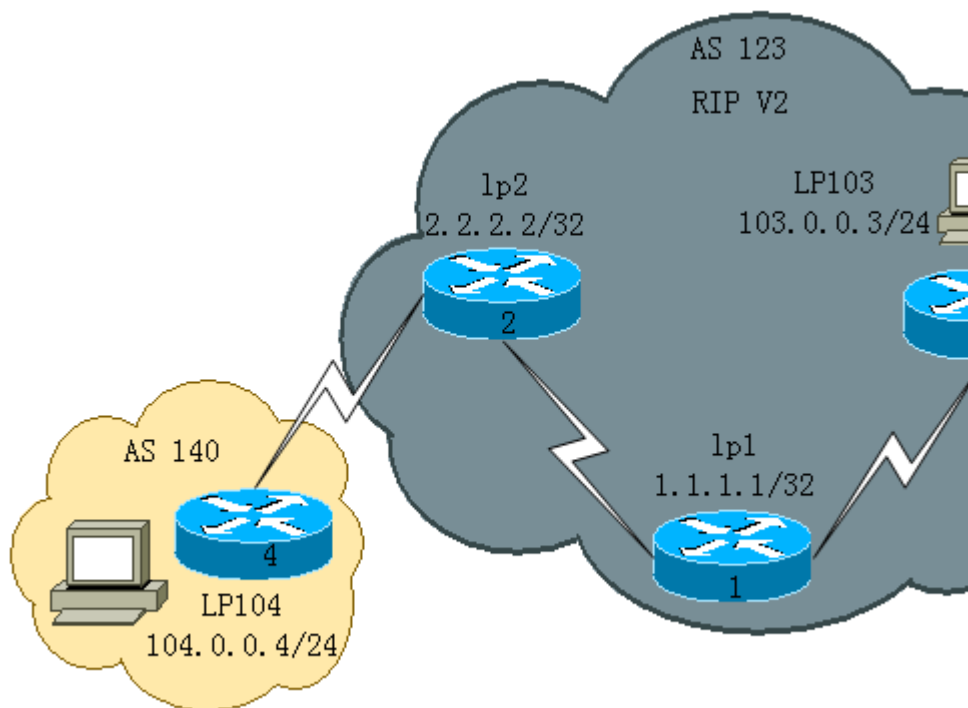
R1不再是黑洞!!

Step7: 在R2上观察路由的递归查询:  
R 105.5.0.0[120/2] via 12.0.0.1  
C 12.0.0.0 is directly connected, serial0

Step8: 测试:

R4#ping 105.5.5.5 source 104.4.4.4 !!!!!!!!!!!

**LAB7: Full-mesh IBGP, No-Sync/(Peer Groups):**



#### Step0:启动BGP进程

```
R1(config)#  
router bgp 123  
bgp router-id 123.0.0.1
```

#### Step1:在R1上,使用Peer-Group(一个模版),对R2/R3建立IBGP邻居:

1-1: 定义peer-group: (模块R1-PG)

```
R1#  
router bgp 123  
neighbor R1-PG peer-group  
neighbor R1-PG remote-as 123  
neighbor R1-PG update-source loopback 1
```

1-2: 对不同的IBGP邻居,调用peer-group:

```
neighbor 2.2.2.2 peer-group R1-PG  
neighbor 3.3.3.3 peer-group R1-PG
```

peer-group 只是一种模版,只影响本路由器的,邻居建立的方法.

在R2/R3上,与R1的邻居建立,仍然可以使用普通方法建立.

#### Step2:确保整个AS123中的所有BGP路由器的,下一跳,同步问题能够解决:

2-1:

```
R1/R2/R3#关闭同步
```

2-2:

R2上,对R1/R3 Say next-hop-self

R3上,对R1/R2 Say next-hop-self

```
r2/r3#nei 1.1.1.1 next-hop-self
```

**Step3:在R1上,观察所有BGP路由:**

```
*>i103.3.3.0/24      3.3.3.3
```

```
*>i104.4.4.4.0/24    2.2.2.2
```

```
*>i105.5.5.5.0/16    3.3.3.3
```

**Step4:观察在R2上的BGP路由的递归查询:**

```
R2#
```

```
B   105.5.0.0[/0]via 3.3.3.3
```

```
R   3.3.3.3[120200/2]via 12.0.0.1
```

```
C   12.0.0.0 is directly connected, serial0
```

**Step5:测试:**

```
R4#ping 105.5.5.5 source 104.0.0.4
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

## 1.7—BGP③

### 1.7—BGP③

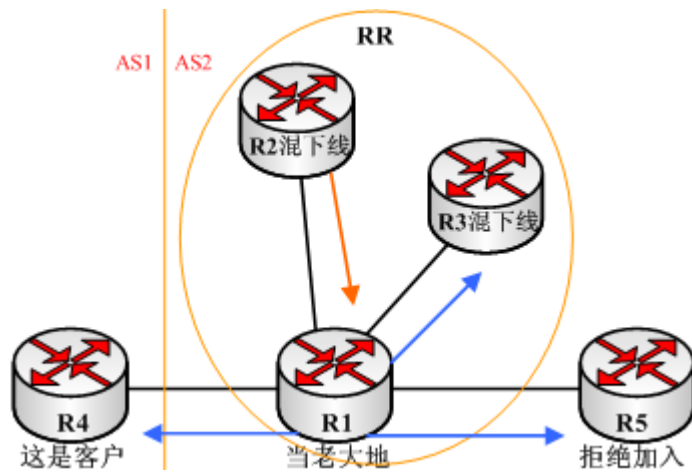
**IBGP的水平分隔原则 (Split Horizon Rule) :**

IBGP的水平分割原则: by default, routes learned via IBGP are never propagated to othe IBGP peers; 默认情况下对于一个BGP路由器R1来说,从一个IBGP邻居R3那里学到的BGP路由,是不会传递给另外的一个IBGP邻居R2的。(提醒:EBGP是没有这种规则的!!!)

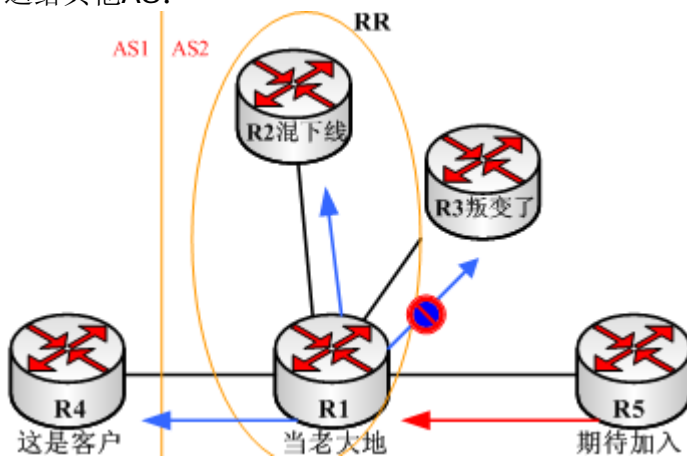
**BGP路由反射器RR (Route Reflector) :**

事实证明:路由反射器就是搞传销地!!!

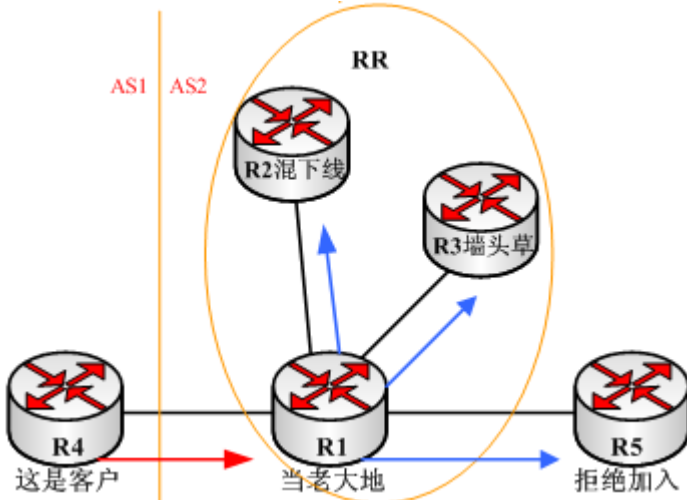
第一种:来自同一AS内的下线的路由,会反射给所有BGP session:



第二种：非下线的路由会发给自己的下线而不会发送给非下线，但可以发送给其他AS：

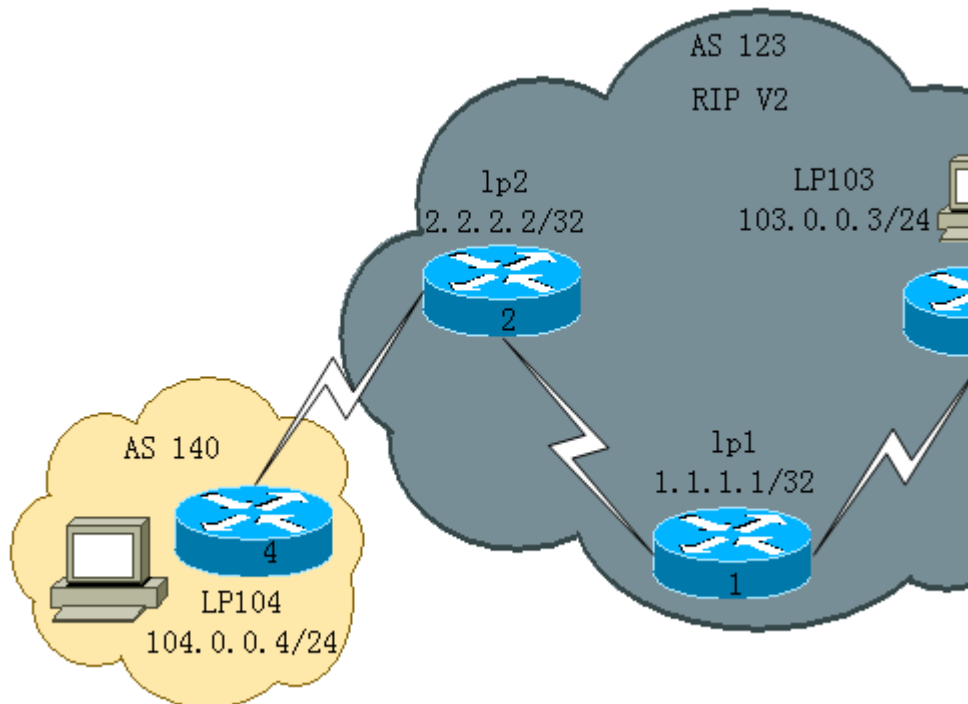


第三种：外部来的EBGP路由，可以发给所有本区内的下线和非下线：



## LAB8: Part-Mesh IBGP +RR (Router-

## REfilector) :



Step1:

删除R2-R3IBGP Session

删除之前: R2有105.5.0.0/16路由

```
R3(config-router)#no neighbor 2.2.2.2
```

```
R2(config-router)#no neighbor 3.3.3.3
```

删除之后

IBGP Split Horizon Rule: /IBGP的水平分割原则:

(已完成实验)

IBGP的水平分割原则,

by default, routes learned via IBGP are never propagated to other IBGP peers.

默认情况下:

对于一个BGP路由器R1来说, 从一个IBGP邻居R3那里学到的BGP路由, 是不是传递给另外的一个IBGP邻居R2的

提醒:EBGP是没有这种规则的!!

Step2:解决方法:BGP Route Reflector /RR  
在R1上,定义R2/R3为自己的"路由反射器的客户端"

2-1:  
如果使用Peer-group:

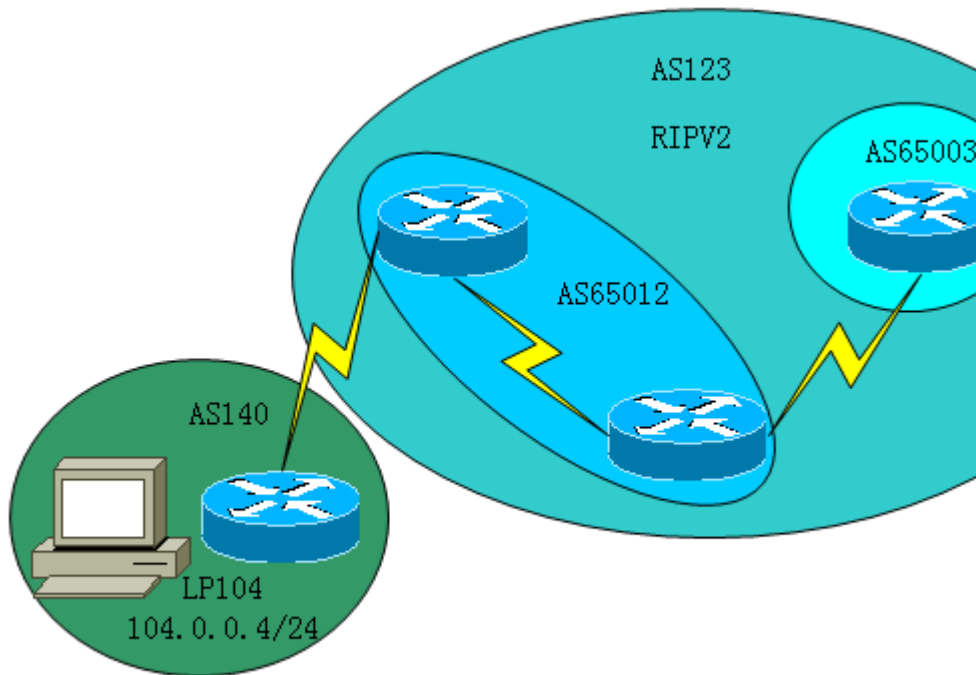
```
R1#neighbor R1-PG route-reflector-client
```

2-2:  
如果没有使用peer-group:

```
R1(router bgp 123)#  
neighbor 2.2.2.2 route-reflector-client  
neighbor 3.3.3.3 route-reflector-client
```

Step3: 查看:  
问题:在R1上,定义R2为自己的"路由反射器的客户端",但R3不是.  
R4能收到105的路由吗?  
能! R5能收到114的路由吗? 能!

## LAB9: 联邦 (Confederation) :



假定：子AS65003/AS65012之间是不运行IGP的、AS65012内部运行的IGP是RIP.

#### STEP1:删除原来的AS123.重新建AS123

R1/2/3#no router bgp 123

R1# (R2, R3建立IGP)

```
router rip
version 2
network 1.0.0.0
network 12.0.0.0
no auto-summary
```

#### step2:启动新的子AS:

R1/R2#

router bgp 65012

R3#

router bgp 65003

#### step3:R1/R2/R3指定自己是属于AS123这个联邦

r1/2/3 (config-router) #

bgp confederation identifier 123

#### step4:在两个子AS相邻的边界路由器上,互相指定对方的子AS号

R1(config-router)#bgp confederation peers 65003

```
R3(config-router)#bgp confederation peers 65012
```

#### step5:在联邦中,互相BGP邻居

构建R3-R5之间的EBGP:

```
R5(config-router)#neighbor 35.0.0.3 remote-as 123
```

```
R3(config-router)#neighbor 35.0.0.5 remote-as 150
```

构建R2-R4之间的EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
```

```
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

提醒: 在 联邦以外的EBGP邻居, 它们能查看到的是联邦的大AS号, 而不是子AS号。

#### R1-R3的联邦EBGP: (联邦子AS间有IGP)

R3:

```
nei 1.1.1.1 remote-as 65012
```

```
nei 1.1.1.1 upd lo 3
```

```
nei 1.1.1.1 ebgp-multihop
```

R1:

```
nei 3.3.3.3 remote-as 65012
```

```
nei 3.3.3.3 upd lo 1
```

```
nei 3.3.3.3 ebgp-multihop
```

#### R1-R2的联邦IBGP

R1#

```
nei 2.2.2.2 remote-as 65012
```

```
nei 2.2.2.2 upda lo 1
```

R2#

```
nei 1.1.1.1 remote-as 65012
```

```
nei 1.1.1.1 upda lo 2
```

#### step6:宣告BGP路由

宣告104和105进BGP

#### step7: 观察一下BGP的路由的传递

R3上有优化路由

```
*> 105 5.5.0/24 35.0.0.5
```

但是由于R1无法通过IGP获得到达35.0.0.0这个网络的路由, 所以R1到此路由的 下一跳不可达, 从而无法优化。

R1

```
* 105 5.5.0/24 13.0.0.3
```

解决办法:

```
R3(config-router)#neighbor 13.0.0.1 next-hop-self
```

```
R1# *>105.5.0.0/16 13.0.0.3
```

#### step8:联邦EBGP和普通EBGP的异同点（观察105.5.5.0）

下一跳:

在联邦的子AS中,所有路由器看到的BGP下一跳,都是相邻大AS的边缘节点,而不是本联邦内子AS的下一跳,这是区别与普通EBGP的.

同步:

联邦EBGP和普通EBGP一样,无需考察同步问题.

#### step9:联邦内的IBGP:(R1-R2)（观察105和104）

```
R2# *i105.5.0.0/16 13.0.0.3
```

```
R1#(config-router)#neighbor 2.2.2.2 next-hop-self
```

R2

```
*>i105.5.5.0/24 12.0.0.1 0 100 0 (65003) 150 i
```

```
R4#*> 105.5.0.0/16 24.0.0.2
```

结论:

联邦子AS之间的EBGP的下一跳,不像普通EBGP那样每经过一个AS,都发生改变,而保留原始的BGP下一跳.

下一跳:

可达,因为两个AS间运行了RIP

同步:

原因是R2从IBGP学到的路由,默认要检查同步

但现在R2不可能通过RIP学到此BGP的路由(指这条路由:i105.1.0.0/16)

#### Step10:联邦内的同步问题:

```
R2/R3# no sy
```

```
R1(config-router-65012)#sy
```

```
R1#show ip bgp (当R1启动"同步")
```

```
* i104.4.4.0/24 2.2.2.2 (来自联邦IBGP)
```

```
* >105.5.0.0/16 13.0.0.3(来自联邦EBGP)
```

```
R2#sh ip bgp
```

```
R2#(config-router)#sync
```

结论:

联邦子AS之间的同步问题:

如果路由来自联邦IBGP,则需要审查同步条件.

如果路由来自联邦EBGP,则不需要审查同步条件.

**step11:**在两个AS间,无IGP的情况

```
R3(config)#no router rip
```

```
R1(config)#router rip
```

```
R1(config-router)#no net 13.0.0.0
```

手工建立到达对方环回口的静态路由;

```
R1:ip route 3.3.3.0 255.255.255.0 13.0.0.3
```

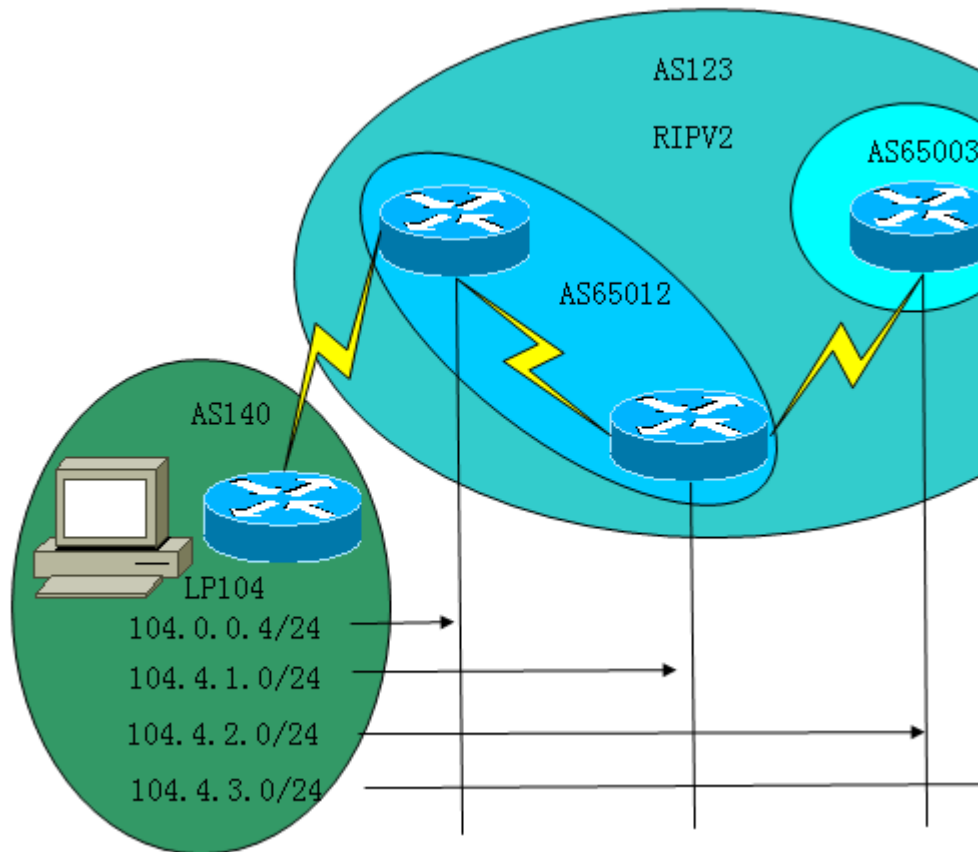
```
R3:ip route 1.1.1.0 255.255.255.0 13.0.0.1
```

到此,R1-R3之间的联邦EBGP关系,可以成功建立

但,R2无法通过IGP学到3.3.3.3的路由,所以下一跳不可达.需在R1指next-hop-self

以下是观察BGP的路由的传递:

**LAB10: 团体 (Community)**  
(相当于一种BGP路由的标识位,常用于标识这条BGP路由应该传播的范围)



**step1:通过prefix,定义出特定的BGP路由**

R4:

```
ip prefix-list B-1 seq 5 permit 104.4.0.0/24
ip prefix-list B-2 seq 10 permit 104.4.1.0/24
ip prefix-list B-3 seq 15 permit 104.4.2.0/24
```

**step2:通过route-map,调用前缀列表设定每类路由的community种类**

```
route-map T-R2 permit 10
match ip add prefix B-1
set community no-advertise (do not advertise to any peer/R4通知R2, 不要发给任何BGP邻居)
!
route-map T-R2 permit 20
match ip add prefix B-2
set community local-as (Do not send outside local AS/联邦的子AS/小AS)
!
route-map T-R2 permit 30
match ip add prefix B-3
set community no-export (do not export to next AS/联邦的大AS)
!
route-map T-R2 permit 40 (match any,set nothing!)
```

```
route-map T-R2 permit 40 (match any,set nothing!)  
!
```

**step3:**在R4上,对R2的BGP路由策略发生,"出方向"的改变

```
R4(config)#router bgp 140
```

```
R4(config-router)#nei 24.0.0.2 route-map T-R2 out
```

**step4:**每个BGP路由器,将community这些标签发送给下一个BGP路由器

每向前走一个BGP Router, 就要"send-community"推一下。

在BGP进程中

```
R4#nei 24.0.0.2 send-community
```

```
R2#nei 1.1.1.1 send-community
```

```
R1#nei 3.3.3.3 send-community
```

```
clear ip bgp *
```

```
R2#sh ip bgp community
```

```
sh ip bgp community no-advertise
```

```
..... local-as
```

```
..... no-export
```

```
..... 104.4.3.0/24
```

**step5:**

如果route-map中, 没有最后的那句空的route-map"route-map T-R2 permit 40",

R4向R2通告的bgp路由只有3条:

```
R4#sh ip bgp neigh 24.0.0.2 advertised-routes
```

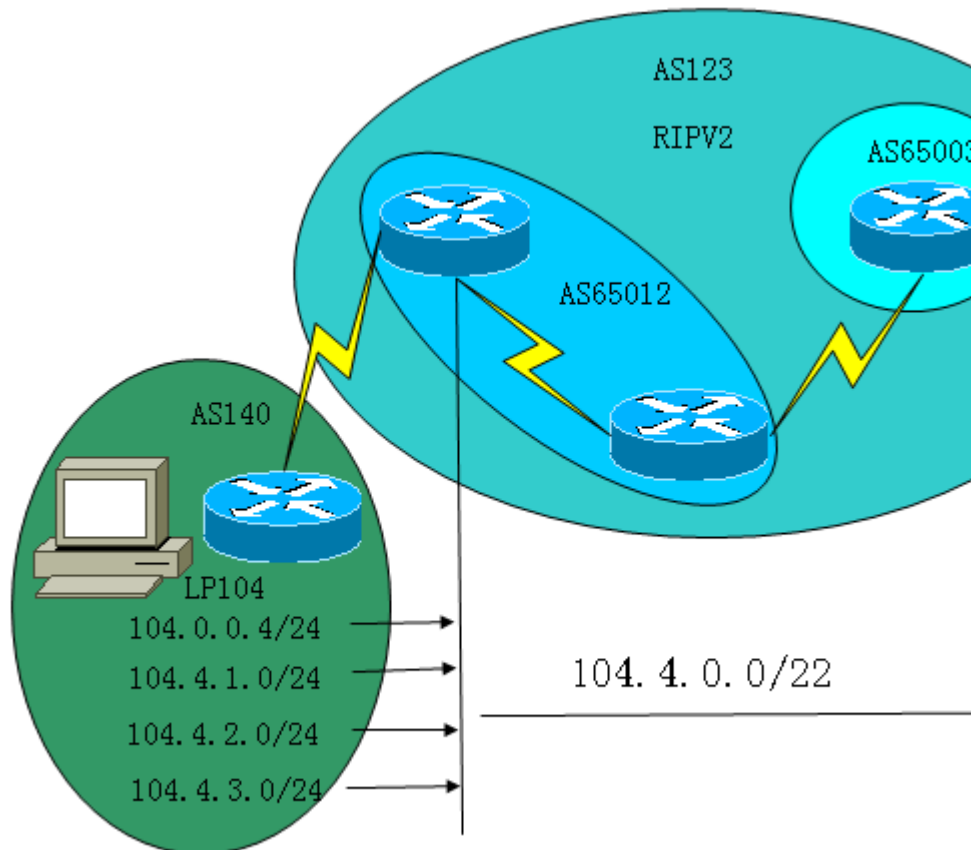
```
* > 104.4.0.0/24
```

```
* > 104.4.1.0/24
```

```
* > 104.4.3.0/24
```

## BGP-Summarization

**LAB11:**非专业汇总(network命令,是不需要宣告明细路由的.)



**step1:手工生成一条需要汇总的,静态的,空接口的路由:**

R4指4条环回: 104.4.1.0/24

104.4.2.4/24

104.4.3.4/24

104.4.0.4/24

R4(config)#ip route 104.4.0.0 255.255.252.0 null 0 <---指这条静态的原因是为了network的时候能在路由

表里发现这条指向null 0 且22位

的路由存在,才

能network成功

**step2:将上述路由,宣告到BGP进程里**

R4(config)#router b 140

R4(config-router)#net 104.4.0.0 mask 255.255.252.0

在R5上,可以查看到明细路由/汇总路由,实际上,明细路由是不需要的:

**Step3:删除原明细BGP路由的宣告:**

R4(config-router)#

router bgp 140

no net 104.4.0.0 mask 255.255.255.0

```
no net 104.4.0.0 mask 255.255.255.0
no net 104.4.1.0 mask 255.255.255.0
no net 104.4.2.0 mask 255.255.255.0
no net 104.4.3.0 mask 255.255.255.0
```

**step4:**

```
R5#show ip bgp
```

```
*>104.4.0.0/22
```

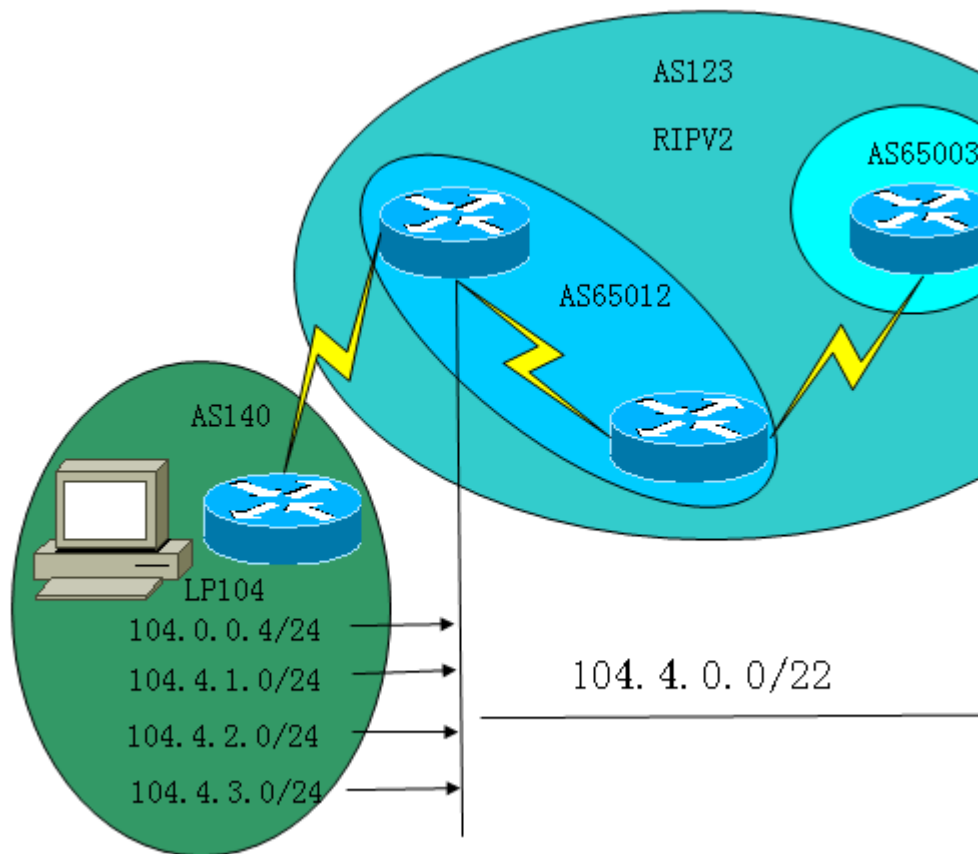
**step5:**

R4(config)#no ip route 140.4.0.0 255.255.252.0 null 0 - 汇总不成功

R4在BGP表中都无法优化,更加不会传给别地BGP路由器了.

也可以由IGP自动生成

## LAB12:BGP的专业汇总（推荐的方法）



**step1:准确地宣告每一条BGP明细路由**

```
net 104.4.0.0 ma 255.255.255.0
net 104.4.1.0 ma 255.255.255.0
net 104.4.2.0 ma 255.255.255.0
```

```
net 104.4.3.0 ma 255.255.255.0
```

(不要用network宣告汇总路由)

```
no net 140.4.0.0 ma 255.255.252.0
```

**step2:使用aggregate命令,实现BGP路由的汇总 (aggregate-address命令是不需要实现配置汇总路由的) :**

```
R4(config-router-BGP)#aggregate-address 104.4.0.0  
255.255.252.0
```

此时,在BGP路由器上,都接收到了明细和汇总的路由.

**step3:为了不让明细路由传播出去,启用summary-only参数.**

```
R4(config-router-BGP)#aggregate-address 104.4.0.0  
255.255.252.0 summary-only
```

此时的R4抑制了所有的明细路由,只发送了汇总路由给R3,实现了BGP路由的汇总

**step4:R4的BGP进程,自动生成了一条用于汇总的空接口路由**

```
r4#show ip route
```

```
B 104.4.0.0/22[200/0]via 0.0.0.0
```

## 1.7—BGP④

注意: 默认路由ip route 0.0.0.0 0.0.0.0 12.1.1.1是不可以作为BGP邻居TCP始发连接的(但回包可以)

要配静态路由: ip route 13.1.1.3 255.255.255.0 12.1.1.1

第三方下一条: 多路以太网访问网络会出现。

BGP属性控制(以下面图为例):

```
route-map commu permit 10
```

```
r2(config-route-map)#set community ?
```

local-AS: 小AS范围内

no-advertise 本路由器, 不传到别的路由器

no-export 本联邦内

```
route-map commu permit 20
```

r2:

```
router bgp 64512
```

```
neighbor 24.1.1.4 route-map commu in
```

```
r2:router bgp 64512
```

```
neighbor 1.1.1.1 send-community 把属性传递下去（默认不传递）
```

```
r1:router bgp 64512
```

```
neighbor 3.3.3.3 send-community（不管属性本身是否具有传递性质，如果要传，首先都要send-community）
```

```
route-map commu permit 10
```

```
no set community
```

```
set community ?（从400到100的路由）
```

```
<1-4294...> community number (32bit)（作用相当于tag）
```

```
aa:nn (16bit:16bit)
```

```
set community 400:100（这个默认会传递出去，不用额外命令）
```

```
sh route-map
```

```
community 26214500（前16位：后16位叠加--不是相加）
```

```
(config)#ip bgp-community new-format
```

```
community 400:100
```

```
r5(config-router)#aggregate-address 172.16.4.0 255.255.252.0
```

```
summary-only（只发汇总路由出去，明细路由被抑制） as-set（把所有明细属性记录下来，防止路由返回路由始发点--虽然还是会发回去，但是因为有as path防环机制，所以始发路由器不接收--EBGP的防环机制的一种）
```

```
suppress-map--调用route-map，可以抑制想要的路由条目
```

```
unsuppress-map--针对某邻居不抑制
```

```
r1(config-router)#neighbor 14.1.1.4 unsuppress-map route-map名称
```

```
advertise-map 只产生匹配路由的聚合，如果该匹配路由消失，那么对整个网段的aggregate都会消失
```

```
aggregate-address 172.16.4.0 255.255.252.0 as-set advertise-map
```

```
AAA
```

```
这时4/5/6/7网段都会被聚合成4.0/22，如果这时5.0消失了，那么聚合不起作用，则4/6/7网段/24位路由发出去
```

选路：

1:weight

2:l-p

3:next hop

4:as-path

5:起源代码

增加AS-PATH:

```
r1:route-map as
```

```
set as-path prepend 600 700
```

```
router bgp 100
```

```
set as-path prepend 600 700
router bgp 100
nei 13.1.1.3 route-map as in
```

as-path:r1上看

策略r1 in 600 700

600 700 300 500 i

策略r3 out 600 700

300 600 700 500 i

如果设了入接口的策略，那么在原来该有的as-path顺序不变的基础上网前加上属性。

如果设了出方向的策略，那么as-path属性先加上策略在把自己的AS号加上发出去。

修改起源代码：做了相同as-path数目后。

```
r1(config)#route-map ori
```

```
set origin incomplete (未知)
```

```
router bgp 100
```

```
nei 12.1.1.2 route-map ori in
```

peer group:

```
router bgp 200
```

```
neighbor aaa(peer-group name) peer-group
```

```
neighbor aaa update-source lo 0
```

```
neighbor aaa remote-as 200
```

```
neighbor aaa route-reflector-client
```

```
neighbor 2.2.2.2 peer-group aaa (调用)
```

```
neighbor 3.3.3.3 peer-group aaa
```

本地优先级：（只能本AS或者联邦内传递）

```
route-map local-p
```

```
set local-preference 200（默认100）
```

```
router bgp 200
```

```
nei 12.1.1.1 route-map local-p in（要影响所有本AS的路由器，所以在边界的IN方向做）
```

```
clear ip bgp * s
```

weight值，只对本路由器有效而且cisco only

MED主要影响别人选路（如果从ibgp学来默认不传出本AS；EBGP就一定会传）：

```
down邻居:neighbor 12.1.1.2 shutdown
```

```
起用: no neighbor 12.1.1.2 shutdown
```

```
r2:ip prefex-list med permit 4.4.4.0/24
```

```
route-map med
match ip add prefix-list med
set metric 200
router bgp 200
nei 12.1.1.1 route-map med out
```

```
r3:ip prefix-list med permit 4.4.4.0/24
route-map med
match ip add prefix-list med
set metric 300
router bgp 200
nei 13.1.1.1 route-map med out
```

metric为空时cisco路由器默认值为0，为最佳选路，但对华为等不一定。

```
router bgp 100
bgp bestpath med missing-as-worst
```

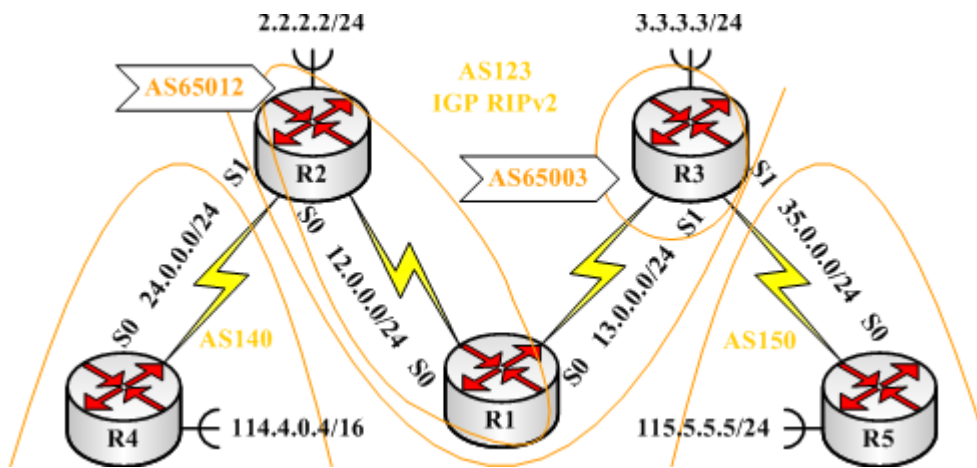
这时认为空的MED为最坏选路

ping: icmp echo请求->icmp echo-reply回复，有去有回  
traceroute ttl=1 ttl=2，然后记录美一跳回包的ip

#### LAB8:

Confederation(联邦)/(community) (减少IBGP连接的数量)

~~~~~



AS65012内部运行的IGP是RIP.

Step1:子AS内的IGP是RIP

整个AS123都运行IGP

Step2:启动子系统的BGP

```
router bgp 65012
```

```
bgp router-id 120.0.0.X
```

```
router bgp 65003
```

Step3:

R1/R2/R3#指定自己是属于AS123这个联邦 / Confederation

```
R1/2/3#bgp confederation identifier 123
```

Step4:在两个联邦子AS相邻的边缘路由器R1/R3上, 互相指定对方的子AS号:

```
R1(config-router)#bgp confederation peers 65003
```

```
R3(config-router)#bgp confederation peers 65012
```

Step5:在联邦中, 互指定BGP邻居

R3-R5

```
R5#neighbor 35.0.0.3 remot-as 123
```

```
R3#neighbor 35.0.0.3 remot-as 150
```

构建R2-R4之间的EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
```

```
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

提醒:

在联邦以外的EBGP邻居，他们能查看到的是联邦的大AS号，而不是子AS号。

r1-r3的联邦EBGP: (联邦子AS间有IGP)

R3 #

```
R3#neighbor 1.1.1.1 remote-as 65012
R3#neighbor 1.1.1.1 update-source loopback 3
R3#neighbor 1.1.1.1 ebgp-multihop
```

```
R1#neighbor 3.3.3.3 remote-as 65003
R1#neighbor 3.3.3.3 update-source loopback 1
R1#neighbor 3.3.3.3 ebgp-multihop
```

R1-R2的联邦IBGP

```
R1#neighbor 2.2.2.2 remote-as 65012
R1#neighbor 2.2.2.2 update-source loopback 1
```

```
R1#neighbor 1.1.1.1 remote-as 65012
R1#neighbor 1.1.1.1 update-source loopback 2
```

Step6:宣告BGP路由:

```
R5# 105.0.0.5/16
R4# 104.0.0.4/24
```

Step7:联邦EBGP和普通EBGP的异同点 (105.0.0.0 / 16)

下一跳: (不同点)

在联邦的子AS中，所有路由器看到的BGP的下一跳都是相邻大AS的边缘节点，而不是本联邦内子AS的下一跳这是区别于普通EBGP的。

联邦EBGP，不会改变BGP下一跳特性。

同步: (相同点)

联邦EBGP和普通EBGP一样，无需考察同步问题

Step8: 联邦内的IBGP: (R1-R2) (104.0.0.0 / 24)

下一跳: 同STEP7

同步: (相同点)

原因是R1从联邦IBGP R2 学到的路由，默认要检查同步但现在R1不可能通过RIP学到此BGP的路由

解决办法:

```
R1#no synchronization
```

LAB9:community/团体

(相当于一种BGP路由的标识位, 常用于标识这条BGP路由应该传播的范围)

R4#

```
增加104.0.0.0 / 24 传到R2就不在往前传
    104.0.1.0 / 24 传到R1就不在往前传
    104.0.2.0 / 24 传到R3就不在往前传
    104.0.3.0 / 24
```

并NETWORK到BGP进程中去。

Step1:通过ACL / prefix-list, 定义出 / 抓出特定的BGP路由:

```
ip prefix-list B-0 seq 5 permit 104.0.0.0/24(permit:表示匹配)
ip prefix-list B-1 seq 5 permit 104.1.0.0/24
ip prefix-list B-2 seq 5 permit 104.2.0.0/24
```

Step2:通过Route-map, 设定每类路由的COMMUNITY种类:

```
route-map T-R2 permit 10
match ip address prefix-list B-0
set community no-advertise(do not advertise to any BGP peer)
```

```
route-map T-R2 permit 20
match ip address prefix-list B-1
set community local-as (do not send outside local-AS 联邦的子AS / 小AS)
```

```
route-map T-R2 permit 30
match ip address prefix-list B-2
set community no-export (do not export to next 大AS)
```

```
route-map T-R2 permit 40
(空的route-map:Match any,Set Nothing!) (Internet)
```

Step3:在R4上, 对R2的BGP路由策略发生, “出方向”的改变:

```
R4(config-router)#neighbor 24.0.0.2 route-map T-R2 out
```

step4:每个BGP路由器, 将community发送给下一个BGP路由器

```
R4#neighbor 24.0.0.2 send-community
```

```
R2#neighbor 1.1.1.1 send-community
```

```
R1#neighbor 3.3.3.3 send-community
```

```
R1#neighbor 3.3.3.3 send-community
```

```
R2#show ip bgp community
```

```
R5#show ip bgp community
```

```
show ip bgp community no-advertise
```

```
show ip bgp community local-AS
```

```
show ip bgp community no-export
```

Step5:如果没有: “router-map T-R2 permit 40”

R4向R2通告的bgp路由只有3条:

```
R4#show ip bgp neighbor 24.0.0.2 advertisedp-routes
```

```
*> 104.0.0.0/16
```

```
*> 104.1.0.0/16
```

```
*> 104.2.0.0/16
```

BGP路由过滤的方法之一.

BGP Summarization

LAB1:非专业汇总(network命令,是不需要宣告明细路由的.)

Step1:手工生成一条需要汇总的,静态的,空接口路由:

```
R4(config)#ip route 104.0.0.0 255.255.252.0 null 0
```

Step2:将上述汇总的 / 22的路由,宣告到BGP进程中:

```
R4(config)#router bgp 140
```

```
R4(config-router)#network 104.0.0.0 mask 255.255.252.0
```

在R5上,可以查看到明细路由/汇总路由,
实际上,明细路由是不需要的:

Step3:删除原宣告到BGP的明细路由:

(使用network命令做BGP汇总,是不需要宣告明细路由的)

```
R4(config-router)#router bgp 140
```

```
R4(config-router)#no network 104.0.0.0 mask 255.255.0.0
```

```
R4(config-router)#no network 104.1.0.0 mask 255.255.0.0
```

```
R4(config-router)#no network 104.2.0.0 mask 255.255.0.0
R4(config-router)#no network 104.3.0.0 mask 255.255.0.0
```

Step4:实现BGP路由汇总:

```
R5#show ip bgp
```

```
*>104.0.0.0/22
```

Step5:如果

```
no ip route 104.0.0.0 255.255.252.0 null 0
```

∴BGP进程,在路由宣告前,
会检查路由表,如果能够查到汇总路由,
才会将此汇总路由宣告到BGP进程中.

∴

```
R4(config)#ip route 104.0.0.0 255.255.252.0 null 0(空接口)
```

也可以由IGP,自动/手工的汇总生成,

LAB2:专业汇总(推荐方法)

~~~~~

Step0:删除原表态的, / 22汇总路由

Step1:宣告准确的明细路由:

```
R4(config)#router bgp 140
R4(config-router)#network 104.0.0.0 mask 255.255.0.0
R4(config-router)#network 104.1.0.0 mask 255.255.0.0
R4(config-router)#network 104.2.0.0 mask 255.255.0.0
R4(config-router)#network 104.3.0.0 mask 255.255.0.0
```

不要使用network命令,宣告汇总路由:

```
R4(config-router)#no network 104.0.0.0 mask 255.255.252.0
```

Step2:使用aggregate-address命令,实现BGP路由的汇总:

是不需要事先配置汇总路由的.自动生成

```
R4(config-router)#aggregate-address 104.0.0.0 255.255.252.0
```

此时的所在的BGP路由器上,都接收到了明细和汇总的路由:

Step3:为了不让明细路由传播出去,调用"summary-only":

```
R4(config-router)#
aggregate-address 104.0.0.0 255.252.0.0 summary-only
```

此时的R4,抑制了所有的明细路由,只发送了汇总路由给R2,实现了BGP路由的汇总。

```
R4#show ip bgp
*> 104.0.0.0/22 (*:valid)
S> 104.0.0.0/24 (s:suppressed)
S> 104.0.1.0/24
S> 104.0.2.0/24
S> 104.0.3.0/24
```

Step4:R4的BGP进程,自动生成了一条用于汇总的空接口路由

## 1.7—BGP⑤

BGP Attributes/BGP属性

(通过BGP的属性,实现对BGP路由的选择 / 操纵)

BGP Route Selection/BGP的选路原则:

1:

The BGP forwarding table usually has multiple pathways from which to choose for each network.

在BGP路由器的BGP表中,可以存在到达某个特定目标网络的,都能满足“同步”和“下一跳可达”的,多条路径.

2:

BGP is not designed to perform load balancing:

Paths are chosen because of policy.

Paths are not chosen based upon Metric/bandwidth.

BGP默认不执行负载均衡,而是严格按照网管的策略/意志,进行BGP选路.

网管是通过BGP属性/Attribute,去表达其策略/意志的,实现BGP路由选择的控制.

对比:

而IGP是通过最小的Metric,实现路由选择的.

对于BGP,可以通过“maximum-path (1~6)”命令,实现BGP的负载均衡。

衡。

3:

The BGP selection process eliminates any multiple pathways through attrition, until a single best pathway is left.

BGP是按照BGP属性, 自上而下, 依次剔除不是最佳的路由:  
直到优选出, 到达目标风格的最佳的那一条BGP路由.

4:

That best pathway is submitted to the routing table manager process and evaluated against the methods of other routing protocols for reaching that network (administrative distance).

BGP把自己认为“最佳的”那条路由, 提交给IP路由表选择的路由, 会和别的路由协议(IGP)所获得的路由进行AD比较

5:

The routing protocol with the lowest administrative distance will be installed in the routing table.

AD最小的那个路由协议所生成的路由, 将被注入/优先进路由表, 成为最终达到该目标网络的路由.

### BGP属性的分类:

#### Well-known attributes: (公认属性)

~~~~~

Well-known **mandatory** (公认, 强制的)

Well-known **discretionary** (公认, 自决的)

Optional attributes: (可选属性)

~~~~~

Optional **transitive** attributes (可传递的, partial)

Optional **nontransitive** attributes(非传递的)

关于“传递”的属性:

1: 不论“可传递”还是“非可传递”，如果设备支持这种属性，那么都会传递。

2: 如果网络设备不支持的属性:

2-1: 对于“可传递”，那么，会被标识为“partial”，来进行继续传递。

2-2: 对于“非可传递”，那么这种属性会被丢弃，但BGP这条路由条目，还是正常传递。

Well-known, mandatory & transitive attribute:

AS path

next-hop

origin

origin: (起源代码)

IGP (i) (RIP/IGRP/EIGRP/OSPF/IS-IS) (路由表后面的那个I)

通过BGP中的network command, 宣告进BGP的.

R2#router bgp 12

network 120.1.0.0 mask 255.255.0.0

EGP (e)

Redistributed from EGP

Incomplete (?)

Redistributed from IGP or Static

BGP路由优化的前提条件:

A:Sync;B:next-Hop

BGP路由选择的主要比较因素的先后次序:

1:Weight (LAB6)

2:L-P (LAB5) 越大越好

-----

3:AS Path (LAB4) 越小越好

4:MED (LAB3)

5:EBGP VS IBGP的对比(其实是AD的对比) (LAB2)

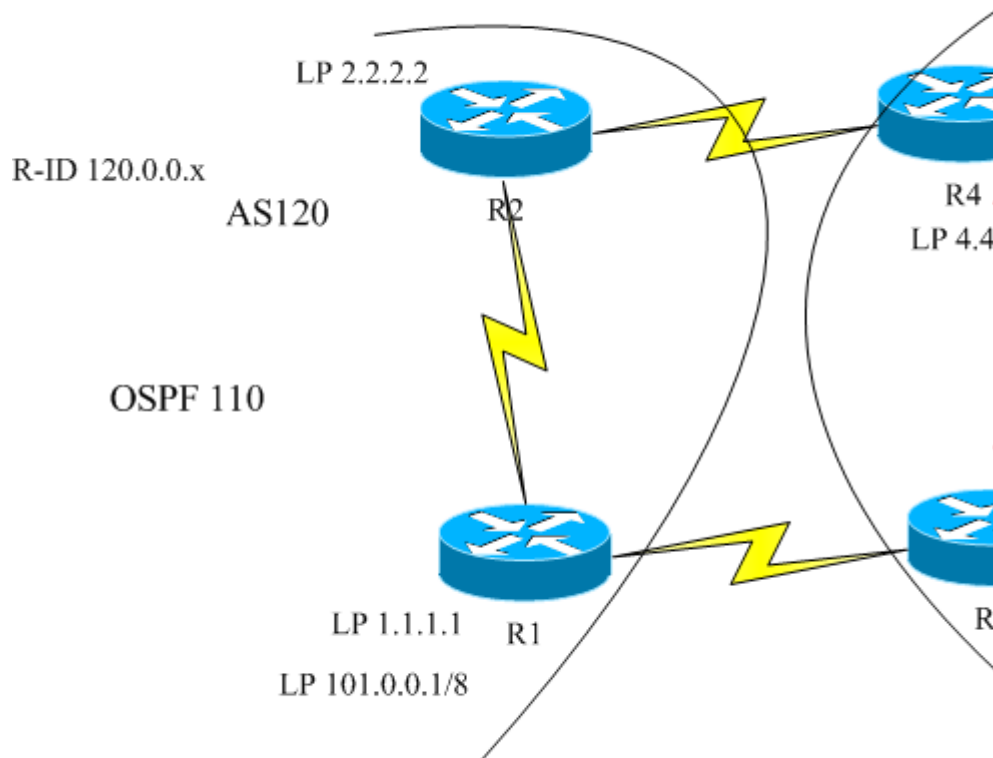
6:Closest IBGP Neighbor (LAB1)

7: lowest neighbor BGP router ID (LAB0)

**LAB0: IGP为RIP**

~~~~~

~~~~~



lowest neighbor BGP router ID

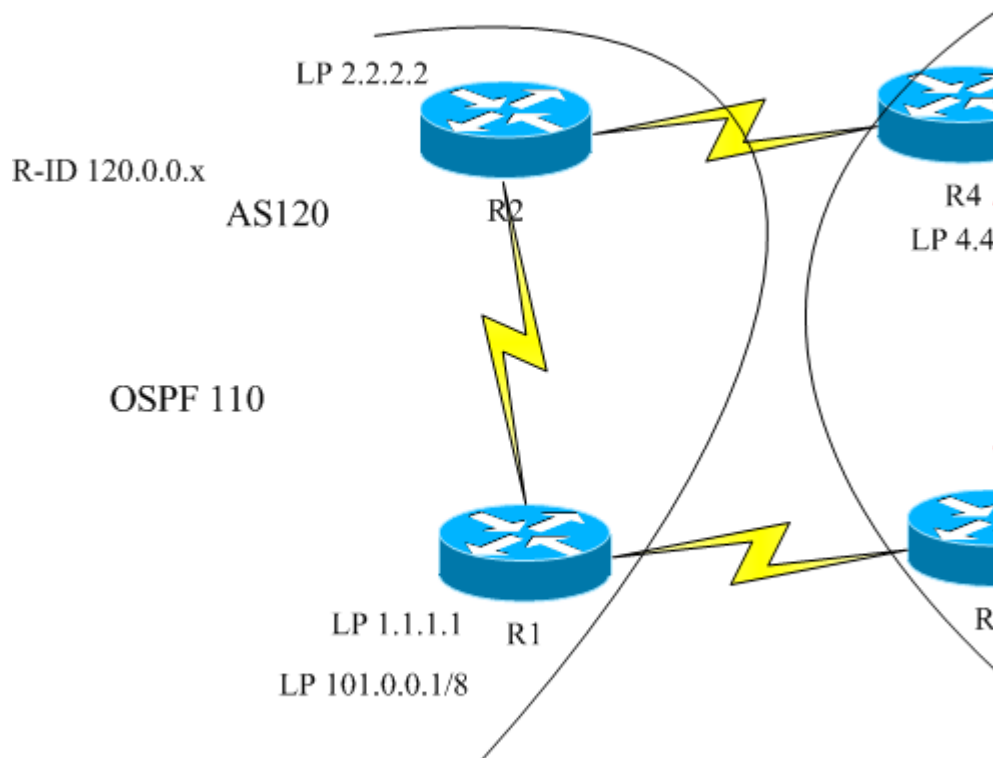
在R3上，调整BGP R-ID，可以控制R5的选路

neighbor 3.3.3.3 shutdown（临时断掉3.3.3.3的邻居）

### LAB1:Closest IGP Neighbor

~~~~~

Step1:将AS345中的IGP更改为:EIGRP



在R5上观察

```
R5#show ip route
```

```
D    3.3.3.3 [90/2297856]
```

```
D    4.4.4.4 [90/409600]
```

```
R5#show ip BGP
```

```
*>i 101.0.0.0      4.4.4.4
```

```
*i                3.3.3.3
```

```
R5#show ip route
```

```
B 101.0.0.0 [200/0] via 4.4.4.4
```

LAB2:EBGP VS IBGP

在R4上观察：

Step1:如果从R2和R3的两个方向上,都收到BGP路由：

```
R4#show ip bgp
* i 101.0.0.0      3.3.3.3 (来自IBGP)
*>                24.0.0.2(来自EBGP)
```

EBGP path over IBGP path

Step2:切断R2/R4的链路:

```
R4#
*>i 101.0.0.0      3.3.3.3
```

LAB3:MED(Multi-exit discriminator) (also called the metric)
(整个AS35都从下面链路走)

MED的特征:

- 1:MED的取值是越小越好
- 2:MED只发送给EBGP邻居,用于建议对方如何离开对方的AS,来访问本AS中的网络
- 3:MED是一种可选的,非传递的属性.
- 4:MED的默认值:0.

Step1:定义BGP路由:

```
R2(config)#ip prefix-list B-101 permit 101.0.0.0/8
```

Step2:在Route-map中,设定路由的MED值:

```
R2(config)#route-map R2-AS345
R2(config-route-map)#match ip address prefix-list B-101
R2(config-route-map)#set metric 100
```

Step3:

```
R2(config-router)#neighbor 24.0.0.4 route-map R2-AS345 out
```

```
clear ip bgp * soft out
```

Step4:

```
show ip bgp 101.0.0.0/8
```

```
R4#show ip bgp
```

network	next hop	metric
* 101.0.0.0		

```
metric
* 101.0.0.0
```

```
R5#
*> i 101.0.0.0      3.3.3.3
```

R4有两条路由可以优化，R3和R5都只有一条

LAB4:AS path

(在R1与R3之间，虚拟一个AS100:导致整个AS345都从上面链路走)

实验需求:

整个AS345的所有路由器, 都从R2走

在R1与R3之间, 虚拟一个AS100

Step1:通过Prefix-list, 定义BGP路由:

```
R3(config)#ip prefix-list B-101 permit 101.0.0.0/8
```

Step2:在Route-map中, 为此路由添加一个虚拟的AS100

```
R3#
route-map V-AS100 permit 10
match ip address prefix-list B-110
set as-path prepend 100
```

step3:在R3的BGP进程中, 对来自R1的BGP路由, 调用route-map V-AS345

```
R3(config-router)#neighbor 10.0.0.1 route-map V-AS100 in
```

Step4:

```
R3#
```

Network	Next-hop	metric	LocPrf	Weight	Path
*>i 101.0.0.0	4.4.4.4	2000	100	0	
120i					
*	13.0.0.1	150		0	
120i 100i					

LAB5:Local Preference(整个AS345都从下面链路走)

LP的特征:

1:LP的取值越大越好, 默认值是100.

LP的特征:

- 1:LP的取值越大越好, 默认值是100.
- 2:LP只发送给本AS内的IBGP邻居, 用于建议他们如何离开本AS, 去访问外网.
- 3:LP的属性是公认的, 自决的, 只会在本AS传递的. !!!

R3#

Step1:通过Prefix-list, 定义BGP路由:

Step2:作用Route-map, 为特定路由设定LP值:

```
route-map V-AS100 permit 10
match ip address prefix-list B-101
set as-path prepend 100
set local-preference 130
```

Step3:在R3的BGP进程中, 对来自R1对路由, 调用route-map V-AS100:

```
R3(config-router)#neighbor 13.0.0.1 route-map V-AS100 in
```

Step4:观察:

R3#

Network	next Hop	Metric	LocPrf	Weight
Path				

```
*>101.0.0.0
```

R4#

```
*>i 101.0.0.0
```

R5#

```
*>i 101.0.0.0
```

LAB6:Weight

~~~~~

Weight的特征:

- 1:cisco私有的属性.
- 2:默认值为0, 越大越好.
- 3:不发送给任何BGP邻居, 只影响本路由器的选路.

方法1:

```
R4:(config-router)#neighbor 24.0.0.2 weight 10
```

方法2:

```
R4:(config-router)#network 101.0.0.0 mask 255.0.0.0 weight 10
```

方法3:

3-1:定义路由

3-2:

```
route-map WEI permit 10
```

```
match ip address prefix-list B-110
```

```
set weight 10
```

3-3:

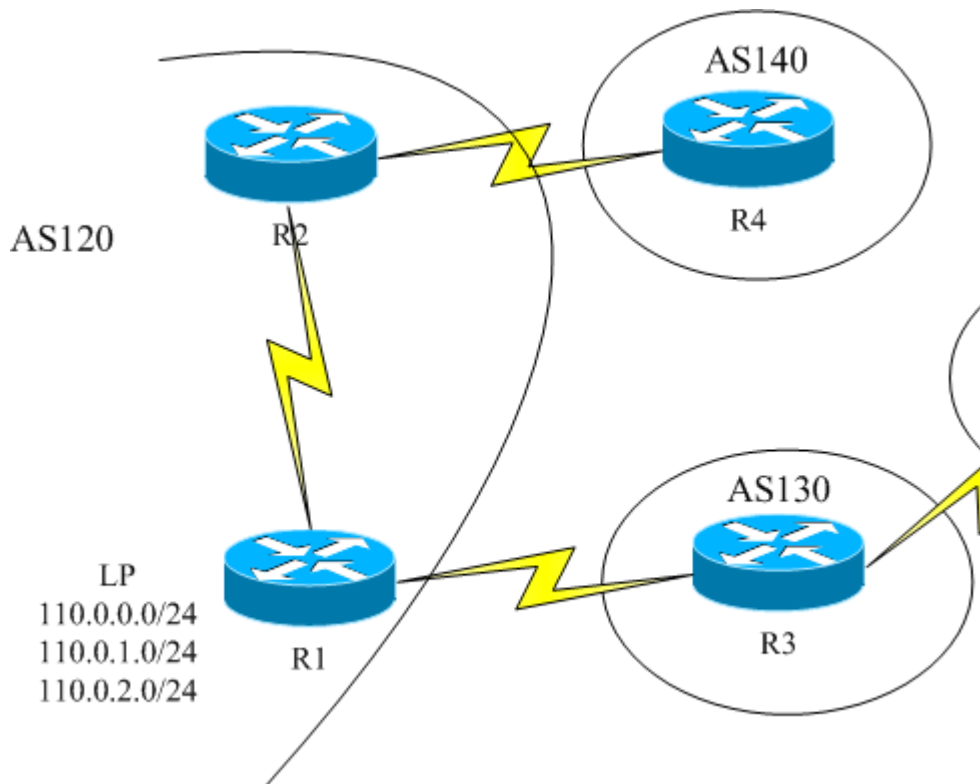
```
R4#(config-router)#neighbor 24.0.0.2 route-map WEI in
```

## 1.7—BGP⑥

BGP中的路由控制/过滤:

LAB1:Distribute-list调用ACL(较落后)

~~~~~



Step1:通过ACL定义BGP路由:

如果不打算建立route-map

那么Access-list里面的

(Deny:不允许)

(permit:允许)

```
R1(config)#access-list 1 permit 110.0.1.0 (0.0.0.0)
```

(ACL的最后,有隐患的Deny Any.)

Step2:通过distribute-list/分布列表

```
R1(config)#router bgp 120
```

```
R1(config-router)#neighbor 13.0.0.3 distribute-list 1 out
```

Step3:使用ACL的反掩码,定义有一定特征的路由:

3-1: 定义有偶数特征的路由:

```
R2(config)#access-list 2 permit 110.0.0.0 0.0.254.0 (偶数)
```

```
R2(config-router)#neighbor 1.1.1.1 distribute-list 2 in
```

```
R1(config)# access-list 1 permit 110.0.1.0 0.0.254.0 (奇数)
```

Redistribute:重分布,将一种路由注入到另外一种路由中.

软清:

R3#clear ip bgp * soft out (使用在:出方向的策略发生改变时)

Step3:观察R3对R1发送了哪些BGP路由:

R3#show ip bgp neighbor 13.0.0.1 advertised-routes

提醒: 在更新的IOS版本中, 出现了 “Distribute-list+prefix-list”

LAB2:通过neighbor命令, 直接调用prefix-list (较先进, 简洁)

~~~~~

Step1:通过前缀列表/prefix-list, 定义需要控制的BGP路由:

(Deny:不允许)

(Permit:允许)

R1(config)#ip prefix-list T-R3 seq 5 permit 110.0.1.0/24

R1(config)#ip prefix-list T-R3 seq 10 permit 110.0.3.0/24

(IP prefix-list 的最后, 有隐含的Deny Any)

如果Prefix-list直接被BGP进程调用, 则应该严格匹配路由表的路由长度!!

IP prefix-list的最后, 有隐患的Deny Any.

Step2:在R1对R3的出方向, 调用prefix-list T-R3, 进行路由过滤:

R1(config-router)#neighbor 13.0.0.3 prefix-list T-R3 out

在BGP进程中, 不能对同一个邻居同时调用access-list和prefix-list

如果只打上:

neighbor 13.0.0.1 prefix-list maple out

而没有建立一个名为maple的prefix-list, 则全部的路由都放行, 即邻居把所有的路由都可以收到!!

### LAB3:neighbor+route-map+ACL

(使用route-map, 功能强大, 实现复杂功能)

**Step1:**通过ACL定义BGP路由:

(permit:匹配)

```
R1(config)#access-list 1 permit 110.1.0.0
```

```
R1(config)#access-list 3 permit 110.3.0.0
```

**Step2:**通过route-map, 调用ACL,

让Route-map决定是否允许这些定义好的路由穿过本AS:

(permit:允许), (Deny:不允许)!!!

R1#

```
route-map Send-R3 permit 10
```

```
match ip address 1 (ACL 1)
```

后面 没有空的Route-map: Deny Any

**Step3:**

```
neighbor 13.0.0.3 route-map Send-R3 out
```

假如增加:

```
route-map Send-R3 permit 20
```

(Match any, set nothing!)

(是否有这一句空的route-map, 决定了:

route-map中, 没有明细定义的路由, 将是否可以传到别的AS中)

(115.3.3.0/24)

### LAB4:neighbor+Route-map+Prefix-list

**Step1:**通过前缀列表/prefix-list, 定义需要控制的BGP路由:

(permit:匹配)

```
ip prefix-list B-130 seq 5 permit 110.0.1.0/24
```

```
ip prefix-list B-130 seq 10 permit 110.0.3.0/24
```

**Step2:**通过Route-map, 决定是否允许特定路由传递给对方:  
(permit:允许), (Deny:不允许)!~!!!

```
route-map SEND-R2 permit 10
match ip address prefix-list B-130
```

**Step3:**R1对邻居R3调用Route-map:

```
R1(config-router)#neighbor 13.0.0.3 route-map SEND-R3 out
```

如果写错了ROUTE-MAP的名字则不会发送任何路由出去

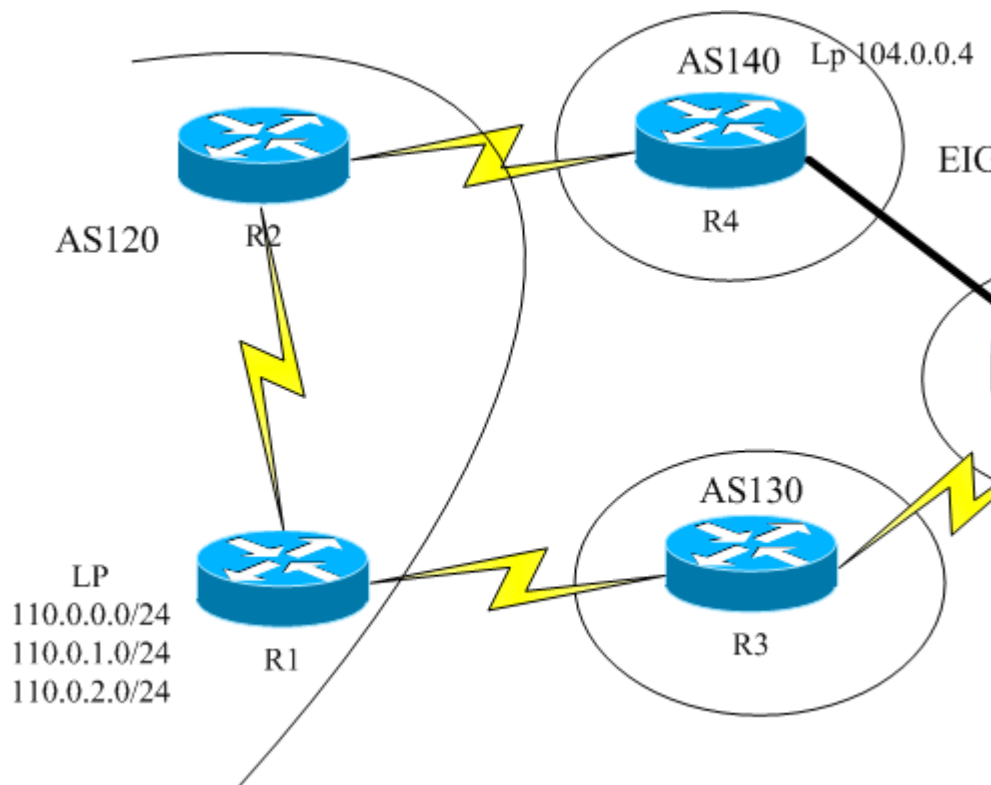
对于上述4个实验, 如果在R3上做in方向的路由过滤, 则需要硬清除.  
Clear ip bgp \* (会Reset TCP 的连接)

### LAB5:BGP BackDoor/后门

用于EBGP与IGP之间的AD竞选时, 人为的让IGP优先进入路由表的一种操作.

20 120

人为的让IGP优先进入路由表的一种BGP路由操纵



R4/R5之间运行IGP:EIGRP



```
R4(config)#ip as-path access-list 130 permit .*
```

### LAB7:BGP Damping

背景知识:

BGP dampening:牺牲收敛性, 强调稳定性。

Damp的惩罚值: 1000 (固定值)

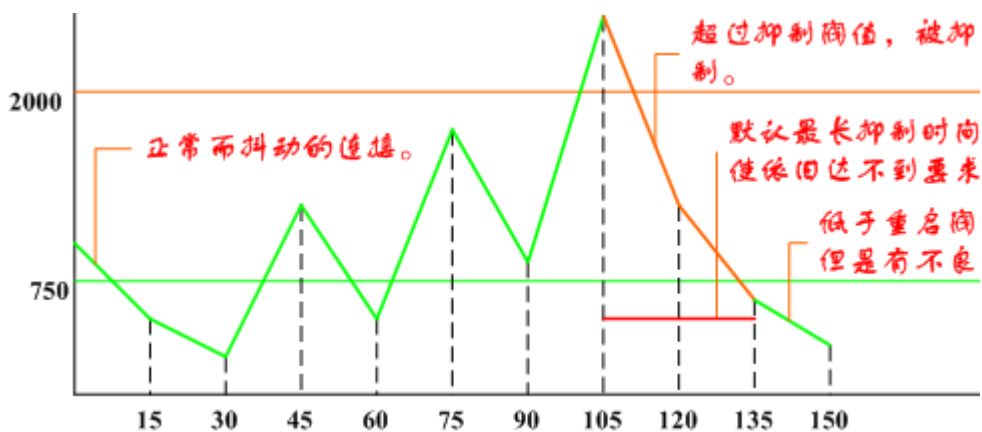
15:(分钟) half-life /半衰期 half-life time for the penalty

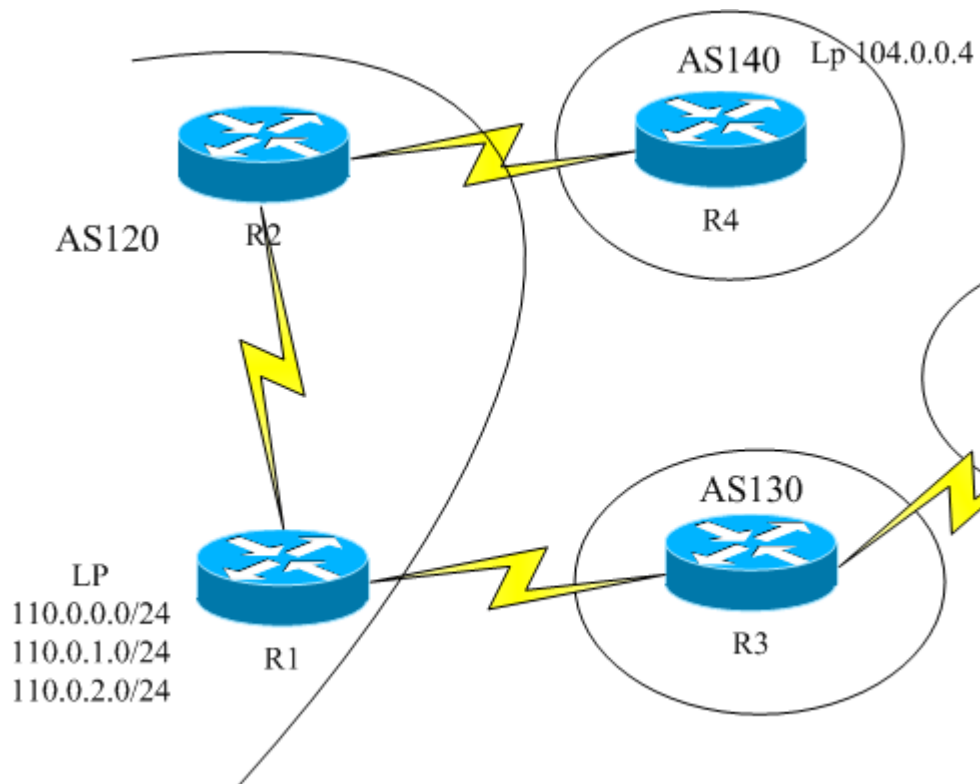
750:reusing /重新启用路由的阈值 penalty to start reusing a route

2000:suppressing /开始进行抑制的阈值 penalty to start suppressing a route

60:(分钟) Maximum duration to suppress (设置为半衰期的4倍)

Maximum duration to suppress a stable route





Step1:通过前缀列表, 定义需要进行Damp的BGP路由:

```
R3(config)#ip prefix-list B-105 permit 105.0.0.0/8
```

Step2:创建Route-map, 对特定路由进行Damping, 并且可以设定Damping的参数.

```
route-map DAMP permit 10
match ip address prefix-list B-10
set dampening 15 750 2000 60
```

Step3:在BGP进程中调用Route-map, 进行BGP dampening

```
router bgp 130
Bgp dampening route-map DAMP
```

旧版本:

```
R3#show ip bgp flap-statistics
```

```
R3#show ip bgp dampened-paths
```

新版本:

```
R3#show ip bgp dampening flap-statistics
```

R3#show ip bgp dampening dampened-paths

clear ip bgp dampening (恢复正常)

## 组播配置

### 新建信息项目...

组播路由器的基本配置

step 1: 开启路由器的组播转发功能:

```
rl(config)#ip multicast-routing
```

step 2: 开启接口的组播转发功能, 任意的组播路由协议

dense-mode:

```
rl(config-if)#ip pim dense-mode
```

sparse-mode:

```
rl(config-if)#ip pim sparse-mode
```

RP的配置

模拟一个接收者

```
pc5(config-if)#ip igmp join-group 224.1.1.1
```

IGMP

```
show ip igmp groups
```

```
show ip igmp interface f0/0
```

### Auto-RP (自动RP)

MA——Mapping Agents (映射代理)

C-RP——Candidate RPs (候选RP)

Step 1: 开启路由器的组播转发功能

```
R5(config)#ip multicast-routing
```

step 2: 开启接口的组播转发功能, 一般用在sparse-dense-mode

```
R5(config-fi)#ip pim sparse-dense-mode
```

step 3: 要指定一台设备的接口成为代理

```
R5(config)#ip pim send-rp-discovery loopback 0 scope 16
```

step 4: 要指定一台设备的接口成为候选RP

```
R5(config)#ip pim send-rp-announce loopback 0 scope 16
```

注意: loopback 0 部分IOS需要运行PIM

BSR (Bootstrap Router 自举路由器)

BSR——Bootstrap Router (自举路由器)

C-RP——Candidate RPs (候选RP)

Step 1: 开启路由器的组播转发功能

```
R5(config)#ip multicast-routing
```

step 2:开启接口的组播转发功能, sparse-mode或sparse-dense-mode

```
R5(config-if)#ip pim sparse-mode
```

```
R5(config-if)#ip pim sparse-dense-mode
```

step 3:要指定一台设备的接口成为自举路由器

```
R5(config)#ip pim bsr-candidate loopback 0
```

step 4:要指定一台设备的接口成为候选RP

```
R5(config)#ip pim rp-candidate loopback 0
```

## pppoe拨号

### pppoe

PPPOE(ppp over ethernet) to ADSL Modem

step 1:VPDN (12.2(13)T or later都是默认配置)

step2:

```
interface e 0/0
```

```
ip add 192.168.1.1 255.255.255.0
```

```
ip nat inside
```

step3:

```
interface e 0/1
```

```
no ip add
```

```
pppoe enable
```

```
pppoe-client dial-pool-number 1(将e0/1放入pool中)
```

step4:

```
int dialer1(虚拟接口)
```

```
ip add negotiated(协商)
```

```
encapsulation ppp
```

ppp authentication pap (callin 只对外面发起的认证相应, 配不配都行)

```
ppp pap sent-username AAA password BBB
```

```
"ppp authentication chap callin
```

```
ppp chap hostname AAA
```

```
ppp chap password BBB"
```

```
ip mtu 1492(pppoe帧头默认8个字节)
```

```
ip nat outside
```

```
ip nat outside
```

dialer pool 1(去pool中, 调用e0/1口, 为什么要弄dailer pool而不是dialer直接调用e0/1?这是因为便于管理)

step5:NAT (全局配置)

access-list 7 permit 192.168.1.0 0.0.0.255 (定义需要进行NAT的内网用户)

```
ip nat inside source list 7 interface dialer1 overload
```

step6: (全局配置)

```
dialer-list 5 protocol ip permit
```

or:

(access-list 10 permit host PC1的IP地址

```
dialer-list 5 protocol ip list 10)
```

dialer-group 5 (何种的数据流量, 才会激活dial接口去进行ADSL拨号)

dialer-group调用dialer-list, dialer-list调用access-list

```
ip route 0.0.0.0 0.0.0.0 dialer1
```

```
end
```

“如果router里面有adsl模块

step VPDN (no need)

```
vpdn enable
```

```
no vpdn logging
```

```
vpdn-group pppoe
```

```
request-dialin
```

```
protoc....
```

step2一样

step3:ADSL interface(ADSL)

```
int atm0
```

```
no shut
```

step4:ATM sub-interface (子接口)

```
int atm 0.1 point-to-point
```

pvc 1/1(VPI/VCI)取决于ISP配置

```
pppoe-client dial-pool-number 1
```

(将这个虚拟的atm 0.1子接口放入pool 1中)

深圳:

VPI/VCI (8/35)  
深圳的是: 8/32

```
step 5:if dial
int dialer 1
dialer pool 1 (去dialer poo 1中, 调用刚放入的atm 0.1子接口)
ip add negotiated (与ISP协商IP地址)
ip mtu 1492
ip nat outside
encap ppp
```

认证与上述一样  
! the isp instructs you about the type of authentication to use.

```
step6:nat:
access-list 1 permit 192.168.1.0 0.0.0.255
ip nat inside source list 1 interface dialer1 overload
ip route 0.0.0.0 0.0.0.0 dialer 1
end
"
```

E1线路知识要点:  
在中国/欧洲使用E1, (欧标)

在北美/日本使用T1, (美标)

- 1、一条E1是带宽为: 2.048M的链路, 用PCM编码(脉冲编码调制)。
- 2、一个E1的帧长为256个bit, 分为32个时隙, 一个时隙为8个bit.  
(Time slots:时隙)
- 3、每秒有8K个E1的帧通过E1接口, 既 $8k \times 256 = 2048k\text{bps}$
- 4、因为每个时隙在E1帧中占8bit,  
因为一个时隙的带宽是 $8\text{bps} \times 8k = 64k\text{bps}$ ,  
即一条E1中含有32个64K信道/时隙。

E1帧结构:  
E1有不成帧, 成帧, 成复帧三种方式:  
1: 在不成帧的E1中(透明模式):  
所有32个时隙都可以用于传输有效数据。

2: 在成帧的E1中:  
第0时隙用于传输帧同步数据,  
其余31个时隙可以用于传输有效数据。

3: 在成复帧的E1中:  
第0时隙用于传输帧同步数据,  
第16时隙用于传输信令,  
只有1到15, 17到31, 共30个时隙可以用于传输有效数据。

E1的2种接口(G. 703):  
同轴电缆: BNC头, 园头, 阻抗: 非平衡的75 ohm  
双绞线: RJ-45头, 阻抗: 平衡的120 ohm

工程中常见的E1连接方法:

相对便宜:

用2611等的广域网接口卡(WAN WIC 1T (DB-60)/2T(Smart serial)), 经V. 35-G. 703协议转换器接E1线。

(用于不分时隙, 完整2M线路使用)

相对较贵:

E1卡: 目前DDN的2M速率线路通常是经HDSL线路拉至用户侧。

E1可由传输设备出的光纤拉至用户侧的光端机提供E1服务。

三。使用E1由三种用法:

1: 透明传输的专线:

将整个2M用作一条链路, 如DDN 2M;

2: CE1:

2M用作若干N\*64K及其组合, 如128K, 256K等;

3: E1最本来的用法:

在用作语音交换机/程控交换(PSDN网络种)的数字中继线(TRUNK)时, 这也是E1最本来的用法。

以下例子为:

E1连接3条64K专线, 帧类型为NO-CRC4, 非平衡链路, 路由器具体设置如下:

hostname shanxi (背景颜色代表匹配与调用)

step1:controller e1 0/1 (进入E1控制器)

step2:framing NO-CRC4

linecode hdb3 (ISP给)

step3:建立逻辑通道组与时隙的映射:

channel-group 1 timeslots 1

channel-group 2 timeslots 2

channel-group 3 timeslots 3

exit

step4:

interface serial0/1:1

ip add 222.119.96.1 255.255.255.252

```
interface serial0/1:2  
ip add 222.119.96.5 255.255.255.252
```

```
interface serial0/1:3  
ip add 222.119.96.9 255.255.255.252
```

step5:

```
ip route 139.20.40.0 255.255.255.0 serial 0/1:1  
ip route 139.20.41.0 255.255.255.0 serial 0/1:2  
ip route 139.20.42.0 255.255.255.0 serial 0/1:3
```

el                    - circuit switch      电路交换——比较老  
frame relay - packet switching      包交换——新

## **mpls vpn**

### MPLS VPN

#### Drawbacks of Traditional IP Routing

传统IP路由的缺点:

1: 每个路由器都要依赖路由协议(OSPF、BGP、RIP或Static), 作出独立

独立的转发策略（读取IP包中的目标IP，查询路由表，根据路由表中的出接口，进行重新封装，转发），这个过程是非常消耗CPU资源的。

### MPLS

- 1: MPLS is a new forwarding mechanism in which packets are forwarded base on labels.（而不是目标IP地址）
- 2: labels may correspond to IP destination network.

（ATM技术的创新：

- 1: 摒弃了繁琐的路由查找，改为简单快速的标签交换
  - 2: 将具有全局意义的路由表改为只有本地意义的标签表
- 这些都可以大大提高一台路由器的转发能力。）

MPLS充分吸取了ATM的精华，但也同时认识到IP网络无法取代，所以成为IP的承载层，但为了与一般的链路层做区别，把自己定义成2.5层。

FEC(Forwarding equivalence class)转发等价类  
(ccnp/sw fec:fast ether-channel GEC)——两个是不一样的。

标签 (label)

是一个比较短的，定长的 (32bit) (ip包不定长，最少20位)

7: LSP (label switching path) 标签交换通道  
一个FEC的数据流，在MPLS域中所经过的路径就是LSP。

8: LDP (label distribution protocol)  
负责label的生成(分配), 交换(分发), 保存于LIB中, 然后根据IGP所生成的RT/路由表, 将最佳的label, 择优录取到LFIB中。

MPLS比传统IP路由快在不用拆IP包头信息，直接通过LABEL交换就可以到目标网络。

LDP有两标准：

- 1: cisco私有标准：TDP(T:tag)
- 2: 业界标准：LDP

（如Trunk技术有两种标准：ISL/802.1Q）

MPLS has two major components:

- 1:Control plane--exchanges layer 3 routing information and labels
- 2:Data plane--forwards packets based on labels.

control plane contains complex mechanisms to exchange routing information, such as OSPF, EIGRP, IS-IS and GBP.

```
1:ip packet->FIB->LFIB->label packet
2:label packet->LFIB->label packet
3:label packet->LFIB->FIB->ip packet
4:label packet->pop (次末跳弹出) ->FIB->ip packet
```

9: 次末跳弹出

PHP optimizes MPLS performance(one less LFIB lookup).

最后的出口路由器:

The pop or **implicit null label(隐含空)** uses a reserved value when being advertised to a neighbor.

#### LAB1:MPLS解决BGP黑洞

```

r4      ->      r2      ->      r1      ->      r3      ->r5
as140    (      as123    )
as150
```

r3与r3建BPG邻居, r1不跑bgp。

step1:网络拓扑规划, IP子网规划。

物理链路的建立测试 (L1/2)

step2:

ISP中的MPLS CORE的IGP: OSPF

IGP的数据库: (同一IGP中, 根据Metric选路) (L3)

```
show ip ospf nei
```

```
show ip ospf database
```

```
RT:sh ip route
```

step3:

```
r2-r3 r5-r3 r2-r4 IPV4 EBGp,
```

```
r5: router bgp 150
```

```
bgp router-id 150.0.0.5
```

```
neighbor 35.0.0.5 remote-as 150
```

```
r3:neighbor 2.2.2.2 update-source lo 0
```

```
neighbor 2.2.2.2 next-hop-self
```

step4:宣告GBP路由:

r1成为BGP黑洞:

```
r4#ping 105.5.5.5 source 104.4.4.4
```

step5:在MPLS域中的所有MPLS路由器: R1/R2/R3:

运行MPLS LDP, 并且生成/交换LABEL (分配/分发)

```
r1#
```

```
r1#
5-1全局命令:
r1:
ip cef (全部接口运行CEF快速交换)
mpls label protocol ldp (选定LDP协议为LDP/TDP)
mpls label range 100 199 (r2用200-299, 可以容易区分标签是谁发的)
mpls ldp router-id loopback 0 (要配置成/32位的环回口)
```

5-2在需要运行LDP的接口中:

```
conf t
interface s 0
mpls ip (激活LDP的运行)
sh mpls int (察看运行MPLS/LDP的接口)
LDP: show mpls ldp nei
LIB: sh mpls ldp bindings ?除了两条EBGP路由, 其他路由都有标签
FIB(CEF):sh ip cef
LFIB:sh mpls forwarding-table
```

对于BGP路由:

只要拥有相同下一跳(3.3.3.3)的BGP路由

(105.5.0.0/16, 105.6.0.0/16), 都是一个FEC (不管地点有多少跳路由),

既然是一个FEC, 那就只分配一条标签, 只会为BGP的下一跳分配标签。

MPLS不会为BGP路由分配标签: (而是借用BGP的下一跳: 3.3.3.3, 因为3.3.3.3路由在BGP配置前就已经有了, 所以早就有3.3.3.3的标签了)

所以MPLS CORE中的路由器, 尽管没有BGP路由:

step6:观察通过标签转发, 解决黑洞

```
R2/R3#
debug ip packet
debug mpls packets
```

```
r4#ping 105.5.5.5 source 104.4.4.4!!!!
```

```
r2#sh ip bgp
```

```
*>i 105.5.0.0/16      3.3.3.3
```

```
r2#show ip route
```

```
0    3.3.3.3[110/3] via 12.0.0.1
```

```
r2#show ip cef 105.5.5.5
```

```
.....3.3.3.3/32,
```

```
.....12.0.0.1, tags imposed:{101}
```

LAB2:

```
.....12.0.0.1, tags imposed:{101}
```

LAB2:

MPLS VPN:

前几步一样

ISP的PE(ISP边缘设备)上使用VRF, 去获取/感知, 客户CE(客户端边缘设备)的路由

VRF三步曲之一: 建VRF:

```
ip vrf s1
```

```
rd 25:1 (近似理解为VRF虚拟路由器的router-id)
```

```
router-target export 100:1
```

VRF三步曲之二: 将连接用户的接口划入VRF:

```
int fa 0/0.35
```

```
ip vrf forwarding s1
```

```
ip add 35.0.0.3 255.0.0.0
```

(注意, 一旦将接口划入VRF中, 原IP地址会消失, 需要重新配置)

检查:

```
r3#sh ip route vrf s1
```

```
c 35.0.0.0.0 ...
```

(此接口的路由, 会从全局路由表, 转入VRF S1中的路由表)

测试:

```
r3#ping vrf s1 35.0.0.5!!!!
```

vrf三步曲之三: 在VRF中, 与CE运行IGP, 获取的CE路由:

PE:R3#

```
router rip
```

```
ver 2
```

```
address-family ipv4 vrf s1
```

```
net 35.0.0.0
```

```
no auto-summary
```

```
ver 2
```

```
exit-address-family
```

ce:r5

```
router rip
```

```
ver 2
```

```
net 35.0.0.0
```

```
net 192.168.5.0
```

```
no auto
```

检查:

```
r3#sh ip route vrf s1 rip (查看是否感知到客户的路由)
```

```
r 192.168.5.0/24 [120/1] via 35.0.0.5, fa 0/0.35
```

step 5: pe r2 使用vrf, igp:ospf, 获取/感知ce r4的路由:

之一: 建VRF:

```
ip vrf s2
rd 25:6
route-target export 100:6
```

之二: 接口划入VRF:

```
int fa 0/0.24
ip vrf forwarding s2
ip add 24.0.0.2 255.255.255.0
```

之三: 获取CE路由:

用静态感知用户的VPN路由:

```
r4(config)#ip route 0.0.0.0 0.0.0.0 24.0.0.2
r2(config)#ip route vrf s2 192.168.4.0 255.255.255.0 24.0.0.4
```

测试:

```
r2#ping vrf s2 192.168.4.4!!!!
```

step 6:在MPLS域中, 使用MP-BGP, 构建MPLS VPNv4 通道:

配置:

```
r2#conf t
router bgp 25
bpg router-id 25.0.0.2
no auto
no sy
no bgp default ipv4-unicast (启动MP-BGP)
```

```
nei 3.3.3.3 remote-as 25
nei 3.3.3.3 update-source lo 0
```

```
address-family vpnv4
nei 3.3.3.3 activate (激活vpnv4邻居关系)
(neighbor 3.3.3.3 send-community extended)
```

检查:

```
r2#sh ip bgp summary
sh ip bgp vpnv4 all summary
2.2.2.2          .....      0
```

step 7:VPN路由传输

VPN路由传输三步曲之一:

7-1: VRF S1中的重分布: (VRF中要作RIP到VPNv4 BGP的重分布)

7-1: VRF S1中的重分布: (VRF中要作RIP到VPNv4 BGP的重分布)

配置:

```
r3#  
router bgp 25  
address-family ipv4 vrf s1  
redistribute rip
```

检查:

r3#show ip bgp (IPv4的BGP, 当前是没有路由的)

```
r3#sh ip bgp vpnv4 all
```

```
*> 35.0.0.0/24    0.0.0.0
```

```
*> 192.168.5.0    35.0.0.5
```

r3#sh ip bgp vpnv4 all nei 2.2.2.2 ad (确认R3已经成功将VPN路由传给R2)

```
r2#sh ip bgp vpnv4 all
```

```
???
```

```
r3#sh ip bgp vpnv4 all 192.168.5.0
```

```
.....
```

VPN路由传输三步曲之二:

7-2:RT import控制导入的VPN路由:

```
ip vrf s2  
route-target import 100:1
```

```
r2#sh ip bgp vpnv4 all
```

```
sh ip bgp vpnv4 vrf v2
```

## GRE over IPsec

理论:

IPsec的不足之处:

- 1: 只能支持单播的IP数据包，别的协议数据包都不支持。
- 2: 只支持tunnel IP only.
- 3: 无法直接支持站点(site)之间的动态路由协议。
- 4: IPsec does not tunnel non-IP (非IP) protocols; GRE does.

does。

为了解决以上IPsec的不足，GRE(Generic Routing Encapsulation)可以很好地解决此类问题：

Gre is good at tunneling:

1:support multiprotocol

2:provides virtual point-to-point connectivity,  
allowing routing protocols to be used.

3:在logical hub-and-spoke topology中，支持：

tunnel mode gre multipoint(多点)

tunnel mode gre ip (默认)

LAB2: GRE Over ipsec s2s vnp

step 0:

r1:lo 1

ip add 1.1.1.1 255.255.255.0

r4#ping 1.1.1.1!!!!

r4#ping 35.0.0.5!!!!

step1:构建GRE的Tunnel:

配置:

r4# int tunnel 1

ip add 10.1.12.1 255.255.255.252

tunnel source 24.0.0.4

tunnel destination 35.0.0.5

(接口UP起来，但不代表tunnel已经配好)

r5#int tunnel 1

ip add 10.1.12.2 255.255.255.252

...

检查: sh int tunnel 1

r4#ip route 10.1.2.0 255.255.255.0 10.1.12.2(tunnel接口)

r5#ip route 10.1.1.0 255.255.255.0 10.1.12.1

然后用扩展ping测试

如果不跑静态路由

no ip route ip route 10.1.2.0 255.255.255.0 10.1.12.2

router rip

ver 2

no auto

net 10.0.0.0

以下，使用IPsec，把tunnel保护起来

# IPsec

IPsec is an IETF standard that employs cryptographic mechanisms on the network layer:

- 1: Authentication of every IP packet
- 2: Verification of data integrity for each packet
- 3: Confidentiality of packet payload

IPsec is the only standard Layer 3 technology that provides:

- 1) Confidentiality
- 2) Data integrity
- 3) Authentication
- 4) Replay detection

IPsec Protocols

IPsec uses three main protocols to create a security framework:

- Internet Key Exchange (IKE):
  - Provides framework for negotiation of security parameters
  - Establishment of authenticated keys
- Encapsulating Security Payload (ESP):
  - Provides framework for encrypting, authenticating, and securing of data
- Authentication Header (AH):
  - Provides framework for authenticating and securing of data

IPsec uses three main protocols:

Internet Key Exchange(IKE)

用来把密码安全地发送到对方及变更密码

Encapsulating Security Payload(ESP)

提供加密，认证和数据的安全性

Authentication Header(AH)

提供认证和安全性，现在已经不用了，因为不能加

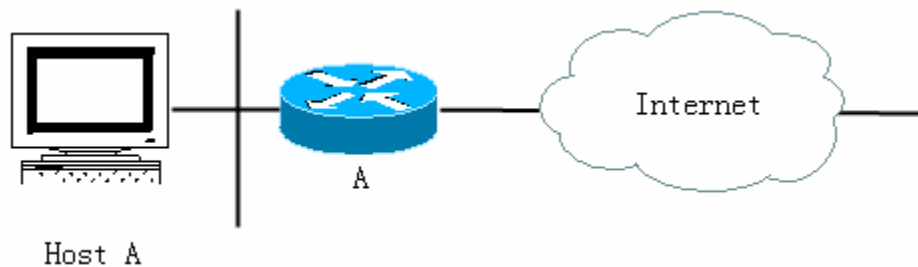
密

Peer authentication methods:

- Username and password

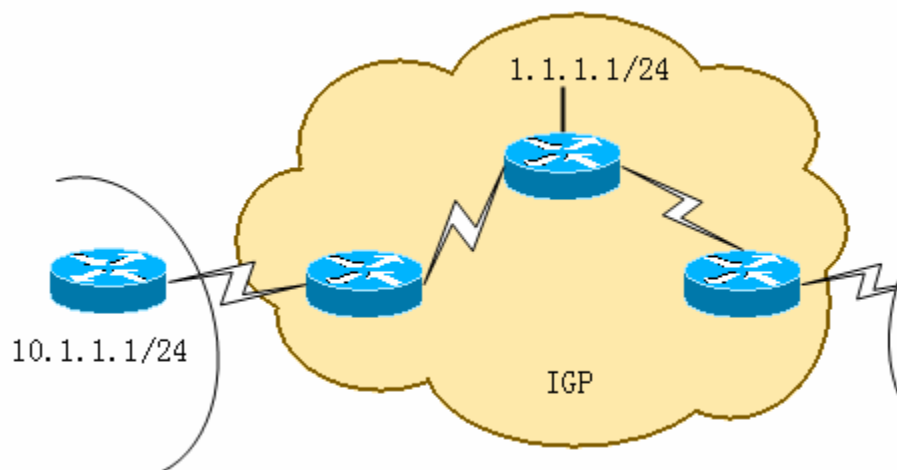
- OTP (one Time password)
- Biometric
- Preshared keys
- Digital certificates

#### Site-to Site IPsec VPN Operation



- 1 : Host A send interesting traffic to Host B
- 2 : Routers A and B negotiate an IKE Phase 1 session.
- 3 : Routers A and B negotiate an IKE Phase 2 session.
- 4 : Information is exchanged via the IPsec tunnel.
- 5 : The IPsec tunnel is terminated.

#### LAB:



Step1:构建模拟的ISP/Internet:

R1/R2/R3:运行EIGRP:

```
router eigrp 90
```

```
no auto-summary
```

Step 2：为用户网络指定默认路由：

```
R4(config)#ip route 0.0.0.0 0.0.0.0 24.0.0.2
```

```
R5(config)#ip route 0.0.0.0 0.0.0.0 35.0.0.3
```

```
R4#ping 1.1.1.1 !!!!!(可以访问公网)
```

```
R5#ping 35.0.0.5 !!!!!(可以访问对方路由器)
```

Step3:在R1/R2上，NAT：

3-1：ACL定义内网用户：

```
R4#
```

排除VPN流量，不作NAT：

```
access-list 101 deny ip 10.1.1.0 0.0.0.255 10.1.2.0  
0.0.0.255
```

上网的流量， 要作NAT：

```
access-list 101 permit ip 10.1.1.0 0.0.0.255 any
```

3-2：

```
R4#
```

```
int loopback 0
```

```
ip nat inside
```

```
int s0
```

```
ip nat outside
```

```
ex
```

R5相同方法：

3-3:基于外口的NAT OVERLOAD

```
ip nat inside source list 101 int s 0 overload
```

测试：

```
R5#ping 1.1.1.1 so 5.5.5.5
```

开始构建IPsec VPN，

Step4:定义感兴趣流量, 要求对称：

```
R4(config)#ip access-list extended vpn
```

```
permit ip 10.1.1.0 0.0.0.255 10.1.2.0 0.0.0.255
```

```
R5(config)#ip access-list extended vpn
```

```
permit ip 10.1.2.0 0.0.0.255 10.1.1.0 0.0.0.255
```

Step5:第一阶段: (完成IKE SA)关联

5-1: 确定IKE Policy: (两边路由器完全一致)

```
R4 / R5(config)#crypto isakmp policy 1
R4 / R5(config-isakmp)#authentication pre-share      (认证
方式: 预先定义的密码)
R4 / R5(config-isakmp)#hash md5                      (完整性校验)
R4 / R5(config-isakmp)#encryption des                (加密方式)
R4 / R5(config-isakmp)#group 2                      (HASH:1024 bit)
//R4 / R5(config-isakmp)#lifetime 86400             (有效时间为一天)
```

检查:

```
R4#show crypto isakmp policy
```

5-2: 确定Pre-share-key: (密码, 以及对方的地址)

```
R4(config)#crypto isakmp key cisco address 35.0.0.5
(cisco就是密码)
R5(config)#crypto isakmp key cisco address 24.0.0.4
```

Step6:第二阶段: (IPsec SA)

6-1: 配置Transform-set:

```
R4 / R5(config)#crypto ipsec transform-set MYSET Esp-des
esp-md5-hmac
```

加密

完整性

```
R4/R5(cfg-crypto-trans)#mode tunnel
```

6-2: 定义crypto map:

```
R4(config)#crypto map MYMAP 10 ipsec-isakmp
R4(config-crypto-map)#match address vpn(上面定义的ACL)
R4(config-crypto-map)#set peer 35.0.0.5
R4(config-crypto-map)#set transform-set MYSET
```

R5的配置相反

6-3: 在外口(公网地址)调用crypto map

```
R4(config)#interface e0/0.35
R4(config-subif)#crypto map MYMAP
```

搞定!

R4#clear crypto sa(安全关联) (全清)

R4#ping 10.1.2.2 source 10.1.1.1 .! ! ! !

R2#show crypto isakmp sa (第一阶段)

R2#show crypto ipsec sa (第二阶段)

R1#show crypto engine connections active(快速察看VPN连接状态)

测试: OK

全部搞定

GRE over IPsec

GRE-----Generic Routing Encapsulation

OSI Layer 3 tunneling protocol:

- 1) uses IP for transport
- 2) Uses an additional header to support any other OSI Layer 3 protocol as payload (e.g., IP ,IPX,AppleTalk)

IPsec的不足之处:

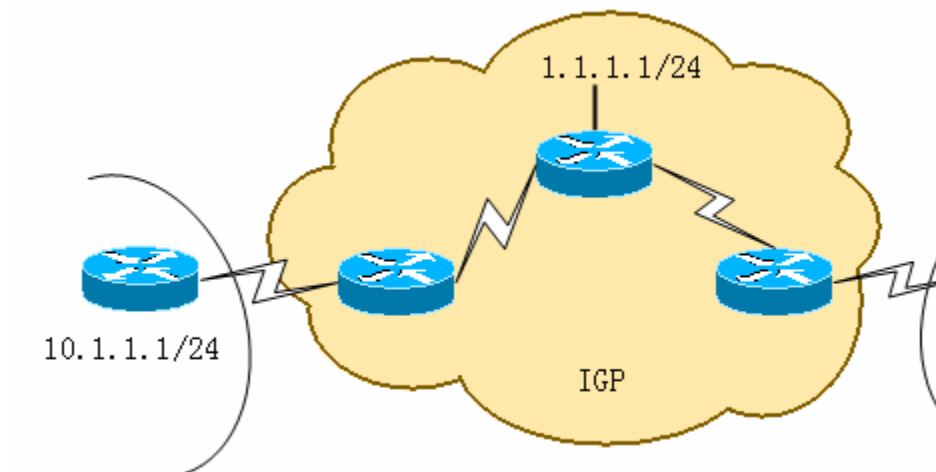
- 1:只能支持单播IP数据包, 别的协议的数据都不能通过.
- 2:只能支持tunnel IP only
- 3:无法直接支持站点 (site)之间的动态路由协议.
- 4:IPsec does not tunnel non-IP protocols;GRE does.

为了解决以上的IPsec的不足, GRE(Generic Routing Encapsulation)可以很好地解决些类问题:

GRE is good at tunneling:

- 1:Support Multiprotocol
- 2:Provides virtual point-to-point connectivity, allowing routing protocols to be used.
- 3:在Logical hub-and-spoke topology中, 支持: " tunnel mode gre multipoint"(多点)

## LAB2:GRE Over IPsec S2S VPN



Step0:CE间已经实现了公网接口的互通.

测试:

```
R4#ping 35.0.0.5 !!!!!
```

Step1:构建GRE的Tunnel:

配置:

R4#

```
interface tunnel 1
 tunnel source 24.0.0.4
 tunnel destination 35.0.0.5
 ip address 192.168.45.4 255.255.255.0
```

R5#

```
interface tunnel 1
 tunnel source 35.0.0.5
 tunnel destination 24.0.0.4
 ip address 192.168.45.5 255.255.255.0
```

检查:

```
R4/R5#show interface tunnel 1
```

```
R4/R5#show ip route c
```

测试:

```
R4#ping 192.168.45.5 !!!!!
```

Step2:在两站点（R4/R5）之间跑路由协议或静态路由：（RIP）

配置：

R4#

```
router rip
  ver 2
  no auto-summary
  net 192.168.1.0
  net 192.168.45.0
```

R5#

```
router rip
  ver 2
  no auto-summary
  net 192.168.2.0
  net 192.168.45.0
```

检查：

R4/R5#

```
show ip route r
```

**GRE VPN**已经成功构建，但GRE不安全，还要做Over IPsec

更改上面第二阶段：

R4/R5#

```
ip access-list extended GRE
  permit gre host source host destination (any any)
```

R4 / R5(config-crypto-map)#match address GRE

```
r4#crypto ip sec transform-set MYSET esp-des esp-md5-hmac
exit
#crypto map mymap 10 ipsec-isakam
r4(config-crypto-map)#match add gre
set peer 35.0.0.5
set transform-set MYSET
exit
调用和第一个实验一样。
```

## VOIP配置

LAB:Voip

step1:  
IGP互通

step2:配置本地电话号码和voice port之间的映射:

```
r1(config)#  
dial-peer voice 119 pots (传统电话机)  
destination-pattern 119
```

```
port 1/0/1
```

```
r2(config)#  
dial-peer voice 110 pots (传统电话机)  
destination-pattern 110  
port 1/0/0
```

step3:配置远端电话号码和ip之间的映射:

```
r1#  
dial-peer voice 110 (对方名称) voip (对方是voip过来的)  
destination-pattern 110  
session target ipv4:12.0.0.2
```

```
r2#  
dial-peer voice 119 (对方名称) voip (对方是voip过来的)  
destination-pattern 119  
session target ipv4:12.0.0.1  
可以打电话了
```

show voice port summary可以查看接口

## DHCP + 重分布

```
router(config)#ip dhcp pool [pool name]
```

-define the pool name

```
router(config-dhcp)#import all
```

```
router(config-dhcp)#network [network address] [subnet mask]
```

-specifies the network and subnet mask of the pool

```
router(config-dhcp)#default-router [host address]
```

-specifies the default router for the pool to use(config the gateway)

```
router(config-dhcp)#domain-name xxx
```

-define the domain name

```
router(config-dhcp)#dns-server 202.96.134.133 202.96.128.68
```

-define the dns server ip address

```
router(config-dhcp)#lease in(无限租期)
```

```
router(config-dhcp)#lease 30 (租期30天, 默认7天)
```

```
router(config)#ip dhcp excluded-address 10.1.1.100
```

```
router(config)#ip dhcp excluded-address 10.1.1.200整一段不分配
```

DHCP中继代理

step1:

```
router(config)#int e0
```

```
router(config-if)#ip helper-address 13.1.1.3 客户路由器上配
```

```
router(config)#ip forward-protocol udp 68
```

pc site:

```
pc site:
pc2(config)#int e 0
pc2(config-if)#ip address dhcp 路由器上模拟PC通过DHCP自动获取地址
```

LAB:重分布的基础实验:

step1:

```
r1(config)#router ospf 110
```

```
re(config-router)#redistribute eigrp 100 subnets
```

把EIGRP 100的路由重分布到OSPF 110

OSPF默认只能重分布主类路由，必须打上subnets才能把非主类的路由重分布进来

```
r1(config-router)#redistribute eigrp 100 subnets metric-type 1
```

指定外部路由的类型，默认是类型2；指定类型1

外部路由类型1，计算整个路径cost

外部路由类型2，只计算重分布时的cost(default-metric)

step3:

```
r1(config-router)#redistribute eigrp 100 metric 300 subnets
```

修改OSPF重分布时的cost(default-metric), 默认是20。

## 1.2 EIGRP 重分布

step1:

```
r1(config)#router eigrp 100
```

```
r1(config-router)#redistribute ospf 110 metric 1544 2000 255 1 1500
```

把OSPF 110的路由重分布到EIGRP 100

EIGRP重分布别的动态路由协议的时候，必须带上metric指定带宽、延时、可靠性、负载、MTU

## 1.3 RIP重分布

step1:

```
r1(config)#router rip
```

```
r1(config-router)#redistribute ospf 110 metric 2
```

把OSPF 110的路由重分布到RIP

RIP重分布别的动态路由协议的时候，必须带上metric指定跳数，默认是infinity无限大（16跳）

```
r1(config)#route-map conn( 写一个route-map去匹配loopback 0)
match int lo 0
```

```
r1(config)#route-map conn( 写一个route-map去匹配loopback 0)
match int lo 0
redistribute connected route-map conn subnets
```

TAG:

```
4      ->2      ->1      ->3      ->5
      ->tag50      <-tag100
```

```
r2(config)#router rip
r2(config-router)#ver 2
r2(config-router)#net 24.1.1.0
r2(config-router)#exit
r2(config)#route-map tag50 permit 10
r2(config-route-map)#set tag 50
r2(config-route-map)#exit
r2(config)#router ospf 100
r2(config-router)#redistribute rip route-map tag50 subnets
```

```
r3(config)#route-map tag100 (默认permit 10)
r3(config-route-map)#set tag 100
r3(config-route-map)#exit
r3(config)#router ospf 100
r3(config-router)#re
r3(config-router)#redistribute eigrp 100 subnets rou
r3(config-router)#redistribute eigrp 100 subnets route-map
tag100
```

```
r2(config)#route-map deny_tag100 deny 10
r2(config-route-map)#match tag 100
r2(config-route-map)#exit
r2(config)#router rip
r2(config-router)#re
r2(config-router)#redistribute osp
r2(config-router)#redistribute ospf 100 met
r2(config-router)#redistribute ospf 100 metric 2 rout
r2(config-router)#redistribute ospf 100 metric 2 route-map
deny_tag100
r3(config)#route-map deny_tag100 permit 20
```

```
r3(config)#route-map deny_tag50 deny 10
r3(config-route-map)#match tag 50
r3(config-route-map)#route-map deny_tag50 permit 20
r3(config-route-map)#exit
r3(config)#router eigrp 90
```

```
r3(config-route-map)#exit
r3(config)#router eigrp 90
r3(config-router)#re
r3(config-router)#redistribute ospf 100 metric 1544 2000 255 1
1500
r3(config-router)#redistribute ospf 100 metric 1544 2000 255 1
1500 rou
r3(config-router)#$ metric 1544 2000 255 1 1500 route-map
deny_tag50
r3(config-router)#
```

### 反掩码计算

access-list/eigrp等 反掩码计算

原则：地址部分，相同的照写，不同的写“0”

反掩码部分，相同的写“0”，不同的写“1”

如

172. 16. 1. 0

172. 16. 2. 0

172. 16. 3. 0

172. 16. 8. 0

172. 16. 9. 0

172. 16. 10. 0

172. 16. 11. 0

172. 16. 0000 0000

0000 0001  
0000 0011  
0000 1000  
0000 1001  
0000 1010  
0000 1011

地址部分0000 0000 ->172.16.0.0

掩码部分0000 1011 ->0.0.11.0

反掩码用了三个bit，所以有8条，因为多匹配了一条，所以要先在访问列表上做deny那条。

如果用route-map调用，可以写成

```
access-list 100 permit 172.16.0.0 0.0.3.0 255.255.255.192  
0.0.0.0
```

如果在ip access-group调用则后面部分为目标地址，如果用route-map则后面为前面地址部分的掩码部分匹配

## IPV6

ipv4用nat可以减缓地址不够用，但是同时破坏端到端模型已经带来更大的运算量。美国拥有一半地址。

所以一些地址不够用的城市迫切需要转换到IPV6。

ipv4下一跳为对方接口地址；ipv6则为对方link-local地址。

### IPv6

- 1:Larger address space
- 2:Simpler header
- 3:Mobility and security
- 4:Transition richness(V4 to V6)

r4 -> r2 -> r1 -> r3 -> r5

r2/1/3跑ipv4 igp, r2-r3做tunnel（可以用loopback如果有冗余链路），r4/2/3/5跑ipv6

IPv4地址有32bits, 以“点分十进制”标示. (192.168.1.1)

IPv6地址有128bits, 以16进制数表示. (冒号, 分16进制数)

2C3D:0000:030F:0000:0000:0000:8A60:130B

每一个16进制数表示4bits,

每4位16进制数, 组成一个字段, 一个字段表示16bits.

IPv6地址共有8个字段.

每个字段之间用冒号分割.

IPv6地址的简写:

对于全零字段":0000:", 可以简写为":0:"

2031:0:030F:0:0:0:8A60:130B

对于多个连零字段":0000:0000:0000:", 可以简写为":::"

2031:0:030F::8A60:130B

注意:

::在一个IPv6地址中, 只能出现一次.

FF01:0:0:0:0:0:1>>>FF01::1

0:0:0:0:0:0:1>>>>>::1

0:0:0:0:0:0:0>>>>>::

每个字段前的0, 可以省略:

2031:0:30F::8A60:130B

2:Simpler header:

~~~~~

IPv4的包头/IPv4 Header

每个word有32bits(4 Bytes), 一个IP包头最少有5 word, 整个包头最少有20个Bytes, 而结构较为复杂, ip地址部分占8Bytes.

∴IPv4的路由转发效率不高.

IPv6的包头/IPv6 Header

每个word有32bits(4 Bytes), 一个IPv6包头最少有10 word, 整个包头固定有40个Bytes, 但是源IP/目标IP已经占32 Bytes,

∴包头结构较为简单, 大大提高了IPv6网络中的路由器的路由/转发效率.

在IPv6网络中, 数据包的类型:

支持:

Unicast/单播, (one 2 one)

Multicast/组播, (one 2 many)

Anycast/任意播, (one 2 nearest) 多个服务器用同一个IP地址, DNS解析一样, 主机会找最近的服务器地址去访问.

而不再使用Broadcast/广播. ——无发广播ARP查询, 代替者为NDP (邻居

而不再使用Broadcast/广播.——无发广播ARP查询, 代替者为NDP (邻居发现协议), 使用组播地址。

IPv6执行更为严格的地址分配机制:

IPv6支持路由汇总, 提高路由效率/Address Aggregation.

IPv6支持自动配置, Auto-Configuration (替代了DHCP) / 也可以用DHCPv6
自动配置EUI-64 (扩展唯一标识) 产生:
支持自动配置的IPv6主机,
可以使用从路由器接收到的路由前缀,
加上自己的数据链路层地址 (MAC地址),
形成了自己的全球唯一的IPv6地址.
如MAC: 00 90 27 17 FC 0F
第一位00:000000UG 当u=1 (0000001G) 的时候为全球唯一, 为0的时候表示本地唯一 (MAC地址相反)。
做法, MAC地址前24比特和后24比特分开, 中间插入16个比特FF FE
00 90 27 FF FE 17 FC 0F
当从一个MAC地址变成IPV6地址的时候U要从0变1; 如果是1要变0。
0000 0000 -> 0000 0010即02

自动获取地址:

```
首先在隔壁路由器配: ipv6 unicast-routing
int e 1
ipv6 enable
ipv6 add autoconfig
```

IPv6支持Renumbrering:

允许用户在完成新的路由前缀网络过渡之前,
继续使用原来的旧的路由前缀一段时间,
以平滑完成网络升级/过渡.

网络协议/被路由协议: IPv4

寻路协议/路由协议: RIP/IGRP/EIGRP/OSPF (v2)/ISIS/BGP

网络协议/被路由协议: IPv6

寻路协议/路由协议: RIPng/OSPF (v3)/ISIS/MP-BGP/EIGRP (For IPv6)

IPv4 to IPv6 Transition:

IPv4 to IPv6 Transition:
Dual stack
tunnels
Translation

LAB1:在IPv4的海洋中, 将IPv6的孤岛实现网络互通.
(通过Tunnel实现)

~~~~~  
Step1:构建IPv4网络(RIP V2), (R1/R2/R3#)

```
R1#  
router rip  
  version 2  
  no auto-summary  
  network 12.0.0.0  
  network 13.0.0.0
```

检查:  
show ip route rip

Step2:在R3/R5 ; R2/R4间, 构建IPv6网络:

```
R4/5(config)#  
  no ip routing(关闭IPv4的路由能力)  
  ipv6 unicast-routing (启动IPv6的单播路由能力)
```

双栈/Dual Stack路由器: (IPv4/IPv6)

```
R2/3(config)#  
  ip routing(默认)  
  ipv6 unicast-routing
```

在IPv6网络中, 配置IPv6的链路地址, 测试链路:

```
R3(config)#in s 1  
R3(config-if)#ipv6 address 2007:0012:0034:0035::3/64  
R3(config-if)#clock rate 2000000
```

```
R3(config-if)#no shutdown
```

链路测试:

```
R3#ping 2007:12:34:35::5 !!!!!
```

```
R3#show ipv6 route connected
```

为了实现IPv6孤岛的互通,  
在双栈路由器R2/R3上,  
构建能承载IPv6通信流量的Tunnel: (Step3/4)

Step3:在R2/R3, 为Tunnel新增一个环回口,  
注意此环回口是要宣告到IPv4网络的IGP (RIPv2) 中的:

```
R2#
```

```
interface Loopback2
 ip address 2.2.2.2 255.255.255.255
```

```
R3#
```

```
interface Loopback3
 ip address 3.3.3.3 255.255.255.255
```

将新建的环回口, 宣告到RIP中:

配置:

```
R2#
```

```
router rip
 network 2.0.0.0 (RIP没有用来控制哪些接口运行RIP的反掩码)
```

```
router eigrp 90
 network 2.0.0.0 (0.255.255.255)
 network 2.2.2.2 (0.0.0.0)
```

检查:show ip route rip

```
测试:R2#ping 3.3.3.3 source 2.2.2.2 !!!!!!!!!!!!!!!
```

Step4:R2/R3, 都以自己的环回口作为Tunnel的源地址,  
以对方的环回口作为Tunnel的目标地址,  
来构建Tunnel接口(Tunnel可以理解为虚拟链路/隧道):

配置:

```
R2#
```

```
interface Tunnel 1
 tunnel source 2.2.2.2
```

```
tunnel destination 3.3.3.3
ipv6 address 2007:12:34:23::2/64
```

```
R3#
interface Tunnel 1
    tunnel source 3.3.3.3
    tunnel destination 2.2.2.2
    ipv6 address 2007:12:34:23::3/64
```

测试:

```
R2#ping 2007:12:34:23::3!!!!!!!
```

```
R2#show ipv6 route connected
R2#show interfaces tunnel 1
```

至此, IPv4的网络中的路由器R1, 都忽略了. (透明了)

```
R3#show ipv6 route connected
C   2007:12:34:23::/64 [0/0]
    via ::, Tunnel1
C   2007:12:34:35::/64 [0/0]
    via ::, Serial1
```

```
R3#show interfaces tunnel 1
R3#show ipv6 interface tunnel 1
```

Step5:

在所有的IPv6网络接口中, 启动IPv6的路由协议OSPFv3  
(提醒: 包括R2/R3间的Tunnel接口)

```
5-1:R2/R3#
interface Tunnel 1
    ipv6 enable (启用ipv6功能, 如果设置IPV6地址, 那么不用写这个自
自动启用功能)
    tunnel mode ipv6ip(IPv6 over IP encapsulation)
```

```
5-2:在所有的IPv6路由器上 (R2/3/4/5), 都启动IPv6的路由协议:"OSPF
v3"
conf t
```

```
ipv6 router ospf 16
router-id 16.0.0.X(格式还是保留为,好像IPv4地址的格式) 注意: 如果router-id不指定, OSPF进程将不会启用。
```

5-3:在所有需要运行IPv6的OSPF接口中,激活IPv6的OSPF的运行:

```
R3(config)#
in s1
  ipv6 ospf 16 area 0

in tunnel 1
  ipv6 ospf 16 area 0
  (Tunnel接口:跑IPv6的OSPF v3, 不跑IPv4的RIP)
```

5-4:检查:

```
R2#show ipv6 ospf interface (brief)
R2#show ipv6 ospf neighbor
R2#show ipv6 ospf database
R2#show ipv6 route ospf
```

Step6:如果IPv6网络运行的路由协议是:RIP for IPv6  
从Step1~4,都是相同的.

```
conf t
no IPV6 ROUTER OSpf 16 (STEP5)
```

启动RIP for v6的进程:

```
conf t
ipv6 router rip RIP-V6
```

在所有的IPv6接口中,激活RIP for v6的运行  
ipv6 rip RIP-V6 enable (Tunnel也跑RIP for IPv6)

```
R5#show ipv6 route rip
```

LAB1:在完全的IPv6网络中, 运行RIP: (2006:12:34:\*\*:\*/64)

~~~~~

Step0:按图配置IPv6网络地址, 测试接口:

R5#

```
interface Serial1
```

```
no ip address(无IPv4地址)
```

```
ipv6 address 2006:12:34:35::5/64
```

Step1:将全网的IP网络都升级为IPv6:

```
no ip routing (关闭IPv4的路由能力)
```

```
ipv6 unicast-routing(启动IPv6的路由能力)
```

Step2:在所有的路由器上, 都启动IPv6的路由协议:"RIP"

```
R2(config)#ipv6 router rip RIP-V6
```

(RIP-V6是路由进程的名字, tag, label)

Step3:在所有需要运行IPv6的RIP, 的接口中, 激活RIP协议的运行:

```
R1(config)#in s0
```

```
R1(config-if)#ipv6 rip RIP-V6 enable
```

Step4:检查IPv6的路由:

```
R5#show ipv6 route
```

```
R5#show ipv6 route rip
```

LAB2:在完全的IPv6网络中, 运行OSPF v3:

~~~~~

Step0:

```
conf t
```

```
no ipv6 router rip RIP-V6(删除IPv6的RIP)
```

Step1:在所有的路由器上, 都启动IPv6的路由协议:"OSPF"

```
R1(config)#ipv6 router ospf 160
```

```
R1(config-rtr)#router-id 100.0.0.1
```

Step2:在需要运行OSPF的接口中, 激活IPv6的OSPF的运行:

```
R3(config)#in s0
```

```
R3(config-if)#ipv6 ospf 160 area 0
```

```
R3(config-if)#ipv6 ospf 160 area 0
```

Step3:

```
R2#show ipv6 ospf interface
```

```
R2#show ipv6 ospf neighbor
```

```
R2#show ipv6 ospf database
```

```
R2#show ipv6 route ospf
```

LAB4: IPv6网络在FR链路上的运行:

~~~~~

Step1: 在R1上, 配置两端口的帧中继交换机:

```
R1(config)#no ipv6 unicast-routing
```

```
R1(config)#no ip routing
```

```
R1(config)#frame-relay switching
```

```
R1(config)#hostname FR-SW
```

Step2: 在接口中, 进行FR的基本配置:

```
interface S 0/1
```

```
    encapsulation frame-relay
```

```
    frame-relay lmi-type cisco
```

```
    frame-relay intf-type dce
```

Step3: 配置FR的路由:

R2 TO R3 的PVC:

```
FR-SW(config-if-S0)#FRAME-relay route 203 interface serial 1  
302
```

R3 TO R2 的PVC:

```
FR-SW(config-if-S1)#FRAME-relay route 302 interface serial 0  
203
```

Step4: IPv6的FR接口 (FR的自动反向ARP, 在IPv6中是不生效的.)

```
R3#
```

```
interface Serial0
```

```
    encapsulation frame-relay
```

```
    ipv6 address 2005:12:34:23::3/64
```

手工映射:

在对方路由器上, 察看link-local address, 在FR映射中也要对此地址进行映射

```
R3#
```

```
show ipv6 interface serial 0
  IPv6 is enabled, link-local address is
FE80::210:7BFF:FE3B:48E7
```

```
R2#
frame-relay map ipv6 2006:12:34:23::3 203 broadcast
frame-relay map ipv6 FE80::210:7BFF:FE3B:48E7 203 broadcast
```

```
R3#
frame-relay map ipv6 2006:12:34:23::2 302 broadcast
frame-relay map ipv6 FE80::260:5CFF:FEF4:DA24 302 broadcast
```

Step5:在接口中, 激活OSPF的运行:

```
R1/R3(config-if-s0)#ipv6 ospf 160 area 0
```

当前OSPF邻居无法建立, 原因是当前的OSPF网络运行模式是:NBMA

```
R1#show ipv6 ospf interface serial 0
```

Step6:更改接口的OSPF的运行模式为P2P:

```
R1/R3(config-if-s0)#ipv6 ospf network point-to-point
```

flash 格式化

```
Router(boot)#show ver
Cisco Internetwork Operating System Software
IOS (tm) 3000 Bootstrap Software (IGS-BOOT-R), Version 11.0(10c),
RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Fri 27-Dec-96 17:33 by loreilly
Image text-base: 0x01010000, data-base: 0x00001000
```

Compiled Fri 27-Dec-96 17:33 by loreilly
Image text-base: 0x01010000, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(10c), SOFTWARE

Router uptime is 0 minutes
System restarted by power-on
Running default software

cisco 2500 (68030) processor (revision N) with 14336K/2048K bytes of memory.
Processor board ID 06018697, with hardware revision 00000000
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash partition 1 (Read/Write)
8192K bytes of processor board System flash partition 2 (Read/Write)

Configuration register is 0x2142

Router(boot)#erase flash

Partition	Size	Used	Free	Bank-Size	State	Copy Mode
1	8192K	0K	8192K	8192K	Read/Write	Direct
2	8192K	512K	7679K	8192K	Read/Write	Direct

[Type ?<no> for partition directory; ? for full directory; q to abort]

Which partition? [default = 1] 2

System flash directory, partition 2:

File	Length	Name/status
1	525100	c2500-is-l.122-15T14bindNNu
		.SVP.MSG [invalid checksum]

[525164 bytes used, 7863444 available, 8388608 total]

Erase flash device, partition 2? [confirm]

Are you sure? [yes/no]: y

Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erased

Router(boot)#conf t

Router(boot)(config)#partition flash 1

Router(boot)#show version

Cisco Internetwork Operating System Software
IOS (tm) 3000 Bootstrap Software (IGS-BOOT-R), Version 11.0(10c),
RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1996 by cisco Systems, Inc.
Compiled Fri 27-Dec-96 17:33 by loreilly
Image text-base: 0x01010000, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(10c), SOFTWARE

ROM: System Bootstrap, Version 11.0(10c), SOFTWARE

Router uptime is 5 minutes
System restarted by power-on
Running default software

cisco 2500 (68030) processor (revision N) with 14336K/2048K bytes of memory.
Processor board ID 06018697, with hardware revision 00000000
X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interfaces.
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2142

Switching

2.0—交换的基础知识

2.0—交换的基础知识

交换机的工作原理：

交换机可以隔离冲突域 (collision domain) → 交换机的每个接口对应一个冲突域，不能隔离广播域；

L2帧里面带有目标MAC地址和源MAC地址，目标MAC排在源MAC地址的前面；

交换机中有类似路由表的CAM表，查看命令为#show mac-address-table；源MAC地址用来构建CAM表；

交换机有三个动作：泛洪 (flood)、过滤 (filter)、转发 (forward)；

交换机的三种转发模式：

直接转发 (Cut-Through)：检查到目标MAC后直接转发；延迟比较低但无法保证准确性；

存储转发 (Store and Forward)：把帧完整收完后再转发；准确性有保障但延迟比较大；

无碎片转发 (Fragment Free)：检查64字节后转发；是折中方案；达到了最大的物理速度的转发被称为线速度转发。

网桥和交换机区别：

网桥一般用软件实现而交换机用硬件实现；网桥只有两个端口而交换机端口比较多。

三层交换机：

三层交换机:

等于路由器+交换机。

文件处理:

用write命令则保存在flash里面; 如果要把配置全部删除, 则要删除:
vlan.dat和config.text;

在路由器中如果不小心把IOS删除了, 在重新启动的时候按住
ctrl+break进入ROM MONITOR模式, 然后在NVRAM中进行配置。

交换机的三大功能:

MAC地址的学习 (MAC Address Learning) → 交换机有一张MAC地址表, 记录了交换机端口和与端口相连的主机的MAC地址 (也就是数据帧的源MAC地址和目标MAC地址) 的匹配 (Mapping) 关系;

数据帧的转发/过滤机制 (Forward/Filter Decision) → 交换机是根据数据帧的目标MAC地址进行转发的: 如果目标MAC地址是已知的单播 (Unicast) 地址则使用目标MAC匹配的接口转发; 如果目标MAC地址是未知的单播地址则向除了入口的所有接口广播, 找到后同上 (第一次Ping会丢一个包, 其实包送到了但是回包被交换机用来建立MAC表映射了, 所以路由器不能接收到回包, 因此路由器认为丢了一个包; 以后Ping就不显示丢包了因为MAC表已经建立); 如果目标MAC地址匹配的是数据入口则丢弃该包; 如果目标MAC地址是组播/广播 (Multicast/Broadcast) 地址则向除了入口的所有接口广播;

避免环路 (Loop Avoidance) → 在冗余网路中避免网络回路 (STP)。

交换的广播地址:

FFFF.FFFF.FFFF.FFFF (就是所有位数全是1, 与Mask同)。

以太口的种类:

Ethernet: 10MPS的接口;

FastEthernet: 100MPS的接口;

GiFastEthernet: 1000MPS的接口。

ARP (Address Resolution Protocol):

ARP用于解析对方的IP与MAC地址的对应关系, 目的是省略配网关的步骤来减少工作量。

ARP的欺骗: the client uses ARP (Address Resolution Protocol) To get the destination it wants to reach, and a router will respond to the ARP request with its own MAC address (详见LAB1)。

LAB1: ARP的协商:

STEP1: ARP的协商:

STEP1: ARP的协商:

通过SW#sh cdp nei来确认链路的通达;

打开Debug: #debug arp, Ping未Ping过也就是未建立MAC映射的目标:

1: creating incomplete entry for IP address 192.168.1.2

interface ethernet 0 ;

2: sent req src (source) 192.168.1.1 xxxx.yyyy.zzzz,
dst (destination) 192.168.1.2 _____._____._____

ethernet 0 ;

3: rcvd rcp src 192.168.1.2 xxxx.yyyy.zzzz,
dst 192.168.1.1 ethernet 0 ;

在PC1上sh arp得: 映射表…… (AGE: 老化时间, 相当于Hold Time, 默认为10min) ;

STEP2: 代理ARP (最大特点: 欺骗) :

R2上sh ip in e 0得: Proxy ARP is enabled ;

首先指定默认网关: (c)#ip default-gateway 192.168.1.2, 然后再访问外网时无需做ARP查询;

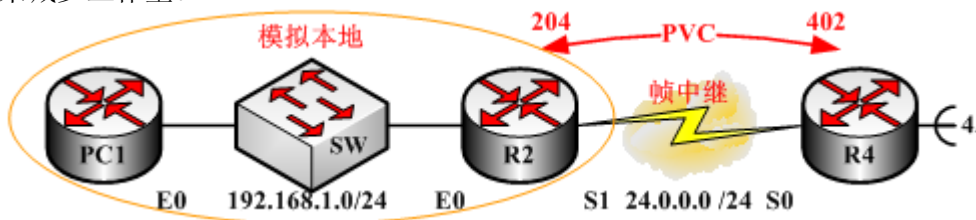
做代理ARP: 首先删除默认网关, 使用默认启动的代理ARP:

1: creating incomplete entry for IP address 4.4.4.4 interface ethernet 0 ;

2: sent req src 192.168.1.1 xxxx.yyyy.zzzz,
dst 4.4.4.4 _____._____._____ ethernet 0 ;

3: rcvd rcp src 4.4.4.4 xxxx.yyyy.zzzz (这是R2的MAC!),
dst 192.168.1.1 ethernet 0 ;

ARP用于解析对方的IP与MAC地址的对应关系, 目的是省略配网关的步骤来减少工作量。



2.1—VLAN/TRUNK/VTP

2.1—VLAN/TRUNK/VTP

注意: 配置VLAN时要退出VLAN配置模式才会执行

如果VLAN被删除或者shutdown, 那么属于这个vlan的接口将被阻塞 (灯一直是橙色, 变不了绿色)。

排错的时候如果发现端口一直变不到绿色, 可以用sh vlan brief查看该端口是否划到了一个被

shutdown或者删除的vlan中。

接口类型:

1. Access (访问端口)

一个Access Port, 只能属于一个Access Vlan

```
int fa 0/1
```

```
switchport mode access
```

```
switchport access vlan 10
```

除了一种情况例外: (还可以加入一个Voice Vlan)

```
vlan 200
```

```
name voice
```

```
switchport voice vlan 200
```

1. Trunk (中继端口)

1. ISL (cisco 私有, 重封装)

Destination MAC | Source MAC | Type | Data | CRC

isl(26bytes) | Destination MAC | Source MAC | Type | Data | CRC | vlan
CRC

2. 802.1q (业界标准, 打标记)

Destination MAC | Source MAC | tag | Type | Data | dot1q CRC

access-port:

可以接受的帧:

802.3

Ethernet II

当vlan id和pvid相同, 也可以接dot1q帧, 如果vlan id和pvid不一样则drop。

可以发送的帧:

802.3

Ethernet II

Trunk port:

接受 802.1q、ISL、(native vlan 802.3和ethernet II)

发送 802.1q、ISL、(native vlan 802.3和ethernet II)

Native Vlan 不是vlan (只是巧合默认序号是1), 是一个Trunk链路上的一个特性,

Trunk针对该vlan的流量不用Trunk封装, 其他vlan用Trunk封装。

```
switch(config-if)#switchport trunk native vlan
```

DTP:Dynamic Trunking protocol

Switchport mode:

Access 手工指定访问端口, 不发DTP(dynamic trunking protocol)

Trunk 手工指定中继端口, 发送DTP

Dynamic 动态协商

auto 被动协商，不主动发DTP，收到DTP可回应。

desirable 主动协商，主动发DTP

Trunk 手工指定中继端口

step1:

sw1(config-if)#switchport trunk encapsulation dot1q (如果交换机只支持一种协议，那么这条命令可以不写)

step2:

sw1(config-if)#switchport mode trunk

Dynamic 动态协商: auto

sw1(config-if)#switchport mode dynamic auto

Dynamic 动态协商: desirable

sw1(config-if)#switchport mode dynamic desirable

关闭DTP的情况:

sw1(config-if)#switchport nonegotiate (关闭DTP，此命令必须mode trunk后才可用)

3种: 对方非cisco交换机

带trunk接口能力网卡的 服务器

单臂路由器

sh int trunk

```
switch(config)#int fa 5/8
```

```
switch(config)#shutdown 配置前先关掉是良好习惯
```

```
switch(config)#switchport trunk encapsulation dot1q
```

```
switch(config)#switchport trunk allowed vlan 1,5,11,1002-1005
```

```
switch(config)#switchport mode trunk
```

```
switch(config)#switchport trunk native vlan 99
```

```
switch(config)#switchport nonegotiate
```

```
switch(config)#no shut
```

VTP三种模式:

Server 可以增加、修改、和删除vlan信息，vlan信息存在flash中的vlan.dat文件中

client 不可以增加、修改和删除vlan信息，vlan信息存在flash中的vlan.dat文件中

Transparent 可以增加、修改和删除vlan信息，vlan信息存在配置当中，默认不发送VTP

sh vtp status:

交换机默认是server模式，域名为null，可以接受任何的vtp，并且可以学习到域名（学习到一个域名后就不可以学习到别的域名的信息）。

如果已有域名，只能接受到该域名信息，如果想更改域名，只能删除vlan.dat。

```
sw1#show flash:
sw1#delete flash:vlan.dat
reload
```

configuration revision很重要，所以新加一个交换机之前必须把模式变为transperent模式（configuration revision自动清零）。

vtp version 1和2的区别，1为默认，2可以透传vtp信息：domain 1 -> domain 2 ->domain 1
server上启用即可
switch(config)#vtp version 2

生成树：

802.1D STP -->PVST（只支持ISL）、PVST+（同时支持ISL、DOT1Q）
802.1W Rapid-STP -->Rapid-PVST
802.1s MSTP -->MSTP（可以成批把VLAN放到一个实例）

CST(commond spanning tree)独立生成树（只有一个实例）
BPDU网桥协议数据单元

修改cost值计算类型。

```
sw1(congig)#spanning-tree pathcost method long
查看详细BPDU信息
sh spanning-tree vlan 1 interface fa0/1 detail
```

根桥收到TCN的时候会flood BPDU给所有SWITCH，然后所有交换机把MAC地址MAX AGE由300秒改为15秒，加速MAC地址表收敛。

在PVST、PVST+中VLAN ID就是VLAN实例ID(extend system id)，如优先级别root id为4096的vlan 1则bridge id priority是4097

不推荐：

```
switch(config)#spanning-tree vlan 1 priority primary（优先级在学到根桥BPDU基础上减两级：一个级别为4096）
secondary（自动降一个级别）
```

这个命令只有一次作用，选定根桥后就无效了，如果后来又加新的进来，优先级不能影响，所以最好手动指优先级为0或者较低优先级。

敲完命令后会自动产生一个priority，如果想要再敲一次这个命令，则需要先no掉原来产生的priority。

推荐：

```
switch(config)#spanning-tree vlan 1 priority 0
```

PortFast模式（接口模式下配）：
swich(config)#int fa 0/47

```
swich(config-if)#spanning-tree vlan 1 portfast
swich(config-if)#keepalive 1（默认是10，这个时间过后端口才会变绿，
但省了端口listen和learning的30秒）
```

UplinkFast（全局模式下blocking交换机配）--只对直连链路有用（当fwd端口坏了的时候，blocked端口切换的时间接近为0，而跳过block->listen->learning状态）

```
swich(config)#spanning-tree uplink-fast
```

BackbonFast（全局模式下所有交换机配）--对非直连链路有用（可以缩短20秒blocking老化时间）：

MST:区域需要三个参数同时匹配：

area name

revision number

VLAN association table 映射表需要一致

如

```
SWA  inst 1    1-500vlan
      inst 2    501-1000vlan
SWB  inst 1    501-1000
      inst 2    1-500
```

则以上两个交换机不在同一个域

extended system id与实例id一致，而在pvst中与vlan id一致。

配置方式

```
sw1/2(config)#vlan 10/20/30/40
```

```
sw1/2(config-vlan)#name ccna/ccnp/ccie/wolf
```

```
inst 1 ->vlan 10/30
```

```
inst 2 ->vlan 20/40
```

每个实例分配两个VLAN，SW1作为INST 1的根，SW2作为INST 2的根

step1:

```
sw1(config)#spanning-tree mode mst
```

step2:

```
sw1(config)#spanning-tree mst configuration进入MST配置模式
```

step3:三个参数一致：

```
sw1(config-mst)#name ccnp
```

```
sw1(config-mst)#revision 11
```

```
sw1(config-mst)#instance 1 vlan 10,30
```

```
sw1(config-mst)#instance 2 vlan 20,40
```

step4:

sw2把sw1的配置重复一遍就完成了

```
sw2(config)#spanning-tree mode mst
```

```
sw2(config)#spanning-tree mst configuration
```

```
sw2(config-mst)#name ccnp
```

```
sw2(config-mst)#revision 11
```

```
sw2(config-mst)#instance 1 vlan 10,30
```

```
sw2(config-mst)#instance 2 vlan 20,40
```

step5:

让sw1成为实例1的根，sw2成为实例1的根。

```
sw1(config)#spanning-tree mst 1 priority 0
```

```
sw2(config)#spanning-tree mst 2 priority 0
```

step6:检查

```
sh spanning-tree {1/2}
```

因为先进，COST都改为长字节型。

EtherChannel

配置一（不好的配置方法，有缺陷）：如果遇到配置不错但是起不来那么只好删除重做。

```
sw1/2(config)#int port-channel 1
```

```
sw1/2(config-if)#exit
```

```
sw1/2(config)#int fa 0/10
```

```
sw1/2(config-if)#channel-group 1 mode on
```

```
sw1/2(config)#int fa 0/11
```

```
sw1/2(config-if)#channel-group 1 mode on
```

配置二：

step1:

```
default int fa 0/10
```

```
default int fa 0/11
```

```
sw1/2(config)#int range fa0/10 , f0/11
```

```
sw1/2(config-if-range)#shutdown
```

step2:

```
sw1/2(config)#channel-group 1 mode on(cisco建议两边用desirable)
```

自动创建interface port-channel 1

step3:

```
sw1/2(config)#no shut（两台同时要尽快，经验！）
```

检验：

```
sh int port-channel 1
```

mode on	是不启用动态协商协议；所以不能和任何别的mode协商
----------------	----------------------------

mode auto	被动的pagp协商
-----------	-----------

desirable	主动的pagp协商
-----------	-----------

passive	被动的lacp协商
---------	-----------

active	主动的lacp协商
--------	-----------

虚拟局域网VLAN的核心目的：

划分vlan的目的：低成本的将一个大的网络划分为小的网络，也称为网络分片/分段 (Segmentation)；

划分vlan的原因：隔离广播域 (广播有ARP、DHCP)；

广播流量不能穿越VLAN的边界：A vlan: a broadcast domain；

为了实现VLAN之间的数据通信需要实现VLAN间路由：A logical network (subnet)；

一个VLAN对应着一个广播域，最好对应一个网络子网 (为VLAN间的路由作准备)；VLAN限制了广播域的范围，VLAN间路由又允许了VLAN间的数据包通信。

HUB： 所有端口都在一个冲突域和一个广播域中；

Switch： 一个端口对应一个冲突域，所有端口都在一个广播域中；

Router： 一个端口对应一个冲突域，一个端口对应一个广播域。

VLAN的分类：

静态VLAN (Static VLAN)： 由MAC表建立的VLAN→匹配信息存储在MAC表；

动态VLAN (Dynamic VLAN)： 由服务器VMPS建立的VLAN→匹配信息存储在VMPS。

CDP (Connect Discover Protocol)：

CDP是CISCO私有的协议，只可以发现直链的设备！

交换环境中的2种连接类型：

Access Links：指的是只属于一个VLAN且仅向该VLAN转发数据帧的端口，也叫做native VLAN；交换机把帧发送到Access-link设备会进行信息帧的鉴别 (Frame Tagging)：当帧到达每个交换机，会首先检查VLAN ID然后决定如何对帧进行处理，当VLAN ID和Access link匹配的时候会移去VLAN标识符；Access-link设备不能与VLAN外通信除非建立了Vlan间路由；

Trunk Links：Trunk指的是能够转发多个不同VLAN的通信的端口，可以在一条网络介质Segment (网线/光纤) 上同时传输多个vlan的信息 (1次最多可

以在一条网络介质Segment(网线/光纤)上同时传输多个vlan的信息(1次最多可以携带1005个VLAN的信息)，必须使用100Mbps以上的端口来进行点对点连接，Trunk使你的单独的1个端口同时成为数个VLAN的端口，这样可以不需要L3设备；如果在Switch之间使用了Trunk, 那么多个VLAN间的信息将从这个连接上通过；如果没有使用Trunk而使用一般的连接，那么将只有VLAN1的信息能通过这个连接传递：VLAN1默认作为管理VLAN。

VLAN标识符 (VLAN Identification Method) 的种类：

VLAN标识符：在交换机的trunk link上, 可以通过对数据帧附加VLAN信息, 构建跨越多台交换机的VLAN. 附加VLAN信息的方法, 最具有代表性的有：

ISL(Inter-Switch Link)：Cisco私有，只能在快速和千兆以太网连接中使用；ISL路由可以用在Switch端口、Router接口和服务器接口卡之上；

dot1Q(IEEE 802.1Q)：由IEEE创建属于业界标准，在有非Cisco设备的连接上不能使用ISL时就必须使用802.1Q；802.1Q识别信息tag位于数据帧的源MAC地址与类型字段之间，大小为4Byte(其中包含12bits的VLAN-ID)；IEEE 802.1Q附加的VLAN信息就像在传递物品时附加的标签：当交换机检测到数据包的目标地址不在本交换机而应当发到某条trunk链路时，先进行trunk封装就是添加上tag并重新计算其FCS校验，完成封装后就可以从trunk链路上发给对方交换机了。

VTP协议 (Vlan Trunk Protocol)：

VTP也是Cisco创建的但是现在已经不为Cisco所私有，是用来自动协商、通告和同步Trunk信息的协议，必须依靠Trunk来传播；

VTP采用触发更新，周期为5min；

VTP有三种模式：**客户模式(Client)**→不能操作VLAN信息但是会转发变动通告/会同步VLAN信息/VLAN信息不会保存在NVRAM里；**服务器模式(Server)**→可以对VLAN进行建立、删除和修改等操作并发出变动通告/会同步VLAN信息/配置好的VLAN信息会保存在NVRAM里；**透明模式(Transparent)**→自己可以创建VLAN/可以转发信息/不能同步信息/配置好的VLAN信息会保存在NVRAM中；

注意：VTP不能完成将端口放入VLAN的工作！这需要在每一个交换机上配置。

Only "global" VLAN information regarding VLAN number, name, and description is exchanged.

Information on how ports are assigned to VLANs on a given switch is kept local to switch

and is not part of a VTP advertisement.

Vlans deleted on one switch may be deleted on all switches in the VTP domain, and thus

all ports removed from that VLAN. Delete VLANs with caution on a switch that is participating in a VTP domain with other switches.

VTP的工作过程：

交换机启动后如果VTP名字为空，就会在网络上查找VTP信息并且把自己的名字改为查找到的VTP名字，和对方是客户端还是服务器是没有关系的。

VTP的工作过程:

交换机启动后如果VTP名字为空,就会在网络上查找VTP信息并且把自己的名字改为查找到的VTP名字,和对方是客户端还是服务器是没有关系的。

VTP的修正号(Configuration Revision)和版本(Version):

用#sh vtp status查看VTP信息(保存vlan.dat文件中),注意#sh ru是看不到VTP信息的!

在运行同一VTP的VLAN中每进行一次配置或修改则会在修正号的基础上加1→客户端的不能导致修正号的变化;VLAN信息是由高版本号向低版本号的覆盖而不管是Client端还是Server端;

VTP的Vt1和Vt2是不兼容的:在一个server上改成版本2那么所有的交换机都会同步到版本2→Transparent例外。

VTP修剪(VTP pruning):

目的是减少广播、组播、单播、保留带宽;打开VTP pruning后只在Trunk link上发送广播;默认情况VTP pruning只能在VLAN2-1005使用,VLAN1是作为管理VLAN存在的!用(v)#vtp pruning打开VTP修剪。

LAB1:VLAN的创建,修改,删除和接口操作:

STEP1: VLAN的创建:

方法1(新方法,主流设备): (c)#vlan 10 →exit ;

方法2(老式机用法): (c)#vlan database →vlan 20 →apply (应用,即保存了所做的配置)→exit ;

(注意:在使用老方法创建,修改vlan时,如果不使用apply,而直接使用ctrl+z推出,则所有配置都不保存)

完成后可以用#sh vl br查看vlan的摘要信息;

STEP2: VLAN的修改:

方法1: (c)#vlan 10 →name ENG →exit ;

方法2: (c)#vlan database →vlan 20 name DRAGON →apply →exit
(可以在建立时name);

STEP3: VLAN的删除:

方法1: (c)#no vlan 10 →exit ;

方法2: (c)#vlan database →no vlan 20 →apply →exit ;

STEP4: 将端口划分入VLAN:

(c-i)#switchport access vlan * ;

可以使用Range一次定义多个接口: in range fastethernet 0/1 - 5, 0/9 (注意空格!);

LAB2: 跨交换机的同一VLAN通讯:

STEP1: 构建Trunk:

Trunk有两种→有自动协商:

802.1Q的Trunk: (c-i)#switchport trunk encapsulation dot1q

→switchport mode trunk ;

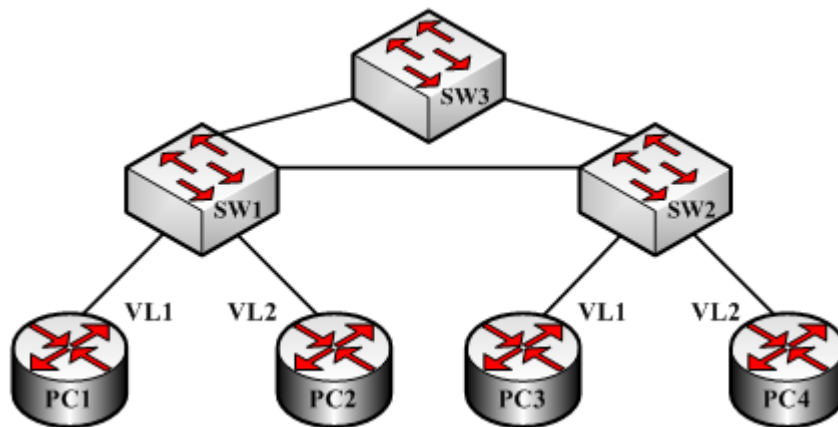
ISL的Trunk: (c-i)#switchport trunk encapsulation isl

→switchport mode trunk ;

两个交换机要配置相同的Trunk;

STEP2: 察看Trunk链路的连接状态:

#sh int tru .



LAB3: VTP的配置:

STEP1: 确认Trunk已经建立:

∴VTP必须依靠Trunk来传播∴要确认Trunk已经建立→用#sh in tr确认;

STEP2: 确定一个VTP域名并确认Server/Client:

一般以网络中用最高端/中心的交换机做Server, 考虑到冗余性可以建立两个Server;

在交换机上(c)#vtp domain CCNP →vtp mode server/client (服务器/客户端);

然后用#sh vtp status查看VTP信息, 注意sh ru是看不到VTP信息的!(VTP信息保存vlan.dat文件中)

STEP3: 查看版本号的变化:

通过在Server端增加/修改/删除vlan来观察Client上VLAN信息的同步和VTP修订版本号的变化;

STEP4: VTP的认证:

使用(c)#vtp password 123来加密VTP区域；可以用#sh vtp password来查看密码；

STEP5: VTP的修剪(在Server端做)：

(c)#vtp pruning 。

2.2—VLAN间路由

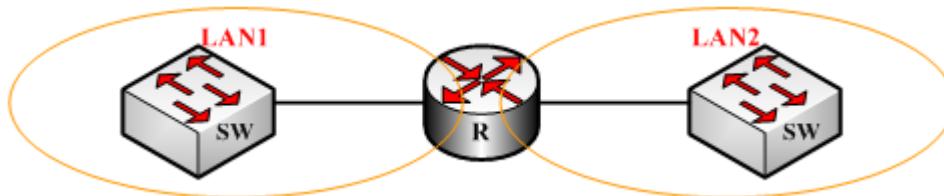
2.2—VLAN间路由

第一代LAN间的通信：

不支持VLAN的交换机：由一个路由器和几个交换机组成，每个交换机的所有端口都同属于一个网段/LAN；在路由器上有几个网段就有几个与之相对应的直链链路；在那个时代还没有VLAN这种技术；

缺点：交换机的所有端口都同属于一个LAN，一个LAN就需要一个路由器端口→导致建网成本很高；

优点：易于理解便于实施。

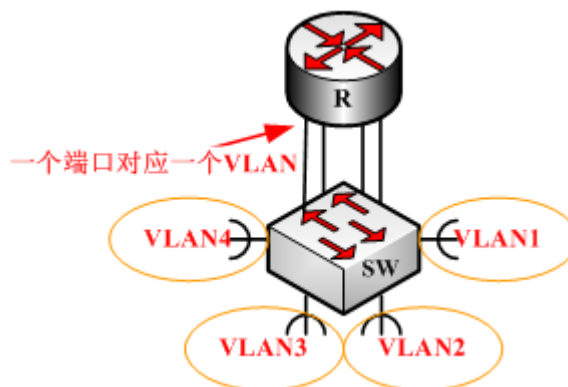


第二代VLAN间的数据通信：

在这个时代出现了能够支持VLAN的交换机，在路由器和交换机之间的链路，以及交换机的那个端口，都是完全属于所对应的那个VLAN的；

缺点：路由器还是一个端口对应一个VLAN, 成本还是很高；

优点：将原来的一个LAN对应一个交换机减少到只需要一台交换机（端口数目足够的话）→大大降低成本。



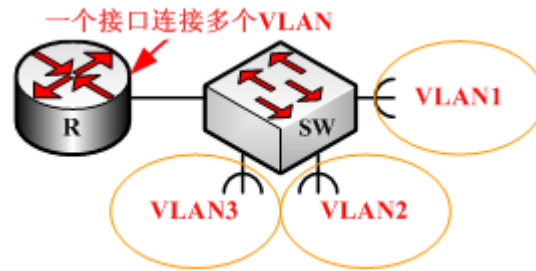
第三代VLAN间的数据通信：

单臂路由(Router On Astick)：在这个时代出现了Trunk技术：在一条物理介质上(网线/光纤)传输多个vlan的信息！是目前的主流。

实现：通过在路由器接口中创建子接口如e0.10和e0.20分别对应vlan10与vlan20。

优点：减少了路由和交换机连接的接口，减少了成本。

缺点：在路由器查询路由表；交换机与路由器之间的TRUNK链路，都可能构成网络性能瓶颈。



独臂路由：

step1:在所有的交换机上，传见VLAN10/VLAN20

STEP2: 把相应接口，划入相应的VLAN中。

STEP3:在交换机与路由器间的链路，也要配置TRUNK

3-1:交换机端：

```
sw2(config)#inter fa 0/5
```

```
switchport mode trunk ----2950上做，因为2950不支持802.1q
```

如果支持802.1q

那么→

```
switchport trunk encapsulation dot1q
```

```
switchport mode trunk
```

3-2: 路由器端：

```
R5#
```

```
IN E 0/0 (主接口)
```

```
no ip add
```

```
no shut
```

```
int e 0/0.10
```

```
encap dot1q 10(vlan10)明确指定为VLAN10服务
```

```
ip add 192.168.10.100 255.255.255.0
```

```
int e 0/0.20
```

```
encap dot1q 20(vlan20)
```

```
ip add 192.168.20.100 255.255.255.0
```

step 4:在R5上，检查直连路由：

```
r5#show ip route
```

```
c    192.168.10.0/24 is directly connected, ethernet0/0.10
c    192.168.20.0/24 is directly connected, ethernet0/0.20
```

如果switch与switch间用isl，路由与switch间用dot1q，那么PC PING路由是可以通的，因为switch与switch间的ISL只影响switch间通讯，同样的router与switch之间的dot1q也只影响路由器与交换机的那段通讯。

step5:交换机之间的trunk:

```
sw1:
int fa 0/11
switchport trunk encap dot1q
switchport mode trunk
```

```
int fa 0/1
switchport access vlan 10
int fa 0/3
switchport access vlan 20
```

```
sw2:
int fa 0/12
switchport trunk encap dot1q
switchport mode trunk
```

```
int fa 0/2
switchport access vlan 10
int fa 0/4
switchport access vlan 20
```

检查trunk链路:

```
show interfaces trunk
```

Port	Mode	Encapsulation	Status	Native vlan
Fa0/11	desirable	n-802.1q	trunking	1
Fa0/12	desirable	n-802.1q	trunking	1
Fa0/24	on	802.1q	trunking	1

step6:

所有VLAN10中的用户，都以E0/0.10为网关:

```
PC1: ip default-gateway 192.168.10.100
pc2: ip default-gateway 192.168.10.100
pc3: ip default-gateway 192.168.20.100
pc4: ip default-gateway 192.168.20.100
```

```
pc1/2#ping 192.168.10.100
pc3/4#ping 192.168.20.100
```

trunk和vlan间通讯是没有关系的，vlan间通讯是靠路由的，这里只是通过物理介质走trunk实现。

step7:跨VLAN/IP网段的通信:

第四代VLAN间的数据通信:(process switching)--被淘汰技术

三层交换(Layer 3 Switch): 将简化的路由器和交换机合二为一成为带有路由功能的交换机(l3 sw);

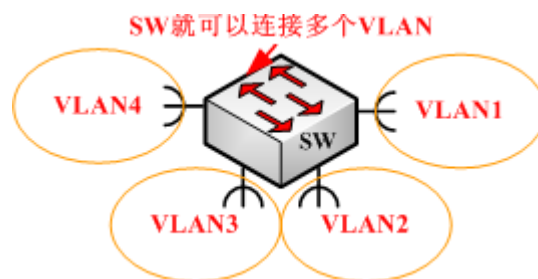
多次路由，多次交换

其实路由器做trunk的时候并没有启动路由协议，所以L3交换机不需要做到标准路由器的所有功能，费用能降低。

缺点: 受限于内置路由器的CPU查询路由表的能力; 每次路由多层交换效率相对低下;

优点: 性能更高，不用trunk而用PCB集成的告诉总线。

另外，不再需要外置路由器→大大降低成本。



第四代--使用L3交换机，作VLAN间路由: (内部路由器)

step1:

```
sw1#ip routing
```

show ip route 发现里面可以跑很多路由协议，功能非常强大，因为是3560交换机

sh run--发现默认有vlan1，可以通过vlan1 ip 网管。

step2:

```
vlan 10
```

```
ip add 192.168.10.100 255.255.255.0
```

```
ip add 192.168.10.100 255.255.255.0
vlan 20
ip add 192.168.20.100 255.255.255.0
```

```
show ip route
无需配置trunk口或者封装协议
```

step3:所有VLAN10/20的用户，都指定本VLAN接口作为网关：192.168.10(20).100。
网关为虚拟接口，PING的时候后先送到网关，然后查路由表，然后再送到另一个VLAN
的网关，然后再从交换机下来。
pc*#clear arp-cache
ping 192.168.10.* !!!!!

第五代VLAN间的数据通信:(fast switching)--也是被淘汰技术

多层交换 (Multilayer switching)：一次路由, 多次交换；默认的，只查一次路由表，效率高；

流掩码 (Netflow mask) 记录了源IP/PORT、目标IP/port和协议号。
第一次访问需要查路由表，后续的包不需要查询路由表，而是在二层形成cache，后面的包如果匹配的话就进行直接交换，
所以叫做一次路由，多次交换。

缺点：

- 1：还是需要查询路由表。
- 2：cache不能开太多。（IP都是/32的主机路由）
组合是在太多，cache记录不完。
- 3：基于Traffic/流。
必须之前有流量，否则不会有记录，所以叫基于Traffic。

第六代VLAN间的数据通信：

基于CEF（CISCO私有协议）的多层交换 (Cisco Express Forwarding Multilayer switching)：没有路由, 直接多次交换；

转发信息数据库FIB（转发信息数据库）、邻接关系、流掩码；可以用SW3560(c)#ip cef或者#sh ip cef查看。

无需路由，多次交换。（基于拓扑）

sw3550(config)#ip cef（全局的所有接口启动CEF）3550不能用，3560不需要用

sw3550(config)#ip cef distributed（全局的所有接口启动分布式CEF）
3560用

sw3550(config)#int vlan 10

sw3550(config-if)#ip route-cache cef（接口级别的CEF控制）

show ip cef 查看转发信息库，如：show ip cef 192.168.10.1/32可以查看很多详细信息。

show adjacency 查看邻居表

show ip route 事实上不需要路由表

show ip route 事实上不需要路由表

多层交换机上的接口分类：交换机端口port /路由接口interface ——把这么叫，但不是定论。

交换端口 (switch ports)：

交换端口是2层的物理端口，一般分为两种：

access port：一般是连接主机/电脑/服务器
sw3(config-if)#switchport mode access
只能属于一个VLAN；在access port传输的数据包是不携带vlan tag标记的
标准以太网帧；

trunk port：在两个交换机之间一定是trunk！在做单臂路由的链路上路由器和交换机之间也是trunk；trunk可以传输多个VLAN的信息并且依靠tag标识位携带的VLAN-ID来标识数据帧。

ISL header 26 byters:DA-TYPE-USER-SA-LEN-AAAA03-HSA-VLAN-BPDU-
INDX-RES

一个数据怎么知道要去什么VLAN就靠ISL包头的“VLAN”信息，15bit，可以支持更多VLAN，但效率也比DOT1Q低，因为每个包除了多增加26个字节的包头还多了4个字节的校验信息。因为这个，ISL会被802.1q淘汰。——注意，是私有的，而DOT1Q是业界标准。

802.1q tag 总共4个字节：ETHERTYPE(0x8100) —PRI-VLAN ID
12bit，默认支持4096个vlan，但是实际上没有交换机能上那么多。

路由接口 (routed ports)：

分为两类：

真实接口A：路由器接口和L3-SW关闭交换能力的接口 (in fa0/1 →no
switchport 相当于做为路由接口 →ip add ……)；

虚拟接口B：子接口、VLAN port和Loopback等；
int vlan10
ip add ……

VTP: (VLAN TRUNK PROTOCOL)

是cisco私有的协议 (GVRP)

VTP信息，只能在TRUNK链路上传递。——除了这个，和VLAN没有别的任何关系。

VTP目的：

提高VLAN网络的配置效率，减少网络配置的工作量：

VTP用于通告、同步VLAN的配置信息，简化关于VLAN的配置。

提醒：

VTP不能用于：将端口放入某个VLAN的操作--但可以把所有VLAN信息自动同步到所有交换机上。（IF# SWITCHPORT ACCESS VLAN 10）
（这需要在每个交换机上单独配置）

VTP MODE：

VTP的SERVER/CLIENT的相同点：

都可以转发VTP通告信息，而且都会向最新版本的VTP信息同步。

不同点：

SERVER:默认模式：可以创建/修改/删除VLAN，可以保存这些信息在NVRAM中。（VLAN.DAT）

client：不能创建/修改/删除VLAN，不能保存这些信息在NVRAM中。

VTP信息是触发更新，或者每5分钟周期性通告一次。

LAB:VTP

step0:确认Trunk链路已经成功建立：

```
show int trunk
```

step1:确定一个相同的VTP域名：

```
sw1/2: vtp domain CCNP
```

step2:确定server/client端：

```
sw1(config)#vtp mode server
```

```
sw2(config)#vtp mode client
```

show vtp status(查看VTP信息--VTP的信息，是无法通过SHOW RUN查看到的，因为sh run是看config.dat的信息的，而vlan信息存在vlan.dat。)

step3:观察vlan信息的同步：vlan添加，修改，删除。

```
vlan 100
```

```
show vtp status->看到版本号增加。configuration revision +
```

step4:VTP修剪(pruning)

```
vtp server:
```

```
sw1(config)#vtp pruning
```

对于没有用户的VLAN信息就不会发送到其他交换机，节省带宽和交换机处理开销。

VLAN信息不传递不代表就没有了这个VLAN，只是不发到别的交换机而已。

step5: VTP安全：

所有交换机：

```
sw1/2(config)#vtp password 123
```

```
sw1/2(config)#vtp password 123
```

sho vtp password才能看到password。

LAB1: 单臂路由(Router On Astick):

STEP1: 确认交换机之间的Trunk链路:

SW3是一个3550交换机; 在SW3的fa0/21、f0/9和f0/5上: (c)#in
fa0/21 → switchport trunk encapsulation dot1q → switchport mode
trunk ;

SW1/SW2是SW2950只支持802.1q! 同样配置Trunk;

完成后在SW3用#sh in trunk查看Trunk链路的建立;

STEP2: 在R5上为各个VLAN构建子接口并配置Trunk:

in e0/0 → no sh → in e0/0.10 → en dot1q 10 → ip add
192.168.10.100 255.255.255.0 → in e0/0.20 → en dot1q 20 → ip add
192.168.20.100 255.255.255.0 ; 然后配置Trunk;

STEP3: 在R5上查看直连路由:

在R2610上#sh ip ro会看到两条直连路由;

STEP4: 把各个交换机的相应接口放进对应VLAN:

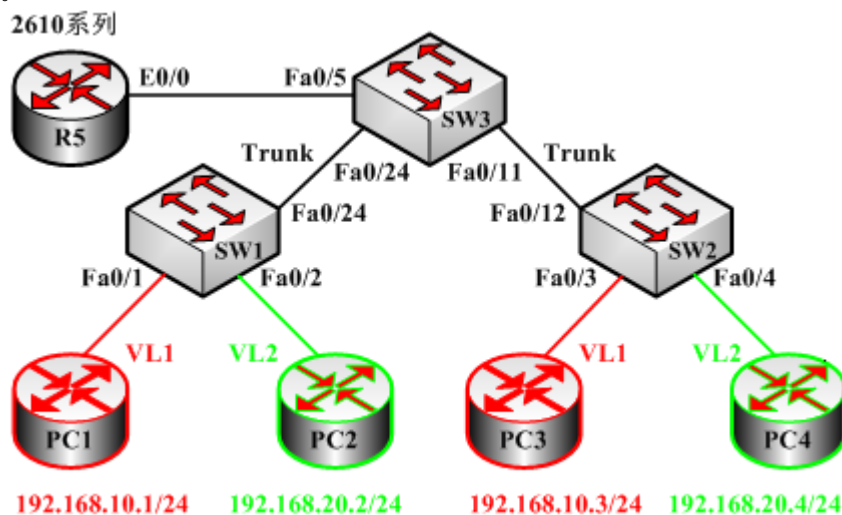
在3个交换机上配置vlan10/20; 完成后用#sh ip brief观察VLAN内通信
和VLAN间通信;

STEP5: 关闭两个子接口的代理ARP功能:

否则自己就通了: in e0/0.10 → no ip proxy-arp → in e0/0.20 → no
ip proxy-arp ;

STEP6: 在各个PC上指定对应的子接口为网关:

在四个PC上: (c)#ip default-gateway 子接口IP → 这样就完成了单臂
路由。



LAB2: 三层交换(Layer 3 Switch):

STEP1: 确认交换机之间的Trunk链路:

同上一个实验;

STEP2: 在SW3上为各个VLAN构建虚拟接口并配置Trunk:

SW3是一个3550交换机拥有路由功能, 因此可以去除R5使用SW3进行VLAN间路由: `in v10 → ip add 192.168.10.10 255.255.255.0 → no ip proxy-arp → in v20 → ip add 192.168.20.20 255.255.255.0 → no ip proxy-arp`; 然后配置Trunk;

STEP3: 把各个交换机的相应接口放进对应VLAN:

在3个交换机上配置vlan10/20; 完成后观察VLAN内通信和VLAN间通信;

STEP4: 在各个PC上指定对应的子接口为网关:

在四个PC上: (c)#`ip default-gateway` `VL接口IP` 然后就实现了VLAN间通信; 注意此时需要清下ARP表#`clear arp-cache`, 否则MAC会封装失败; 还有需要在SW3上启动路由功能(c)#`ip routing`才能看到路由表!

2.3—STP生成树

2.3—STP生成树

单点失效 (single point of failure) 及其解决方法:

当两个Segment之间只有一条物理连接时就有可能出现单点失效→ 单方面的故障导致全网Down; Segment的三种概念: 在STP领域表示一段物理介质(网线/光纤)、在封装领域表示经过L4封装的数据、在路由领域表示被L3设备所分割的逻辑子网;

避免单点失效的方法→构建冗余网络。

冗余网络三大问题:

冗余网络的三大问题都由其**核心问题**也就是**网络环路**(打环)引起;

多帧复制 (Multiple Frame Copies) → 未知MAC地址的数据帧在交换机间不断的复制转发复制转发……直到找到该目标地址并成功构建MAC映射, 而这时该目标已经收到了N个同样的数据帧了;

MAC表地址跳动 (MAC-Address Instability) → 因为几个接口都能够连接同一个PC因此MAC表中该设备的MAC值匹配的接口不断在几个接口间跳动;

广播风暴 (Broadcast Storms) → 广播会永无休止的发送下去占用越来越多的资源直到资源耗尽网络堵塞交换机Down机;

解决方法: STP。

生成树协议STP (Spanning Tree Protocol) :

STP核心: Provides a loop-redundant network topology, by placing certain in the blocking state (在冗余网络中将特定的端口置于阻

生成树协议STP (Spanning Tree Protocol) :

STP核心: Provides a loop-redundant network topology, by placing certain in the blocking state (在冗余网络中将特定的端口置于阻塞状态, 来实现既没有环路, 也可以冗余的网络)。

, k

STP的四大原则(越小越好):

lowest root BID ;
lowest path cost to root bridge ;
lowest sender BID ;
lowest port ID 。

生成树协议STP (Spanning-Tree Protocol) 的四大结论:

One root bridge per network → 每个交换网络, 都有一个唯一的根桥RB (根交换机: Bridge-ID最小的交换机或优先级最小的);

One root port per non-root bridge → 每个非根桥, 都有一个唯一的根端口RP (根端口: 去往根桥开销最小的端口);

One designated port per segment → 每条网络介质, 都必定有一个唯一的指定端口DP (指定端口, 根端口都是转发数据包Forwarding状态的);

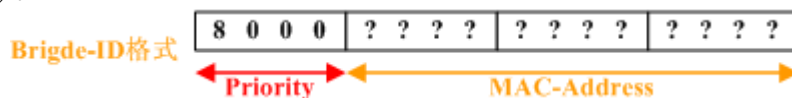
Nondesignated ports are unused → 没有获得任何标记的端口NDP (指定端口、根端口以外的端口), 都被禁用, 不转发数据包。

注意: 交换机开机时认为自己是根桥! 根桥上没有根端口, 所有端口 (参与了STP运算的所有端口, 通常是交换机与交换机相连的端口) 都是指定端口。

Bridge-ID:

是8BIT的ID, 由**优先级**(Priority/2Bits) + MAC地址(6Bits)组成, 优先级一般默认为0x8000 (32768) 并且只取整;

优先级取值: 0x0000 (0)、0x1000 (4096)、0x2000 ()、0x3000 ()、0x4000 ()、0x5000 ()、0x6000 (24576)、0x7000 (28672)、0x8000 (32768)、0x9000 (36864)。

**各种接口的STP默认开销:**

link speed	Revised IEEE Spec Cost	Previous IEEE Spec Cost
10Gbps	2	1
1Gbps	4	1
100Mbps	19	10
10Mbps	100	100

桥接协议数据单元BPDU (Bridge Protocol Data Unit):

在交换网络中由根桥RB (Root Bridge) 发送, 用于STP的计算和收敛; 发送周期为2秒;

BPDU的作用: 选举根桥、确定本地是否形成环路、阻塞特定端口防止环路、监控生成树的状态;

BPDU的作用：选举根桥、确定本地是否形成环路、阻塞特定端口防止环路、监控生成树的状态；

BPDU有两类：TC和TCN；TCA是TC的一种，当拓扑变化时检测到变化的交换机会通过RP向根桥发送TCN；而根桥收到TCN后会发送TCA表示收到，同时发送TC要求该交换机把MAC表的age时间改为15秒以学习新的表：由于TC是RB产生的，所以要更改各个计时器时只能在RB上更改。

端口的不同状态：

堵塞（Block，默认20s）：丢弃从该接口收到的正常数据帧；不会在该接口上学习mac-address-table；不会从该接口发送正常数据帧；[会从该接口接收对端发送的BPDU](#)；不会从该接口发送BPDU；

监听（Listen，默认15s）：丢弃从该接口收到的正常数据帧；不会在该接口上学习mac-address-table；不会从该接口发送正常数据帧；会从该接口接收对端发送的BPDU；[会从该接口发送BPDU（根桥RB、根端口RP、指定端口DP等的选举在该阶段完成，如果接口没有成为DP则重新回到Block）](#)；

同步（Learning，默认15s）：丢弃从该接口收到的正常数据帧；[会在该接口上学习mac-address-table](#)；不会从该接口发送正常数据帧；会从该接口接收对端发送的BPDU；会从该接口发送BPDU；

转发（Forward，直到down机→TK里面是15、30s的）：[从该接口接受正常数据帧](#)；会在该接口上学习mac-address-table；[从该接口发送正常数据帧](#)；会从该接口接收对端发送的BPDU；会从该接口发送BPDU。

交换机对BPDU的处理：

如果交换机从一个接口接收到优先级高的BPDU，会把该BPDU保存下来并且该接口不再往外发送BPDU；

在收敛时只有根桥产生BPDU，其余交换机只能从RP接收BPDU后才从DP发出去；这样非根桥可能从DP或者NDP接受到BPDU；

如果交换机从DP接收到优先级低的BPDU会丢弃，并给源MAC发送自己较新的BPDU；如果从NDP收到优先级低的BPDU会只丢弃了事。

STP下的链路失败类别：

直接失败：Block端口立刻进入Listen状态，收敛时间为30秒；

间接失败：要等待20秒后才能判断端口失败。

LAB1：四大结论第一（One root bridge per network）：

STEP1：察看交换机的网管IP 所对应的那个MAC地址：

每一个STP交换网络都只有一个根桥RB；引用原则1→Lowest Bridge-ID的那个交换机就是RB；

Bridge-ID由本交换机的优先级（2字节）和网管MAC地址（6字节）组成一共8个字节（[网管MAC地址：可以理解为被telnet的那个地址](#)）；

Bridge-ID由本交换机的优先级(2字节)和网管MAC地址(6字节)组成一共8个字节(网管MAC地址：可以理解为被telnet的那个地址)；

用#sh ve查看MAC地址得：Base ethernet MAC address: 00:0F:FE:2A:80:F6；还有sh cdp nei也能察看到MAC地址；

交换机的STP优先级默认是0x8000(32768)；

STEP2: 查看STP相关信息(本机BID)：

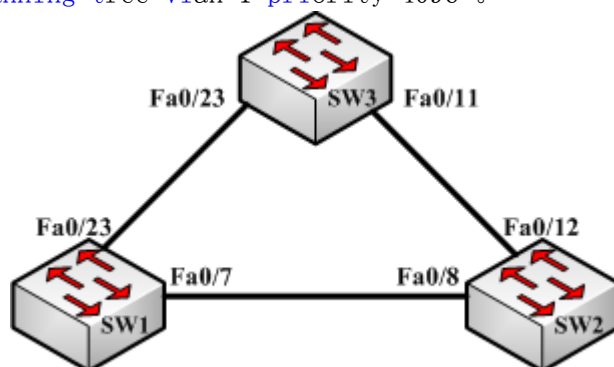
用#sh spanning-tree查看，Root-ID就是本STP网络中根桥的Bridge-ID→根据Root-ID找到根桥；

STEP3: 通过优先级控制根桥选举：

根桥：sw1(config)#spanning-tree vlan 1 root primary(默认PRI=24576/0x6000)；备份根桥：sw2(config)#spanning-tree vlan 1 root secondary(默认PRI=28672/0x7000)；

还可以更改优先级：

通过#spanning-tree vlan 1 priority 4096。



LAB2: 四大结论第二 (One root port per non-root

bridge)：

STEP1: 查看Cost：

每个非根桥NRB都有一个唯一的根端口RP；不是RB的交换机都是NRB；引用原则2→Lowest path cost to root bridge的那个交换机接口就是RP；

用#sh spanning-tree查看，得知当前的RP是9，Cost为19（当前去往根桥的cost值）；

STEP2: 控制RP的选举：

用(c-i)#speed 10(单位是Mbps)；

可以用#sh in status →sh spanning-tree detail查看；

也可以通过直接调试Cost值来控制：(c)#spanning-tree cost 50。

LAB3: 四大结论第三 (One designated port per

segment) :

STEP1: 查看Cost:

每一个网段有一个指定端口DP; 引用原则3→Lowest sender Bridge-ID;

STEP2: 控制DP的选举:

SW1(c)#spanning-tree vlan 1 priority 36864(优先级Prio=0x9000)。

LAB4: 四大结论第四 (Nondesignated ports are unused

) :

STEP1: 查看Cost:

没有获得任何标记的端口NDP被禁用; 引用原则4→Lowest port ID; 用#sh spanning-tree in ~~10/22~~ detail查看; →断后NDP是SW2的Fa0/7!
→BPDU入口不能堵的啊!!!

STEP2: 控制NDP的选举:

SW3(c-i)#spanning-tree port-priority 32 (这个值必须是16的整倍数); 再查看。

STEP3: SW2与SW1的连接恢复后会怎样?

先堵7后堵8→SW2被堵死了……把SW2改为RB后再恢复呢?

s

LAB5: 观察STP的转发延时:

STEP1: 直链检测错误 (35秒):

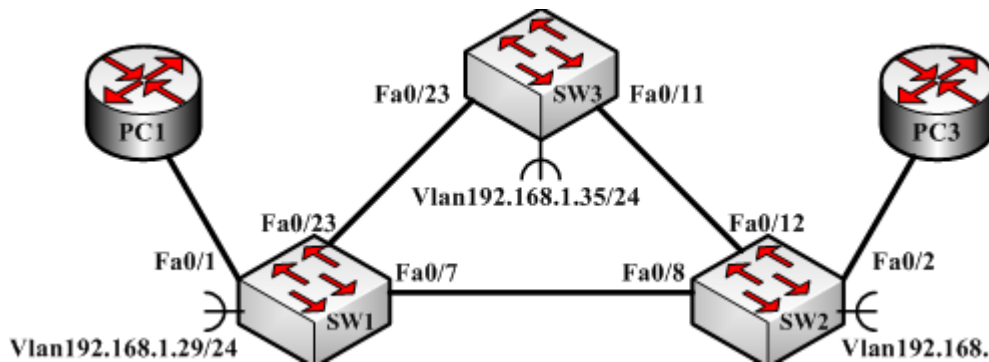
PC1 ping sw3的网管IP, shutdown SW3 23口观察:

让SW1成为根桥, 即把优先级改为最低(c)#spanning-tree vlan 1 priority 0 ; 然后PC3 ping PC1的同时shutdown掉SW2的Fa0/23, 因为SW2的Fa0/8是阻塞的, 所以down之前是从SW2的Fa0/23口发送数据到PC1的;

STEP2: 非直链检测错误 (50秒):

PC3 ping sw3的网管IP, shutdown SW3 9口观察:

SW1/SW2的链路带宽由原来的100M改为10M后Cost值变为100, 而SW1/SW3和SW2/SW3为100M, 那么Fa0/23是阻塞的; ping的同时shutdown掉SW2的Fa0/8后观察时间。



2.4—EN_STP

2.4—EN_STP

增强型生成树协议 (EN_STP)：

Spanning Tree port states:

blocking 20s+listening 15s+learning 15s最后才forwarding

这就是为什么交换机冗余切换会丢几个包的原因—但这些时间都是不一定的，可能blocking会经过很短时间就直接进入listening。

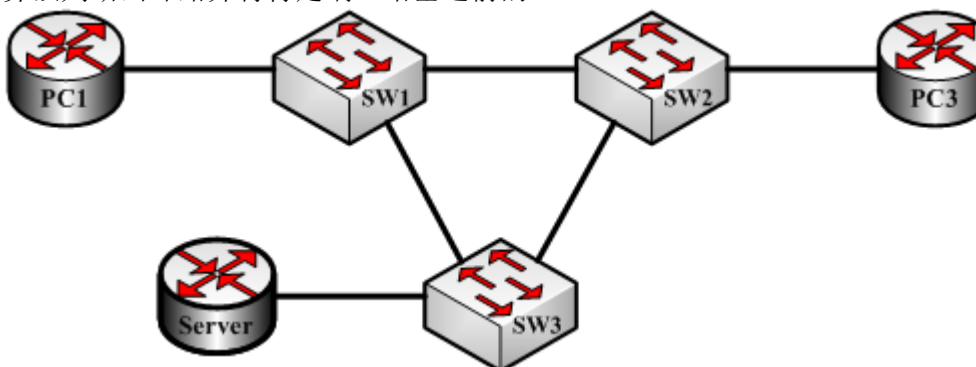
方法一：PortFast：

PortFast只适用于交换机与主机(电脑)直连的端口；

不应该在交换机与交换机、路由器、Hub等互连的网络设备的端口使用→会导致短暂的STP环路(STP Loop)并立刻导致广播风暴！

建立命令：(c)#int f0/1(交换机上直连主机的端口) →switchport mode access(接入链路) →spanning-tree portfast；

对于portfast端口，如果其接收到BPDU包，往往意味着本该连接主机的端口已经不正确的连接到了交换网络→这意味着很可能已经形成了STP环路(STP Loop)并导致了广播风暴→交换网络立刻会**暂短的瘫痪**→这是发生在STP算法判断出环路并将特定端口堵塞之前的！



方法二：uplink-fast：

在所有接入层交换机上配置uplink-fast，用于加快因为直链故障/直链检测错误（如果sw1与sw3之间链路段了，sw3可以直接检测到，这种情况就叫做直链检测错误）所导致的缓慢的STP网络收敛速度；direct link failure →uplink-fast应该在所有接入层/非核心层交换机上配置（根桥和备份根桥不要做），否则很可能会引起STP网络的环路；
配置命令：(c)#spanning-tree uplinkfast →速度由35s变为0s(跳过了接口的listening/learning状态)。

方法三：backbone-fast:

在所有的交换机上配置backbone-fast，用以使全网的交换机在遇到非直链检测错误（如果sw1与sw2之间断，对于sw3来说是不能直接检测到的，这种情况就叫做：非直链检测错误）时快速收敛；indirect link failure
；
配置命令：(c)#spanning-tree backbone →速度由50s变为35s 。

前面三种都是CISCO私有的，如果网络有别的品牌交换机则使用上会出问题。

IEEE 802.1D Media Access Control (MAC) bridge
IEEE 802.1Q Virtual Bridged Local Area Networks
IEEE 802.1W Rapid STP(RSTP)(业界的开放新标准)
 RSTP provides faster convergence than "IEEE
802.1D" STP when topology changes occur.

方法四：portfast bpduguard:

交换机端口的portfast bpduguard，是指在交换机的某端口接收到BPDU包后立刻关闭端口，以避免了更大范围的广播风暴的措施；portfast bpduguard要在连接主机的端口上配置；
基于接口的命令：(c-i)#spanning-tree bpduguard enable ；
基于全局的命令：(c)#spanning-tree portfast bpduguard default
；
是指在所有已经启动了portfast端口中,启动bpduguard 。

方法五：portfast-bpdufilter:

要在特定的portfast端口上配置，非PortFast接口无效；
基于接口的命令：(c-i)#spanning-tree bpdufilter enable ；
基于全局的命令：(c)#spanning-tree portfast bpdufilter default
。

（以上的几种特性,都是cisco私有的）

RSTP(802.1w Rapid STP):

RSTP是业界的开放性标准。

RSTP provides faster convergence than "IEEE 802.1D" STP when topology changes occur.

RSTP的4种端口角色:

root port : (forwarding);
designated port: (forwarding);
alternate port: 收到别的交换机转发来的BPDU, 此端口是blocking;
backup port: 收到本交换机转发来的BPDU, 此端口是blocking。

RSTP的3种端口状态:

discarding ;
learning ;
forwarding 。

RSTP链路分类:

全双工链路为: point to point 链路;
半双工链路为: share link链路——不论是HUB连交换机或者PC连交换机都叫share link。

RSTP的两种端口类型:

Edge Port & Non-Edge Ports

Edge port : 用于连接主机的端口:

Functions similarly to portfast,
(Immediately transitions to forwardin)

Unlike portfast,
an edge port that receives a BPDU,
immediately loses its edge port status and becomes a normal
spanning tree port.

可以这么说: 802.1w比CISCO(或802.1d)先进, 当接受到BPDU的时候由portfast变成stp端口, 可以进行blocking或者forwarding。

启动RSTP的命令:

```
SW(c)#spanning-tree mode pvst/rapid-pvst 。 (per-vlan rapid  
spanning tree)  
show sapnning-tree  
show spanning-tree vlan 1  
选举方式与STP生成树那节课是一样的。
```

standard 802.1q是CST(单生成树): ——CISCO是不会这么做的。

缺点: 所有的VLAN都是按照同一个STP来运行;
所有的交换机都只维护一个STP实体(instance);
所有VLAN的数据流量都压向一个根桥交换机——可能会压垮一个交换机;
无法做基于VLAN的L2负载均衡;

所有VLAN的数据流量都压向一个根桥交换机—可能会压垮一个交换机；
无法做基于VLAN的L2负载均衡；
优点：对交换网络的开销较小→所有的VLAN都共享一个STP实体。

Cisco私有的STP mode PVST (Per Vlan STP):

优点：可以为每个VLAN配置一个STP；
不同交换机可以是不同VLAN的根桥；
可以实现基于VLAN的L2负载均衡；
缺点：交换机维护很多的STP实体→对交换网络的开销比较大；
任意一个VLAN的拓扑变化都可能波及到很多交换机重新运算→收敛稍慢。

MST(802.1s Multiple Spanning Tree):

MST是业界开放性标准；就是对standard 802.1q/PVST的折衷方案→基于“组/instance”的STP；
(一个组可以对应多个结构相同的VLAN，大大节省交换机开销)
MST的主要配置步骤：首先对VLAN进行分组(instance)；每个组都可以有独立的STP、根桥，实现了L2负载均衡\冗余，互为备份。

LAB1:通过PVST实现VLAN间的冗余和L2负载均衡:

STEP1: 选定STP的运行模式:

在交换机上(c)#spanning-tree mode pvst/rapid-pvst (两种都可以pvst为802.1d;rapid-pvst为802.1w) ;

STEP2: 通过配置VTP, 使全网的交换机都有相应的VLAN:

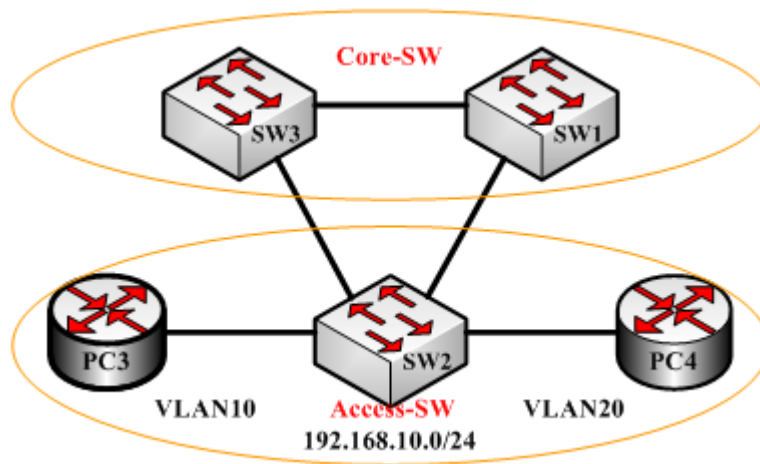
```
sw1/2/3(config)#vtp domain CCNP ;  
sw1/2/3(config)#vtp password 123  
sw3/2(config)#vtp mode server ;  
sw1/2/3(config)#vtp pruning清除旧的vlan信息  
sw3(config)#vtp mode client ;  
然后在其中的一个VTP server上增加vlan 10和vlan20;
```

STEP3: SW3和SW1分别成为VLAN10和VLAN20的根桥:

```
sw3(config)#spanning-tree vlan 10-15 root primary ;  
sw1(config)#spanning-tree vlan 20-25 root primary ;  
同时两交换机要互为备份 (互为对方的备份根):  
sw3(config)#spanning-tree vlan 20-25 root secondary ;  
sw1(config)#spanning-tree vlan 10-15 root secondary ;
```

STEP4: 在SW2上观察对于特定VLAN哪个是根端口:

```
sw2#sh spanning-tree vlan 10 ;  
sw2#sh spanning-tree vlan 20 。
```



LAB2: 通过MST, 实现L2的VLAN间的负载均衡:

STEP1: 选定交换机的stp模式为MST:

```
sw1/2/3(c)#spanning-tree mode mst ;
```

STEP2: 将vlan划分到不同的“instance/组”中: --先要创建VLAN。

```
sw1/2/3(c)#spanning-tree mst configuration ;
```

```
sw1/2/3(c-mst)#instance 1 vlan 10-15 ;
```

```
sw1/2/3(c-mst)#instance 2 vlan 20-25 ;
```

以上步骤所有交换机都要做。

STEP3: 在核心交换机上为不同的组配置不同的优先级, 实现两个组的互为备份/负载均衡:

确定每个组的根桥:

```
sw3(c)#spanning-tree mst 1 root primary ;
```

```
sw1(c)#spanning-tree mst 2 root primary ;
```

互为备份:

```
sw3(c)#spanning-tree mst 2 root secondary ;
```

```
sw1(c)#spanning-tree mst 1 root secondary ;
```

STEP4:

```
sw1/2/3#sh spanning-tree mst configuration ;
```

instance	vlan mapped
0	1-9, 16-19, 26-4096 (默认都在0号组)
1	10-15
2	20-25 ;

STEP5:

```
sw3#sh spanning-tree mst 1/2 .
```

EtherChannel:网络聚合——如: 把两条100M的冗余线路捆绑成一条逻辑链路, 既然使一条, 那么STP就不起作用了。

路，既然使一条，那么STP就不起作用了。

Logical aggregation of similar links

Load balances

Viewed as one logical port

Redundancy

作用：

1: 增大链路带宽

2: Redundancy

3: Load balances

example:

FEC(Fast Ether Channel) (100Mbps)

GEC(Giga Ether channel) (1000Mbps)

LAB: 通过FEC, 实现交换机之间，高带宽，冗余，负载均衡的链路

提醒：

所有参与channel-group的接口的配置参数，必须完全一致。

（譬如：速率，全双工模式等）

step1:先恢复接口的默认配置：

```
SW1/2(CONFIG)#interface range fa 0/11 -12
```

```
SW1/2(CONFIG)#speed 100
```

```
SW1/2(CONFIG)#duplex full
```

step2:

```
SW1/2(CONFIG)#channel-group 1 mode on（不让自动协商，否则容易出问题）
```

```
no shut
```

step3:观察交换机会自动生成一个虚拟的接口：

```
sh run
```

```
(interface port-channel 1)
```

```
sw1#show int port-channel 1
```

```
...BW 200,000Kbps
```

```
show etherchannel 1 port-channel
```

```
show etherchannel 1 summary
```

2.5—冗余VLAN

2.5—冗余VLAN

High-Availability (AH)

First hop routers on the LAN redundancy Network/首跳冗余网络（出口第一个网络）：

建立：Fault-tolerant/容错网络

避免：Single Points of Failure/单点失效

A:网络拓扑冗余（成本最高的）：

B:硬件的冗余：

交换引擎的冗余，电源冗余，线卡（大模块，高端设备用）冗余，风扇冗余，线路冗余，ISP冗余（电信和网通，而且出口要不同）。

C:软件/协议的冗余：

HSRP (RFC2281)

VRRP (RFC2383)

GLBP (Gateway Load Balancing Protocol)

LAB1:Default gateway（不运行代理ARP）

step1:

PC4(config)#ip default-gateway 192.168.1.2(设定默认网关)

PC5(config)#ip default-gateway 192.168.1.3

step2:

关闭两个出口路由器R2/3，的内口（以太口）的代理ARP：

R2/3 (CONFIG)#INT E0

NO IP PROXY-ARP

```
SHOW IP INT E 0
...PROXY ARP IS DISABLED
```

STEP3:在外网运行动态路由协议RIP:

```
R1/2/3#
ROUTER RIP
VER 2
NO AUTO
R1:NET 1.0.0.0/12.0.0.0/13.0.0.0
R2:NET 12.0.0.0
R3:NET 13.0.0.0
SHO IP ROUTE RIP
```

```
R1:DEBUG IP PACKET
R4:PING 1.1.1.1
R1:UN ALL
```

STEP4:R2/3上, 做NAT:(基于NAT路由器外口地址的端口复用)

4-1: 定义内网的用户群:

```
ACCESS-LIST 1 PERMIT 192.168.1.0 0.0.0.255
```

4-2:定义NAT的内口/外口:

```
int s 0
ip nat outside
int e 0
ip nat inside
```

4-3: 进行基于NAT路由器外口地址的端口复用:

```
ip nat inside source list 1 interface serial 0 overload
```

step 5:

观察指定默认网关的单点失效。

```
pc4:ping 1.1.1.1 !!!!! <control>+<shief>+<6>终止ping
```

```
r1:int s 0
```

```
shut
```

```
ping 1.1.1.1 .....
```

LAB2:Proxy ARP/代理ARP:

The client uses ARP to get the destination it wants to reach, and a router will respond to the ARP request with its own MAC address.

step 1:

PC主机上, 无需配置网关:

```
pc4/5# no ip default-gateway
```

```
r2/3#int e 0
```

```
ip proxy-arp
```

step 3:

```
PC4/5#
```

```
show arp
```

```
clear arp-cache (windows:arp -d)
```

```
debug arp
```

```
r2 mac:x.x.c4d3
```

```
r3 mac:x.x.9dcc
```

r4可以从r2或者r3走，但是只会有一个ARP映射产生，后来的会覆盖先来的。

step4:观察通过代理ARP实现的冗余网络，是不满足高可靠性的要求的。

First-Hop redundancy Protocol (FHRP) 首跳冗余

active router & stand by router

在LAN用第一跳构建冗余网络 (First hop routers on the LAN Redundancy Network) :

建立: 容错网络 (Fault-tolerant) ;

避免: 单点失效 (Single Points of Failure) 。

硬件冗余:

1: 拓扑冗余;

2: 交换引擎的冗余、电源冗余、线卡冗余、风扇冗余、线路冗余。

软件/协议的冗余:

HSRP (RFC2281); c

VRRP (RFC2383);

IRDP (RFC1256);

GLBP (Gateway Load Balancing Protocol);

SRM (Single Router Mode);

SLB (Server Load Balancing) 。

冗余VLAN的近似通用配置:

1 配置虚拟路由器

2 配置优先级

3 配置抢占路由器

Routing protocol (路由协议):

The client listens to dynamic routing protocol updates (for example, from IGP RIP/OSPF) And forms its own routing table 。

IRDP(ICMP Router Discovery Protocol):

IRDP client—the client runs an ICMP(Internet Control Message Protocol) router discovery client ;

缺陷：网络收敛性较慢；而且兼容性差→受限于主机的操作系统→少有操作系统支持！

热备份路由协议HSRP(Hot Standby Router Protocol):

HRSP是CISCO私有的，特点是收敛快；

The HSRP(Hot Standby Router Protocol) is a FHRP(First-Hop Redundancy Protocol), Designed to allow for transparent fail-over of the first-hop IP router 。

HSRP provides high network availability by providing first-hop routing redundancy for IP hosts on Ethernet, with a default gateway IP address.

VRRP(Virtual Router Redundancy Protocol):

VRRP是业界标准；其组播地址是：224.0.0.18 ；

配置命令：(c)#Interface Ethernet 1/0 →Ip address 192.168.1.2 255.0.0.0 →Vrrp 1 description VL-1 →Vrrp 1 priority 100 →Vrrp 1 preempt →Vrrp 1 ip 192.168.2.100 →Vrrp 1 authentication cisco (认证) →Vrrp 1 timers advertise 2 (作用类似于Hello包) 。

GLBP(Gateway Load Balancing Protocol):

GLBP是CISCO私有的协议，和它竞争的是HSRP和VRRP；GLBP使用的组播地址是：224.0.0.12 ；

The advantage of GLBP is that it additionally provides load balancing over multiple routers(gateways) using a single virtual ip address and multiple virtual MAC addresses;

配置命令：Interface fastethernet 0/0 →Ip address 10.21.8.32 255.255.255.0 →Glb 10 priority 100 →Glb 10 preempt →Glb 10 ip 10.0.0.1 。

LAB1: Deafault Gateway:

STEP1: 按图构建拓扑:

要在边界路由器面向PC的接口(R2/R3的E0) 关闭自动ARP否则会构建Deafault-Gateway无法达到实验要求：int e0 →no ip proxy-arp ；

STEP2: 配置Deafault-Gateway:

Deafault-Gateway的特点是配置简单和单点失效，其配置在主机(R4/R5)上完成：

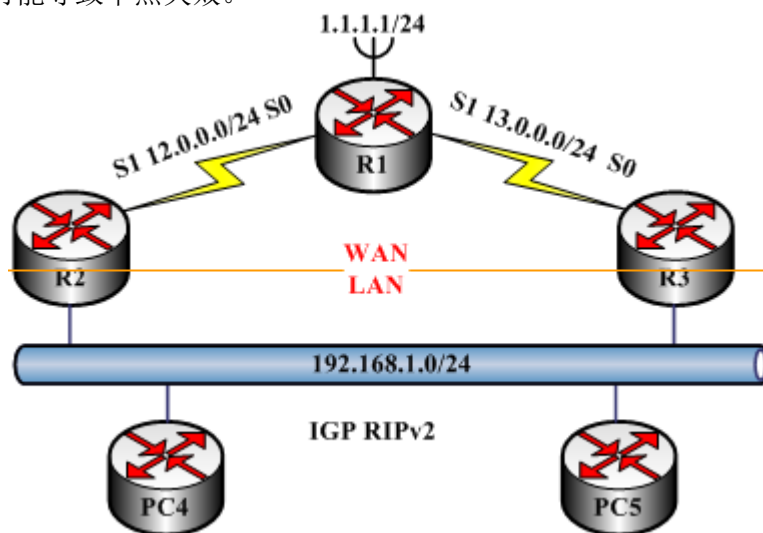
Deafault-Gateway的特点是配置简单和单点失效，其配置在主机(R4/R5)上完成：

(c)#no ip routing →ip default gateway 192.168.1.2 ；

STEP3：要在边缘路由器（R2/R3）运行NAT：

(c)#access-list 1 permit 192.168.1.0 0.0.0.255（首先定义要进行NAT的网段）→int s0 →ip nat outside →int e0 →ip nat inside（接着定义NAT的内/外口）→ip nat inside source list 1 int s0 overload（最后进行NAT转换并端口复用）；

然后测试，在PC4和PC5上ping通R1：断开一边后另外一边仍然是通的，而且断开边的PC也不会走另一条链路；得到结论：无法检测到设备/链路故障，有可能导致单点失效。



LAB2：代理ARP：

STEP1：接上个实验：

ARP的欺骗：the client uses ARP(Address Resolution Protocol) Toget the destingation it wants to reach , and a router will respond to the **ARP request** with its **own MAC address** ；

PC上无需配置网关：(c)#no ip default-gateway；

STEP2：在网关路由器上启动ARP（默认启动但是刚刚关了）：

(c)#int e0 →ip proxy-arp ；

STEP3：测试ARP的运作：

分别在PC4/PC5 上ping1.1.1.1进行测试；

清理ARP进程的命令：#clear arp-cache 。

LAB3: 构建HSRP:**STEP1: 构建拓扑:**

要关掉ARP;

STEP2: 配置虚拟路由VR:

在R2/R3上: (c)#in e0 → standby 1 ip 192.168.1.100 (定义虚拟路由器/网关的IP) → standby 1 priority 105 (定义HSRP优先级控制active竞选, 默认100, R3取默认值) → standby 1 preempt (指定抢占竞选模式: 优先级高的路由成为active) ;

STEP3: 在内网指定VR为默认网关:

ip default gateway 192.168.1.100

STEP4: 查看HSRP的状态:

用#sh standby brief查看; 在PC4/PC5上ping1.1.1.1 ;

STEP5: 查看协商状态:

#debug standby后第一次ping; 注意清理ARP表#clear arp-cache ;

STEP6: 跟踪HSRP路由器的外口:

在R2/R3上: (c-i)#standby 1 track s0 (跟踪外口: 外口断了路由器能检测到, 而且能自动切换链路);

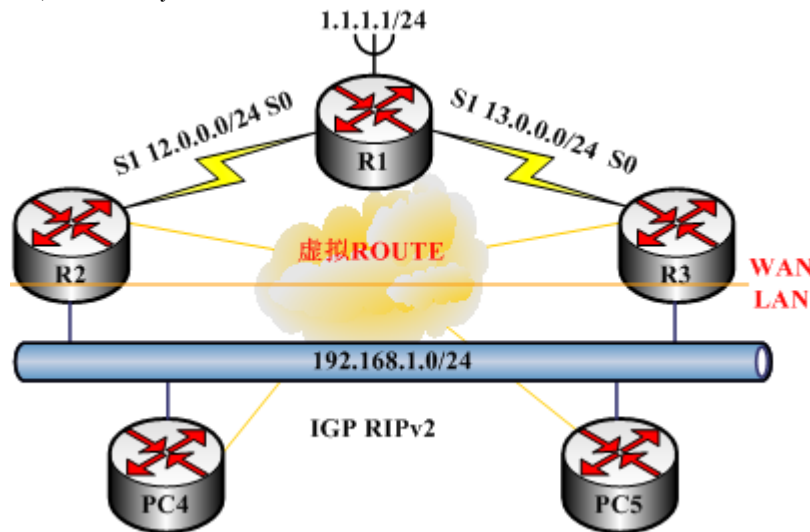
STEP7: 其他命令:

(c-i)#standby 1 authentication 123 (HSRP的认证);

(c-i)#standby 1 mac-address 00c0.abcd.1234 ;

(c-i)#standby 1 name vlan-eng ;

(c-i)#standby 1 timers 3 10 。



上课笔记: LAB:HSRP

STEP 1:关闭R2/3的代理ARP功能:

```
int e 0
```

```
no ip proxy-arp
```

step2:构建一个虚拟路由器在内网指定VR为默认网关 (R2/3共同维护)

r2为Active网关, r3为standby网关

2-1:

r2/3:standby 1(组号) name vr-1

standby 1 ip 192.168.1.100 (定义虚拟路由器/网关的IP)

2-2:定义参与HSRP的路由器的优先级:

int e 0

r3:standby 1 priority 100 (默认100)

r2:standby 1 priority 105 (定义HSRP优先级, 越高越可能成为Active Router)

2-3: HSRP抢占:

R2/3(CONFIG-IF)#standby 1 preempt (抢占: 谁的优先级高, 谁就Active)

ospf DR选举无抢占性

step3:PC4, 指定VR-1为默认网关:

PC4(config)#ip default-gateway 192.168.1.100 (指定虚拟路由器为网关)

step4:观察HSRP的简要工作状态:

r2/3#show standby brief 可以查看谁是ACTIVE谁是STANDBY

ping 192.168.1.100 !!!!!

step5:观察HSRP的切换:

r2/3#debug standby

pc4 ping 1.1.1.1

关闭以太网, 内口(e0)一没问题, 可以切换后正常通信。

standby hello包为每三秒一次, r3三次没收到r2 hello包就把自己转换为active, 所以会有五个包的lost, 平均两秒一个包。

关闭串口, 外口(s0)一有问题, standby路由器因为正常收到r2 hello包, 所以不转发数据, 而r1这时关闭了与r2的串口连接。

step6: 跟踪HSRP路由器的外口:

r2/3(config)#int e 0 在e0口做

r2/3(config-if)#standby 1 track serial 0

跟踪本路由器的外口:

如果外口失效, 本机自动将自己的HSRP优先级默认减10。

线路恢复正常后HSRP优先级自动增加10, r2会迅速从standby变成active, 甚至不会丢包。

step 7: Advanced HSRP

r2/3(config-if)#standby 1 authentication 123 (HSRP的认证)

r2/3(config-if)#standby 1 MAC-address 00C0.1234.ABCD (指定虚拟MAC)

r2/3(config-if)#standby 1 timers 3(hello包) 10(超时)

LAB:不同子网(VLAN)间的负载均衡: (R2600)

```
r2:
int e 0/0.10
en dot1q 10
ip add 192.168.10.2 255.255.255.0
standby 10 ip 192.168.10.100
standby 10 ip priority 105
standby 10 ip preempt
standby 10 name vr-10
standby 10 track s 0/0
```

```
int e 0/0.20
en dot1q 20
ip add 192.168.20.2 255.255.255.0
standby 20 ip 192.168.20.100
standby 20 ip priority 100
standby 20 ip preempt
standby 20 name vr-20
standby 20 track s 0/0
```

```
r3:
int e 0/0.10
en dot1q 10
ip add 192.168.10.3 255.255.255.0
standby 10 ip 192.168.10.100
standby 10 ip priority 100
standby 10 ip preempt
standby 10 name vr-10
standby 10 track s 0/0
```

```
int e 0/0.10
en dot1q 10
ip add 192.168.20.3 255.255.255.0
standby 20 ip 192.168.20.100
standby 20 ip priority 105
standby 20 ip preempt
standby 20 name vr-20
standby 20 track s 0/0
```

更先进的拓扑（看2-13图）

```
sw1:
int vlan10
ip add 192.168.10.1 255.255.255.0
standby 10 ip 192.168.10.100
standby 10 ip priority 105
standby 10 ip preempt
```

```
standby 10 name vr-10
```

```
int vlan20
ip add 192.168.20.1 255.255.255.0
standby 20 ip 192.168.20.100
standby 20 ip priority 100
standby 20 ip preempt
standby 20 name vr-20
```

```
sw2:
int vlan10
ip add 192.168.10.3 255.255.255.0
standby 10 ip 192.168.10.100
standby 10 ip priority 100
standby 10 ip preempt
standby 10 name vr-10
```

```
int vlan20
ip add 192.168.20.3 255.255.255.0
standby 20 ip 192.168.20.100
standby 20 ip priority 105
standby 20 ip preempt
standby 20 name vr-20
```

VRRP:原理与HSRP一样

```
ra#int e 0
ip add 192.168.1.3 255.255.255.0
vrrp 1 description vr-1
vrrp 1 priority 105
vrrp 1 preempt
vrrp 1 ip 192.168.1.100
```

```
rb#int e 0
ip add 192.168.1.2 255.255.255.0
vrrp 1 description vr-1
vrrp 1 priority 100
vrrp 1 preempt
vrrp 1 ip 192.168.1.100
```

GLBP: (CISCO私有) (Gateway load balancing protocol)

The advantage of GLBP is that it additionally provides load balancing over multiple routers(gateways), mapping a single virtual ip address to multiple virtual MAC addresses;

```
ra#int fa 0/0
```

```
ip add 192.168.10.2 255.255.255.0
glbp 10 priority 105
glbp 10 preempt
glbp 10 ip 192.168.10.100
```

```
rb#int fa 0/0
ip add 192.168.10.3 255.255.255.0
glbp 10 priority 100
glbp 10 preempt
glbp 10 ip 192.168.10.100
```


LAB4: HSRP:

STEP1: 构建拓扑:

要关掉ARP;

STEP2: 配置虚拟路由VRA:

在边界路由器的接口上配置HSRP的虚拟路由A:

在R2上: (c)#in e0 →standby 1 name VR-A →standby 1 priority 105 preempt →standby 1 ip 192.168.1.100 →standby 1 track s0 ;

在R3上: (c)#in e0 →standby 1 name VR-A →standby 1 priority 100 preempt →standby 1 ip 192.168.1.100 →standby 1 track s0 ;

注意优先级:

HSRP	R2	R3
VR-A	Active	100
VR-B	100	Active

STEP2: 配置虚拟路由VRB:

在边界路由器的接口上配置HSRP的虚拟路由B:

在R2上: (c)#in e0 →standby use-bia (启动多个组) →standby 2 name VR-B →standby 2 priority 100 preempt →standby 2 ip 192.168.1.200 →standby 2 track s0 ;

在R2上: (c)#in e0 →standby use-bia (启动多个组) →standby 2 name VR-B →standby 2 priority 105 preempt →standby 2 ip 192.168.1.200 →standby 2 track s0 ;

STEP3: 在不同的分组用户中指定不同的VR作为网关:

PC4(config)#ip default-gateway 192.168.1.100 ;

PC5(config)#ip default-gateway 192.168.1.200 。

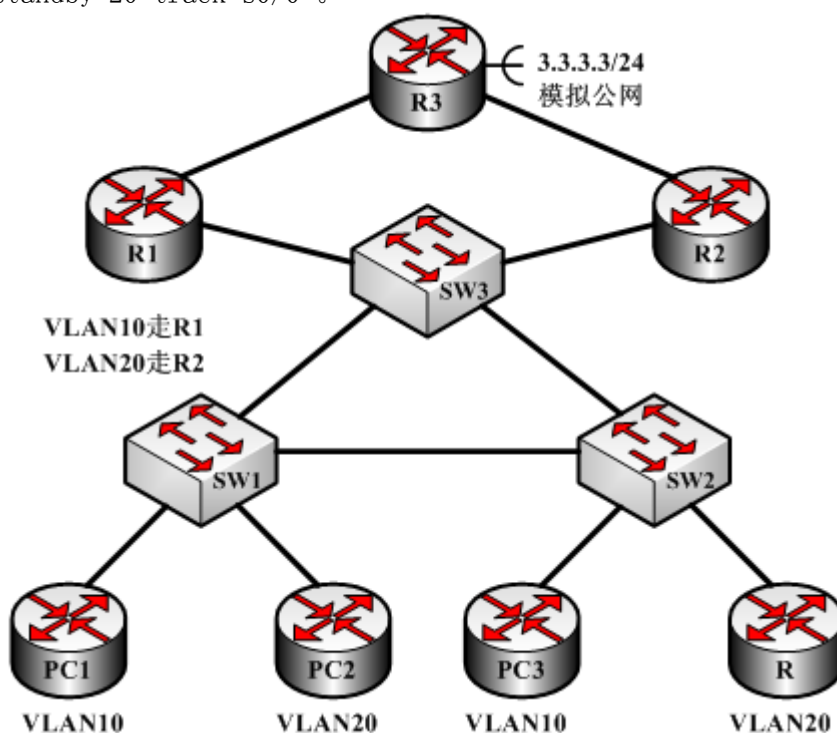
LAB5:不同子网(vlan)间的负载均衡(R2600)

其他同:

```

R1(config)#in e0/0 ;
no shut ;
int e0/0.10 ;
en dot1q 10 ;
ip add 192.168.10.1 255.255.255.0 ;
standby 10 ip 192.168.10.100 ;
standby 10 priority 105 ;
standby 10 preempt ;
standby 10 name VR-10 ;
standby 10 track s0/0 ;
in e0/0.20 ;
en dot1q 20 ;
ip add 192.168.10.1 255.255.255.0 ;
standby 20 ip 192.168.20.100 ;
standby 20 preempt ;
standby 20 name VR-20 ;
standby 20 track s0/0 。

```

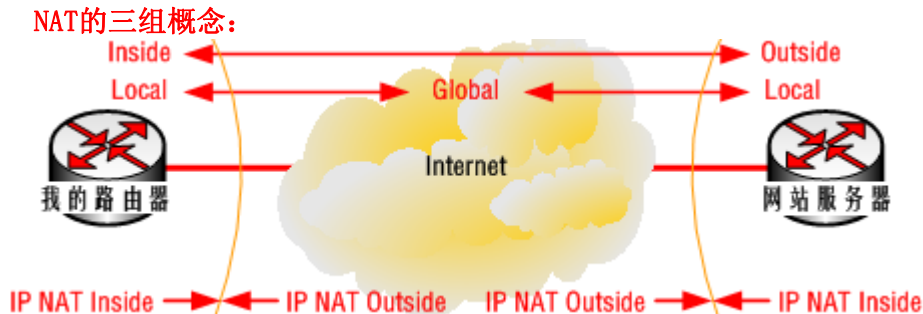


2.6—NAT

2.6—NAT

网络地址转换协议NAT (Network Address Translation) :

交换和远程都要用，先上什么就放在哪一块讲，具体来说NAT还是属于远程的。



LAB1: NAT的基本配置 (原理性的NAT) :

STEP1: 按图构建拓扑:

R2/R4配置为PC模拟用户，R5的环回路口模拟公网；宣告RIP网络时不宣告192.168.1.0/24网段，这是因为现实中私网地址是不会被宣告到公网中的；R5可以宣告所有，公网的边界路由器R3上则宣告35.0.0.0/24网段和13.0.0.3/24接口；

STEP2: 在私网的边缘路由器上做指向公网的默认静态路由:

R1(c)#ip route 0.0.0.0 0.0.0.0 serial 1 ;

STEP3: 在PC上指定网关:

(c)#ip default-gateway 192.168.1.1然后Ping确认能与网关通讯；这个时候PC2/PC4能够将包发往R5,但是R5上没有到PC2/PC4的路由(没有宣告)所以无法回包；现实中的ISP会在路由器上做ACL将私网地址过滤掉，所以PC2/PC4的包也不能到R5；

STEP4: 在公网边界路由器配置ACL过滤私网地址:

首先过滤私网地址: (c)#access-list 1 deny 10.0.0.0
0.255.255.255 →access-list 1 deny 192.168.0.0 0.0.255.255
→access-list 1 deny 172.16.0.0 0.15.255.255 →access-list 1
permit any any →int s0(在ISP连接用户的路由器接口上调用) →ip access-group 1 in ;

然后允许ISP卖出的地址连入: (c)#access-list 13 permit 13.0.0.0
→router-map R-13 permit 10 →match ip add 13 →set metric 1
→router rip (要重分布) →redistribute connected route-map R-13 ;

STEP5: 配置NAT:

注意实验需求要在私网边界内口关闭ARP: (c)#int e0 →no ip proxy-arp ;

proxy-arp ;

然后配置NAT的内/外口: (c-i)#ip nat inside/outside ;

接着配置NAT地址转换公网地址: (c)#ip nat inside(我方) source(所发起的) static 192.168.1.2(私网地址) 13.0.0.10(公网地址);

STEP6: 一些查看命令:

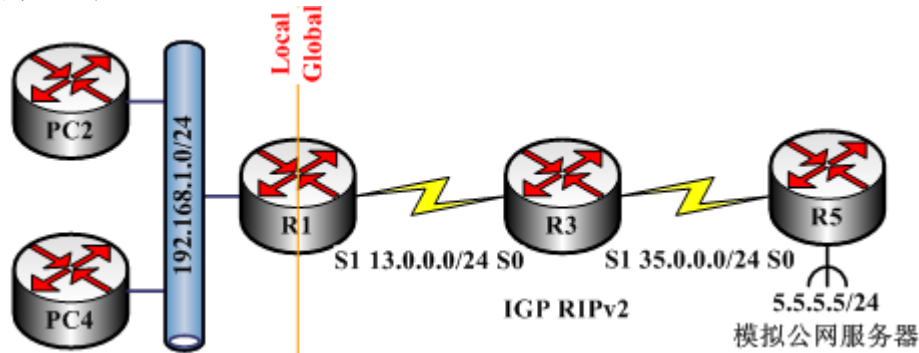
R1#show ip nat translations(察看NAT的转换表);

R1#debug ip nat (察看NAT的转换过程)的结果:

发送: s=192.168.1.2->13.0.0.10 d=5.5.5.5 ;

接收: s=5.5.5.5 d=13.0.0.10->192.168.1.2 ;

结论: 静态NAT不实用, PC4也要做NAT, 如果用户多的话, 要购买相当多的NAT公网地址。



LAB2: 中小型企业通常使用基于接口的端口复用(Overload):

STEP1: 定义NAT的外口/内口:

.....(同LAB1);

STEP2: 通过ACL, 定义准备进行NAT的内网的用户群:

(c)#access-list 1 permit 192.168.1.0 0.0.0.255 ;

STEP3: 进行基于接口的NAT端口复用:

(c)#ip nat inside source list 1(通过ACL指定内网用户) interface serial 1(NAT外口) overload(端口复用: 新IOS里面默认启用) ;

测试: 实现了192.168.1.*整个网段的用户共享一个IP地址连接到Internet。

LAB3: 使用Route-Map做基于接口的端口复用:

STEP1: 定义NAT的外口/内口, 定义内网用户群:

.....(同LAB2);

STEP2: 使用Route-map调用ACL:

STEP1: 定义NAT的外口/内口, 定义内网用户群:

.....(同LAB2);

STEP2: 使用Route-map调用ACL:

(c)#route-map X → match ip address 1 ;

STEP3: 进行基于接口的NAT端口复用:

(c)#ip nat inside source route-map X interface serial 1
overload ;

测试: 实现了192.168.1.*整个网段的用户共享一个IP地址连接到Internet。

LAB4: 大型企业用较多的公网IP利用POOL对不同的部门进行NAT转换:

STEP1: 构建拓扑并定义NAT的外口/内口:

修改IP后.....(同LAB3);

STEP2: 通过ACL, 定义准备进行NAT的内网的用户群:

(c)#access-list 10 permit 192.168.10.0 0.0.0.255 → access-list
20 permit 192.168.20.0 0.0.0.255 ;

STEP3: 在PC上指定网关:

实际工作中不需要此步, 但是实验中为了更全面的了解课程内容所以关闭了ARP就要配了;

首先为各个网段配置同一网段的网关IP, 可以使用secondary IP或者子接口配置: (c)#in e0 → ip add 192.168.10.1 255.255.255.0 → ip add
192.168.20.1 255.255.255.0 secondary ;

然后在PC设定网关: (c)#ip default-gateway 网关IP ; 注意养成好习惯: PC2/PC4都要测试能否ping通自己的网关!!!!

STEP5: 在私网边界路由器将从ISP买到的公网IP放入地址池(pool):

掩码可以使用两种格式, 这是等价的: (c)#ip nat pool PL-A
~~13.0.0.8~~(起始IP) ~~13.0.0.11~~(终止IP) prefix-length 24/netmask
255.255.255.0(IP长度:ISP给定) → ip nat pool PL-B 13.0.0.12
13.0.0.0.15 prefix-length 24 ;

STEP6: 为不同的部门进行基于不同POOL的NAT转换:

(c)#ip nat inside source list 10 pool PL-A overload → ip nat
inside source list 20 pool PL-B overload ;

然后可以用一些命令查看: R1#sh ip nat translation ;

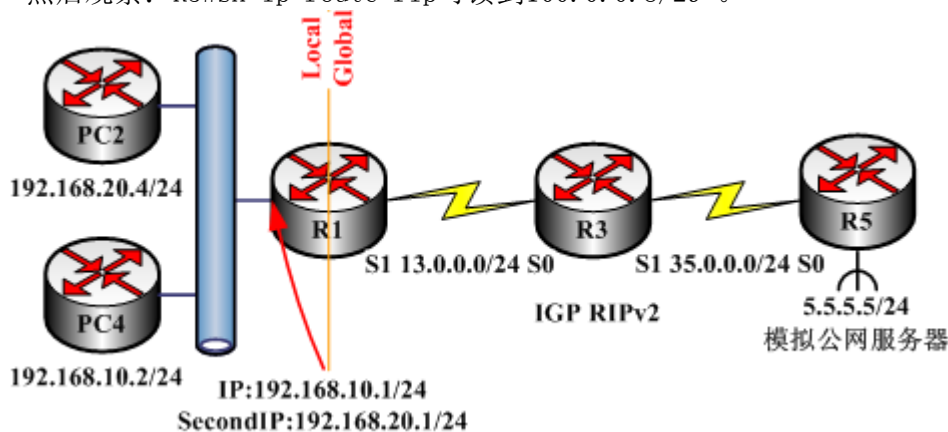
STEP7: 在公网边界路由器的操作:

为了让R5有回包路由, 在R3上重分布静态路由到RIP中:

(c)#ip route 100.0.0.0 255.255.255.248 13.0.0.1 (定义静态路由)
→ ip prefix-list 100 permit 100.0.0.8/29 → route-map R-100 → match
ip add prefix-list 100 → set metric 1 (定义重分布的范围) → route rip
→ redistribute static route-map R-100 (重分布);

然后观察: R5#sh ip route rip可读100.0.0.8/29 。

然后观察：R5#sh ip route rip可读到100.0.0.8/29 。



LAB5：通过NAT实现镜像服务器的负载均衡(实际工程少用→了解即可)：

STEP1：定义NAT外口/内口：

同LAB4；实验需求：公网上能访问这两台服务器，而且负载均衡地到PC2/4；

STEP2：通过ACL定义我方公网地址成为来自外网访问的目标地址：

(c)#~~access-list~~ ~~10~~ permit 13.0.0.10 0.0.0.0(精确匹配，只转换一个地址)；

STEP3：定义内网服务器群的地址池：

(c)#ip nat pool ~~p1~~ 192.168.10.2 192.168.10.3 prefix-length 24 type rotary(轮转类型)；

STEP4：定义NAT：

(c)#ip nat inside destination list ~~10~~ pool ~~p1~~；

STEP5：为了实验目的：

PC上：(c)#~~line~~ vty 0 4 →no login (不需要密码就可以被TELNET→危险→现实请谨慎)；

然后在R5上：(c)#~~telnet~~ ~~13.0.0.10~~ 可以看到是轮流的。

LAB6：使用一个接口的公网IP对外提供多个不同的服务器：

STEP1：定义NAT外口/内口：

同LAB5；注意：静态的TCP端口映射，静态的端口复用；

拓朴使用一个接口的公网IP对外网提供多个不同的服务器：如果你向访

访问我的FTP（TCP10000号端口或21口）或者WWW服务器（20000号端口或80口）；

STEP2: 在私网边界路由器构建服务器:

FTP服务器: (c)#ip nat inside source static tcp ~~192.168.10.2~~ 21 ~~13.0.0.1~~ 21 ;

WWW服务器: (c)#ip nat inside source static tcp ~~192.168.10.4~~ 80 ~~13.0.0.1~~ 80 ;

∴没有服务器∴实验中无法达成;

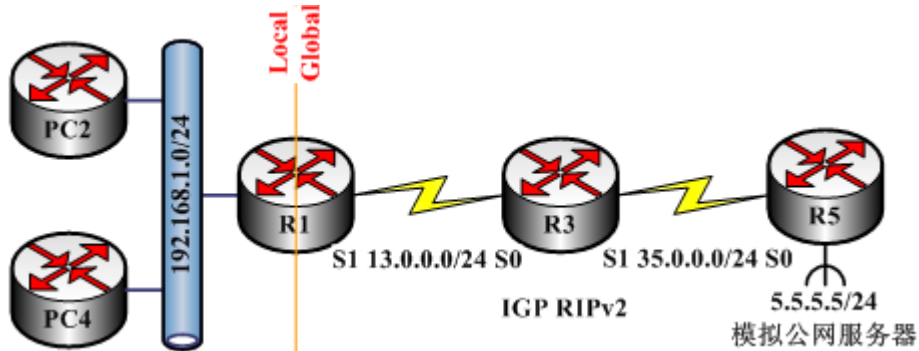
实验中可以把私网端口改为23号端口代替（公网端口不要改）:

FTP服务器: (c)#ip nat inside source static tcp ~~192.168.10.2~~ 23 ~~13.0.0.1~~ 21 ;

WWW服务器: (c)#ip nat inside source static tcp ~~192.168.10.4~~ 23 ~~13.0.0.1~~ 80 ;

R1: (c)#line vty 0 4 →no login ;

R5: (c)#TELNET ~~13.0.0.1~~ (进入了R1, 无端口号) →telnet ~~13.0.0.1~~ 21 (进入了FTP服务器) →telnet ~~13.0.0.1~~ 80 (进入了WWW服务器);



nat

```
r1(config)#do sh run
Building configuration...
```

```
Current configuration : 737 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
```

```
no ip domain lookup
!
!
!
!
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ip nat inside
!
interface Serial0
 no ip address
 shutdown
!
interface Serial1
 ip address 13.1.1.1 255.255.255.0
 ip nat outside
!
 ip nat inside source list 1 interface Serial1 overload
no ip http server
ip classless
ip route 192.168.1.0 255.255.255.0 Serial0
ip route 192.168.1.0 255.255.255.0 Serial1
!
!
!
access-list 1 permit 2.2.2.0 0.0.0.255
access-list 1 permit 10.1.1.0 0.0.0.255
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

静态NAT配置

```
ip nat inside static source-IP des-IP
指定什么端口是inside什么是outside
int e 0
ip nat inside
int s 0
```

```
ip nat outside
```

动态NAT配置

```
#ip nat pool "name" "start-ip" "end-ip" (只有公网地址作池才有意义) mask x.x.x.x
```

```
access-list "access-list-number" permit source [source-wildcard]
```

```
ip nat inside source list
```

example:

```
ip nat pool ccna 12.1.1.4 12.1.1.5 mask 255.255.255.0
```

```
ip access-list standard NAT
```

```
10.1.1.4 0.0.0.1
```

```
ip nat inside source list NAT POOL CCNA
```

```
clear ip nat translations * 清空nat表, 让地址池重置
```

Overloading端口复用PAT--主流应用

```
router(config)#access-list access-list-number permit source source-wildcard
```

```
ip nat source list NAT interface serial 0 overload
```

```
sh ip nat translation
```

3550和29的配置

```
sh ver:35
```

```
Switch#sh ver
```

```
Cisco IOS Software, C3550 Software (C3550-IPSERVICESK9-M), Version 12.2(25)SEE1, RELEASE SOFTWARE (fc1)
```

```
Copyright (c) 1986-2006 by Cisco Systems, Inc.
```

```
Compiled Mon 22-May-06 08:08 by yenanh
```

```
Image text-base: 0x00003000, data-base: 0x00DC0370
```

ROM: Bootstrap program is C3550 boot loader

Switch uptime is 5 minutes

System returned to ROM by power-on

System image file is "flash:/c3550-ipservicesk9-mz.122-25.SEE1.bin"

This product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:

<http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

<http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

If you require further assistance please contact us by sending email to
export@cisco.com.

Cisco WS-C3550-48 (PowerPC) processor (revision H0) with 65526K/8192K
bytes of memory.

Processor board ID CAT0733Z2K3

Last reset from warm-reset

Running Layer2/3 Switching Image

Ethernet-controller 1 has 12 Fast Ethernet/IEEE 802.3 interfaces

Ethernet-controller 2 has 12 Fast Ethernet/IEEE 802.3 interfaces

Ethernet-controller 3 has 12 Fast Ethernet/IEEE 802.3 interfaces

Ethernet-controller 4 has 12 Fast Ethernet/IEEE 802.3 interfaces

Ethernet-controller 5 has 1 Gigabit Ethernet/IEEE 802.3 interface

Ethernet-controller 6 has 1 Gigabit Ethernet/IEEE 802.3 interface

48 FastEthernet interfaces

2 Gigabit Ethernet interfaces

The password-recovery mechanism is enabled.

384K bytes of flash-simulated NVRAM.

Base ethernet MAC Address: 00:0D:BD:CF:FA:00

Motherboard assembly number: 73-5701-09

Power supply part number: 34-0967-01

Motherboard serial number: CAT07330AHT

Power supply serial number: DTH07310QPW

Model revision number: H0

Motherboard revision number: A0

Model number: WS-C3550-48-SMI

System serial number: CAT0733Z2K3

Configuration register is 0x10F

sh ver:29

Switch(config)#do sh ver

Cisco Internetwork Operating System Software

IOS (tm) C2950 Software (C2950-I6K2L2Q4-M), Version 12.1(22)EA10,

RELEASE SOFTWARE (fc2)

Copyright (c) 1986-2007 by cisco Systems, Inc.

Compiled Tue 08-May-07 12:18 by myl

Image text-base: 0x80010000, data-base: 0x8067C000

ROM: Bootstrap program is C2950 boot loader

Switch uptime is 9 minutes

Switch uptime is 9 minutes
System returned to ROM by power-on
System image file is "flash:/c2950-i6k2l2q4-mz.121-22.EA10.bin"

This product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:

<http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

If you require further assistance please contact us by sending email to export@cisco.com.

cisco WS-C2950-24 (RC32300) processor (revision J0) with 19925K bytes of memory.

Processor board ID FOC0822S11E

Last reset from system-reset

Running Standard Image

24 FastEthernet/IEEE 802.3 interface(s)

32K bytes of flash-simulated non-volatile configuration memory.

Base ethernet MAC Address: 00:0D:28:61:62:00

Motherboard assembly number: 73-5781-11

Power supply part number: 34-0965-01

Motherboard serial number: FOC06500G8M

Power supply serial number: PHI071900ED

Model revision number: J0

Motherboard revision number: A0

Model number: WS-C2950-24

System serial number: FOC0822S11E

Configuration register is 0xF

RemoteAccess

3.0—远程的基础知识

RA概述:

remote access:

广域网的远程连接, 按L1分类:

1:通过电路交换网络实现的专线:(circuit switching)

~~~~~

~~~~~

1.1:通过真实的专用物理链路,实现的专线:
一般是通过同步串行链路(V.35)连接的,其支持的二层封装协议主要包括:
HDLC/PPP/SLIP

1.2:通过TDM网络(时分多路复用)实现的专线:
典型有:E1
其支持的二层封装协议与物理专线一样

2:按需拨号的电路交换网络(On-Demand circuit Switched)

~~~~~

一般用于网络的链路备份。

常见的业务包括:ISDN/PSTN(其中PSTN通常是以异步串行连接)  
(ISDN也可以通过异步串行连接)  
ISDN分为两种业务:(在中国ISDN交换机类型:Basic-net3)(全数字信道)  
BRI:2B+D  
PRI:30B+D

**B信道:**是用户用于传输数据的信道(用户数据信道),其带宽为64KBPS

**D信道:**是信令信道,不传输用户数据。

PRI:D带宽是64kbps; BRI:D带宽是16kbps

PSTN:56kbps(48~52kbps)(模拟信道)

## 3:包交换/分组交换(packet switching)

~~~~~

在开始传输数据之前,必须事先建立一条VC(虚电路),用于数据包的传输。

用户的终端设备(CPE)通过同步串行链路连接ISP(局端CO)
常见业务:X.25/FR/ATM

4:宽带接入/Broadband Access

~~~~~

有线宽带:

xDSL:主要是传统的电信运营商所经营的网络,主要使用传统的固话网(双绞线),作为最后一公里的末端输入。  
(语音+data)

Cable:主要是传统的有线电视运营商所经营的网络,主要使用传统的CATV网络,通过线路的双向改造,作为最后一公里的末端接入。(同轴电缆)  
(视频+data)

powerLine Modem:  
(电能+Data)

powerLine Modem:  
(电能+Data)

无线宽带:  
CDMA/GPRS/PHS/3G

### HDLC:

HDLC一般不推荐，原因有两个：

1：Cisco的HDLC帧头格式，携带了一个cisco的私有位：  
其好处：实现HDLC的环境中，支持多协议：IP/IPX/AT  
(appletalk)  
其缺点：只能跟CISCO的设备互通，不能兼容各厂商设备。

(原因是：标准的HDLC只支持单协议：IP)  
CISCO默认在串口中，以HDLC为2层封装协议。

2：HDLC协议，本身不支持认证，无法保证安全性

建议使用PPP，PPP有多种可选模块，可以提高网络安全性，提升性能。  
(PPP可支持认证)

SLIP，相当于是PPP前身，功能单一，趋向淘汰

在CISCO设备上，串行链路默认使用CISCO HDLC，在华为的设备上，默认使用PPP

### PPP (Point-to-point protocol)

PPP是业界开放性的标准，支持多协议环境，所有的厂商都可以支持。

HDLC/PPP的对比：  
HDLC不支持多协议，PPP支持多协议  
HDLC不支持认证，PPP支持认证

LCP (link control Protocol)  
负责对L1的物理层链路，进行链路的建立，控制，维护，



测试:

R2#ping 3.3.3.3!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

R1#PPP链路的对方接口的32位主机路由:

C                13.0.0.3/32

R1(config)#in serial 1

R1(config-if)#no peer neighbor-route(取消本路由表里面的32位明细路由, 仅对自己有效)

### PPP authentication

~~~~~

PAP/CHAP (PPP的认证, 是链路的认证)

PAP (password authentication protocol)

~~~~~

两次握手, 建议在网络工程中使用双向认证。

两次握手:

- 1: 被认证方, 将对方所定义的账号/密码, 以明文方式, 发送给主认证方。
- 2: 主认证方, 把收到的账号/密码, 与自己的数据库进行核对后, 发回认证成功与否的信息。

PAP的缺点: 账号/密码以明文方式, 在链路上传输, 不安全

### LAB2: PAP认证 (R1-R3)

~~~~~

step1:确认链路已经是封装为PPP链路

step2:在本路由器的数据库中, 为对方构建账号/密码:

R1 (config) #username xx password xx

R3(config)#username yy password yy

step3:选定PPP的认证方式为PAP

R1/3 (config-if)#PPP authentication PAP

step4:将 “自己在对方数据库中的账号/密码, 发送给对方, 供对方进行校验

R1(config-if)#ppp pap sent-username XX password XX

R3(config-if)#ppp pap sent-username YY password YY

(密码用户名大小写敏感)

观察:

观察:

R3#debug ppp authentication (PPP的认证)

LAB3: 使用”主机名“作为”用户名“的简化的PAP认证:

step1:

R1(config)#username R3 password R3

R3(config)#username R1 password R1

step2:

R1(config-if)#PPP pap sent-username R1 password R1

R3(config-if)#ppp pap sent-username R3 password R3

PPP chap认证 (challenge handshake authentication protocol)

3次握手:

chap的优点:

从不在链路传送密码, challenge (X) 和response (Y) 都是随机数, 这两者间是不可逆运算的, 可以确保密码不被破译, 保证网络的安全性。

1: 主认证方的路由器, 发出随机数 (X=9)

2: 被认证方的路由器, 将接收的随机数, 和事先定义好的密码=7, 一起放入MD5的加密器, 进行HASH算法的计算加密, 把得到的数值y=32, response的形式, 发送给主认证方。

3: 主认证方, 同样进行与第2步相同的操作, 将得到的数值y', 与从被认证方发来的Y, 进行比较:

如果一致, 发出认证成功信息:

如果不一致, 发送认证失败信息

LAB4: CHAP认证:

step1: 确认链路已经是封装为PPP链路

step2: 为对方建账号/密码: (特别注意: 密码必须一致)

R1(config)#username GZ password G-S

R2(config)#username SH password G-S

step3: 选定认证方式是CHAP:

R1/2(config-if)#ppp authentication chap

step4: 选定某组账号密码, 进行CHAP认证

R2(config-if)#ppp chap hostname GZ

R2(config-if)#ppp chap password G-S

step4:选定某组账号密码，进行CHAP认证

```
R2(config-if)#ppp chap hostname GZ
```

```
R2(config-if)#ppp chap password G-S
```

```
R1(config-if)#ppp chap hostname SH
```

```
R1(config-if)#ppp chap password G-S
```

打开接口，观察CHAP认证过程

” authentication failed ” 原因是双方密码不一致

“MD/DES compare failed”

step5:在CHAP认证中，密码必须一致：

```
R1(config)#username xx password xx
```

```
R2(config)#username xx password xx
```

认证成功

LAB6：使用主机名作为用户名，简化的CHAP认证

step1:直接使用路由器的主机名，进行CHAP认证：

```
R1(config)#username R2 password xx
```

```
R2(config)#username R1 password xx
```

step2:在PPP接口中，只需要以下命令：

```
interface serial 0
```

```
encap ppp
```

```
ppp authentication chap
```

PPP/compression PPP 压缩

PPP压缩可以提高PPP链路的带宽利用率。

支持3种压缩算法：

1: mppc

2: predictor

3: stac

step1:按图配置好IP，确认L1/L2/L3（RIP）

step2: 将R2-R4链路更改为PPP链路

step3:在R2-R4的PPP链路的接口中，启动PPP压缩：（3选1）

3-1: (if-s0) compress mppc

3-2: (if-s0) compress predictor

3-3: (if-s0) compress stac

3-4: TCP头压缩：（语音的数据包，其数据净荷很小，头大身小情况）

```
(config-if) #ip tcp header-compression
```

3.1—HDLC/PPP

同步串行链路 (Serial Point-to-Point Link) 的封装

3.1—HDLC/PPP

高级数据链路控制HDLC (High-Level Data Link Control) :

是由ISO开发的面向位的同步数据链路层协议，由SDLC协议发展而来。HDLC通过使用帧字符和校验和而规定了在同步串行线路上的封装方式；一般不推荐HDLC；

HDLC不支持多协议IP；Cisco为了使其支持多协议而在其帧头格式上增加了一个私有位：支持多协议了IP/IPX/AT (appletalk) 但是不再兼容其他品牌；

HDLC还不支持认证因此无法保证网络安全；

Cisco的设备默认封装HDLC。

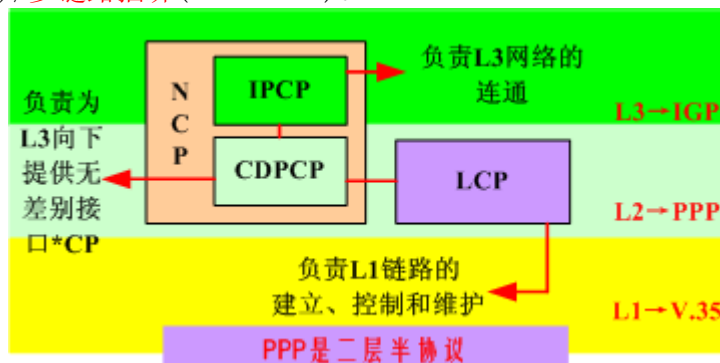
点到点协议PPP (Point-to-Point Protocol) :

PPP是业界开放性的标准，支持多协议环境，兼容所有品牌；华为的设备默认封装PPP；

PPP的前身是串行线路协议SLIP (Serial Line Interface Protocol)，因功能单一而趋向淘汰；

PPP分成两个部分：链路控制协议LCP (Link Control Protocol) 负责对L1物理层进行链路的建立、控制和维护；网络控制协议NCP (Network Control Protocol=IPCP+CDPCP) 负责为L3网络层向下提供无差别的接口(*CP)；因此可以说PPP是二层半协议也就是具有部分L3功能的L2协议；

LCP包含4大网络模块：认证(Authentication)/压缩(Compress)/回拨(Callback)/多链路捆绑(Multilink)。



PPP的认证:

∴PPP是L2协议∴PPP的认证是链路认证→一旦认证成功就接通链路在链路Down掉以前都不再认证了；PPP认证有两种：

链路Down掉以前都不再认证了；PPP认证有两种：

明文密码认证PAP (Password Authentication Protocol)：二次握手
(被认证方发送账号密码→主认证方对比账号密码后根据结果作出回应)；PAP的账号密码在网上传播，不安全；

密文密码认证CHAP (Challenge Handshake Authentication Protocol)：三次握手(主认证方发送乱码challenge→双方同时使用MD5不可逆算法运算并且被认证方发送结果→主认证方对比运算结果后根据结果作出回应)；CHAP的账号密码不会在网上传播，安全性好，推荐使用。

PPP的协商过程：

- 1: Interface Serial0, changed state to up (L1 up);
- 2: LCP: State is Open ;
- 3: PPP的认证 (这是可选项目，如果进行认证，就必须成功，才有NCP的工作)；
- 4-1: sel IPCP: State is Open (IP);
- 4-2: sel CDPCP: State is Open (cdp: show cdp neighbor);
- 5: Line protocol on interface serial1, changed state to up (L2 up)。

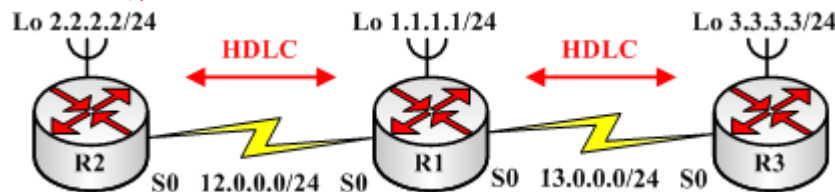
LAB1：使用HDLC封装点对点链路：

STEP1：在接口之间封装HDLC（默认的）：

构建拓扑，确认L1/L2/L3通达（L1：V.35/L2：HDLC/L3：routed网络协议IP和routing寻路协议RIP）；

然后查看即可（sh in s 0）：L1 is up, L2 is up……Encapsulation HDLC；

STEP2：运行RIP：



LAB2：使用PPP封装点对点链路：

STEP1：在串行链路的接口之间封装PPP：

接上一个拓扑，在R1/R3之间的接口：(c-i)#encapsulation ppp；

可以打开Debug后SH/no SH接口来查看PPP的协商：#debug ppp

encapsulation : L1 State is up → LCP State is open → IPCP State is open → CDPCP State is open → L2 State is up ;

再查看接口信息 (sh in s 0) : L1 is up, L2 is

up.....Encapsulation PPP, LCP is open open: IPCP, CDPCP.....;

观察完后最好关闭Debug: #no debug ppp encapsulation ;

STEP2: 在R3和R1之间互Ping:

能通的! 为什么? 路由的工作原理.....

在R1#PPP链路的对方接口的32位主机路由: C 13.0.0.3/32 ; 可以用

(c)#in s1 → no peer neighbor-route消去(仅对本R有效)。

路由(Route v. 动词): 核心词Forwarding转发, 过程为路由器从入接口收到信息后剥掉L2帧头FRAME(源地址和目标地址是前一个发此信息帧的路由器和接收此信息帧的本路由器), 然后查看L3层包头PACKET(源地址和目标地址是发此信息包的根源路由器和要接收此信息包的最终路由器), 接着查询自己的路由表找到目标地址的出接口和下一个路由器(下一跳), 并按照两者之间的封装方式重新封装L2帧头FRAME(源地址和目标地址是发送此信息帧的本路由器和下一个要接收此帧的路由器)并从出接口发出此帧; 不断封装→解封装→封装→解封装→封装: 跳数(Hops) 因此影响到路由的开销。

LAB3: PPP的认证:

STEP1: 进行PAP认证:

首先确定R1/R3链路已经是PPP封装 (sh in s 0) : L1 is up, L2 is up.....Encapsulation PPP ;

然后在双方的路由器上为对方建立账号/密码: (c)#username R3N password R3P 和..... ;

接着在双方接口上选定认证方式为PAP: (c-i)#ppp authentication pap ;

最后在双方接口上输入账号/密码: (c-i)#ppp pap sent-username R1N password R1P 和..... ;

可以打开Debug后SH/no SH接口来查看PPP的协商: #debug ppp encapsulation ;

在R3和R1之间互Ping能通的, 为什么;

STEP2: 进行CHAP认证:

这次在R1/R2上做, 首先确定链路之间已经是PPP封装, 不是就改 (c-i#en p) ;

然后同样是在双方的路由器上为对方建立账号/密码: (c)#username R2 password R1R2P 和.....; 特别注意: 双方的密码必需一致哦, 不然MD5运算的结果必定不同的, 还有账号要用对方的路由器名字; 可以分别试一试;

接着在双方接口上选定认证方式为PPP: (c-i)#ppp authentication chap ;

如果名字不同: : (c-i)#ppp authentication chap → ppp chap hostname R2 → ppp chap password R1R2P 好麻烦的.....

可以打开Debug后SH/no SH接口来查看PPP的协商: #debug ppp

可以打开Debug后SH/no SH接口来查看PPP的协商：#debug ppp encapsulation 。

LAB4: PPP的MLP(MultiLink Protocol):

STEP1: 构建拓扑:

将冗余链路上接口的原有配置都删除；需要在DCE端配时钟→R2的两端都是哦！

MLP是L2冗余，对比RIP的L3冗余：L3冗余收敛速度受路由协议的收敛速率影响通常较慢；

实验前用#sh controllers serial查看：

```
HD unit 0, idb = 0x939294, driver structure at
0x940860.....buffer size 1524 HD unit 0, V.35 DCE cable, clockrate
64000.....cpb = 0xE1, eda = 0x5078, cda = 0x508C.....RX ring with 16
entries at 0xE15000 ;
```

STEP2: 在链路两端的接口封装PPP并运行MLP:

(c)#in s0 →en ppp →ppp multilink ;

STEP3: 在双方路由器上创建虚拟模板接口:

(c)#Interface virtual-template 1 →ip add 24.0.0.2
255.255.255.252 →ppp multilink ;

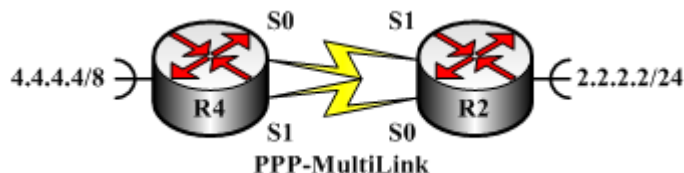
STEP4: 在MLP中, 调用虚拟模板:

R2/R4(c)#multilink virtual-template 1

设置虚拟接口后带宽加倍用#sh in virtual-a 1 得：Interface
address is 35.0.0.1/30.....BW 3088 Kbit.....

MLP两端上有了32位主机路由由(host route): : C 12.0.0.1/32 is
directly connected, Serial 0 ;

可以在接口中, 关闭PPP的主机路由Interface serial 0 →No peer
neighbor-route 。



3.2—帧中继①

3.2—帧中继①

帧中继 (Frame-Rekay) :

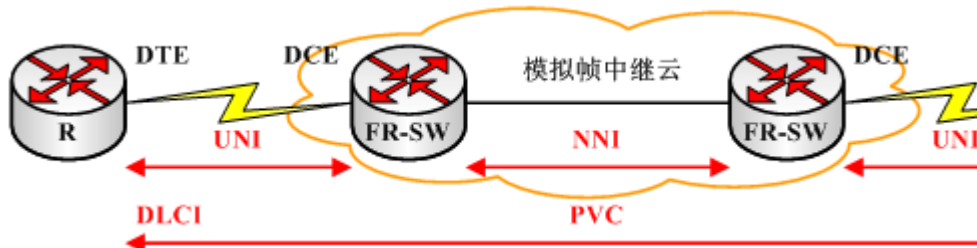
帧中继是面向连接的服务；是业界标准的纯L2数据链路层协议，它在所

所连接的设备之间采用HDLC封装，可以处理多条虚电路VC；帧中继比X.25更为有效，现在一般都认为应该用帧中继取代X.25；

虚链路VC（Virtual Circuits）：永久性虚链路PVC（Permanent VC）和交换式虚链路SVC（Switch VC）；帧中继在**运行前**必须构建好虚链路。

帧中继的接口类型：

帧中继的接口有用户网路接口UNI（User-Network Interface）和NNI（Network-Network Interface）两大类，UNI又分成DCE和DTE（**帧中继的客户端永远是DTE端**）；PVC两端的接口还应配置DLCI（Data-Link Connection Identifier）号用于FR的寻路。



帧中继的信令（Signaling）模式：

帧中继的信令使用本地管理接口协议LMI（Local Management Interface），有三种模式：

Cisco兼容；

ANSI T1.617 Annex D；

ITU-T Q.933a Annex A；

12.0以后的IOS都拥有检测LMI格式的能力→可以不指定→但是指定了就一定要一致！

PVC的连接模式：

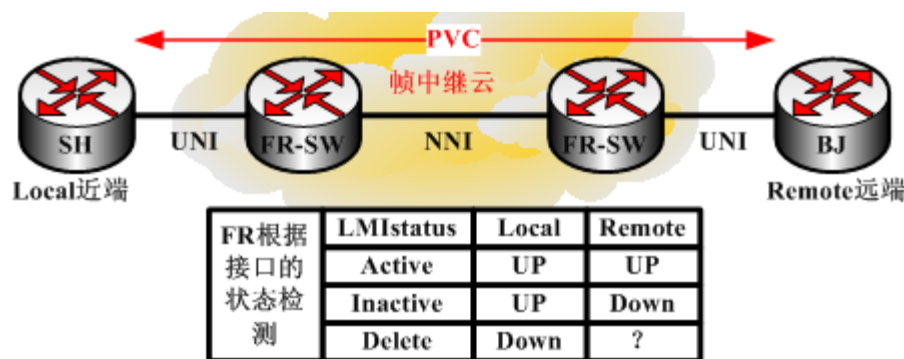
FR可以看做P2P链路→P2P（FR）：只可以有一个物理链路（PVC）但是可以有多个逻辑链路（DLCI）；

冗余连接（Full-Mesh）：每两个节点之间都有直接相连的连接，N个节点的Full-Mesh网路有 $N*(N-1)/2$ 个连接；Full-Mesh的可靠性极好但是成本极高，是高端用户的**必然选择**；

星型网路（Hub&Spoke）：每个节点（Spoke）都只与主机（Hub）直连，N个节点的Hub&Spoke网路有N-1个连接；Hub&Spoke成本低但是冗余性不好，而且存在因为水平分隔导致的一个接口有多条PVC时路由不能全面通达的问题（用帧中继子接口解决）。

PVC状态：

选举（active）、自己没配好（deleted）、对方没有配好（inactive）。



帧中继的映射表（Mapping Table）：

描述的是本机接口的DLCI号（FR的L2地址）与对端接口的IP（Route的L3地址）的匹配关系：对方的L3-IP 与本地的L2-DLCI进行映射。

LAB1：帧中继的基本配置（ISP部分）（基于每个物理连

接的一条PVC的Hub&Spoke架构）：

STEP1：配置帧中继交换机：

没有FR-SW，用路由器模拟；

首先把一个路由器R2变成帧中继交换机：(c)#no ip routing → frame-relay switching；

然后在接口封装帧中继：(c-i)#encapsulation frame-relay；

接着指定接口类型，而且由于FR-sw总是充当DCE所以还需要配置时钟，当然接口也要打开：(c-i)#frame-relay intf-type dce → clock rate 2000000 → no shutdown；

接着可以指定FR的LMI模式：(c-i)#frame-relay lmi-type cisco → 注意两边要一致；

STEP2：在帧中继交换机上配置FR路由：

要在两边接口有去有回的配置：(c-i)#frame-relay route ~~input~~ PVC interface ~~出接口~~ ~~出~~ outgoingPVC；然后可以用#show frame-relay route查看。

LAB2：帧中继的基本配置（USER部分）（基于每个物理

连接的一条PVC的Hub&Spoke架构）：

STEP1：配置用户端接口：

首先在接口封装帧中继：(c-i)#encapsulation frame-relay；

接着指定接口类型并打开：(c-i)#frame-relay intf-type dte（已默

默认, 可以不指定) → `no shutdown`;

嘿嘿, 路由的接口可别忘记配IP了, 否则……另外FR-SW也是交换机→两边要在同一个网段;

还有需要的话还要同步同一条PVC两端接口的LMI模式: (c-i)#`frame-relay lmi-type cisco`;

STEP2: 测试链路:

查看命令: #`show frame-relay pvc` 和 #`show frame-relay map`。

#`show frame-relay pvc` : DLCI=?……PVC status=ACTIVE……in s0

;

#`show frame-relay map` : Serial 0 (UP): ip? dlci? (映射表)

……dynamic (ARP学得), Broadcast (广播模式)……active (状态)……

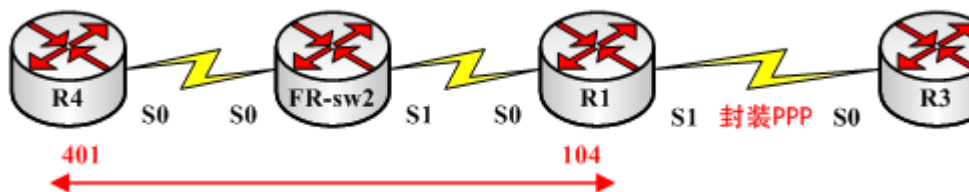
设置环回路口运行IGP然后Ping: ! ! ! ! !

还不通! ? 没交钱吧……

STEP3: FR的自动方向ARP:

考试中往往要求关闭ARP: (c-i)#`no frame-relay inverse-arp`;

然后手工建立映射表也就是在接口配置FR路由: (c-i)#`frame-relay map ip` 目的IP 出PVC broadcast。



LAB3: 三端口的FR-SW配置 (Full-mesh):

STEP1: 配置帧中继交换机对路由的接口:

首先把一个路由器R2变成帧中继交换机: (c)#`no ip routing` → `frame-relay switching`;

然后在接口封装帧中继: (c-i)#`encapsulation frame-relay`;

接着指定接口类型, 而且由于FR-sw总是充当DCE所以还需要配置时钟, 当然接口也要打开: (c-i)#`frame-relay intf-type dce` → `clock rate 2000000` → `no shutdown`;

STEP2: 在帧中继交换机上配置对路由器的FR路由:

要在两边接口有去有回的配置: (c-i)#`frame-relay route` 入PVC interface 出接口 出PVC; 然后可以用#`show frame-relay route`查看;

STEP3: 配置用户端接口:

首先在接口封装帧中继: (c-i)#`encapsulation frame-relay`;

接着指定接口类型并打开: (c-i)#`frame-relay intf-type dte` (已默认, 可以不指定) → `ip address 100.0.0.0 255.255.255.0` → `no shutdown`;

;

STEP4: 构建虚拟管道Tunnel:

由于以太网不运行FR, 所以在SW2/SW3之间的Ether口配置IP并构建

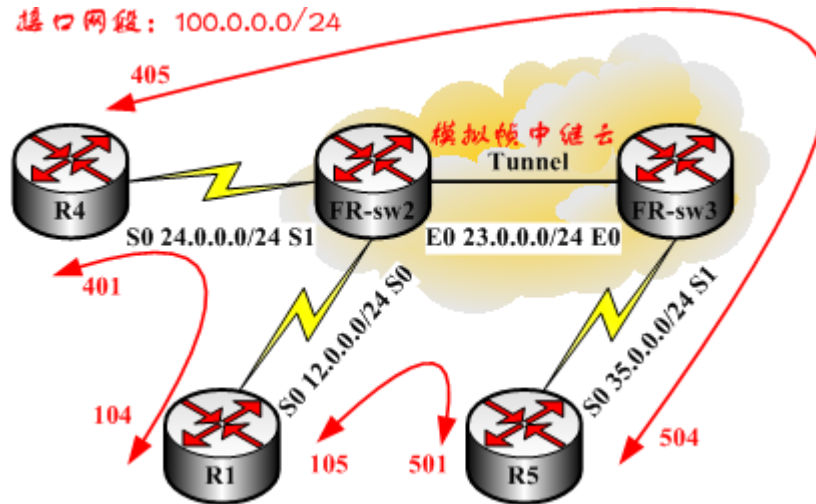
Tunnel（作用是在原帧基础上封装以太网包头→二次封装，不拆封的）以实现实验目的：(c)#`interface tunnel 1` → `tunnel source 23.0.0.2` → `tunnel destination 23.0.0.3` 和……；

STEP6: 配置R4到R5的PVC的FR路由：

首先对接tunnel管道：(c-i)#`frame-relay route` 入PVC interface 出 Tunnel IDLCI；

ISP(FR-SW)上#`show frame-relay route`查看FR路由：tunnel 1 1000 s1 504 active, serial 1 501 tunnel 1 1001 active……互Ping测试，通！！！！

接口网段：100.0.0.0/24



LAB4: ARP&Full-Mesh的PVC构建IGP网络：

STEP1: 构建拓扑：

接上个LAB；

STEP2: 运行IGP：

3-1: RIP over Full-mesh&ARP PVC FR网络：……建成；

3-2: EIGRP over Full-mesh&ARP PVC FR网络：(c)#`router eigrp 100` → `network 0.0.0.0` (所有接口，都运行EIGRP) → `no auto-summary`；∵LAB2是建立的Broadcast模式∴邻居关系顺利建成；

3-3: OSPF over Full-mesh&ARP PVC FR网络(c)#`router ospf 100` → `router-id 100.0.0.?` → `network 0.0.0.0 255.255.255.255 area 0` (所有接口都运行OSPF，OSPF反掩码不可以省略)；用#`show ip ospf interface`观察当前运行OSPF的接口有那些，并且特别注意接口的OSPF运行模式(network type)：FR的主接口默认是NBMA(non_broadcast)默认不主动发送组播hello包；用#`show ip ospf neighbor`查看邻居：因为OSPF运行模式是NBMA所以OSPF无法建立邻居 → 解决方案是将OSPF的接口运行模式改为BROADCAST → (c)#`interface serial 0` → `ip ospf network broadcast/point-to-point` (MA/BMA网络)。

LAB5:使用多点子接口MP替代物理接口构建IGP网络:

多点子接口和物理接口的配置方法完全一致(同LAB4);

考虑到网络将来的发展,扩展性→与LAB4相比工程中推荐使用LAB5 (full-mesh);

主接口配置: (c)#`in s 0 → en fr → no ip add → no sh` ;

子接口配置: (c)#`in s 0.100 multipoint (MP子接口) → ip add 100.0.0.1 255.255.255.0 → fr map ip 100.0.0.4 104 b → fr map ip 100.0.0.5 105 b` 。

3.2—帧中继②

3.2—帧中继②

子接口的种类:

首先是物理接口,子接口都构建在物理接口下,配置了子接口时物理接口不必配置IP;在FR中物理接口的默认模式为NBMA,

点对点子接口P2P (Point-to-Point): 在FR中P2P子接口的默认模式为P2P;

多点子接口MP (MultiPoint-to-MultiPoint): 在FR中MP接口的默认模式为NBMA;

点对多点子接口P2MP (Point-to-MultiPoint): 在FR中P2MP接口的默认模式为P2P;

注意: 子接口的种类一旦配置后即**不可更改**!

P2P子接口ping该子接口所在网段中的所有节点都能通(包括本机节点);

MP子接口ping该子接口所在网段中的所有节点前都必须手工做映射 Mapping(包括本机节点)。

子接口判断原则:

判断创建几个子接口: 一个子接口对应一个子网→在路由器的一个FR物理接口中对应着几个子网,就应该创建几个子接口;

判断子接口的类型(P2P/MP): 如果一个子接口的对方只有一个点那么接口类型是P2P; 如果一个子接口的对方是多于一个点那么接口类型是MP;

建议: **P2P子接口适用于Hub&Spoke**的网络拓扑(一条PVC对应一个IP子网,有多少个分支点/分公司就应该创建多少个P2P子接口,相当于一个点对点的串行接口); **MP子接口适用于Full-mesh**的网络拓扑(多条PVC对应一个IP子网,相当于一个MA接口)。

于一个MA接口)。

FR的水平分割:

在FR的主接口中无论是P2P还是MP, 其水平分割都是默认打开/Enable的; 而主接口/物理接口中其水平分割都是默认关闭的;

因此在运行DV协议时会出现分支点都只有到中心点的路由而没有分支点之间的路由的现象; 解决方法就是在中心点的子接口关掉它: (c-i)#no ip split-horizon 。

LAB1: 使用P2P构建Hub&Spoke网络(工程中推荐,

Hub&Spoke最佳选择方案):

STEP1: 按图构建拓扑:

按图构建, 配置FR-SW;

STEP2: 配置主接口:

在R1、R4、R5上: (c-i)#no ip address →encapsulation frame-relay (无需IP) →no frame-relay inverse-arp (关闭FR的自动反向映射) →no shutdown ;

STEP3: 配置FR的P2P子接口:

在R1、R4、R5上: (c)#interface serial 0.104 point-to-point →ip address 100.0.0.10 255.255.255.252 →no frame-relay inverser-arp (子接口和主接口要分别关闭反向ARP) →Frame-relay interface-dlci 104 和 ;

STEP4: 察看FR的映射:

在R1上#show fram-relay map 得: Serial0.104(up) broadcast(P2P子接口默认是允许广播通过);

STEP5: IGP over Hub&Spoke:

4-1: RIP (R1/R4/R5都运行RIP协议);

4-2: EIGRP (R1/R4/R5都运行EIGRP协议);

4-3: OSPF (R1/R4/R5都运行OSPF协议):

在R1: #show ip ospf interface serial 0.104得: Network type POINT_TO_POINT;

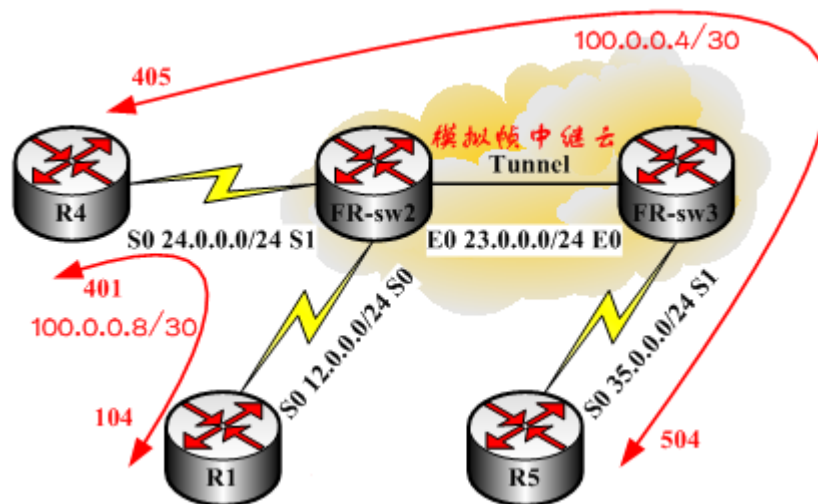
STEP6: 关于本实验中所有分支点的下一跳:

R4# any Router via 100.0.0.1 ;

R5# any Router via 100.0.0.5 ;

所以分支点之间的路由的下一跳是可达的→路由通达!

提醒: 注意OSPF的DR/BDR选举!!!



LAB2：使用MP构建Hub&Spoke网络（通过静态映射）：

STEP1：按图构建拓扑：

按图构建，配置FR-SW；

STEP2：配置主接口：

在R1、R4、R5上：(c-i)#no ip address → encapsulation frame-relay (无需IP) → no frame-relay inverse-arp (关闭FR的自动反向映射) → no shutdown ；

STEP3：配置FR的MP子接口：

在R1、R4、R5上：(c)#interface serial 0.100 multipoint-to-multipoint → ip address 100.0.0.1 255.255.255.0 → no frame-relay inverse-arp (也要关闭反向ARP) → frame-relay map ip 100.0.0.1 104 broadcast 和 …… ；

STEP4：IGP over Hub&Spoke (用#debug ip packet查看un all关闭)：

4-1：RIP (R1/R4/R5都运行RIP协议)：

通后R1、R5可以ping通R4但是不能互通→水平分割导致路由信息报文不能经过R4；关掉它解决：(c-i)#no ip split-horizon ；

发现仍然不通→直链路由优先级高→FR-SW无法识别Ether封装的帧→en failed→下一跳不可达；在分支点上分别做映射解决：R1(c-i)#frame-relay map ip 100.0.0.1 104 和 R5(c-i)#frame-relay map ip 100.0.0.1 504 ；

4-2：EIGRP (R1/R4/R5都运行EIGRP协议)：

EIGRP的水平分割和其他DV协议不同它是基于AS的，要在接口指定AS关闭：(c-i)#no ip split-horizon eigrp 99 ；

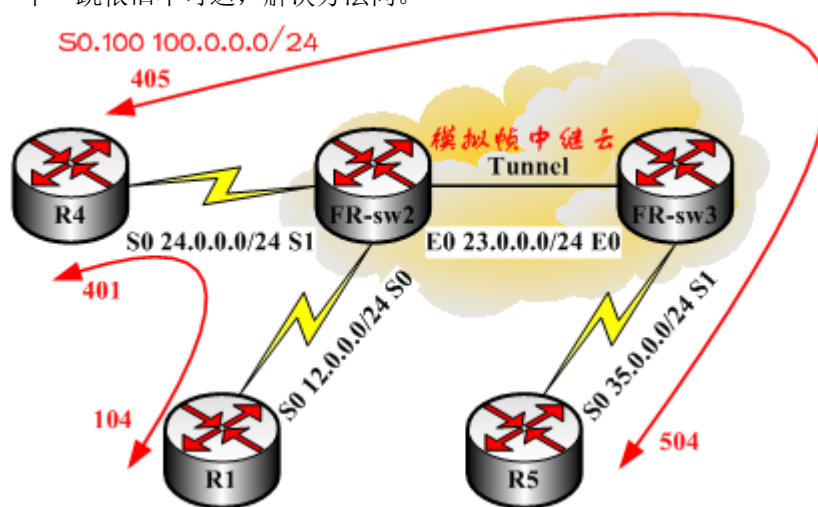
下一跳依然不可达，但是分支点的10之间可以互通（EIGRP的自动下一跳）→已经满足题目要求→映射不是必须建立→一定要求建立的话同上；

4-3：OSPF (R1/R4/R5都运行OSPF协议)：

首先是邻居问题，OSPF的接口运行模式为NBMA不允许广播流量通：映射时是否加broadcast无影响→邻居无法建成；通过单播更新也就是在Hub上使用 Neighbour 命令解决；

否加broadcast无影响→邻居无法建成；通过单播更新也就是在Hub上使用
Neighbour命令解决；

接下来发现DR/BDR混乱→影响OSPF的LSDB不正常；通过指定Hub为DR无BDR解
决：(c-i)# ip ospf priority 100/0 (DR/DR-other) ；
∴是LS协议所以没有水平分割问题；
下一跳依旧不可达，解决方法同。



LAB3：使用P2MP构建Hub&Spoke网络（通过静态映射）：

射）：

STEP1：按图构建拓扑：

按图构建，配置FR-SW；

STEP2：配置主接口：

在R1、R4、R5上：(c-i)#no ip address →encapsulation frame-relay (无需IP) →no frame-relay inverse-arp (关闭FR的自动反向映射) →no shutdown ；

STEP3：配置FR的NBMA的P2MP子接口：

在R1、R4、R5上：(c)#interface serial 0.100 point-to-multipoint non-broadcast (NBMA) →ip address 100.0.0.1 255.255.255.0 →no frame-relay inverser-arp (也要关闭反向ARP) →frame-relay map ip 100.0.0.4 104 broadcast 和 ；

STEP4：IGP over Hub&Spoke (用#debug ip packet查看un all关闭)：

4-1：RIP (R1/R4/R5都运行RIP协议)；

4-2：EIGRP (R1/R4/R5都运行EIGRP协议)；

4-3：OSPF (R1/R4/R5都运行OSPF协议)；

首先是邻居问题，OSPF的接口运行模式为NBMA不允许广播流量通：映射时是否加broadcast无影响→邻居无法建成；通过单播更新也就是在Hub上使用
Neighbour命令解决；

接下来是DR/BDR：NBMA无DR选举因此不成问题；

∴是LS协议所以没有水平分割问题；

自动下一跳→下一跳可达；

STEP5：扩展→BMA的P2MP运行OSPF：

全自动地！！！一切OK：邻居问题OK、DR问题OK、水平分割问题OK、下一跳依然OK。

3.3—ISDN

3.3—ISDN

综合业务数字网ISDN(Integrated Services Digital Network)：

ISDN主要有两种接口类型：分为BRI (2B+D=2×64+16Kbps)和PRI (带宽：T1/E1=28/30B+D=23/30×64+64Kbps)；

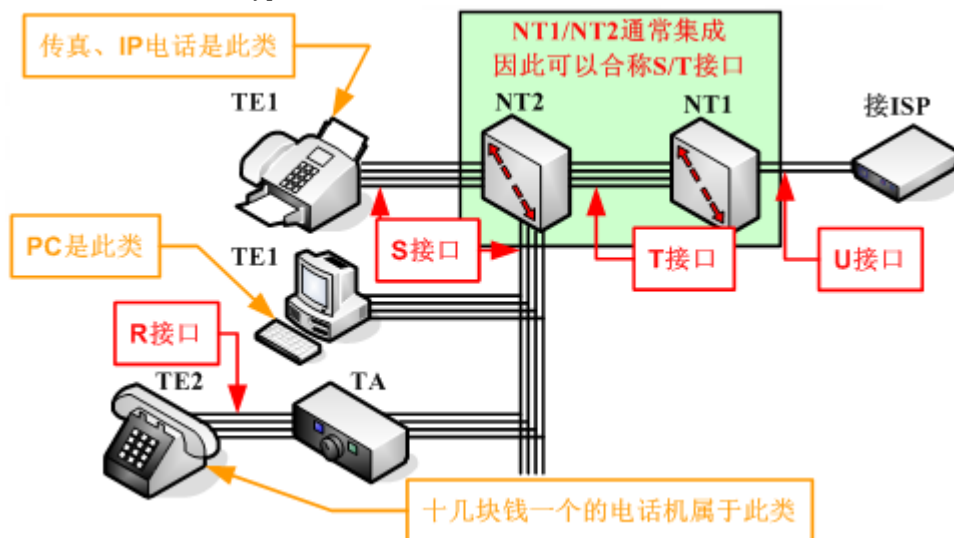
ISDN一般通过同步串行链路(V.35)连接，也可以采用异步串行路。

ISDN协议栈	D	B
L3	Q.931	2P
L2	Q.921	HDLC/PPP
L1	I.430/I431	

ISDN的交换机类型：

isdn switch-type basic-net3 (中国/欧洲标准)；

isdn switch-type basic-ni (北美标准)。



ISDN的接口命令：

物理命令：

isdn switch-type basic-ni (ISDN交换机类型，接口会自动继承全局配置)；

encapsulation hdlc (默认的L2封装HDLC)；

SPID xx (SPID号，用户服务种类的标识位，中国不需要)；

SPID xx (SPID号, 用户服务种类的标识位, 中国不需要);
 逻辑命令:
 Ip address 35.0.0.3 255.0.0.0 (配置IP地址);
 dialer map ip 35.0.0.5 broadcast 810888 (对方L3地址与对方L2地址的映射);
 dialer-list 3 protocol ip permit → int b0 → dialer-group 3
 (定义能够触发ISDN起拨的数据流, 此为全部可以)。

LAB1: ISDN的基本配置:

STEP1: L1/L2通:

(c)#isdn switch-type basic-ni (此命令在全局配置等同于在所有的接口配置) → interface bri0 → no shutdown;

用R3#show isdn status命令查看: Layer1 status:ACTIVE, Layer2 status:State = MULTIPLE_FRAME _ESTABLISHED (多帧已建立);

拨号测试: R3(c-i)#isdn (test) call interface bri0 810888; R4 (c-i)#isdn (test) disconnect interface bri 0 all; R3#show isdn active;

STEP2: L3通:

所有的配置命令都在ISDN接口中:

物理命令: (c-i)#isdn switch-type basic-ni → encapsulation hdslc;
 逻辑命令: (c-i)#Ip address 34.0.0.3 255.0.0.0 → dialer map ip 34.0.0.4 broadcast 810888 → dialer-list 3 protocol ip permit → int b0 → dialer-group 3;

L3测试: Ping 34.0.0.4 ! ! ! ! !

STEP3: 按需拨号DDR (Dial on Demand Route):

首先确保路由器两端来去都有正确路由: 使用默认路由;

然后确认上述路由下一跳的可达性(确认有到达下一跳的映射): (c-i) #dialer map ip 34.0.0.4 810888; 使用Show dialer maps察看映射表;

然后修改拥有触发起拨权限的数据流(即定义感兴趣流): (c)#access-list 10 permit 10.0.0.0 0.255.255.255 (access-list 10 permit 10.0.0.0 0.255.255.255 等价于access-list 10 permit 10.0.0.0 0.0.0.0即精确匹配) → Dialer-list 3 protocol ip list 10 → interface bri 0 → dialer-group 3;

ISDN的idle time是只能让“感兴趣流”复位的; 即使是ISDN上正在通信的非“感兴趣流”也不能让idle time复位!

DDR行为规则		之前	之后	
能自动起拨的	10.0.0.0	断	通	2
		通	通(此时会复位idle time)	3
不能自动起拨的	20.0.0.0	断	断	1
		通	通(即使通, idle time没复位)	4

STEP4: PPP的认证:

首先封装PPP: (c-i)#encapsulation ppp;

		通	通(即使通, idle time没复位)	4
--	--	---	----------------------	---

STEP4: PPP的认证:

首先封装PPP: (c-i)#encapsulation ppp ;

接着进行CHAP认证: (c)#username R3 password 0 R34 → interface
bri 0 → ppp authentication chap ;

STEP5: PPP的捆绑:

R3(config-if)#ppp multilink load-threshold 165 (255*0.65=
165, 255是最大值) (链路负荷/利用一个B信道的65%时, 启动第二B信道);

STEP6: PPP的压缩:

在R3-R5的PPP链路, 的接口中, 启动PPP压缩(3选1): (c-i)#compress
mppc/predictor/stac ;

TCP压缩(语音的数据包, 其数据净荷很小, 头大身小情况): (c-i)#ip tcp
header-compression ;

STEP7: PPP的回拨:

首先在R3请求回拨: (c-i)#ppp callback request ;

(↓ 以下全在R5做 ↓)

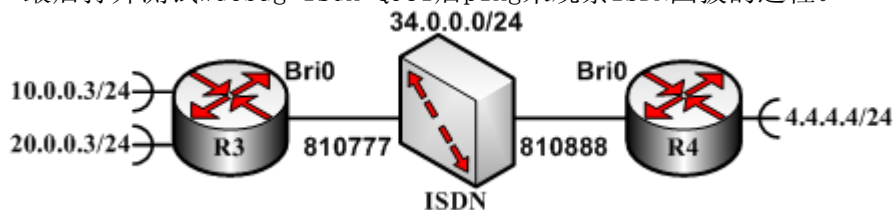
接着在R5接受回拨: (c-i)#ppp callback accept ;

同时要启用安全机制: (c-i)#dialer callback-secure ;

定义回拨用户组: (c)#map-class dialer CB → dialer callback-
server username ;

然后定义并且在R3输入用户名: R5(c-i)#dialer map ip 34.0.0.3
name R3 class CB broadcast 810777 和R3(c-i)#dialer map ip 34.0.0.4
name R4 broadcast 810888 ;

最后打开测试#debug isdn Q931后ping来观察ISDN回拨的过程。

**LAB1: ISDN做链路备份:****STEP1: 构建拓扑运行IGP:**

配置Bri口后(c)#router eigrp 90 → net 0.0.0.0 ;

STEP2: 定义起拨流量:

在R3/R4上: (config)#dialer-list 3 protocol ip permit → int b0
→ dialer-group 3 ;

STEP3: 定义备份接口:

在R3的S1上: (c-i)#backup interface bri 0 (定义ISDN是主链路接口
的备份, Bri口成为备份接口后down掉);

用#sh isdn static和#sh int bri 0查看Bri的备份状态;

STEP4: 使用虚拟的dialer接口做备份(dial-profile):

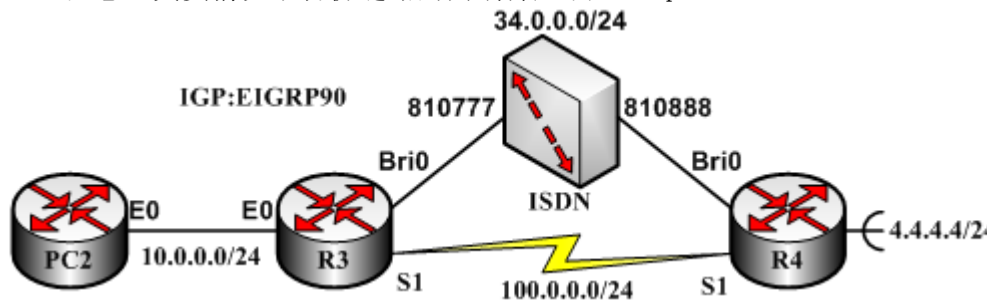
物理接口: (c-i)#encapsulation ppp →dialer pool-member 10(将物理接口放入dial-pool10中) →isdn switch-type basic-ni ;

逻辑接口(Dial接口): (c)#in dialer 3 →ip add 34.0.0.3
255.255.255.0 →en ppp →dialer pool 10 (在dial-pool10中调用物理接口)
→dialer string 810888 →dialer-group 3 →ppp authentication chap ;

定义为备份接口: (c-i)#backup int dialer 3 ;

可以设置一些参数: (c-i)#backup delay 5 15 (主链路断开5秒后开始拨号; 主链路恢复15秒后断开ISDN); (c-i)#backup load 50 20 (主链路的带宽利用率高于50%时启动ISDN; 主链路+备份链路的带宽利用率低于20%时断开ISDN);

注意: 真实情况下备份链路的两端都应该backup-if。



3.4—乱七八糟①

3.4—乱七八糟①

E1线路知识要点:

中国/欧洲使用E1(欧洲标准); 在北美/日本使用T1(北美标准);

一条E1是带宽为:2.048M的链路, 用PCM编码.

一个E1的帧长为256bit, 由32个大小8bit的时隙(timeslots)/信道组成; 一个E1接口每秒有8K个帧通过, 即E1的带宽为 $8k \times 256 = 2048k\text{bps}$;

因为每个时隙为8bit, 所以一个时隙的带宽是 $8\text{bit} \times 8k = 64k\text{bit}$, 即一条E1中含有32个64k信道/时隙。

E1帧结构:

E1有不成帧、成帧和成复帧三种方式:

不成帧的E1(透明模式)中: 所有32个时隙都可以用于有效数据的传输;

成帧的E1中: 0时隙用于帧同步数据的传输, 其余31个时隙都可以用于有效数据的传输;

成复帧的E1中: 0时隙用于帧同步数据的传输, 16时隙用于信令的传输, 其余31个时隙(1~15和17~31)都可以用于有效数据的传输。

E1的中继线(Trunk):

原始的使用方法/场合与成复帧时相同: 0时隙用于帧同步数据的传输,

原始的使用方法/场合与成复帧时相同：0时隙用于帧同步数据的传输，16时隙用于信令的传输，其余31个时隙（1~15和17~31）都可以用于有效数据的传输。

E1的三种典型使用方法：

透明传输的专线：将整个2M用作一条链路，如DDN2M；

CE1：信道化E1（channel-E1）；就是把2M的带宽用作若干个64K组合，一般写成N*64，如128K、256K等；你可以只利用其中的几个时隙也就是只利用n个64k；CE1最多可有31个信道承载数据；必须接在cel/pri接口上；

E1最本来的用法：在用作语音交换机/程控交换机（PSTN）的数字中继线（Trunk）时的最本来的用法。

E1的两种接口（G. 703）：

同轴电缆：BNC头、圆头，阻抗：非平衡的75ohm；

双绞线：RJ-45头，阻抗：平衡的120ohm。

工程中常见的E1连接方法：

PRI是其中最常用的一种接入方式，采用PRA信令：

相对便宜的连接方法：用2611系列的广域网接口卡（1T/2T：WAN WIC DB-60/SmaSerial）经过V.35-G.703协议转换器连接E1线；

相对较贵的连接方法：使用E1卡（目前DDN的2M速率线路通常是经HDSI线路拉至用户侧）；E1可由传输设备出的光纤连接用户侧的光端机提供E1服务

（连接方式：ISP的光传输网络设备→光纤→光端机→2根BNC同轴线→G703转V35转换器→同步串行链路→路由器的同步串口；ISP的光传输网络设备→光纤→光端机→2根BNC→DB15-G703转V35转换器）；

阻抗转换Cable：

BNC-DB15、BNC-RJ48。

绝对线路号：

Con口线路号为0；

异步串口线路号为1~X（固定路由上看情况；模块化路由器上为X=8×网络模块数量→网络模块接口8个一组→八爪鱼→没有时预留4×4个→Aux最少129）；

Aux的绝对线路号为X+1；

LAB1：使用E1线路实现多个64K专线连接：

STEP1：了解相关命令：

进入controller配置模式：contriller t1/e1 number ；

选择帧类型：framing crc4/no-crc4 ；

选择line-code类型：linecode ami/b8zs/hdb3 ；

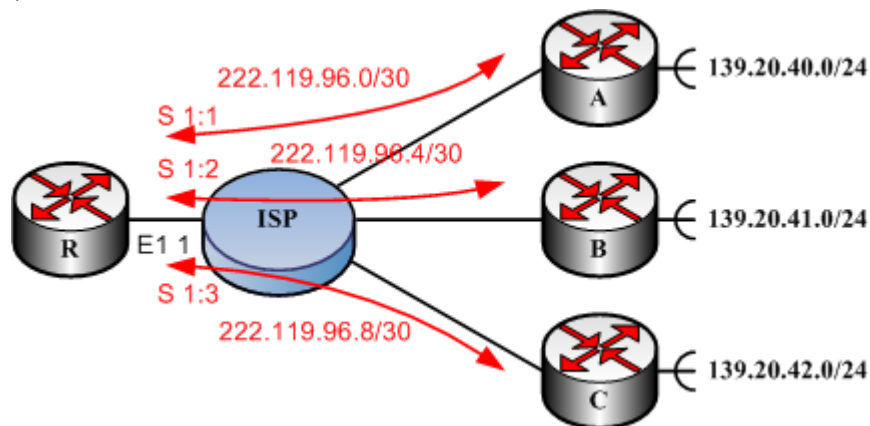
STEP2：具体配置：

STEP2: 具体配置:

E1连接3条64K专线, 帧类型为NO-CRC4, 非平衡链路, 路由器具体设置如

下:

```
hostname shanxi ;
!
controller E1 1 （进入E1控制器） ;
!
framing NO-CRC4 ;
linecode hdb3 ;
!
channel-group 1 timeslots 1 （建立逻辑通道组与时隙的映射，
timeslots 1（类似PVC）对应下面的s1:1（类似Sub-if））；
channel-group 2 timeslots 2 ;
channel-group 3 timeslots 3 ;
exit ;
!
in s1:1 ;
ip add 222.119.96.1 255.255.255.252 ;
!
in s1:2 ;
ip add 222.119.96.5 255.255.255.252 ;
!
int s1:3
ip add 222.119.96.9 255.255.255.252 ;
!
ip route 139.20.40.0 255.255.255.0 s1:1 （配置默认路由）；
ip route 139.20.41.0 255.255.255.0 s1:2 ;
ip route 139.20.42.0 255.255.255.0 s1:3 ;
!
```



LAB2: 带外网管 (Modem Connections) : 通过异步

Modem/(PSTN程控交换机/语言交换网/7号信令网/SS7) 远程控制路由器的AUX口:

STEP1: 在开始通讯前的准备工作:

使用Con线通过Console口进行近程Telnet配置来控制Aux和Modem之间的通讯;

先查看Aux绝对线路号: Con口线路号为0、异步串口线路号为1~X (X=8 × 网络模块数量 → 网络模块接口8个一组 → 八爪鱼)、Aux的绝对线路号为X+1;

进入Aux口: (c)#line 65/aux 0 (使用绝对线路号/相对接口名进入Aux口) → flow control hardware (硬件流控) → transport input all (容许所有协议) → modem inout (容许出/入方向的呼叫) → password 123 (配置密码) → login (登录许可, 同Con口的配置, no login表示不设防);

然后设置enable密码即可;

STEP2: 使用AT命令集配置Modem:

笔记本通过Com口、RJ-45转DB-25转换器在Modem的DB-25接口配置, 每个品牌的AT命令集不同, 需要查询文档; 常用的:

AT (开始配置) → AT& (查看配置) → AT&f (恢复默认, 有些为AT&F1) → AT&S0=3 (设置应答指示响铃次数, AT&S0=0表示不响铃) → AT&W (保存设置) → AT&E0 (关闭字符回显) → AT&E1 (打开字符回显);

STEP3: 按图配置;

STEP4: 拨号 → 使用超级终端。



LAB3: PPPoE (ADSL的服务: Route As PPPoE's client) :

STEP1: VPDN(虚拟拨号专网):

these vpdn commands are not needed with cisco ios software release 12.2(13)

```
sh ru 查看:
vpdn enable ;
no vpdn logging ;
!
vpdn-group pppoe ;
```

```
no vpdn logging ;
!
vpdn-group pppoe ;
request-dialin ;
protocol pppoe ;
```

STEP2: internal ethernet network:

```
in e0 →ip add 192.168.1.1 255.255.255.0 →ip nat inside ;
```

STEP3: xDSL interface (ADSL卡上的接口):

```
in ATM0 →no ip address →no atm ilmi-keepalive →dsl
operation mode auto →bundle-enable →hold-queue 224 in (allthing
were defaulted, 不需要配);
```

STEP4: ATM sub-interface:

```
in ATM0.1 point-to-point →pvc 1/1 (VPI/VCI) →pppoe-client
dial-pool-number 1 (将虚拟ATM子接口放入pool中) →pvc 1/1 is an
example value, that must be changed →to match the value used by
the isp ;
```

STEP5: IF Dial:

the pppoe client code ties into a dialer interface upon which a virtual-access interface is cloud.

```
in dialer 1 →ip address negotiated(与ISP协商IP地址) →ip mtu
1492 (ethernet mtu default=1500 ; 1492+PPPOE headers=1500) →ip nat
outside →encapsulation ppp →dialer pool 1 (去dialer pool 1 中, 调用
刚放入的ATM的子接口) ;
```

STEP6: 认证 (CHAP OR PAP) :

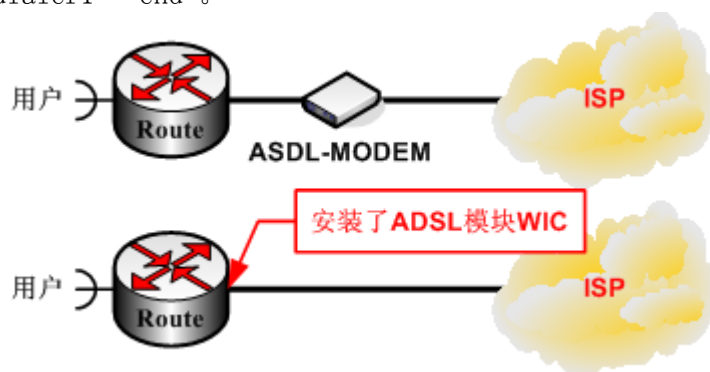
the isp instruets you about the type of authentication to use.

PAP: ppp authentication ppp callin →ppp pap send-username
~~username~~ password ~~password~~

CHAP: ppp authentication chap callin →ppp chap hostname
~~username~~ →ppp chap password ~~password~~ ;

STEP7: NAT:

```
access-list 1 permit 192.168.1.* 0.0.0.255 →ip nat inside
source list 1 interface dialer1 overload →ip route 0.0.0.0
0.0.0.0 dialer1 →end 。
```



3.5一乱七八糟②

3.5一乱七八糟②

IPv6的概述:

IPv4的地址有32Bits, 以点分十进制表示; IPV4包头有20个字节, 有4个段, 每个段有8; IPv4有32位一共可以提供42亿个地址, 一个映射消耗内存64K;

IPv6的地址有128Bits, 以十六进制表示; IPV6包头有40个字节, 有8个段, 每个段有16; ∴IPv6的扩展包头包含ESP和AH字段∴V6中的端到端有安全性。

服务	Ipv4	Ipv6
地址空间	32bit	128bit
自动配置	DHCP	无状态自动配置/DHCP
安全	Ipssec	可以做端到端的ipsec加密
移动性	Mobile-ip	Mobile-ip with direct routing
QoS	区分服务/集成服务	区分服务/集成服务
组播	IGMP/PIM/MBGP	MID/PIMMBGP, scope idetilser

IPv6的地址格式:

每4位的16进制数组成一个字段, 共8个字段, 之间用冒号分割;

前面有0如 030F 可简写为 30F ;

全零字段 :0000: 可以简写为 :0: ;

多个全零字段 :0000:0000:0000: 可以简写为 :: (注意: 一个IPv6地址, 只能出现一次双冒号) ;

例如: 2031:0000:030f:0000:0000:0000:8760:130B可简写为
2031:0:30F::8760:130B 。

IPV6编址及寻址:

未指定 00...0 (128bit) ::/128

环回 00...1 (128bit) ::1/128

全局单播 0010 2000/3 范围: 第一个字段的前3位
(2000~3FFF)

组播地址 1111 1111 FF00::/8 范围: 前8位

链路本地地址 1111 1110 10 FE80::/10范围: 前10位 链路本地地址: link-local地址, 用来标示一条链路上的唯一设备, 而不是一个设备上的唯一链路

站点本地: 1111 1110 11 站点本地相当于IPV4中的私有地址
(旧的私有地址)

unique local unicast address 1111 1100 FC00::/8 1111 1101
FD00::/8unique local unicast address:新的IPV6私有地址

IPv4和IPv6所能支持的路由协议:

(v2)/ISIS/BGP ;

网络协议:IPv6 → 寻路协议/路由协议:RIP/OSPF (V3)/ISIS/BGP (For IPv6) 。

IPv6的地址转换:

双栈 (Dual stack) : 一个路由器上同时拥有两种IP ;

隧道 (tunnels) : 与FR同 ;

地址转换 (Translation) : 就是IPv6的NAT 。

在IPV6网络中, 数据包的类型:

支持:

unicast/单播, (one 2 one)

multicast/组播(one 2 many)

anycast(one 2 nearest)

-One-to-nearest(allocated form unicast address space).

-Multiple devices share the same address.

-All anycast nodes should provide uniform service.

-Source devices send packets to anycast address.

-Routers decide on closest device to reach that destination.

-Suitable for load balancing and content delivery services.

IPv6支持路由汇总, 提高路由效率/Address Aggregation.

IPv6支持自动配置, Auto-Configuration (替代了DHCP)

支持自动配置的IPV6主机,

可以使用从路由器接收到的路由前缀,

加上自己的数据链路层地址 (MAC地址),

形成了自己的全球唯一的IPV6地址。

IPv6支持Renumbering:

允许用户在完成新的路由前缀网络过渡之前,

继续使用原来的旧的路由前缀一段时间,

以平滑完成网络升级/过渡。

网络协议/被路由协议: IPV4

寻路协议/路由协议: RIP/IGRP/EIGRP/OSPF (V2)/ISIS/BGP

网络协议/被路由协议: IPV6

寻路协议/路由协议: RIPng/OSPF (V3)/ISIS/MP-BGP/EIGRP(For IPv6)

IPv4 o IPv6 Transition:

Dual stack既运行IPV4也运行IPV6

tunnels (tunneling encapsulates the ipv6 packet in the ipv4 packet.)

Translation

LAB1:在IPv4的海洋中, 将IPv6的孤岛实现网络互通。(通过TUNNEL实现)
r4--r2--r1--r3--r5 (其中R2与R3为边界, 同时运行IPv4和IPv6; R4/5运行IPv6; R2运行IPv4)

```
step1:构建ipv4网络 (rip v2), (r1/r2/r3#)
r1/r2/r3#
router rip
ver 2
no auto-summary
net 12.0.0.0
net 13.0.0.0
```

检查: show ip route rip

```
step2:在R3/R5--R2/R4间构成IPv6网络。
R4/5(config)#
no ip routing (关闭IPv4的路由能力)
ipv6 unicast-routing (启动IPv6的单播路由能力)
```

```
双栈/DUAL stack路由器: (IPv4/IPv6)
R2/3(config)#
ip routing(默认)
ipv6 unicast-routing
```

在IPv6网络中, 配置IPv6的链路地址, 测试链路:

```
R3(config)#int s 1
r3(config-if)#ipv6 add 2007:0012:0034:0035::3/64
r5(config-if)#ipv6 add 2007:0012:0034:0035::5/64
```

```
show ip route
show ipv6 route
链路测试:
r3#ping 2007:12:34:35::5!!!!
```

为了实现IPv6孤岛的互通, 在双栈(V4/V6)路由器R2/R3上, 构建能承载IPv6通信流量的Tunnel:(step3/4)

```
step3:在R2/R3, 为Tunnel新增一个环回口,
注意此环回口是要宣告到IPv4网络的IGP(RIPv2)中的:
R2#int lo 2
ip add 2.2.2.2 255.255.255.255
r3#int lo 3
ip add 3.3.3.3 255.255.255.255
```

将新建的环回口，宣告到RIP中：

配置：

R2#

```
router rip
version 2
net 2.0.0.0
net 3.0.0.0
```

检查：show ip route rip

测试：r2#ping 3.3.3.3 source 2.2.2.2 !!!!

step4:R2/R3，都以自己的环回口作为tunnel的源地址，
以对方的环回口作为tunnel的目标地址，
来构建Tunnel接口（Tunnel可以理解为虚拟链路/隧道）：

配置：

r2#

```
interface tunnel 1
tunnel source 2.2.2.2
tunnel destination 3.3.3.3
ipv6 add 2007:12:34:23::2/64（为TUNNEL配置IPV6地址）
```

r3#

```
int tunnel 1
tunnel source 3.3.3.3
tunnel dest 2.2.2.2
ipv6 add 2007:12:34:23::3/64
```

测试：

r2#ping 2007:12:34:23::3 !!!!!

sh ipv6 route connected

sh int tunnel 1

至此，IPV4的网络中的路由器R1，都忽略了。（透明了）

step5:

在所有的IPV6网络接口中，启动IPV6的路由协议OSPFv3

（提醒：包括R2/R3间的Tunnel接口）

5-1:R2/R3#

```
int tunnel 1
ipv6 enable
tunnel mode ipv6ip (IPv6 over IP encapsulation)
```

5-2:在所有的IPv6路由器上(R2/3/4/5)，都启动IPv6的路由协议：“ospf

v3”

conf t

```
ipv6 router ospf 16 (进程号)
router-id 16.0.0.* (格式还是保留为，类似IPv4地址的格式)
```

5-3:在V4里面需要NET接口，但是在V6里面不同：在所有需要运行IPv6的OSPF接口中，激活IPv6的OSPF的运行：

```
r3/r5/r4/r2(config)#
int s 1
ipv6 ospf 16 area 0
```

```
int tunnel 1
ipv6 ospf 16 area 0 (Tunnel接口：跑ipv6的ospf v3，不跑ipv4的rip)
r2/r3的loopback口不要运行ospf v3，因为它们用来建立tunnel的，它们之间运行ipv4的rip
```

检查：

```
r2#sh ipv6 ospf interface (brief)
r2#sh ipv6 ospf neighbor
r2#sh ipv6 ospf database
r2#sh ipv6 route ospf
```

step6:如果IPv6网络运行的路由协议是：RIP FOR IPV6
从step1-4，都是相同的。

conf t

```
no ipv6 router ospf 16
```

启动rip for v6的进程：

```
ipv6 router rip RIP-V6 (这个是名字，自己起的，相当于描述)
```

在所有的IPv6接口中，激活RIP FOR V6的运行

```
ipv6 rip rip-v6 enable (Tunnel也跑rip for ipv6)
```

```
r5#show ipv6 route rip
```

搞定！

IPv6基本配置：

首先自动生成本地链路（link-local）IPv6地址；

接口下ipv6 enanle（容许IPv6的运行）→ipv6 address fe80:: link-local（更改IPv6的link-local地址）→ipv6 unicast-routing（启用IPv6路由功能）→ipv6 nd suppress-ra（不发送链路RA公告）。

MAC地址转化为IPv6地址规则：

MAC地址共48位，从中间断开各为24位，在中间加上FFFE，再把加上后的地址第7位的0改成1，如果第7位是1则改成0（MAC地址：第2位是0则全球唯一，是1则本地唯一；IPv6地址：第7位为0本地唯一，为1全球唯一）。

质量服务QOS (quality of service) :

应用程序变慢了 (Application X is slow) → 带宽不够;

广播流量不连续、抖动了 (Video broadcast occasionally stalls) → 延时过大;

IP电话的信息比卫星电话还要不如了 () → 延时不稳定;

打电话时语音质量变差了 () → 延时不稳定;

ATM拥塞而不响应了 () → 丢包了;

综上所述: QOS在网络拥塞 (Congestion) 的情况下生效。

产生拥塞的原因:

带宽不够 (Lack of bandwidth) → 解决方案: 增加带宽; 压缩流量; 对L2层帧的净荷进行L3/L4包头压缩;

延时过大 (Too much delay) → 解决方案: 同上;

抖动Jitter (Variable delay) → 解决方案: 同上;

丢包Drops (Prevent packet loss) → 解决方案: 同上; 优先保证特别敏感的数据流为其预留带宽 (集成服务); 定义流量优先级在带宽接近用尽时丢弃优先级低的包 (分级服务)。

由几条不同链路组成的网络中:

有效带宽 (Available Bandwidth): 链路的最小带宽;

有效延时 (Available Delay): 所有的延时之和。

现代数据通讯网络上的四种典型数据分类及特征:

	Throughput	Delay	Loss	Jitter
Interface(e.g.Telnet)	LOW	LOW	Low	Not important
Batch(e.g.FTP)	High	Not important	Low	Not important
Fragile(e.g.SNA)	Low	Low	None	Not important
Voice(e.g.Phone)	High	Low and Predictable	Low	Low
Video(e.g.IPPhone)	High	Low and Predictable	Low	Low

QOS的发展:

尽力传输 (Best Effort): no Qos, default behavior;

集成服务 (Integrated Services): 如资源预留协议RSVP (Resource Reservation Protocol);

分级服务 (Differentiated Services): 为用户流量定义优先级, 确保优先级高者的流量, 带宽不足时依次丢弃优先级低的流量; 当前的主流服务;

MPLS (Multi-Protocol Labeled Switching): 高层交换机不查看路由表而是直接根据L2包头进行选路, 速度最快!

组播 (Multicast):

组播的不利因素: 组播是基于UDP协议的→可靠性和安全性的问题 (TCP只能单播, 要求两个终端必须连接起来); 最大努力传输→丢包的必然性; 没

(TCP只能单播, 要求两个终端必须连接起来); 最大努力传输→丢包的必然性; 没有拥塞避免→缺少TCP慢启动进制导致拥塞的出现; 副本的出现→一些多播处理机制导致网络中大量副本的出现; 传输次序丢失。

组播的应用:

多媒体会议 (Mbone);
数据分发 (允许采用PUSH方式进行文件和数据库的更新);
实时数据组播 (股市、汇率等的公布);
游戏和仿真。

组播的类型:

One-to-Many: 单一播源连接多个客户端;
Many-to-Many: 源=客户端;
Other: 其他类型, 比如说多个对一个。

Multicast			
Whiteboard	Audio	Video	Other
.....	G7.xx PCM	H.261 MPEG
Reliable transport (SRM)	RTP/RTCP		Other transports
UDP			
IP			

组播地址的分类:

ClassD IP: high-order 4 bit are set 1110**** : 224.0.0.0~239.255.255.255 ;
224.0.0.1 : all route on multicast subnet;
224.0.0.2 : all route on subnet;
224.0.0.4 : all DVMFP routes ;
224.0.0.13 : all PIMv2 routes ;
224.0.0.5、224.0.0.6、224.0.0.9、224.0.0.10 : for routing protocol ;
224.0.0.39 : RP宣告;
224.0.0.40 : RP发现。

RPF检查:

组播的范围由存活检查TTL (Time To Life) 决定:

TTL阈值 (Thresholds): 控制组播的范围 → 路由器检测到流量的阈值小于某值后不转发该流量 → 即他就是最后一个用户。

最短路径树SPT (Shortest Path Tree):

有源树: 直接寻路 (类似路由寻路) → 生成SPT → 开始通讯;
共享树: 竞选RP → 生成共享树 → 消息发给RP由RP向所有共享树节点

点发放组播信息 → 开始通讯。

密集模式PUSH/PIM-DM (Dense Mode Protocols)：推销式的

采用PUSH模式发送组播信息 → 存在Flooding和Prune泛洪式进行RP宣告 → **扩散和剪枝**（被接受着接受方转发扩散，被拒绝则发送方修剪也就是停止发送3分钟后再发送）。

稀疏模式PULL/PIM-SM (Sparse Mode Protocols)：订购式的

采用PULL模式，是一种join的模式 → 由最后一条的路由器pull对RP进行宣告要加入SPT的节点，没有扩散和剪枝，同时支持有源树和共享树，使用中介RP。

IGMP (Internet Group Management Protocol)：

IGMPv1：周期性发送存活查询，成员离开时不会发送报文，Dead-time后才收敛 → 浪费带宽；

IGMPv2：周期性发送存活查询，成员离开时发送报文，立刻收敛 → 不再浪费带宽；

IGMPv3：Cisco私有。

QOS

链路上有效的**带宽**就是链路所有的路径节点中，最少的带宽值。
..... **延时**.....，所有延时之和。

网络出现拥塞的情况：

Lack of bandwidth（带宽不足）

Too much Delay（延时）

Variable Delay jitter（抖动）

Drop（丢包）

解决方法：

1：解决Lack of bandwidth方法：

1-1向ISP购买更多的链路带宽

1-2压缩不那么重要的数据流量，优先保证重要的数据流量通过。

1-3压缩2层的数据frame的净荷，压缩L3/L4的报头。

2: 降低延时的方法:

同上

3: Prevent Packet loss的方法

3-1: 同上

3-2: 优先保证, "特别敏感的"数据包的带宽(资源预留)

3-3: 在链路/接口临近拥塞出现之前, 先随即丢一小部分不重要的数据. 来防止拥塞的出现.

QoS 实行的4大类方法

Best effort (no QoS, default behavior) 尽力而为服务模型

Integrated Services model 综合服务模型简称Intserv

Differentiated Services model 区分服务模型简称Diffserv

MPLS (Multi-Protocol Labeled Switching)

Integrated Services model (RFC1633) IntServ.

Resource reservation

Admission control

Resource Reservation Protocol (RSVP) (RFCs 2205 to 2215).

Common Open Policy Service (COPS) (RFCs 2748 to 2753).

3: Different Services Model

Traffic Terminology

- **Flow**: a single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port and protocol id.
- **Traffic stream**: an administratively significant collection of one or more flows which traverse a path. A traffic stream may consist of a set of active flows which are selected by a particular classifier.
- **Traffic profile**: a description of the temporal properties of a traffic stream such as average rate, peak rate and burst size.

Blocks of IP QoS Mechanisms

QoS的工作流程:

Step1:Classifier

区分不同的数据包

access lists, route maps, class maps,

Step2:Meter/测量(速率)

Token bucket/令牌桶

rate limiting, shaping, scheduling,

Step3:Marker/标记

CAR, class-based marking,

CAR, class-based marking,
通常为特定的数据包, 打上以下的一种标签:
IP Precedence
DSCP
QoS group
MPLS experimental bits
Frame Relay DE bit
ATM CLP bit
IEEE 802.1Q or ISL CoS

Step4:Conditioner/调节器-----主要分为2种技术:

- 1 **Dropping mechanisms**
 - **Committed Access Rate (CAR) and Class-based Policing** can drop packets that exceed the contractual rate
 - **Weighted Random Early Detection (WRED)** can randomly drop packets when an interface is nearing congestion
- 2 **Shaping mechanisms:**
 - **Generic Traffic Shaping (GTS)**
 - **Frame Relay Traffic Shaping (FRTS)**
 - **Class-based Shaping**
 - **Hardware shaping on ATM VC**

Step5:Forwarding/转发器

- 1:Process Forwarding/进程转发
- 2:Fast Forwarding/快速转发
- 3:Cisco Express Forwarding(CEF)

Step6:软件队列:

队列种类:

FIFO, priority queuing (PQ), custom queuing (CQ)

WFQ, DWFQ, CoS-based DWFQ, QoS-group DWFQ

Class-based WFQ, Class-based LLQ

Step7:能否加入软件队列?

有多技术可以控制能否进对:

- 1:Tail drop(最常用的)(伪丢弃:满了就排不进去)
- 2:WFQ has an improved tail-drop scheme.
- 3:WRED randomly drops packets when nearing congestion.
(在链路/接口临近拥塞出现之前, 先随机丢弃一小部分不重要的数据包, 来防止拥塞出现)

Step8:Scheduling/调度器

调度器负责按照一定的算法, 从多个软件队列中, 调入数据包, 送进硬件队列.

Step9:硬件队列:

在硬件队列中, 按照“先进先出/FIFO”原则, 将数据送入接口, 进行转发.

Classification and Marking

Policy-based routing (PBR)

QoS Policy Propagation on BGP (QPPB)

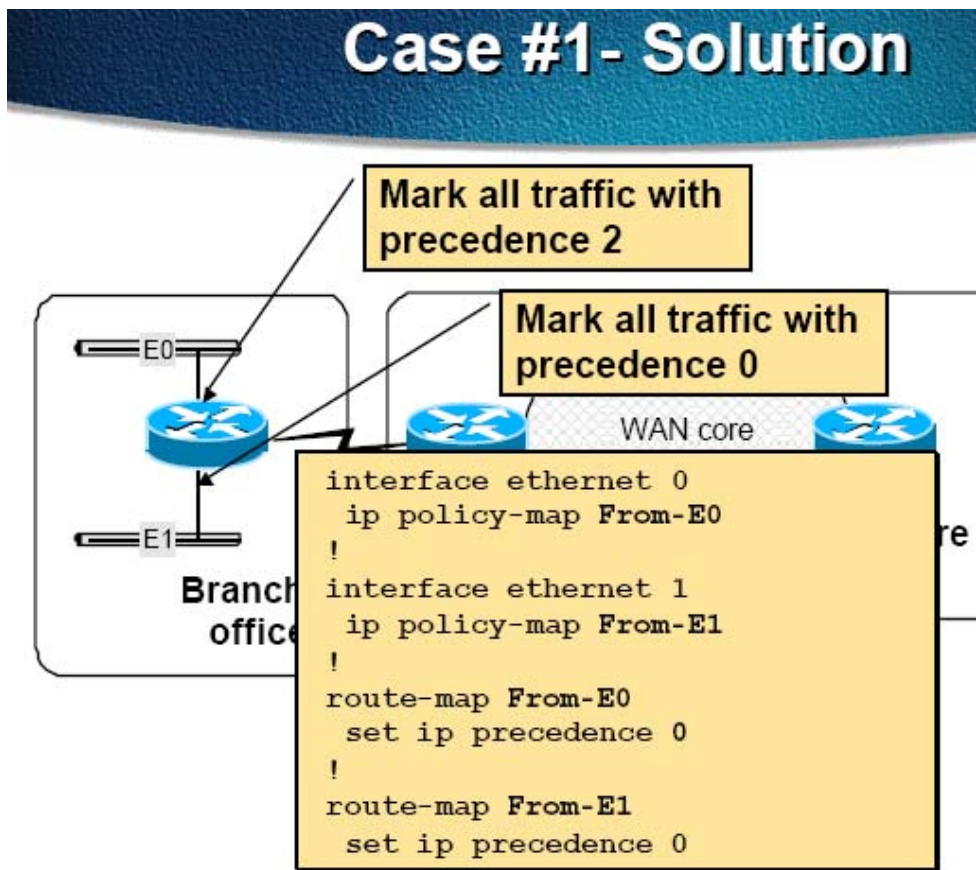
Traffic Classification and Marking

- This module describes the two mechanisms that are used for classification and marking only:
 - Policy-based Routing (PBR)
 - QoS Policy Propagation through BGP (QPPB)
- Other classification and/or marking mechanisms are described in other QoS modules

PBR: (只在入口生效)

classification---分类器

1. 大多数的QoS机制都包括了分类器
2. 自动分类的有WFQ
3. 手工分类的有PQ, CQ, CBWQ



LAB1

Step1:

```
route-map From-E0 permit 10
 set ip precedence routing
```

LAB2

Step1:通过ACL, 进行数据分类:(Classification) 表示名字

```
ip access-list standard From-BOSS
 permit 192.168.1.1
!
```

```
ip access-list extended Telnet
permit tcp any any eq telnet(23)
```

Step2:通过Route-map, 进行数据标记(Marking)

```
route-map NET-CTL permit 10
match ip address From-BOSS
set ip precedence priority/1 (值越高越优先)
!
route-map NET-CTL, permit 20
match ip address Telnet
set ip precedence immediate/2
!
route-map NET-CTL, permit 30
match ip address Telnet
set ip precedence routine/0
```

Step3:在入口中调用Route-map

```
interface Ethernet0
ip policy route-map NET-CTL
```

R2(config-route-map)#match length 1000 1500(最小1000 最大1500) (配置L3的包长度)

~~~~~  
~~~~~

Queuing Mechanisms/队列机制(只对出口生效)

Hardware Queue/Transmit queue (TxQ)

FIFO Queue

跳过软件队列的前提条件:

硬件队列都没有满/Full, 没有发生拥塞.

Software Queue:

软件队列的3大组件:

classification/分类

insertion policy/入队

service policy/scheduling/调度机制

FIFO Queuing:

~~~~~  
~~~~~

硬件队列默认就是FIFO, 默认在队列中存储40Packets, 不建议更改.

在接口带宽高于2Mbps的接口上, 默认是FIFO.

在接口带宽低于或等于2Mbps的接口上, 默认是WFQ.

如果需要低于或等于2Mbps的接口上, 启动FIFO,

使用命令:

```
interface serial1
no fair-queue(启动FIFO)
```

检测:

```
R2#Show int s1
Serial1 up , line protocol is up
  MTU 1500 bytes, BW 1544 bit, DLY 20000 usec,
    Queueing strategy: weighted fair
    Queueing strategy: fifo
R2#show queue serial 1
```

Priority Queuing/优先级队列/PQ

必须先传输完高优先级的队列的数据,才能传输低优先级的队列.
可能导致低优先级的队列的数据的“饿死”

4个PQ队列的队列长度的比例:

```
R2(config)#priority-list 2 queue-limit 20 40 60 80
                        高 中 正常 低
```

LAB 3 Priority Queuing

~~~~~  
~~~~~

Step1:通过ACL区分不同数据:

```
access-list 20 permit 3.0.0.0 0.255.255.255
!
access-list 110 permit tcp any any eq telnet
(不支持命名ACL)
```

Step2:根据不同数据, PQ(PQ的编号的取值1~16)

```
priority-list 2 protocol ip high list 110
priority-list 2 interface serial1 medium
priority-list 2 protocol ip normal list 120
priority-list 2 default low
```

Step3:在数据出口调用PQ-2

```
interface serial0
priority-group 2
```

检测:

```
R2#show queueing priority
```

Cusom Queuing:/CQ/用户队列:

最多支持16个队列

Step1:

```
access-list 110 permit tcp any any eq telnet
access-list 150 permit ip any any precedence flash-override/4
access-list 150 permit ip any any dscp af41
access-list 150 permit ip any any tos min-delay/8
```

Step2:

```
queue-list 5 protocol ip 1(几号队)list 150
```

```
queue-list 5 queue 1 byte-count 800 limit 30          (一个
队列, 一次循环中, 最少能传的字节. 默认:1500)
                                Bytes      Packets
```

```
queue-list 5 lowest-custom 2 -----设队列2为最低的. 指定上
面的队列1是优先级队列
```

```
queue-list 5 protocol ip 2 list 110(Telnet)
```

```
queue-list 5 interface Ethernet 0 3
```

```
queue-list 5 default 4(FTP)
```

Step3:在接口中调用CQ5:

```
interface serial 0
 custom-queue-list 5
```

IP的分类:

```
Source ip
destination ip
soutce tcp port
destination tcp port
source udp port
destination udp
proctotol tag
fragment
packet size
```

```
source interface
```

大于2M的接口，用的队列都是FIFO队列。

DSCP differentiated services code point

tag值在数据库中才能看到，或明细路由中

```
set tag
```

```
match tag
```

Wolf-NP+ 笔记

路由器安全设置详解

路由器安全设置详解

```
hostname Perimeter-Router
```

! 路由器名称

```
enable secret ena-secret
```

! 特权访问口令为

```
ena-secret
```

```
interface serial 0
```

! 定义接口

```
description To Internet
```

! 目的描述

```
ip address 161.71.73.33 255.255.255.248
```

! 设置IP地址

```
ip access-list 101 in
```

! 定义入站过滤器

```
ip access-list 102 out
```

! 定义出站过滤器

```
access-list 101 permit tcp any any established Note 1
```

! 允许所有tcp业务

流入，会话始于园区网内

```
access-list 101 permit tcp any host 144.254.1.3 eq ftp
```

! 允许 ftp 到不洁

网

```
access-list 101 permit tcp any host 144.254.1.3 eq
```

! 允许 ftp 数据到

不洁网中的ftp服务器

```
access-list 101 deny ip 127.0.0.0 0.255.255.255 any
```

! 阻止来自Internet

并以RFC

```
access-list 101 deny ip 10.0.0.0 0.255.255.255 any
```

!保留地址为源的数

据包入站

```
access-list 101 deny ip 172.16.0.0 0.240.255.255 any
```

```
access-list 101 deny ip 192.168.0.0 0.0.255.255 any
```

```
access-list 101 deny icmp any any echo-reply
```

! 拒绝任何应答

```
access-list 101 deny icmp any any host-unreachable
```

! 拒绝任何无法接通

的主机

```
access-list 101 deny udp any any eq snmp
```

! 拒绝引入的SNMP

```
access-list 101 deny udp any eq 2000
```

! 拒绝引入的

```
openwindows
```

```
access-list 101 deny udp any any gt 6000
```

! 拒绝引入的X-

windows

```
access-list 101 deny tcp any any eq 2000
```

! 拒绝引入的

```
openwindows
```

```
access-list 101 deny tcp any any gt 6000
```

! 拒绝引入的X-

```

access-list 101 deny udp any any eq 69          ! 拒绝引入的tftpd
access-list 101 deny udp any any eq 111          ! 拒绝引入的SunRPC
access-list 101 deny udp any any eq 2049         ! 拒绝引入的NFS
access-list 101 deny tcp any any eq 111          ! 拒绝引入的SunRPC
access-list 101 deny tcp any any eq 2049         ! 拒绝引入的 NFS
access-list 101 deny tcp any any eq 87           ! 拒绝引入的连接
access-list 101 deny tcp any any eq 512          ! 拒绝引入的 BSD
UNIX “r” 指令
access-list 101 deny tcp any any eq 513          ! 拒绝引入的 BSD
UNIX “r” 指令
access-list 101 deny tcp any any eq 514          ! 拒绝引入的 BSD
UNIX “r” 指令
access-list 101 deny tcp any any eq 515          ! 拒绝引入的 lpd
access-list 101 deny tcp any any eq 540          ! 拒绝引入的
uucpd
access-list 101 permit ip any any                ! 其它均允许

access-list 102 permit ip 144.254.0.0 0.0.255.255 any ! 只允许有源的包
access-list 102 deny ip any any                  ! 园区网到Internet
的地址

aaa new-model                                    ! 在全范围实现AAA
aaa authentication login default tacacs+          !默认登录方法经由
tacacs+
aaa authentication login staff tacacs+ local      !通过tacacs+鉴别工
作人员用户名... 如果无法连接服务器，退而求其次的方法是本地鉴别
aaa authorization exec tacacs+ local              ! 鉴别通过后，授权
运行 exec shell
aaa authorization commands 0 tacacs+ none         ! 鉴别与指定特权等
级相关的运行模式指令
aaa authorization commands 1 tacacs+ none         ! 如果无可用的
tacacs+ 服务器，
aaa authorization commands 15 tacacs+ local       ! 15级权限指令就需
要本地鉴别，其它不需要任何鉴别
aaa accounting update newinfo                    ! 每当有新的记帐信
息需要报告时，中间记帐记录将被送到服务器
aaa accounting exec start-stop tacacs+           ! 对终端会话进行记
帐
aaa accounting network start-stop tacacs+        ! 对所有 PPP, SLIP
和ARAP连接记帐
username staff password 7 staffpassword          ! 创建本地口令并以
加密格式存储
tacacs-server host 144.254.5.9                    ! 定义tacacs+ 服务
器地址
tacacs-server key thisisasecret                  ! 定义共享的
tacacs+ 密码

line con 0
exec-timeout 5 30                                ! 确认控制台会话结

```

line con 0	
exec-timeout 5 30	! 确认控制台会话结
束时间	
login authentication staff	! 只有用户名工作人
员可接入控制台	
line aux 0	
transport input none	! 没有telnet进入
no exec	! 该端口没有得到运
行提示	
line vty 0 3	
exec-timeout 5 30	! 确认 telnet 会话
束时间	
login authentication default	! 通过 tacacs+ 登
录鉴别	
privilege level 15	! 获得15级权限
line vty 4	
exec-timeout 5 30	! 确认 telnet 会话
束时间	
login authentication staff	! 鉴别为工作人员
rotary 1	
privilege level 1	
logging on	! 开启syslog
logging 144.254.5.5	! 定义syslog服务器
地址	
logging console information	! 定义登录的信息

WOLF攻击课经典语录

WOLF攻击课经典语录

攻击课主旨：

你们有权保持沉默，但你们所攻击的每一个问题都将成为我在IE道路上成功的路标。

开场白：今天我站在这里被大家攻击，首先要感谢党感谢人民感谢CCTV感谢TTMV感谢CISCO 感谢WOLF.fk ME NEW!

注：问 ---攻击者

答 ---被攻击者

问：在运行OSPF的区域里是先选举DR还是先选举BDR？

答：本来这概念的问题我是拒绝回答的，在RFC2328第73页倒数第十四行第二句话明确的说明，在需要选举DR、BDR的区域是先选举BDR的，如果你们还不明白，那去把先有鸡还是先有蛋的问题回顾下。仔细想了下有点乱，开始对达尔文的《进化论》产生怀疑。

问：我在OSPF区域一次网络收敛之后重新指定了DR，为什么路由器还和以前的DR建立邻居？

答：这是曾经有一腿的问题！因为以前他们建立起邻居。清下路由表就好了。透彻！思路清晰比喻恰当，曾经有一腿

问：为什么要关闭BGP同步？

答：关闭同步是为了使其同步，因为要同步所以要关闭同步。参照美军第二十二条军规如下：如果你要是想在服役期间退役那么请写一份声明，声明自己有神经病。

问：还是不懂！！

问：什么是自治系统，自治系统号的代表什么？

答：一个自治系统就是有着相同管理机制的多个路由器的集合，自治系统号就是标志这些区域，公有的自治系统号范围是1-64512私有的自治系统号是64512-65535，公有的是要向ISP申请的是要花钱买的。

问：大概是什么价码？

答：17块钱包月，免基本路由费，全球无漫游，接收路由免费发送路由条目一毛钱一条！！！！

问：无话可说……

问：BGP的选路原则是什么？

答：1 HIGHEST WEIGHT

2 HIGHEST LOCAL PRERENCE

3 ROUTE ORIGINATED BY THE ROUTER NEXT HOP=0.0.0.0

4 SHORTEST AS PATH

5 LOWEST ORIGINATED IGP>EGP>INCOMPLETE

6 LOWEST MED

7 EBGp PATH OVER IBGP PATH

8 PREFER THE PATH THROGH THE CLOSEST IGP NEIGHBOR

9 RREFER OLDEST ROUTER FOR EBGp PATH

10 PREFER THE PATH WITH THE LOWEST NEIGHBOR BGP ROUTER ID

11 BGP LOWEST ROUTER ID

12 MINIMUM CLASTER LIST LENGTH

13 LOWEST NEIGHBOR ADDRRESS

问：不明白啊，能不能形象的比喻下？？

答：正如我找女朋友一样，

1 不能生孩子的不要

2 不会做饭的不要

3 不洗脚的不要

4 不温柔的不要

5 不孝敬老人的不要

6 长的太漂亮的不要

7 长的不漂亮的不要

8 平胸的不要

9 低于165CM高于175CM的不要

10抽烟的不要

11吸毒的不要

12有狐臭的不要

13别人不要的我不要

问：……

↑ 谬误之处 ↑

在OSPF中，DR与BDR是同时选举的，你可以加timestamps, DR与BDR是同一时间出现的。

```
xq(config)#service timestamps debug datetime msec
xq(config)#service timestamps log datetime msec
xq#debug ip os ev
```

BGP的同步

必须关闭同步使其同步

关闭同步的意思是指使用no syn这条命令关闭其路由器的同步功能

使其同步的意思是指 在去往一目的地的路由必须在 IBGP与IGP的路由表中有相同的路由前缀，即IBGP与IGP的路由达成一致，当然，还要下一跳可达。

思科OSPF设计的建议

每个区域的路由器不超过50个。

每个路由器的邻居不超过60个

每个路由器的所在的区域数不超过3个。

路由器不要成为超过一个LAN的DR或BDR。

另，OSPF选择DR与BDR是根据网络拓扑来的。

TCP/IP卷一

第二章的命令

arp ip address hardware address 静态地址映射IP地址类型（别名）到硬件地址

arp timeout seconds 设置Cisco路由器保留ARP表项的时间值

clear arp-cashe 强制从ARP表中删除所有动态表项

debug ip icmp 显示在路由器上出现的ICMP事件

ip address ip-address mask 为接口分配IP地址和掩码

ip netmask-format[bitcount|decimal|hexadecimal] 配置路由器，使路由器可以用位计数器、点分十进制和十六进制方式显示IP（地址、掩码）对

ip proxy-arp 启用代理ARP

ip redirects 启用ICMP重定向功能

RIP的相关命令

debug ip rip 简要地显示路由器收发的RIP信息

ip address ip-address mask secondary 在接口上指定一个IP地址作为辅助地址

neighbor ip-address 通过指定接口邻居的IP地址来建立邻居关系

network network-number 指定一个需要运行RIP的网络

[type number] 指定路由选择表中一个与指定的访问列表匹配的路由条目，将自己的度量值增加一个offset指定的偏移量
output-delay delay 设定一个指定延迟长度的延迟间隙，以便协调高速路由器和低速路由器之间的延迟问题
passive-interface type number 在指定类型和序号的接口上阻止RIP广播
timers basic update invalid holddown flush 更改指定的计时器的值

卷一静态路由笔记

在互连网络上实施静态路由选择的步骤：

step1: 为互连网络中的每个数据链路确定地址（包括子网和网络）

step2: 为每个路由标识所有非直连的数据链路。

step3: 为每个路由器写出关于每个非直边数据链路的路由说明。

各项AD

管理距离：

Connect 0

S:1

Enhanced Interior Gateway Routing Protocol summary route EIGRP 5

EBGP 20

<INTERNAL> EIGRP 90

IGRP 100

OSPF 110

IS-IS 115

RIP V1/V2 120

ODR 160

<EXTERAL> EIGRP 170

IBGP 200

Unknown 255

Route Source	Default Distance
Connected interface	0
Static route*	1
Enhanced Interior Gateway Routing Protocol (EIGRP) summary route	5
External Border Gateway Protocol (BGP)	20
Internal EIGRP	90
IGRP	100
OSPF	110
Intermediate System-to-Intermediate System (IS-IS)	115
Routing Information Protocol (RIP)	120
Exterior Gateway Protocol (EGP)	140
On Demand Routing (ODR)	160
External EIGRP	170
Internal BGP	200
Unknown**	255

Protocol ID		Port ID 端口号
9	IGRP	
88	EIGRP	
89	OSPF	
	RIP	UDP 520
	BGP	TCP 179
6	TCP	
17	UDP	

Routing Protocol	Protocol No.	Port No.	Update Reliability
IGRP	9		Best effort delivery
EIGRP	88		1-to-1 window
OSPF	89		1-to-1 window
RIP		UDP 520	Best effort delivery
BGP		TCP 179	Uses TCP windowing

故障处理

第1章 故障处理方法

第1章 故障处理方法

一、网络的复杂性

一般网络包括路由、拨号、交换、视频、WAN（ISDN、帧中继、ATM、...）、LAN、VLAN、...

二、故障处理模型

1、 界定问题（Define the Problem）

详细而精确地描述故障的症状和潜在的原因

2、 收集详细信息（Gather Facts）R>信息来源：关键用户、网络管理系统、路由器/交换机

1) 识别症状：

2) 重现故障：校验故障依然存在

3) 调查故障频率：

4) 确定故障的范围：有三种方法建立故障范围

？由外到内故障处理（Outside-In Troubleshooting）：通常适用于有多个主机不能连接到一台服务器或服务器集

？由内到外故障处理（Inside-Out Troubleshooting）：

？半分故障处理（Divide-by-Half Troubleshooting）

3、 考虑可能情形（Consider Possibilities）考虑引起故障的可能原因

4、 建立一份行动计划（Create the Action Plan）

5、 部署行动计划（Implement the Action Plan）

用于纠正网络故障原因。从最象故障源处，想出处理方法每完成一个步骤，检查故障是否解决

6、 观察行动计划执行结果（Observe Results）

7、 如有行动计划不能解决问题，重复上述过程（Iterate as Needed）

三、记录所做修改

在通过行动计划解决问题后，建议把记录作为故障处理的一部分，记录所有的配置修改。

第2章 网络文档

第2章 网络文档

一、网络基线

解决网络问题的最简单途径是把当前配置和以前的配置相比较。

基线文档由不同的网络和系统文档组成，它包括：

？网络配置表

？网络拓扑图

？ES网络配置表

？ES网络拓扑图

创建网络的注意事项：

1) 确定文档覆盖的范围；

2) 保持一致：收集网络中所有设备的相同信息；

3) 明确目标：了解文档的用途；

4) 文档易于使用和访问;

5) 及时维护更新文档。

二、网络配置表

网络配置表的通常目标是提供网络中使用的硬件和软件组成的列表,其组成有:

分级 项目

杂项信息 设备名、设备型号、CPU类型、FLASH、DRAM、接口描述、用户名口令

第1层 介质类型、速率、双工模式、接口号、连接插座或端口

第2层 MAC地址、STP状态、STP根桥、速端口信息、VLAN、Etherchannel配置、封装、中继状态、接口类型、端口安全、VTP状态、VTP模式

第3层 IP地址、IPX地址、HSRP地址、子网掩码、路由协议、ACL、隧道信息、环路接口

在多数情形下,存储这些信息的最佳方式是电子表格或数据库,电子表格用于较小的网络,数据库用于较大的网络。

三、网络拓扑图

网络拓扑图是图示网络的各组成部分之间如何在逻辑上和物理上相互连接。

1、网络拓扑图的组成

分级 项目

杂项信息 设备名、设备型号、设置间连接、接口描述

第1层 介质类型、接口号

第2层 MAC地址、VLAN、封装、中继状态、接口类型、DLCI

第3层 IP地址、子网掩码、路由协议

对于大型的网络,可以制作多个网络拓扑图,每个网络拓扑图反映一个分离的部分。

2、建立网络拓扑图

四、发现网络配置信息

1、收集路由器和第3层交换机网络配置信息

show version ; 显示设备型号、Flash、DRAM、IOS版本

show ip interface brief ; 显示接口简要信息(类型、状态、协议状态、IP地址)

show interface e0/0 ; 显示某接口详细信息(MAC、IP、MASK、...)

show ip protocols ; 显示IP路由协议信息

show ip interface e0/0 ; 显示接口的IP协议信息(状态、IP地址、ACL、...)

2、收集交换机配置信息

交换机网络配置表包含的信息:设备名、型号、位置、Flash、DRAM、CATOS版本、管理地址、VTP域、VTP模式、端口号、端口速率、端口双工、VLAN、STP状态、速端口状态、中继状态、...

show version ; 显示IOS或CATOS版本、DRAM、Flash

show vtp domain ; (CatOS) 显示VTP域和VTP模式

show vtp status ; (IOS)

show interface ; (CatOS) 显示管理接口信息

show port ; (CatOS) 显示每个端口的简要信息(号、VLAN、双工、...)

show interface ; (IOS)

show trunk ; (CatOS) 显示中继信息(模式、封装、允许端口、剪裁、...)

show interface trunk ; (IOS)

show spantree 45 ; (CatOS) 显示端口的STP模式、类型、状态、速端口、

show trunk ; (CatOS) 显示中继信息 (模式、封装、允许端口、剪裁、...)
show interface trunk ; (IOS)
show spanntree 45 ; (CatOS) 显示端口的STP模式、类型、状态、速端口、...)

show spanning-tree 45 ; (IOS)

3、发现相邻CISCO设备的信息

CDP (Cisco Discovery Protocol) 是CISCO的专用协议, 用于识别直接相邻的CISCO设备信息, CDP工作在第2层。

Show cdp neighbor ; 显示相邻CISCO设备的简要信息 (ID、相邻接口、平台、...)

Show cdp neighbor detail; 显示相邻CISCO设备的详细信息 (包含第3层信息)

五、创建网络文档的过程

- 1、 LOGIN ; 登录到设备进入特权模式。
- 2、 接口发现 ; 发现关于设备的所需信息
- 3、 Document ; 在网络配置表中记录发现的信息。
- 4、 Diagram ; 从网络配置表传输所需信息到网络拓扑图
- 5、 设备发现 ; 判断是否有相邻设备没有记录文档。

第3章 ES文档和故障处理

第3章 ES文档和故障处理

一、ES网络配置表

ES网络配置表是ES的硬件和软件组成的列表。ES网络配置常包括以下项目:

分级 项目

杂项信息 系统名、系统厂商/型号、CPU速率、RAM、存储器、系统功能

第1、2层 介质类型、接口速率、VLAN、MAC、网络接头

第3层 IP地址、缺省网关、子网掩码、WINS、DNS、

第7层 操作系统 (版本)、基于网络的应用程序、高带宽应用程序、低延时应用程序、特定考虑

二、ES网络拓扑图

ES网络拓扑图的典型项目有: 系统名、网络连接、物理位置、系统目标、VLAN、IP地址、子网掩码、操作系统、网络应用程序

大多数ES网络拓扑图都建立在网络拓扑图中, 其中还可加入ES网络配置表数据的子集。

三、收集ES网络配置信息

通用命令:

- 1) ping host/ip-address ; 发送和接收ICMP响应, 校验网络的连通性
- 2) arp -a ; 查看修改ES的MAC-IP映射表 (同一子网)
- 3) telnet host/ip-address ; 登录远程ES或特定TCP端口

Windows平台命令

- 1) ipconfig /all ; 查看修改ES的IP信息 (适用所有Windows平台)
- 2) winipcfg ; 查看修改ES的IP信息 (仅适用于Win9x平台)
- 3) tracert host/ip-address ; 校验到主机的连接并显示路径上的设备IP
- 4) route print ; 显示本设备IP路由表的内容
- 5) netstat ; 显示当前网络连接

4) route print ; 显示本设备IP路由表的内容

5) netstat ; 显示当前网络连接

Unix、Linux和Mac OS系统命令

1) ifconfig -a ; 查看UNIX和MAC主机的IP信息

2) traceroute host/ip ;

3) route ?n ;

4) cat /etc/resolv.conf ; 查看DNS服务器信息

四、通用的故障处理过程

1、通用的故障处理过程:

1 收集症状: 收集网络、用户、ES的症状

1) 分析现存症状

2) 判断所属

3) 窄化范围

4) 判定症状

5) 记录症状

1 分离问题

1) Bottom-Up troubleshooting

从物理层开始向上排查, 直到应用层。常用于怀疑问题发生在物理层, 或在处理复杂网络问题时使用。

2) Top-Down troubleshooting

从应用层开始向下排查故障, 用于怀疑问题发生在软件部分。

3) Divide-and-Conquer troubleshooting

选择OSI模型的特定层(数据链路层、网络层、传输层)开始故障处理, 确定问题是在该层、还是上层或下层。适于具有丰富的经验的人员使用。

常用traceroute命令检查下4层(从物理层到应用层)。

1 纠正问题

2、ES故障处理命令

1) ping

连续Ping: ping ?t 192.168.0.1 ; Windows系统

ping ?s 192.168.0.1 ; Unix环境

记录路由: ping ?r 192.168.0.1 ; Windows

ping ?s ?nRv 192.168.0.1 ; Unix

2) Trace Route

Tracert 10.0.0.1 ; Windows系统

Tracerout 10.0.0.1 ; Unix

Ping记录路由器的出接口, 而traceroute通常记录进入的接口。

3) Arp

显示第2层和第3层地址的映射表: Arp ?a ; Windows/Unix

4) Route

显示路由表: route print ; windows系统

route ?n ; Unix

5) Netstat

显示到ES的当前连接及端口: netstat ?n ; Windowx & Unix

6) Ipconfig&Ifconfig

显示ES的IP配置: ipconfig /all ; windows

ifconfig ?a ; unix

7) Nbtstat

显示当前名称解析缓存: nbtstat ?c ;

清除当前名称解析缓存: nbtstat ?r ;

第4章 协议属性

第4章 协议属性

一、OSI参考模型

应用层

表示层

会话层

传输层

网络层

数据链路层

物理层

二、全局协议分类

1、面向连接的协议:

windows size: 在需要目标系统确认的传输的数据包数。

队列数据传送: 对进入和发送的PDU指定序号, 在目的地再按序号重排数据;

流控: 确保发送的速率不超过目标接收的速率, 通过为传输建立窗口尺寸实现;

错误控制: 确保接收到的数据连续并无错, 如有丢失或损失的PDU, 则不发送ACK包。

面向连接的协议有: ATM、TCP、Novell SPX、Apple Talk ATP;

2、非连接的协议

不包括连接设置和终止, 没有流控和错误控制。

非连接的协议有: UDP、Apple Talk DDP、Novell IPX;

三、第2层: 数据链路层

1、Ethernet/IEEE802.3

2、Token Ring/IEEE802.5

四、PPP

五、SDLC

六、Frame Relay

七、ISDN

八、第3、4层: IP路由协议

1、IP

2、ICMP

3、TCP

4、UDP

第5章 Cisco测试命令和TCP/IP连接故障处理

第5章 Cisco测试命令和TCP/IP连接故障处理

一、故障处理命令

一、故障处理命令

1、show命令:

1) 全局命令:

show version ; 显示系统硬件和软件版本、DRAM、Flash
show startup-config ; 显示写入NVRAM中的配置内容
show running-config ; 显示当前运行的配置内容
show buffers ; 详细输出buffer的名称和尺寸
show stacks ; 提供路由器进程和处理器利用率信息, 用stack decode
show tech-support ; 显示几个show命令的输出
show access-lists ; 查看访问列表配置
show memory ; 用于测试内存问题

2) 接口相关命令

show queueing [fair|priority|custom]
show queue e0/1 ; 查看接口上队列的设置和操作
show interface e0/1 ; Cisco缺省的Ethernet封装方法是ARPA
show ip interface e0/1 ; 显示指定接口的TCP/IP配置信息

3) 进程相关命令

show processes cpu ; 显示路由器CPU的使用率和当前的进程
show processes memory ; 显示路由器当前进程的内存使用情况

4) TCP/IP协议相关命令

Show ip access-list ; 显示IP访问列表(1-199)
Show ip arp ; 显示路由器的ARP缓存(IP、MAC、封装类型、接口)
Show ip protocols ; 显示运行在路由器上的IP路由协议的信息
Show ip route ; 显示IP路由表中的信息
Show ip traffic ; 显示IP流量统计信息

2、debug命令

DEBUG不应在CPU使用率超过50%的路由器上运行。

1) 限制debug输出

在使用DEBUG获得所需数据后, 要关闭Debug

使路由器对所有消息都配置使用时间戳:

Router#service timestamps debug datetime msec localtime

Router#service timestamp log datetime msec localtime

缺省, error和debug信息仅发送到console, telnet到路由器上看不到debug和log的信息。想在telnet中看到debug和log信息:

Router#terminal monitor

Router#terminal monitor ; 关闭信息输出

Router#undebug all ; 关闭debug进程及所有相关信息的输出

可以应用ACL到debug以限定仅输出要求的debug信息。

如仅查看从10.0.1.1到10.1.1.1的ICMP包:

Router(config)#access-list 101 permit icmp host 10.0.1.1 host 10.1.1.1

Router#debug ip packet detail 101

2) 全局debug命令:

3) 接口debug

4) 协议debug

5) IP debug

5) IP debug

debug ip packets

3、logging命令

输出error和其它信息到console、terminal、路由器内部buffer或一台syslog服务器:

Router>show logging

Cisco路由器有8种可能的logging级: 0-7

Logging级别 名称 描述

1 Emergencies 系统不能用的信息

2 Alerts 直接行动

3 Critical 紧急情形

4 Errors 错误信息

5 Warnings 警告信息

6 Notifications 正常但重要的情形

7 Informational 信息

8 Debugging 调试

缺省地, console、monitor、buffer的logging被设置为debugging级, 而trap (syslog) 服务器的logging被设置为informational。

4、执行路由核心复制

core dump包含一份当前系统内存中信息的精确拷贝。捕捉包含在内存中信息的方法有:

1) 配置路由器在崩溃时执行Core Dump, 存储到TFTP、FTP、RCP服务器:

对TFTP协议, 只需指定TFTP服务器IP, 不需要任何附加的配置:

Router(config)#exception dump 192.168.1.1 ; TFTP服务器的IP地址

对FTP协议的配置:

Router(config)#exception dump 192.168.1.1 ; FTP服务器的IP地址

Router(config)#ip ftp username Kevin

Router(config)#ip ftp password aloha

Router(config)#ip ftp source-interface e0

Router(config)#exception protocol ftp

对RCP协议的配置:

Router(config)#exception protocol rcp

Router(config)#exception dump 192.168.1.1 ; RCP服务器的IP地址

Router(config)#ip rcmd remote-username Kevin

Router(config)#ip rcmd rcp-enable

Router(config)#ip rcmd rsh-enable

Router(config)#ip rcmd remote-host Kevin 192.168.1.1 kevin ;

2) 在系统没有崩溃的情况下, 执行Core Dump命令。

Router#write core

Core Dump仅在Cisco工程师测试和解决路由器问题时有用。

5、ping命令

ping用于测试整个网络可达性和连通性。可在用户EXEC模式和特权EXEC模式下使用。

IP的ping使用ICMP协议提供连通性和可能性信息, 缺省只发送5个echo信息。

扩展Ping的选项有：源IP地址；服务类型；数据；包头选项。

Ping的响应字符集

字符 解释 字符 解释

! Received an echo-reply message Q Source quench

. Timeout M Unable to fragment

U/H Destination unreachable A Administratively denied

N Network unreachable ? Unknown packet-type

P Protocol unreachable

6、tracert命令

tracert用于显示到达目标的包路径。可在用户模式和特权模式下使用。

Tracert的响应：

字符 解释 字符 解释

Xx msec The RTT for each packet * Timeout

H Host unreachable U Port unreachable

N Network unreachable P Protocol unreachable

A Administratively denied Q Source quench

? Unknown packet type

二、LAN连接问题

1、获得IP地址

主机可以动态或静态获得IP地址。

1) DHCP：DHCP比BootP多了地址池和租期。

2) BootP：

3) Helper Addresses：指定集中放置的DHCP服务器的IP地址

Ip helperaddress ip-address ；

No ip forward-protocol udp 137 ；

4) 路由器上的DHCP服务：配置路由器为一台DHCP服务器

5) DHCP和BootP故障处理

Show dhcp server ；

Show dhcp lease ；

2、ARP

ARP映射第2层MAC地址到第3层地址。

Show arp ； 显示路由器的ARP表

Debug arp ；

1) ARP代理：缺省Cisco路由器的ARP代理是启用的

在下列情况下，CISCO路由器将用自身的MAC地址响应ARP请求：

? 接收到ARP的接口上的Proxy ARP是启用的；

? ARP请求的地址不在本地子网；

? 路由器的路由表中包含ARP请求地址的子网；

3、TCP连接示例

三、IP访问列表

1、标准ACL：基于IP包的源IP地址允许或禁用

2、扩展ACL：提供源地址、目标地址、端口号、会话层协议进行过滤。

3、命名ACL：可以是标准ACL，也可以是扩展ACL。

命名ACL与编号ACL的区别：命名ACL有一个逻辑名，可以删除命名ACL中单独一行。

```
Ip access-list extended Example-Named-ACL
Deny tcp any any eq echo
Deny tcp any any eq 37
Permit udp host 172.16.10.2 any eq snmp
Permit tcp any any
```

第6章 TCP/IP路由协议故障处理

第6章 TCP/IP路由协议故障处理

一、缺省网关

当包的目的地址不在路由器的路由表中，如路由器配置了缺省网关，则转发到缺省网关，否则就丢弃。

Show ip route ; 查看Cisco路由器的缺省网关

二、静态和动态路由

三、处理[k_protocol/04937.htm](#) target="_blank">RIP故障

RIP是距离矢量路由协议，度量值是跳数。RIP最大跳数为15，如果到目标的跳数超过15，则为不可达。

RIP V1是有类别路由协议，RIP V2是非分类路由协议，支持CIDR、路由归纳、VLSM，使用多播（224.0.0.9）发送路由更新。

RIP相关的show命令：

Show ip route rip ; 仅显示RIP路由表

Show ip route ; 显示所有IP路由表

Show ip interface ; 显示IP接口配置

Show running-config

Debug ip rip events ;

常见的RIP故障：RIP版本不一致、RIP使用UDP广播更新

四、处理IGRP故障

IGRP是Cisco专用路由协议，距离矢量协议。IGRP的度量值可以基于五个要素：带宽、延时、负载、可靠性、MTU，缺省只使用带宽和延时。

IGRP相关的show命令：

Show ip route igrp ; 显示IGRP路由表

Debug ip igrp events ;

Debug ip igrp transactions ;

常见的IGRP故障：访问列表、不正确的配置、到相邻路由器的line down

五、处理EIGRP故障

EIGRP是链路状态协议和距离矢量混合协议，是CISCO专用路由协议。EIGRP使用多播地址224.0.0.10发送路由更新，使用DUAL算法计算路由。EIGRP的度量值可以基于带宽、延时、负载、可靠性、MTU，缺省仅使用带宽和延时。

EIGRP使用3种数据库：路由数据库、拓扑数据库、相邻路由器数据库。

EIGRP相关的show命令：

Show running-config

Show ip route

Show ip route eigrp ; 仅显示EIGRP路由

Show ip eigrp interface ; 显示该接口的对等体信息

Show ip eigrp neighbors ; 显示所有的EIGRP邻居及其信息

Show ip eigrp topology ; 显示EIGRP拓扑结构表的内容

Show ip eigrp traffic ; 显示EIGRP路由统计的归纳

Show ip eigrp events ; 显示最近的EIGRP协议事件记录

EIGRP相关的debug命令:

Debug ip eigrp as号

Debug ip eigrp neighbor

Debug ip eigrp notifications

Debug ip eigrp summary

Debug ip eigrp

常见的EIGRP故障: 相邻关系、缺省网关等的丢失、老版本IOS的路由、stuck in active。

处理EIGRP故障时, 先用show ip eigrp neighbors查看所有相邻路由器, 然后再用show ip route eigrp查看路由器的路由表, 再用show ip eigrp topology查看路由器的拓扑结构表, 也可用show ip eigrp traffic查看路由更新是否被发送。

六、处理OSPF故障

OSPF是链路状态协议, 维护3个数据库: 相邻数据库、拓扑结构数据库、路由表。

OSPF相关的show命令:

Show running-config

Show ip route

Show ip route ospf ; 仅显示OSPF路由

Show ip ospf process-id ; 显示与特定进程ID相关的信息

Show ip ospf ; 显示OSPF相关信息

Show ip ospf border-routers ; 显示边界路由器

Show ip ospf database ; 显示OSPF的归纳数据库

Show ip ospf interface ; 显示指定接口上的OSPF信息

Show ip ospf neighbor ; 显示OSPF相邻信息

Show ip ospf request-list ; 显示链路状态请求列表

Show ip ospf summary-address ; 显示归纳路由的再发布信息

Show ip ospf virtual-links ; 显示虚拟链路信息

Show ip interface ; 显示接口的IP设置

OSPF相关的debug命令:

Debug ip ospf adj ;

Debug ip ospf events

Debug ip ospf flood

Debug ip ospf lsa-generation

Debug ip ospf packet

Debug ip ospf retransmission

Debug ip ospf spf

Debug ip ospf tree

常见的OSPF故障: OSPF的每个area不超过100台路由器, 整个网络不超过700台路由器; 通配符掩码配置不当;

七、处理BGP故障

BGP (包括IBGP和EBGP) 的关键配置是邻居关系, BGP使用TCP建立相邻关系。

BGP相关的show命令:

Show ip bgp ; 显示BGP所学习到的路由
Show ip bgp network ; 显示特定网络的BGP信息
Show ip neighbors ; 显示BGP邻居信息
Show ip bgp peer-group ; 显示BGP对等组信息
Show ip bgp summary ; 显示所有BGP连接的归纳
Show ip route bgp ; 显示BGP路由表

BGP相关的debug命令:

Debug ip bgp 192.1.1.1 updates
Debug ip bgp dampening
Debug ip bgp events
Debug ip bgp keepalives
Debug ip bgp updates

典型的BGP故障:

八、再发布路由协议

九、TCP/IP症状和原因

症状 原因

本地主机不能与远程主机通讯 1) DNS工作不正常2) 没有到远程主机的路由
3) 缺少缺省网关4) 管理拒绝 (ACL)

某个应用程序不能正常工作 1) 管理拒绝 (ACL) 2) 网络没有正常配置以处理
该应用程序

启动失败 1) BootP服务器没有MAC地址的实体2) 缺少IP helper-address3)
ACL4) 修改NIC或MAC地址5) 重复的IP地址6) 不正常的IP配置

不能ping远程主机 1) ACL2) 没有到远程主机的路由3) 没有设置缺省网关
4) 远程主机down

缺少路由 1) 没有正确配置路由协议2) 发布列表3) 被动接口4) 没有通告
路由的邻居5) 路由协议版本不一致6) 邻居关系没有建立

相邻关系没有建立 1) 不正确的路由协议配置2) 不正确的IP配置3) 没有配
置network或neighbor语句4) hello间隔不一致5) 不一致的area ID

高的CPU利用率 1) 不稳定的路由更新2) 没有关闭debug3) 进程过重

路由触发活跃模式 1) 不一致的间隔2) 硬件问题3) 不稳定的链路

十、TCP/IP症状和行动计划

问题 行动计划

DNS工作不正常 1) 配置DNS主机的配置和DNS服务器, 可以使用nslookup校验DNS
服务器的工作

没有到远程主机的路由 1) 用ipconfig /all检查缺省网关2) 用show ip
route查看是否相应路由3) 如果没有该路由, 用show ip route查看是否有缺省
网关4) 如有网关, 检查到目标的下一跳; 如无网关, 修正问题

ACL 有分离的问题与ACL相关, 必须分析ACL、或重写ACL并应用。

网络没有配置以处理应用程序 查看路由器配置

Booting失败 1) 查看DHCP或BootP服务器, 并查看是否存在故障机的MAC实体
2) 使用debug ip udp校验从主机接收的包3) 校验helper-address正确配置
4) 查看ACL是否禁用包

缺少路由 1) 在第1台路由器上用show ip route查看所学到的路由2) 校验相邻
路由器3) 有正确的路由network和neighbor语句4) 对OSPF, 校验通配符掩码

相邻路由器3) 有正确的路由network和neighbor语句4) 对OSPF, 校验通配符掩码5) 检查应用到接口上的distribute list6) 验证邻居的IP配置7) 如果路由被再发布, 验证度量值8) 验证路由被正常的再发布

没有构成相邻关系 1) 用show ip protocol neighbors列表已构成的相邻关系

2) 查看没有构成相邻关系的协议配置3) 检查路由配置中的network语句4) 用show ip protocol/interface查看特定的接口信息, 如Hello间隔

第7章 处理串行线路和帧中继连接故障

第7章 处理串行线路和帧中继连接故障

一、处理串行线路故障

1、HDLC封装

High-level Data Link Control (HDLC) 是用于串行链路的一种封装方法, HDLC是Cisco路由器串行接口的缺省封装方法。

处理串行链路故障的第一步就是查看链路两端要使用相同的封装类型。

Show interface serial 1 ; 查看接口信息

Clear counters serial number ; 复位接口的计数器到0

正常情况下, 接口和line都是up的。

线缆故障、载波故障和硬件故障都可导致接口down, 通过校验电缆连接、更换硬件(包括电缆)、检查载波信令定位问题。

接口up, line down: CSU/DSU故障、路由器接口问题、CSU/DSU或载波的时间不一致、没有从远端路由器接收到keepalive信令、载波问题。应验证本地接口和远端接口的配置。

接口重启的原因:

? 数秒内排队的包没有被发送;

? 硬件问题(路由器接口、线缆、CSU/DSU);

? 时钟信令不一致

? 环路接口

? 接口关闭

? 线协议down且接口定期重启

show controllers serial 0 ; 显示接口状态、是否连有电缆、时钟速率

show buffers ; 查看系统buffer池, 接口buffer设置

debug serial interface ; 显示HDLC或Frame Relay通信信息

2、CSU/DSU环路测试

有四种类型的环路测试:

? 在本地CSU/DSU上测试本地环路;

? 在远端CSU/DSU上测试本地环路;

? 从本地NIU到远端CSU/DSU测试远端环路;

? 从远端NIU到本地CSU/DSU测试远端环路;

用PPP封装的串行链路上, PPP用协商Magic Number检测环回网络。

3、串行线中总结:

1) 症状和问题:

症状或情形 问题

Interface is administratively down;line protocol is down 1) 接口被从命令行关闭2) 不允许重复的IP地址, 两个使用相同IP地址的接口将down

Interface is down;line protocol is down 1) 不合格的线缆2) 没有本地提供商的信令3) 硬件故障(接口或CSU/DSU、线缆)4) 时钟

Interface is up;line protocol is down 1) 未配置的接口:本地或远程2) 本地提供商问题3) Keepalive序号没有增加4) 硬件故障(本地或远端接口、CSU/DSU)5) 线路杂音6) 时钟不一致7) 第2层(如LMI)

Interface is up;line protocol is up(looped) 链路在某处环路

Incrementing carrier transition counter 1) 来自本地提供商的信号不稳定2) 线缆故障3) 硬件故障

Incrementing interface resets 1) 线缆故障,导致CD信号丢失2) 硬件故障3) 线路拥塞

Input drops,errors,CRC,and framing errors 1) 线路速率超过接口能力2) 本地提供商问题3) 线路杂音4) 线缆故障5) 不合格线缆6) 硬件故障

Output drops 接口传输能力超过线路速率

2) 问题和行动

问题 解决行动方案

本地提供商问题 1) 检查CSU/DSU的CD信号和其它信号,看链路是否在发送和接收信息2) 如果没有CD信号或有其它问题,联系本地提供商处理故障

不合格或故障的线缆 1) 使用符合设备要求的线缆2) 使用breakout盒检查3) 交换故障线缆

未配置的接口 1) 使用show running-config校验接口配置2) 确认链路两端使用相同的封装类型

Keepalive问题 1) 验证keepalive被发送2) 配置了keepalive发送,debug keepalive3) 验证序号在增加4) 如果序号不增加,运行环路测试5) CSU/DSU环路,序号仍不增,则硬件故障

硬件故障 1) 更换硬件

接口在环路模式 1) 检查接口配置2) 如果在接口配置有环路,移除3) 如果接口配置被清除,清除CSU/DSU环路模式4) 如CSU/DSU不在环路模式,可能是提供商置环

接口administratively down 1) 检查是否有重复的IP地址2) 进行接口配置模式,执行no shutdown

线路速率大于接口能力 1) 使用hold-queue减少进入的队列尺寸2) 增加输出的队列尺寸

接口速率大于线路速率 1) 减少广播流量2) 增加输出的队列3) 如有需要,使用队列算法

二、处理帧中继故障

DLCI用于在帧中继中标识虚拟链路,DLCI仅仅是本地信令,DLCI与第3层IP地址相映射。

处理帧中继的步骤:

- 1) 检查物理层,线缆或接口问题;
- 2) 检查接口封装;
- 3) 检查LMI类型;
- 4) 校验DLCI到IP的映射;
- 5) 校验Frame Delay的PVC;
- 6) 校验Frame Delay的LMI;
- 7) 校验Frame Delay映射;

8) 校验环路测试;

1、帧中继的show命令

show interface

show frame-relay lmi ; 显示LMI相关信息 (LMI类型、更新、状态)

show frame-relay pvc ; 输出PVC信息、每条DLCI的LMI状态、...

show frame-relay map ; 提供DLCI号信息和所有FR接口的封装

2、帧中继的debug命令

debug frame-relay lmi ; 显示LMI交换信息

debug frame-relay events ; 显示协议和应用程序使用DLCI的细节

3、帧中继总纳

1) 症状和问题

症状或情形 相关问题

Frame Relay link is down 1) 线缆故障2) 硬件故障3) 本地服务商问题4)

LMI类型不一致5) Keepalive没有被发送6) 封装类型不一致7) DLCI不一致

从Frame Relay网络不能ping远端主机 1) DLCI指定了错误的接口2) 封装类型不一致3) ACL问题4) 接口配置错误

2) 问题和行动

问题 解决行动方案

线缆故障 1) 检查线缆并测试接头2) 更换线缆

硬件故障 1) 执行环路测试,以分离硬件2) 将线缆连接到路由器的另一同样配置的接口,如OK,则需更换硬件

本地服务提供商问题 1) 如环路测试使LMI状态up,但不能连接远端站点,联系本地载波2) 包含载波问题,就好象FR配置错误,如DLCI不一致或封装不一致。

LMI类型不一致 1) 校验路由器的LMI类型与PVC上的每个设备都一致2) 如使用公共提供商网络,不能访问LMI,与提供商联系

Keepalive问题 1) 使用show interface查看是否keepalive被禁用,或校验keepalive被正常配置2) 如果keepalive设置错误,进入配置模式并在接口上指定keepalive间隔

封装类型 1) 校验两端路由器的封装方式相同,如有非Cisco路由器,必须用IETF。用show frame-relay命令显示封装信息2) 用encapsulation frame-relay ietf更换封装方式,与可用frame-relay map设置某个PVC的封装。

DLCI不一致 1) 用show running-config和show frame-relay pvc显示指派给某接口的DLCI号2) 如DLCI号配置正常,联系供应商校验FR交换机是否有了相同的DLCI

ACL问题 1) 使用show ip interface显示应用到接口上的ACL2) 分析ACL,如有需要,删除或修改它

第8章 处理ISDN故障

第8章 处理ISDN故障

一、ISDN基本原理

二、常见ISDN故障

ISDN问题分成3类: 配置不当的路由器、物理线缆和ISDN协议、配置不当的交换

ISDN问题分成3类：配置不当的路由器、物理线缆和ISDN协议、配置不当的交换机。

1、配置不当的路由器

配置不当由于不同原因：typographical错误、从服务提供商提供的错误信息、本路由器配置不正确

- 1) SPID (Service Profile Identifiers) :如SPID和LDN配置错误，将有ISDN连接问题。SPID仅用于北美，只有服务提供商要求时才设置。
- 2) CHAP: CHAP认证在使用PPP封装的接口上使用。两端路由器的CHAP配置一定要相同。在PPP中，用户名和口令是大小写敏感的。
- 3) Dialer Map实体: Dialer map关联高层地址到相关的电话号码。每种协议需要一条dialer map语句。
- 4) 访问列表: ACL可用于ISDN连接以阻止某类型流量触发连接。
- 5) PPP:

2、物理层连接

- 1) BRI: 在现有电话线上提供数字服务。
- 2) ISDN BRI信道: $2B+D(2*64+16+48=192\text{kbps})$; ISDN BRI的物理帧为48bits, 链路每秒发送4000帧。
- 3) 本地环路: 客户和CO之间的链路，连接ISDN设备到ISDN交换机。
- 4) 物理层: 参考点 (R、S、T、U); 设备 (LT/ET、NT1、NT2、TE1、TE2、TA)

三、配置不当的电话交换机

在新安装ISDN时，必须考虑服务提供商ISDN交换机配置错误的可能性。

1、第2层故障处理:

ISDN第2层故障处理的目标: q. 921协议和PPP。

- 1) q. 921: ISDN的第2层在q. 921中定义。Q. 921信令在D信道上用LAPD协议传输。处理q. 921故障最常用命令是debug isdn q921, 问题常与TEI (terminal endpoint identifier)、SAPI (service access point identifier) 和SABME (set asynchronous balanced mode extended) 有关。

TEI=127表示广播; TEI=64-126保留用于动态分配。

SAPI=0表示当前第3层信令; 63表示用于TEI值分配的管理SAPI; 64为呼叫控制。

- 2) PPP: PPP使用LCP设置和维护链路; NCP配置和维护网络层协议。

2、第3层故障处理:

ISDN第3层也叫q. 931, 使用debug isdn q931命令可查看call setup、connect、release、cancel、status、disconnect和、user information。

ISDN第3层连接在本地路由器 (TE) 和远端ISDN交换机 (ET) 之间。

ISDN呼叫建立的过程:

- 1) SETUP: 在本地TE和远端ET之间发送信息
- 2) CALL_PROC: 呼叫处理信令
- 3) ALERT:
- 4) CONNECT
- 5) CONNECT_ACK:

3、交换机类型:

配置ISDN时，必须用isdn switch-type命令指定本地环路的交换机。

四、ISDN故障处理命令

- 1、ping: 在DDR中，ping命令触发一个呼叫，在第2个B信道up前，路由器已完成了ping。

配置ISDN时，必须用isdn switch-type命令指定本地环路的交换机。

四、ISDN故障处理命令

- 1、ping: 在DDR中，ping命令触发一个呼叫，在第2个B信道up前，路由器已完成了ping。
- 2、clear interface bri n: 重置接口上不同的计数器并中止接口上的连接。
- 3、show interface bri n: 显示关于ISDN BRI D信道的信息
- 4、show interface bri n 1 2: 显示ISDN BRI的B信道信息。
- 5、show controller bri: 显示接口硬件控制器信息和U接口，供Cisco的TAC处理故障。
- 6、show isdn status: 显示ISDN接口状态和各层详细信息。
- 7、show dialer: 显示关于DDR连接的信息，包括拨号、成功的连接、IDLE时间、呼叫数。
- 8、show ppp multilink:

五、调试ISDN

- 1、debug bri: 提供有关BRI B信道的信息，包括带宽信息
- 2、debug isdn q921: 获取关于接口D信道的信息，D信息用于在交换机和本地ISDN设备间传输信令。
- 3、debug dialer: 呼叫连接的原因和连接的状态。
- 4、debug isdn q931: 监视发生在第3层的事件。
Cause ID显示呼叫被拒绝的原因;
CallRef ID发送和返回的信息，用于分析路由器和交换机之间不同呼叫的特定会话。
- 5、debug ppp negotiation: 提供建立PPP会话的实时信息，可察看CHAP和PAP验证
- 6、debug ppp packet: 报告实时PPP包流，包括包的类型和所用的B信道

第9章 交换以太网故障处理

第9章 交换以太网故障处理

一、Switch、Bridge、Hub

广播域: 由Router控制

冲突域: 由Switch或Bridge控制

Switch和Hub比较:

类型 Switch Hub

Unicasts 仅发送到目标 发送到所有端口

Broadcasts 发送同VLAN中的所有端口 发送到所有端口

Aggregate bandwidth 等于每个端口的带宽×端口数 等于介质速率

Full/half-duplex 可全双工连接 仅半双工

Support for mixed media:Token Ring,Ethernet,FDDI... 依靠switch,可在不同帧类型和物理介质之间传输 仅支持同一介质

混合介质的支持 依赖于桥配置

处理帧 硬件(ASIC) 软件或

端口数量 从4到超过100 通常16个以下

帧类型转换 依靠桥配置

二、Catalyst故障处理工具

二、Catalyst故障处理工具

1、Catalyst命令行接口：

命令行接口有Native模式和Hybrid模式。本机模式配置第3层和第2层在一起；混合模式在不同CLI下配置第3层和第2层，常为基于set的CLI。

2、混合模式下的CLI：

- 1) show system: 关于switch的高级总结信息，包括供电状态、uptime和管理设置
 - 2) show port: 显示指定端口或一个模块上所有端口的信息（VLAN、速率、双工、状态、类型、...）
 - 3) show log: 报告重要事件，包括所有模块的重启、trap、供电失败、...
 - 4) show logging buffer: 等同于路由器的show log命令，根据logging级别，报告端口up或down、STP、...
 - 5) show interface: 报告管理模块上IP配置和SC0接口上VLAN信息。（sl0、sc0）
 - 6) show cdp: 显示相邻CISCO设备信息
 - 7) show config: 等同于show running-config命令，显示交换机除MSFC等外所有模块上所有设置，仅显示非默认设置。Show config all显示所有设置。
 - 8) show test: 仅显示switch管理模块状态，包括接口卡、供电、内存等。
 - 9) show mac: 显示大量计数，包括每端口帧流量、发出和进入的帧的总数量、丢弃、...
 - 10) show vtp domain:
 - 11) show cam: 显示与端口相关联的MAC地址
 - 12) 重复的MAC地址
 - 13) show spanntree: 显示每个VLAN的SPT进程状态
 - 14) show version: 显示硬件和软件版本号，包括内存、系统UP时间统计等
- ### 3、RMON (Remote Monitoring)
- RMON基于RMONProbe，从电路（物理介质）上采集数据信息。Router和Switch并不支持所有级别的RMON信息，更多的监控可以用SPAN (Switched Port Analyzer 交换端口分析，也叫Port Mirroring端口监控) 实现。
- ### 4、指示灯：
- 管理引擎上包含有负载LED，可以提示交换机的当前负载。在启动过程中，LED将闪烁；正常情况下，LED常绿；橙色LED提示有问题；红色LED提示有故障。

三、用STP控制环路

STP算法在802.1D中定义，用于在多交换机时控制重复路径，避免网络环路。

Cisco使用Port fast和Uplink fast时，要防止产生网络环路。

四、VLAN

VLAN有基于端口的静态VLAN和基于MAC的动态VLAN

1、ISL: Cisco专用协议，用于连接两台设备以支持多个VLAN。

- ? ISL只能在支持ISL的产品上使用
- ? ISL必须是点对点的
- ? ISL仅用于100Mb全双工
- ? ISL要求路由器的IOS和内存升级；
- ? ISL可以支持Token Ring；
- ? ISL添加30Bit到原始帧；
- ? ISL在帧的末尾包含CRC。

? ISL在帧的末尾包含CRC。

2、802.1Q: 用于连接非Cisco中继到Cisco设备

。

3、VTP: VTP使用多播通知VTP域中所有其它交换机关于域中VLAN的信息。

? VTP服务器:

? VTP客户机

? 透明VTP:

五、线缆问题

物理层标准:

线缆 10Mb 100Mb

3类线距离 100m 不可用

5类距离 100m 100m

多模光纤距离 2000m 2000m

单模光纤距离 高达100km 高达100km

1、线缆问题:

1) 万用表 (Multimeters) 和电缆测试器 (Cable Testers)

万用表 (Multimeters) 和伏欧表 (Volt-ohm) 用于验证电缆连通性, 只能用于测试铜线或其它基于电信号的电缆, 不能用于测试光纤。

电缆测试器 (Cable Testers) 既可测试电缆也可测试光缆, 提供给用户更多的被测试电缆的信息, 如: 连通性、断路、短路、距离过长、噪音、MAC信息、线路负载、...

2) 时域反射器 (TDRs) 和光时域反射器 (OTDRs)

TDR是更复杂的电缆测试器, 可用于定位电?械奈钢砒侍猓?叠庠谡裁次恢枚下貳

6.塘貳14砒?纫斐O室蟆?br />

2、交叉线

交叉线用于两台主机直接相连、连接两台网络设备。

以太网使用1、2、3、6四芯 (白橙、橙、白绿、绿), 而T1电路使用RJ-45的1、

2、3、5四芯

六、交换机连接故障处理

发生在交换机上常见的故障有速率和双工设置,

1、SPAN (交换端口分析器):

也叫Port Mirroring (端口监视器) 交换机拷贝所有被发送到工作站接口的包到另一接口, 这个接口没有被指定VLAN。

Set span enable ; 配置SPAN

使用SPAN既监视接收的、发送的或所有的包。

2、多层交换特性卡 (MSFC) 和Catalyst路由:

MSFC是一个在子板的Cisco路由器, 安装在管理模块上, 提供VLAN间路由。

在CLI下访问MSFC: session

3、路由器和交换机间VLAN:

路由器提供VLAN间的通信。

1) 广播管理:

路由器不转发广播, 交换机控制广播仅转发到是源端口所VLAN成员的端口。

2) 策略控制: 交换机没有策略, 而路由器提供连接VLAN的安全和策略控制

3) VLAN交换: 经过路由器转发一个包到同VLAN的不同接口

4) VLAN传输: 使用不同VLAN协议的两VLAN间或VLAN协议传输到非VLAN第2层协

协议。

5) 路由：在不同VLAN或非VLAN网络间通信

6) 路由器上VLAN故障处理：

show vlans

show arp

show interface

show cdp neighbor

debug vlan packet

debug spanntree

7) show vlans：在路由器上执行，显示路由器VLAN配置的细节，包括：VLAN名、接口、IP地址、VLAN封装协议、接口协议。

8) debug vlan packet：判定在中继上发送到路由器的数据的VLAN。

3、VLAN设计和故障处理

VLAN设计时注意事项：

1) 网络直径要少于8台交换机；

2) VLAN必须在某个限制内进行编号；

七、混合/本地模式命令转换

混合模式 本机模式 解释

Clear vlan No vlan 从配置中删除VLAN

Set cam agingtime Mac-address-table aging-time 设置保留MAC地址的超时值

Set port duplex Duplex 在特定端口上配置双工

Set port name Description 设置端口名

Set port speed speed 设置端口速率

Se tspan Monitor session 设置SPAN端口

Set spanntree Spanning-tree 设置STP信息

Set vlan Switchport access vlan 分配某端口到给定VLAN

Show cam dynamic Show mac-address-table dynamic 显示MAC到端口关系

Show port Show interface 显示端口信息

Show span Show monitor 显示SPAN端口

Show test Show diagnostic 显示启动测试结果

Show version Show version 显示交换机IOS版本信息

Show vlan Show vlan 显示VLAN信息

Show vtp domain Show vtp status 显示VTP信息

第10章 分离并纠正物理层和数据链路层故障

第10章 分离并纠正物理层和数据链路层故障

1、识别物理层问题的症状

物理层组件包括：接口 / 端口、模块、线缆、中继器、网卡、转换器等。

物理层问题将导致链路上数据完全或间断的丢失，应用程序失败，数据传输速率低。

设备的端口和特定部件的LED在正常工作时稳定，故障时LED状态将关闭、闪烁或其它颜色。

物理层问题的常见症状：

2、识别数据链路层问题的症状

数据链路层问题包括：不正常的帧类型（不相符的封装）、重复的MAC地址、换

数据链路层问题包括：不正常的帧类型（不相符的封装）、重复的MAC地址、换
换?鹞?层设备的不当行为。

第2层和第3层测试工具（CDP、PING）可以帮助检验并校验数据链路层问题。

3、用于分离物理层和数据链路层问题的命令和应用程序：

1) ES命令：

Ping host|ip-address ;

Arp ?a ;

Netstat ?rn ;

Ipconfig /all ;

Tracert ;

Winipcfg ;

Ifconfig ?a ;

Traceroute ;

2) Cisco IOS命令

Ping ;

Traceroute ;

Debug ;

Show version ;

Show ip interface brief ;

Show interface e 1 ;

Show cdp neighbor detail ;

Show controllers ;

Debug ppp|isdn|serial|asynch|frame-relay

Show arp ;

Debug arp|lapb|stun ;

4、纠正发生在物理层和数据链路层的命令和应用程序

arp ?d ;

interface ;

no shutdown ;

encapsulation ;

clock rate ;

controller ;

duplex full|half|auto

speed 10|100|auto

1) 纠正T1/E1问题的命令

channel-group channel-no timeslots timeslot-list speed 56|64

clock source line|internal

framing sf|esf; framing crc4|no-crc4

linecode ami|b8zs; linecode ami|hdb3

pri-group timeslote range

第11章 分离并纠正网络层问题

1、网络层问题的症状

2、分离网络层问题的ES命令

1) 通用命令：

ping

```
arp ?a
netstat
2) WINDOWS
Route print
Ipconfig /all
Tracert
Winipcfg
3) UNIX&MAC
Ifconfig ?a
Traceroute
Route ?n
3、分离网络层问题的Cisco IOS命令
1) 通用:
ping
trace
debug
show running-config
2) ARP
Show ip arp
Debug arp
3) 路由表
show ip route
debug ip routing
4) IP接口
Show ip interface brief
5) BGP
Show ip bgp
Show ip bgp summary
Show ip bgp neighbors
Debug ip bgp
6) IP流量
Show ip traffic
Debug ip icmp
Debug ip packet
7) IP访问列表
Show ip access-list
```

快捷键

17.4 使用IOS的编辑命令

命令 描述

Ctrl + A 移至命令行开头

Ctrl + E 移至命令行末尾

Ctrl + F 前移一个字符

Ctrl + B 后移一个字符

Esc + B 前移一个单词

Esc + F 后移一个单词

17.5 使用IOS的命令历史功能

命令 描述

Ctrl + P 或 ↑ (向上箭头) 重用前一条命令

Ctrl + N 或 ↓ (向下箭头) 重用下一条命令

Ipsec VPN

IPsec is

an IETF standard that employs cryptographic

1、Authentication(认证)

2、data integrity (校验数据的完整性)

3、confidentillity of packate payload (对数据包的加密)

IPSEC 使用三个protocols

Internet Key Exchange (IKE)

Encapsulating Seccurity Payload (ESP)

Authentication Head (AH)基本不用

five step of ipsec

1、interesting

LAB1:ipsec S2S vpn(site to site)

step1、构建模拟的ISP/internet

R2/3/4 router eigrp 90

step2:在用户上，做指向ISP的默认路由。

step3:定义VPN的感兴趣流量

R1:

```
ip acces-list extended VPN permit 192.168.1.0 0.0.0.255 192.,168.5.0
0.0.0.255
```

```
ip router 0.0.0.0
```

GRE

generic routing encapsulation(GRE)

支持多协议

不安全：不能提供加密、认证、

IPSEC 的不足之处

- 1、只能支持单播IP数据包，别的协议数据包都不支持
- 2、只支持
- 3、无法直接支持站点之间的动态路由协议。

为了解决以上IPSEC协议的不足之处。使用GRE

GRE is good at tunneling:

- 1.support multi
- 2.provides virtual ptp cinnectivity
allowing routing protcol to be used
- 3.

step0:

CE之间已经实现了公网接口的互通。

step1:

R1#

```
int tunnel 1
  tunnel sou 12.0.0.1
  tunnel des 45.0.0.5
  ip add 192.168.15.1 255.255.255.0
```

R1#

```
int tunnel 1
  tunnel sou 45.0.0.5
  tunnel des 12.0.0.1
  ip add 192.168.15.5 255.255.255.0
```

检查:

R1:show int tunnel 1

Wolf笔记

WOLF-1贾雷

CCNA

day-3

cisco设备的启动要点:

- 1、检测硬件(保存在rom)
- 2、载入软件(IOS) (保存在Flash)
- 3、调入配置文件(密码, IP地址, 路由协议都保存在此) (此文件保存在NVRAM)

0x2102: 正常调入配置文件到内存中

0x2142: 不调入配置文件到内存中

即表示 : 第六位(bit6)=0时就调入: 第六位(bit6)=1时就不调入

CISCO设备的文件系统:

RAM/NVRAM/FLASH/TFTP

LAB1: 路由器文件的备份:

~~~~~  
~~~~~

step1: 按图连接好电源线, Console, Ether-Net网线。

step2: 配置IP地址, 确认连接的正确性, 并运行数据通信测试。

step3: 在本机运行TFTP Server (TFTP是UDP的69号端口) (tftpd32)

step4: 察看FLASH中的IOS文件

ROUTER#show flash:

ROUTER#dir flash: (": "表示一个设备, 有时不打会出问题)

step5:

ROUTER#dir nvram (察看nvram中有哪些文件)

ROUTER#more nvram : startup-config (具体察看某个文件的内容)

step6:

```
ROUTER#copy flash: tftp:
ROUTER#copy nvram: startup-config tftp
```

LAB2:路由器的文件升级

~~~~~  
~~~~~

step1/2/3 与LAB1完全一致

step4:
ROUTER#copy tftp:flash:

2140 表进入ROMMON模式（按ctrl+break进入）

2141 表进入MINI IOS模式

2142 表进入正常模式

LAB3: 路由器的密码恢复

step1: 与lab1完全一致（无需网线）

step2: 配置路由器的密码
config t
enable password cisco1
enable secret cisco2

step3: 保存配置

（将当前的内存中的配置信息，保存到NVRAM中的start-config）

```
copy run..start-config
write
```

step4:将配置寄存器设置为0x2102，在线路由器的标准配置

```
R1(config)#config-register 0x2102
```

step5:当忘记了密码时，就在启动路由器的5秒内按CTRL+BREAK，进入ROMMON模式

```
>
>
>
```

step6:

```
>o 察看当前寄存器的取值
>o/r 0x2142 修改寄存器取值为0x2142
```

step7:

```
>i 重启路由器
```

step8:

在system configuration dialog 中输入no

step9:

```
ROUTER#dir nvram
```

```
ROUTER#more nvram:startup-config
```

step10:

将startup-config copy到内存中

```
ROUTER#copy nvram:startup-config running-config
```

(通过手工将startup-config copy 到running-config时, 接口全是关闭的, 需要手工打开)

step11:删除原密码

```
R1 (config) #no enable passwor
```

```
R1 (config) #no enable secret
```

step12:将所有需要工作的接口, 手工打开

```
R1(config)#int e0
```

```
R1(config-if)#no shut
```

step13:

```
R1#config-register 0x2102
```

step14:保存配置

```
R1#copy running-config startup-config
```

```
R1#write
```

LAB4: 交换机的文件备份

~~~~~  
~~~~~

step1: 按图连接好电源线, Console Eether网线

step2: 配置IP地址, 确认连接的正确性, 并进行数据通信测试

```
switch(config)#int vlan 1
```

```
switch(config)#ip add 192.168.1.180 255.255.255.0
```

```
switch(config)#no shut
```

step3:在本机运行tftp

step4: 备份交换机的IOS

```
switch#copy flash:c..... tftp:
```

备份交换机的配置文件

```
switch#copy flash:config.text tftp:
```

LAB5:交换机的IOS的升级

~~~~~  
~~~~~

step1/2/3 与LAB1完全一致

step4:升级IOS

switch#copy tftp:flash:

LAB6:交换机的密码恢复:

~~~~~  
~~~~~

~~~~~ step1: 先连接好Console, 将交换机断电, 然后按着交换机上的“MODE”不放, 再重新接入电源, 直到控制界面出现

交换机初始化失败为止, 然后跟着它要求的初始化步骤执行

step2: 对flash进行初始化

switch: flash\_init

switch: load\_helper

switch: dir flash: 查找flash中的config.text

switch: rename flash:config.text flash:co.t (将配置文件改名)

step3: 让交换机继续启动, 进入空白配置的交换机

switch: boot

step4: 把配置文件copy到内存

switch#copy flash:co.t running-config

step5: 删除原密码

switch(config)#no enable password

switch(config)#no enable secret

step6: 打开交换机的接口 (因为手工加入的是关闭的)

int vlan 1

no shut

step7: 保存

switch#write

step8: 删除不需要的原配置文件

switch#delete flash:co.t

switch交换的三种方式:

1、直通式: 交换机一收到数据帧就立刻开始转发

- 2、存储转发：交换机要完整地接收到数据帧，并且经过校验后，才开始转发
- 3、Fragment-free：交换机对接收到是前64BYTES，进行校验，如果正确就开始转发。

交换机的地址学习：（交换机MAC表）

交换机总是学习数据帧的源地址（MAC），将此地址与进入与进入交换机的端口进行映射。

交换机对于数据帧的转发：

- 1、如果目标MAC是单播地址，交换机就按照MAC表进行单播。（已知MAC）  
（除了目标MAC所对应的那个接口，之外的所有接口，都是不转发数据帧的）
- 2、如果目标MAC是组播/广播地址，交换机就会向除了入口以外的接口进行广播。
- 3、如果目标MAC是单播地址，但是交换机没有此MAC的对应映射端口，交换机就会  
向除了入口以外的接口进行广播。（未知MAC）
- 4、如果目标MAC地址是单播地址，但此目标MAC所对应的接口就是数据帧进入交换机的端口，交换机会丢弃数据帧。

在BPDU信息当中有一段区域叫做bridge ID, Bridge ID=bridge priority（优先级）+MAC（相加之和），交换机缺省  
优先级为32768（16进制为8000）。如果两个bridge ID相等将以MAC地址为准，  
以MAC地址最小的bridge ID值就小

当交换机运行stp协议，交换机的端口会存在如下状态：

blocking（阻塞）、listening（监听）、learning（学习）、forwarding（转发）

blocking（阻塞）--listening（监听） 间隔20秒时间

listening（监听）--learning（学习） 间隔15秒时间

learning（学习）--forwarding（转发）间隔15秒时间

从listening到forwarding所经历的时间被称作转发延迟（默认30秒）

port在各个状态的功能：

blocking：不能接收转发任何数据

listening：不能接收转发数据，但是可以接收BPDU信息

learning：不能接收转发数据，但是能够通过接收BPDU信息创建MAC地址表

forwarding：可以接收转发数据

LAB7：交换机的基本配置

~~~~~  
~~~~~

step1: 为交换机配置网管ip (本网段网管)

```
SW1(config)#int vlan 1
        ip add 192.168.1.1 255.255.255.0
        no shut
```

step2:配置交换机的网关: (跨网段网管)

```
SW1(config)#ip default-gateway 192.168.1.10
```

step3:控制接口的速率

```
SW1(config)#int f0/1
        switchport mode access(接入链路, 一般连接主机)
        speed 10
        speed 100
        speed auto(默认)
```

step4:控制接口的双工模式:

```
sw1(config-if)#duplex full
sw1(config-if)#duplex half
sw1(config-if)#duplex auto(默认)
```

LAB8: Port/Security (端口安全)

step1:

```
sw1(config)#int f0/2
        switchport mode access(接入链路)
        switchport port-security(启动端口安全)
        switchport port-security mac-address 00c0.abcd.1234(绑定
一个MAC)
        switchport port-security violation shutdown(关闭接口, 上
报)
        switchport port-security violation restrict(不转发数据,
上报)
        switchport port-security violation protect(不转发数据)
```

step2:在一个端口上绑定多个MAC地址

```
switchport port-security macimum 3
switchport port-security mac-address 00c0.abcd.1234
switchport port-security mac-address 00c0.abcd.1235
switchport port-security mac-address 00c0.abcd.1236
```

step3:

```
switchport port-security maximun 100
switchport port-security mac-address sticky
```

## VLAN:

实现vlan的主要目的: (segmentation) 将网络分片, 减少每个网络的规模, 即缩减每一个广播域的大小。

一个vlan, 对应着一个广播域, 对应着一个逻辑子网: A VLAN=A Broadcast domain=logical network(subnet)

HUB: 所有端口都在一个冲突域, 一个广播域中;

Switch: 一个端口对应一个冲突域, 所有端口都在一个广播域中;

Router: 一个端口对应一个冲突域, 一个端口对应一个广播域;

交换环境中的2 种连接类型:

1. access links: 指的是只属于一个VLAN, 且仅向该VLAN 转发数据帧的端口, 也叫做native VLAN. switches 把帧发送到 access-link 设备之前, 移去任何的VLAN 信息. 而access-link 设备不能与VLAN 外通信, 除非数据包被路由

2. trunk links: 指的是能够转发多个不同VLAN 的通信的端口. trunk link 必须使用100Mbps 以上的端口来进行点对点连接, 1 次最多可以携带1005 个VLAN 信息. trunk link 使你的单独的1个端口同时成为数个VLAN 的端口, 这样可以不需要层3 设备. 当你在switches 之间使用了trunk link, 多个VLAN 的信息将从这个连接上通过; 如果在你switches 之间没有使用 trunk link而使用一般的连接, 那么只有VLAN1 的信息通过这个连接被互相传递. VLAN1 默认作为管理VLAN。

vlan分类:

静态Vlan

动态Vlan

sw1(config)#default int fa0/1(让一个接口恢复默认配置)

LAB9: 静态Vlan的创建, 修改, 删除

~~~~~  
~~~~~

step1: 观察Vlan的配置信息:

SW1#sh vlan brief (默认情况下, 交换机的所有的端口都在vlan 1)

pc2#sh arp

pc2#debug arp

pc2在ping通pc1后再观察 “sh arp”

step2: 创建vlan:

```
sw1#conf t
    vlan 10(创建vlan10)
```

sh vlan brief观察

step3:将端口放入vlan 10:

```
sw1(config)#int f0/1
    switchport access vlan 10
```

测试:

在pc2上再ping pc1, 无法ping通!

(在不同vlan中的用户, 默认是不能通信的)

step4:

```
sw1(config)#int f0/2
    switchport access vlan 10
```

此之后再进行测试, 可通; (证明相同的vlan中的用户是可以互通的)

step5:修改vlan:

```
sw1(config)#vlan 10
sw1(config-vlan)#name ENG
sw1(config-vlan)#EXIT
```

```
sw1(config)#VLAN 20
sw1(config-vlan)#name sale
sw1(config-vlan)#exit
```

step6:删除vlan

```
sw1(config)#no vlan 20
```

LAB10:vlan间的路由

~~~~~  
~~~~~

step1:按图配置vlan, ip

step2:将R1/R2的路由能力关闭

```
pc2(config)#no ip routing
```

pc1同上

step3:在多层交换上, 创建两个与VLAN一一对应的vlan接口, 以此两接口作为两个vlan的网关

```
sw1(config)#int vlan 10
sw1(config-if)#ip add 192.168.10.100 255.255.255.0
sw1(config-if)#no shut
```

```
sw1(config)#int vlan 20
sw1(config-if)#ip add 192.168.20.100 255.255.255.0
sw1(config-if)#no shut
```

step4:

所有在VLAN10中的网关，都以if vlan 10 为网关：

```
pc1(config)#ip default-gateway 192.168.10.100
```

所有在VLAN20中的网关，都以if vlan 20 为网关：

```
pc2(config)#ip default-gateway 192.168.20.100
```

测试：每个用户都必需ping通自己的网关！！！！！！

step5:启动交换机的路由能力：

```
sw1(config)#ip routing
```

```
sw1#sh ip route
```

step6#不同vlan间的用户数据通信

测试：两个pc互通！！！！！！！！

## Trunk:

在一条网络介质（网线/光纤）上，同时传输多个vlan的信息

trunk的种类：

- 1、cisco的私有标准：ISL
- 2、开放性的业界标准：802.1Q

LAB11:TRUNK

step1:

802.1Q的Trunk:

```
int f0/21
```

```
switchport trunk encapsulation dot1q
```

```
switchport mode trunk
```

ISL的Trunk:

```
int f0/21
```

```
switchport trunk encapsulation isl
```

```
switchport mode trunk
```

两个交换机配置相同。

step2:察看Trunk链路的连接状态:

```
SW1#sh int trunk
```

step3:

```
pc3(config)#ip default-gateway 192.168.10.100
```

PC3设置了网关后,可以实现跨vlan,跨交换机的通信的。

## VTP (VLAN Trunk Protocol)

vtp只是用于构建富有效率的,便于管理的Vlan的网络

(让网络中的一个交换机VTP Server设定好Vlan信息,别的交换机都自动从这个交换机这里,动态地学习到VLAN信息)

cisco私有的协议

vtp信息,只能在trunk链路上传输。

VTP 是Cisco 创建的,但是现在已经不为Cisco 所私有.VTP 的主要目的是在一个交换性的环境中管理所有配置好的VLAN 使所有的VLAN 保持一致性VTP,允许增加,删除和重命名VLAN,然后这些修改后的信息传播到整个VTP 域里的所有switches 上。

LAB12: VTP

~~~~~  
~~~~~

step1:确定网络中的VTP Server/Client

```
sw1(config)#vtp mode server
```

```
sw2(config)#vtp mode client
```

step2:确定在一个相同VTP域名: (CCNA)

```
sw2(config)#vtp domain CCNA
```

注意: 所有vtp信息在sh run是看不到的

step3:检查vtp:

```
sw1/2#sh vtp status
```

step4:观察VTP信息的同步过程:

通过添加vlan观察版本号

Configuration Revision 2

当每改动一个属性时(如增加vlan,删除vlan,改名等动作),版本号就会改变

## STP (Spanning-Tree Protocol)

- 1、每个交换网络，都有一个根桥（根交换机：BID最小的交换机）
- 2、每个非根桥，都只有一个根端口。（根端口即是去往根桥开销最小的端口）
- 3、每条网络介质，都必定有一个指定端口。（指定端口，根端口都是转发数据包的）
- 4、除了指定端口，根端口以外的端口，都被堵塞，不转发数据包。

### LAB13: STP根桥的选定

~~~~~  
~~~~~

所有交换机默认的BID优先级都是0x8000（32768）

step1: 察看各个交换机的MAC地址

```
sw1#sh version  
Base ethernet MAC Address:00:0F:.....
```

step2:察看优先级:

```
sw1#sh spanning-tree vlan 1  
Bridge id priority 32768 (priority 32768 sys-id-ext 1)
```

step3:控制交换网络的根桥:

根桥:

```
sw1(config)#spanning-tree vlan 1 root primary(24576/0x6000)
```

备份根桥:

```
sw2(config)#spanning-tree vlan 1 root secondary(28672/0x7000)
```

```
sw1#sw1#sh spanning-tree vlan 1
```

通过spanning-tree vlan 1 priority 4096（0x1000）直接更改优先级

R1#sh controllers serial 1查看是DTE还是DCE!!!!

## day-4

### CDP

#### LAB1:CDP

~~~~~  
~~~~~

step1:

R1#sh cdp neighbors

step2:在接口级别控制CDP的运行:

R3(config)#int s1

R3(config-if)#cdp enable(启动接口的CDP能力, 默认)

R3(config-if)#no cdp enable(关闭接口CDP)

step3:在全局上, 控制所有接口CDP运行

R3(config)#cdp run(在全局上启动CDP)

R3(config)#no cdp run(在全局上关闭CDP)

step4:

SW1#sh cdp neighbors detail

SW1#sh cdp neighbors f0/1 detail

CDP timer:CDP 包传给每个活跃接口的时间间隔, 默认是60 秒, 即每60s发送一次cdp更新

CDP holdtime:某设备从相邻设备收到的包的保持时间, 默认是180 秒

## Telnet:通过Telnet, 实现远程网管

#### LAB2: Telnet

~~~~~  
~~~~~

step1:

在进行telnet之前, 首先确认能够与被telnet的设备正常通信 (ping)

step2:

R1#ping 192.168.10.2!!!!!!!!!!!!!!!!!!!!!!

step3:

R1#telnet 192.168.10.2(不成功)

出现这些字样: Password required, but none set(原因: VTY没有设置密码)

解决办法:

```
R2(config)#line vty 0 4
R2(config-line)#password 1234
R2(config-line)#login
R2(config-line)#exit
```

step4:再telnet

R1#telnet 192.168.10.2(此时只能进入用户模式，因为没有设置enable密码)

出现这些字样: User Access Verification  
password:1234

R2>

R2>

R2>en

此时无法进入，因为R2没设enable密码

step5:第三次再R2设置了enable后，再telnet

```
R2(config)#enable password cisco
```

R1#telnet 192.168.10.2

Trying 192.168.10.2 Open

password:

R2>

R2>

R2>en

password:

R2#

R2#exit

回到R1

step6:对交换机的Telnet是一样操作的，即在交换机上也需先设line密码，enable密码

step7: 察看已连接的telnet进程:

7-1: 在telnet发起方，察看自己发起的进程

```
R1#sh sessions
```

```
Conn      host
```

```
* 1 192.168.10.29
```

\*表示此telnet是活动的

7-2: 在被telnet方，察看自己被telnet的进程

```
sw2#sh users
```

| Line    | users | host(s) | idle | location     |
|---------|-------|---------|------|--------------|
| 1 vty 0 |       | idle    |      | 192.168.10.1 |

step8:发起telnet的设备，临时“挂起”（回到本设备上）：（telnet进程是没有中断的）

用快捷键可临时挂起：1、ctrl+shift,不松手再按6；

2、松手后再按下 ‘x’

通过这两步就可以在telnet时临时挂起。

当想回到被telnet的机器上时，只需在特权模式下按下“回车”即可回到被telnet方。

step9:彻底中断telnet

9-1: 发起方主动中断：

```
sw2#
```

```
sw2#exit
```

```
R1#
```

即用“exit”即可中断

也可用R1#disconnect 1 选择性的中断某一条telnet

9-2: 被telnet方强制中断

```
sw2#clear line 1
```

```
[confirm]
```

```
[ok]
```

```
sw2#
```

step10:在多个被telnet设备之间进行切换：

```
R1#sh sessions
```

```
conn  host
```

```
* 1    192.168.10.2
```

```
2    192.168.10.29
```

```
R1#2
```

就可进入ip地址为192.168.10.29的主机。

## ACL/访问列表

主要作用：控制数据包，对其进行过滤。

还可以完成: QOS, DDR, 路由过滤;

ACL的分类:

标准ACL: 控制数据包的源IP地址。

1~99, (1300~1999)

扩展ACL: 控制数据包的源IP, 目标IP, 协议号, 源端口号, 目标端口号.....

100~199, (2000~2699)

命名ACL (标准/扩展ACL)

## CCNP

### BGP

#### **bgp-day1**

AS:自治系统

AN as is a collection of networks under a single technical administration.

(独立的一个技术/管理域)

IGP:

包括RIP/IGRP/EIGRP/ODR/OSPF/ISIS 运行在同一个AS中

IGP是通过cost/metric判断路由的优, 约小越好

IGP的主要任务:

更快/更好的正确描述路由信息, 尽快地将数据包送到目的地

强调收敛速率

BGP: (v4)

运行在不同的AS之间, 用于对路由的控制/策略

BGP的主要任务: 网管的人为意图的集中体现, 强调策略控制, 不强调收敛

BGP是通过BGP属性/attributes, 判断路径的优劣, 从中进行选择

BGP可以通过网管所定义的策略, 实现根据路由的操纵

AS: 独立的技术/管理域, 通常是一个大型公司, 或者组织, 或者国家

BGP本身就是一种策略路由/PBR

BGP本身就是一种策略路由/PBR  
实现网管的人为的集中体现

BGP是一种AS BY AS 的高级的路径向量/DV协议  
BGP认为:每经过一个AS就是一跳  
RIP认为:每经过一个router是一跳

BGP号:1~65535 (64K)BGP区域号, 其中64512~65535为私有号(理解时为63k-64k这1K之间).  
BGP的宿主: 视SP为宿, 自己为主, 接多个SP为多宿主。

应该使用BGP的情况:

1. ISP

当允许AS 1 的数据包穿越AS 2, 但是不允许AS 1 的用户访问AS 2内部的时候

2. Multihome/多宿主

对于一个用户的AS, 如果他同时连接到多个AS/ISP

3. PBR

当需要对BGP路由/数据, 进行人为控制/操纵的时候

不该使用BGP的情况:

1. 与ISP只有单连接, 没有同时连接到多个ISP

2. 如果网络硬件设备的档次不够(内存/CPU)

3. 对BGP路由操纵理解有限, 无法预计启动BGP的后果

4. 链路带宽不足

BGP特征:

1. BGP是高级DV协议

2. BGP工作在TCP/IP协议栈的4层:TCP 179 端口

3. BGP是触发/增量更新的协议

4. BGP通过周期性的发送keepalive消息, 保证TCP连接的可靠性

5. BGP使用多种"BGP属性/Attribute"来衡量路径的优劣

6. BGP是为巨型网络设计的, 意味着这种路由协议, 可能带来海量的路由和数据

BGP协议的三张表:

1. BGP的邻居表:

BGP的邻居可以直接相连, 也可以凌空建立

2. BGP转发表

同一个目标网络的多条可能路径

2-2:但是,BGP这种路由协议,默认情况下,是不进行负载均衡的,也就是说,到达一个目标网络,只允许有一条可用

路径

2-3:BGP路由器通过“BGP属性”,从多条候选路由中,优选出一条,它自己认为最好/最优/的路径,作为到达这个

目标网络的“最佳路由”,这条路由也会被称之为“优化”了(“优化”的路由,会标上“>”)

2-4:只有优化了的路由,才能作为BGP选送出的,最佳的,去参加,到达这个目标网络的“AD竞争/竞选”

3. 路由表:

上面BGP提交的“优化”路由,在经过“AD竞选”之后,如果获胜,才能成功地送进路由表

各路由协议的管理距离:

|              |     |    |
|--------------|-----|----|
| 直联           | 0   |    |
| static:      | 1   |    |
| EIGRP :      | 5   | 汇总 |
| EBGP:        | 20  |    |
| EIGRP:       | 90  |    |
| IGRP:        | 100 |    |
| OSPF:        | 110 |    |
| IS-IS:       | 115 |    |
| RIP:         | 120 |    |
| ODR:         | 160 |    |
| EIGRP (EX) : | 170 |    |
| IBGP:        | 200 |    |

BGP的4种消息类型:

1. open 相当于Hello
2. keepalive 保持
3. update CNLRI网络层可达信息
4. notification (发生错误时,就发送一个notification包)

BGP states(状态):

|              |           |
|--------------|-----------|
| Idle         | 初始状态      |
| Active       | TCP三次握手   |
| Open sent    | 发open包    |
| Open confirm | open包确认状态 |
| Established  | 邻居建立成功    |

Peers=neighbor

EBGP peer 不同AS之间的邻居

IBGP peer 相同AS之间的邻居

any two routers that have formed a tcp connection to exchange  
bgp routing information are  
called bgp peers or neighbor

#### 1. EBGP neighbor

两个BGP neighbor分别属于两个不同的AS, EBGP neighbor通常是直接相连的

#### 2. IBGP neighbor

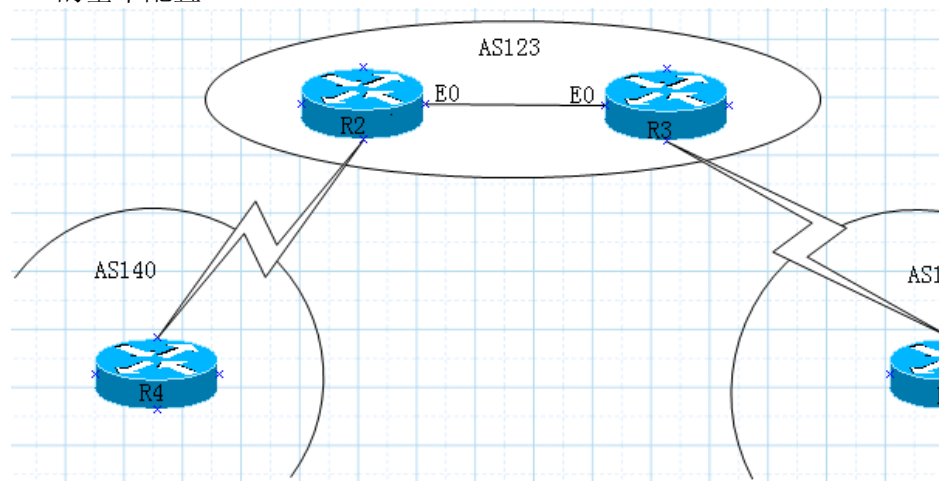
两个BGP neighbor同时属于一个相同的AS.

一个IBGP邻居可以是直接的, 也可以是非直连的

Router-id 作用: 选路、同步、防止回路。各路由器中R-id最好不同,  
同一AS下R-id必须不同。

BGP的更新源就是TCP连接的源地址。

BGP的基本配置:



#### LAB1:BGP的基本配置

step1:确认L1/L2的联通性

step2:确保L3的路由的通达(通过IGP实现):

(一般都是在同一个AS中完成)(两个AS之间是不运行IGP的)

~~~~~  
~~~~~

在AS123中所有路由器(R2/R3), 运行IGP:rip v2

```
router rip
```

```
v 2
```

```
net 23.0.0.0
no auto-summary
```

测试L3网络的通 ping!!!!!!1

step3:启动BGP, 并且立刻指定全局唯一的bgp router-id. (一个路由器只能使用一个BGP进程)

```
R2(config)#router bgp 123
R2(config-router)#bgp router-id 123.0.0.2
其他路由器相同
```

step4:构建BGP neighbor

```
R5-R3:EBGP
R5:neighbor 35.0.0.3 remote-as 123
R3:neighbor 35.0.0.5 remote-as 150
```

R3-R2:IBGP

```
R3:neighbor 23.0.0.2 remote-as 123
R2:neighbor 23.0.0.3 remote-as 123
```

R2-R4:EBGP

```
R2:neighbor 24.0.0.4 remote-as 140
R4:neighbor 24.0.0.2 remote-as 123
```

R4#debug ip bgp 观察BGP邻居建立的过程及其5个状态

- 1.idle:routers is searching
- 2.connect:completed three-way tcp 179 handshake (可通过sh tcp brief查看tcp连接的摘要信息)
- 3.open sent:open message sent
- 4.open confirm:router received agreement
- 5.established:peering is established;

BGP routing begins

R5#sh ip bgp summary (查看BGP邻居的简要信息)

| neighbor | v | as  | state/prfxrcd |
|----------|---|-----|---------------|
| 35.0.0.3 | 4 | 123 | (空白)/ 0       |
| 14.0.0.4 | 4 | 100 | 3             |

0: 没有从这个邻居收到BGP路由, 但是和这个邻居的BGP邻居关系已经建好

3: 从这个邻居收到三条BGP路由

step5:通过NETWORK命令, 宣告BGP路由

step5:通过NETWORK命令,宣告BGP路由

```
R5:router bgp 150
```

```
net 105.1.0.0 mask 255.255.0.0
```

接口所在的网络号      这个网络的准确的路由长度

network的含义:

在IGP中:

1. 激活接口, 在这个接口中运行此IGP路由协议  
(在IGP中, 路由协议是向运行协议的接口, 发送路由更新)
2. 这个路由器的IGP路由协议, 正在为此接口所在的网段进行路由

在BGP中:

- 1: 这个路由器的BGP路由协议, 正在为此BGP网段, 进行路由  
(在BGP中, BGP路由协议是向BGP neighbor, 发送路由更新, 没有了激活接口的含义)

在bgp路由器上, 检查自己的BGP路由, 是否已经成功地宣告到BGP进程中:

```
R5#sh ip bgp (查看转发表, BGP数据库)
```

Network

Next Hop(是指更新源)

```
*>105.1.0.0/16
```

```
0.0.0.0(源自本路由器)
```

查看R3向R2发送了哪些BGP路由条目:

```
R3#sh ip bgp neighbor 23.0.0.2 advertised-routes
```

step6: IBGP路由的优化

6-1: (R3)

BGP路由器, 把自己学到的BGP路由, 转发给别的BGP邻居的条件:

每个BGP路由器, 对于特定的某条BGP路由, 必须是自己已经**优化**的路由, 才具备转发给别的BGP邻居的能力

注意: 这个必要条件, 不是充分条件!

即使自己已经优化, 但此路由器, 可能转发, 也可能不转发

6-2: (R2)

IBGP路由的必要优化条件: (以下是必要条件, 而不是充分条件)

1: 下一跳可达

2: 路由的同步

6-3: 察看R2上, 在R3没有任何更改前, 当前的BGP路由:

注意观察:

1: 本路由器, 是否能够到达这个下一跳

- 1: 本路由器, 是否能够到达这个下一跳
- 2: 本路由器, 是否能够通过IGP, 学到这跳路由

如果BGP中的路由不能优化, 就不可能送进路由表

6-4: next hop问题

(问题本质: R2上 没有到达“这条BGP路由的下一跳”的路由: 35. 0. 0. 0/24)

因为在BGP路由通告中, 当BGP路由器收到相邻的AS发来的路由,  
其下跳是: 相邻的AS的边缘节点 (EBGP的更新源)

所以整个AS123的所有路由器 (R2/R3), 现在收到的AS150中的所有BGP路由, 的下一跳, 都是35. 0. 0. 5

但是, 因为两个AS之间的链路是不运行IGP的, 所以R2没有到达这个下一跳的路由

解决办法: 更改下一跳为本AS内, 可以通过IGP学到的路由

```
R3# router bgp 123
    neighbor 23.0.0.2 next-hop-self
```

6-5: 同步问题

(问题的本质: R2是否能够通过IGP, 得到这条105. 1. 1. 0/24路由)

BGP路由器, 对于从IBGP邻居学的路由, 默认要求同步.

所谓同步, 是指:

如果R2能够通过IGP (RIP), 学到这条路由 (105. 1. 1. 0/24), 那么就同步 (也就是: 满足同步要求)

如果R2不能够通过IGP, 学到这条路由, 那就不满足同步

但是,

因为实际上R2是不可能通过IGP, 学到另外的一个AS中的BGP路由的,  
所以只能关闭同步规则, 也就是不遵循同步规则

解决办法: 关闭同步规则

```
R2#router bgp 123
    no synchronization
```

clear ip bgp \* (硬清除)

这样R2就可以优化了

step7:EBGP的路由优化问题

1:下一跳问题

因为BGP路由的下一跳是其直连路由,所以不存在下一跳不可达的问题

ebgp没有同步的问题

## bgp-day2

BGP 黑洞

BGP的更新源

原则1:在默认情况下,BGP路由器以自己路由表中,到达对方BGP邻居地址的路由所指示的出接口,

作为自己的BGP更新源

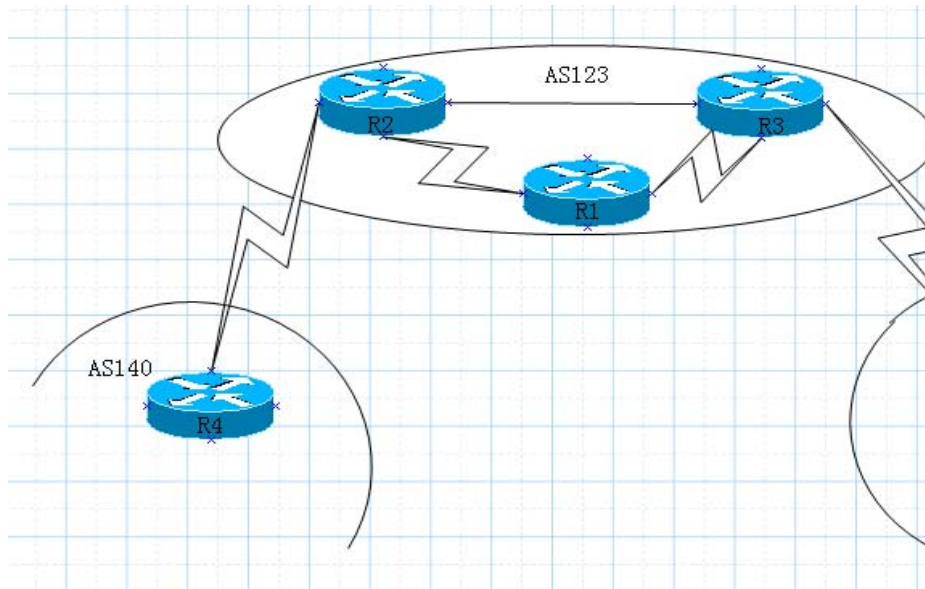
原则2:

当BGP路由器,收到邻居发来的BGP信息时,会检查其源地址,然后和自己宣告的neighbor的目标地址进行比较,

如果一致,这个BGP session才可能建立起来.

简单的说:R1用neighbor 命令所指的地址,若是R2的更新源,那么TCP链接就能建立成功. 另外,双方是都发请求TCP链接的,即有两条,会随机取一条.

~~~~~  
~~~~~



#### LAB1:验证通过物理接口, 构建IBGP邻居的不稳定性

step1:确认L1/L2的通达性

step2:确认L3的IGP的通达性

step3:通过物理接口, 在R2/R3之间构建IBGP邻居

结论:

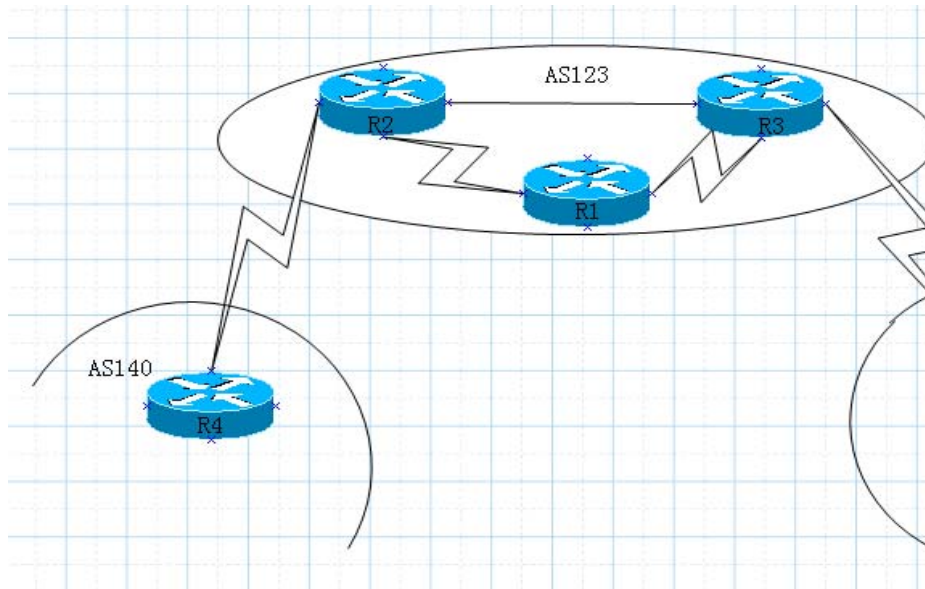
在IBGP中, 如果使用物理接口构建邻居, 是很不稳定的.

很可能因为某条物理链路的抖动, 导致IBGP邻居的FLAPPING/抖动,

建议:

使用环回口/Loopback接口, 构建IBGP邻居.

~~~~~  
~~~~~



#### LAB2: 以loopback作为更新源

step1: 分别在R2, R3构建环回

step2: 删除原先通过物理接口建立的ibgp邻居

step4: 通过环回口构建IBGP邻居

4-1:

以对方的环回口, 作为IBGP的目标地址:

```
R2#neighbor 3.3.3.3 remote-as 123
```

```
R3#neighbor 2.2.2.2 remote-as 123
```

4-2:

注意: 删除原物理接口所做的IBGP邻居时, 相应的下一跳将自动删除.

所以更改了IBGP邻居后需要把下一跳也做相应更改

```
R2(config-router)#nei 3.3.3.3 next-hop-self
```

```
R3(config-router)#nei 2.2.2.2 next-hop-self
```

4-3:

以自己的环回口, 作为IBGP连接的源地址:

```
R2#neighbor 3.3.3.3 update-source loopback 2
```

```
R3#neighbor 2.2.2.2 update-source loopback 3
```

step5:

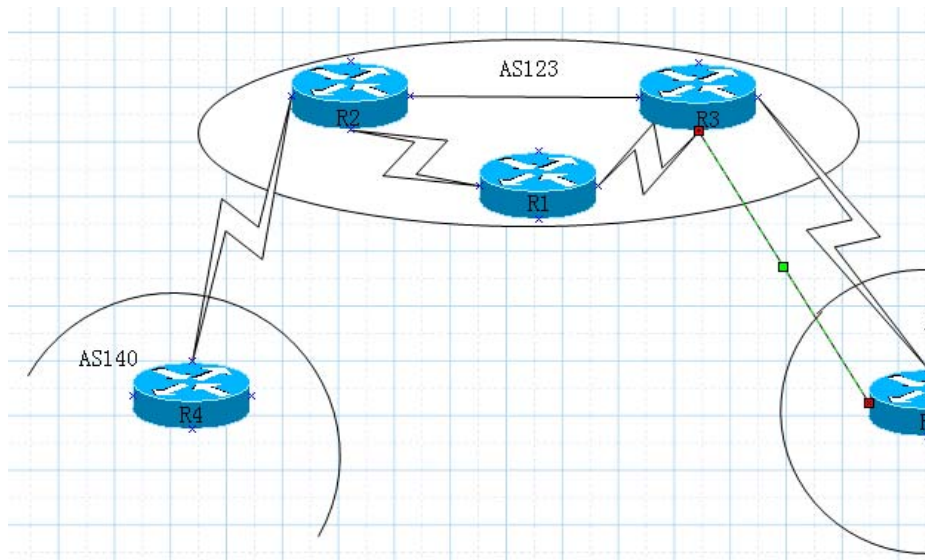
任意切断本AS123中的物理链路,

只要两个IBGP路由器R2/R3之间, 还有最后一条能够到达对方的环回口的路由, IBGP邻居都不会中断

建议: 凡是构建IBGP, 默认都使用环回做更新源, 以构建稳定的IBGP

在本LAB中, 无需关心路由优化, 和BGP路由是否能通达的问题,  
只需观察到R2/R3间的稳定的IBGP session

~~~~~  
~~~~~



**LAB3:** 以LoopBack接口作为EBGP更新源, 构建稳定的EBGP Session

Step1: 在两AS间, 构建多条冗余链路:

Step2: 为两AS间的EBGP路由器, 构建环回口

Step3: 在各自路由器上, 指定到达对方环回口的静态路由

因为有多条冗余链路,

所以有两条到达对方环回口的, 静态路由

```
R5(config)#ip route 3.3.3.3 255.255.255.255 35.0.0.3
          ip route 3.3.3.3 255.255.255.255 100.0.0.3
```

```
R3(config)#ip route 5.5.5.5 255.255.255.255 35.0.0.5
          ip route 5.5.5.5 255.255.255.255 100.0.0.5
```

Step4: 建立EBGP邻居:

```
R3#router b 123
  neighbor 5.5.5.5 remote-as 150
```

```
R5#router b 150
  neighbor 3.3.3.3 remote-as 123
```

Step5: 告知对方, 自己的更新源:

Step5:告知对方,自己的更新源:

```
R3#neighbor 5.5.5.5 update-source loopback 3
R5#neighbor 3.3.3.3 update-source loopback 5
```

Step6:更改EBGP的TTL值(Time to Live )

因为EBGP的TTL值默认是1,

所以EBGP的TTL值最少要设为: 2

而实际上EBGP多跳这个命令,在不指定其取值时,会自动默认指定为255

```
R5#neighbor 3.3.3.3 Ebgp-multihop 2
```

```
R3#neighbor 5.5.5.5 Ebgp-multihop (255)
```

TTL: Time to Live 是L3的IP包头中的一个特定字段,IP包每经过一个路由设备,其TTL会自动减1

如果TTL减到为0,即使路由器有去往目标的路由,也不会继续转发这个IP包.

Step7:测试

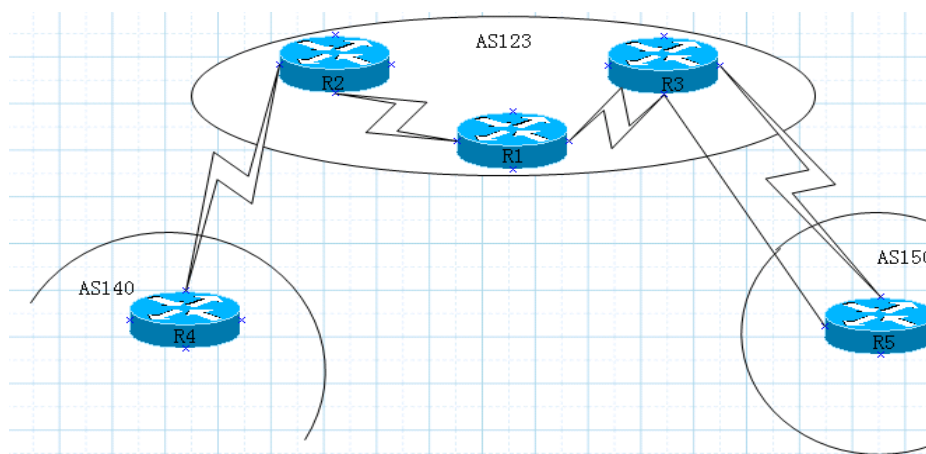
```
R3#ping 5.5.5.5 source 3.3.3.3 repeat 1000000 size 15000
```

结论:一般情况下,EBGP的邻居关系,是不需要使用环回口构建邻居的默认都直接使用物理接口

在只有单链路的时候,都是使用物理接口构建邻居

只有在两AS之间,存在多条冗余链路的时候,才需要考虑使用环回口构建EBGP邻居,以确保其EBGP的稳定性

~~~~~  
~~~~~



LAB4:观察BGP Black hole 的形成

Step1: 确认在本拓扑中的3个BGP Session

特别注意: R2和R3之间没有tcp是没有物理连接的

Step2: 在R3上, 对R2说, 下一跳是我自己, 帮助R2解决了下一跳问题:

```
neighbor 2.2.2.2 next-hop-self
```

R2也一样.

Step3: 观察黑洞:

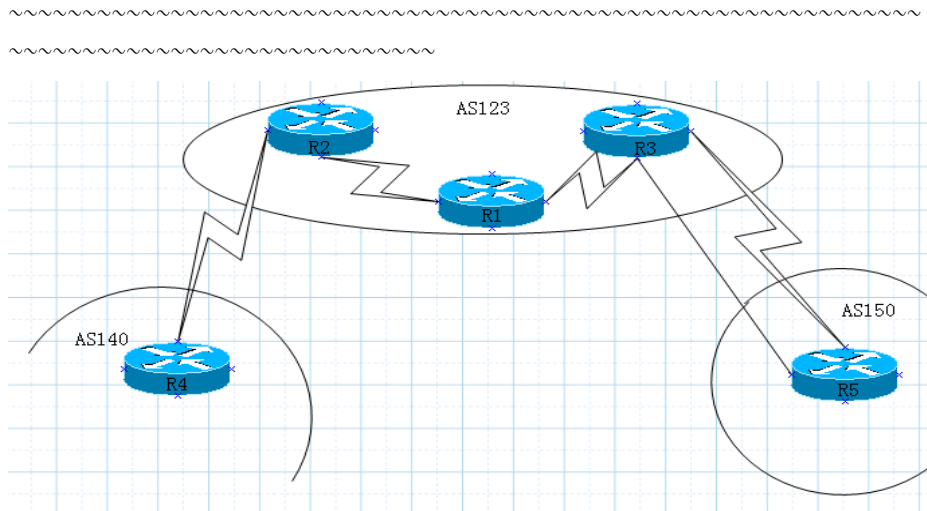
R1,

因为没有运行BGP,所以没有BGP路由,丢弃了去往BGP路由的数据包,从而,R1成为路由黑洞.

BGP路由黑洞的解决方案:

解决BGP路由黑洞,可供选择的解决方案 / Solution:

1. redistribute Selected BGP into IGP
2. full-mesh IBGP
3. Part-mesh IBGP + Reflector (BGP路由反射器)
4. confederation (联邦)
5. MPLS



#### LAB5:

Part-mesh IBGP, **Redistribute selected BGP into IGP**,  
with Sync(同步/synchronization)

step1: 定义需要重分布到IGP中的,BGP路由:

```
R2(config)#ip prefix-list bgp-104 permit 104.1.1.0/24
```

```
R3(config)#ip prefix-list BGP-105 PERmit 105.1.0.0/16
```

step2: 通过Route-map,控制重分布到IGP的范围

```
R2(config)#route-map 222
```

```
R2(config-route-map)#match ip add prefix-list BGP-104
```

```
R2(config-route-map)#set metric 1
```

```
R3(config)#route-map 111
R3(config-route-map)#match ip add prefix-list BGP-105
R3(config-route-map)#set metric 1
```

不要配置:”route-map BGP-RP permit 20”  
一旦配置,意味着所有一切BGP路由都进入RIP

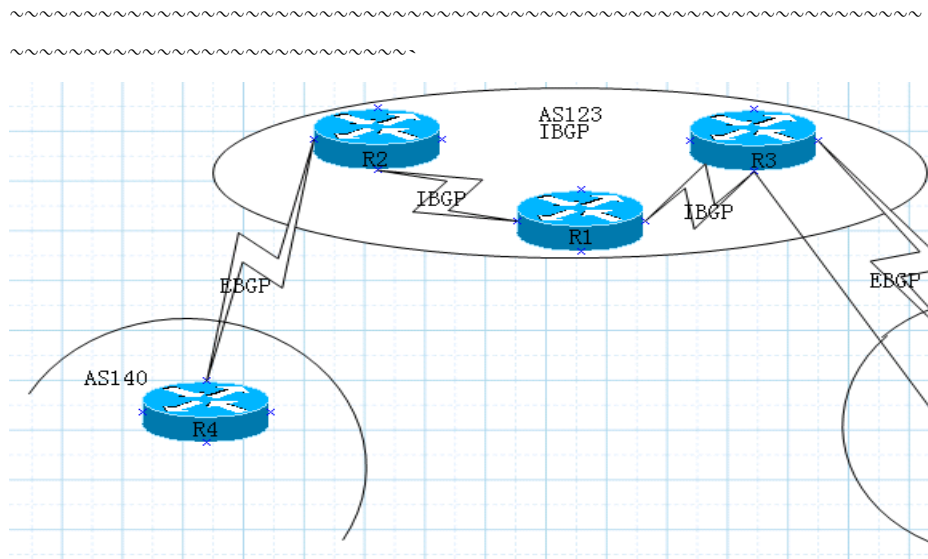
Step3:按照route-map BGP-RP所定义的条件,将BGP路由注入RIP:

```
Router rip
Redistribute bgp 123 route-map BGP-RP
```

```
R2# (递归查询)
B 115.0.0.0 [120/2] vis 3.3.3.3
R 3.3.3.0 [120/2] via 12.0.0.1
C 12.0.0.0 is directly connected ,Serial 0
```

Step4: R2遵循同步原则(即不关闭同步),仍然是可以对这条BGP路由优化的

原因是:  
R2既可以通过IBGP学到105. 1. 0. 0/16  
也可以通过IGP (rip), 学到此路由  
所以同步要求满足



LAB6:FULL-mesh with NO-S (peer-group)

step1:在R1上, 使用peer-group, 对R2/R3建IBGP邻居

1-1:定义peer-group

```
neighbor R1-PG peer-group
neighbor R1-PG remote-as 123
neighbor R1-PG update-source lo 1
```

1-2:对不同的IBGP邻居, 调用peer-group  
nei 2.2.2.2 peer-group R1-PG  
nei 3.3.3.3 peer-group R1-PG

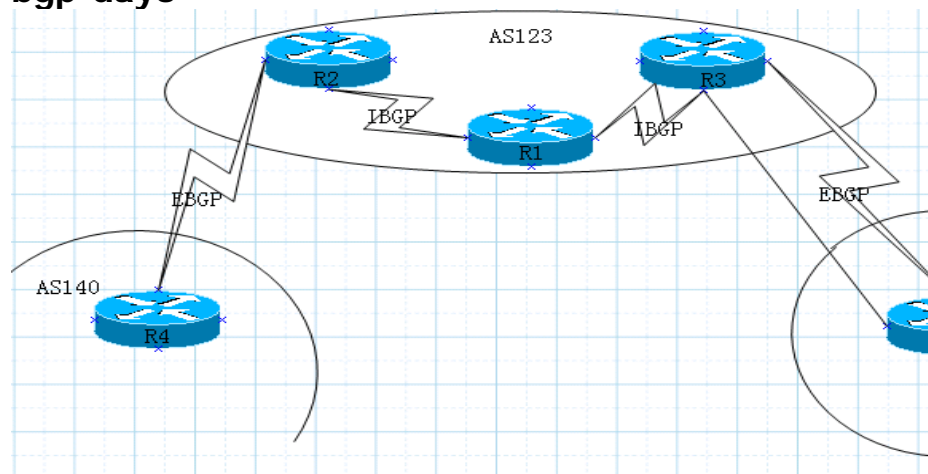
R2/R3/R1邻居建立, 仍然可以使用普通方法建立

step2:确保整个AS123中的所有BGP路由器的, 下一跳, 同步问题能解决  
R1/R2/R3#关闭同步

R2上, 对R1/R3 say next-hop-self  
R3上, 对R2/R1 say next-hop-self

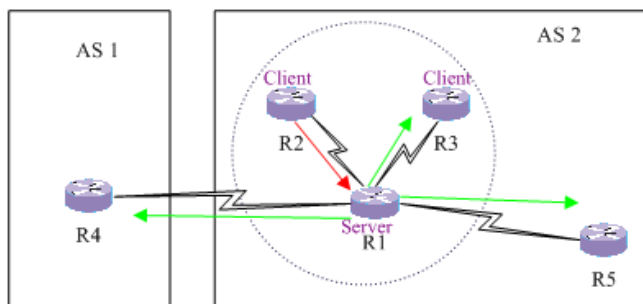
~~~~~  
~~~~~

### bgp-day3



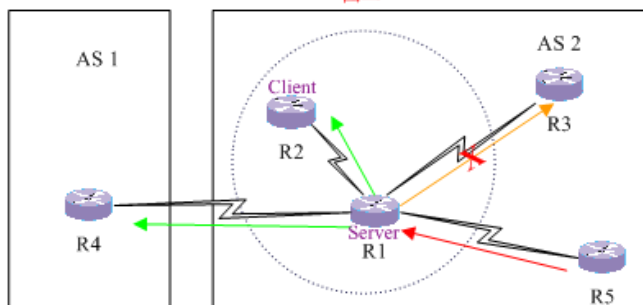
BGP 的Reflector (反射器原理图)

图一



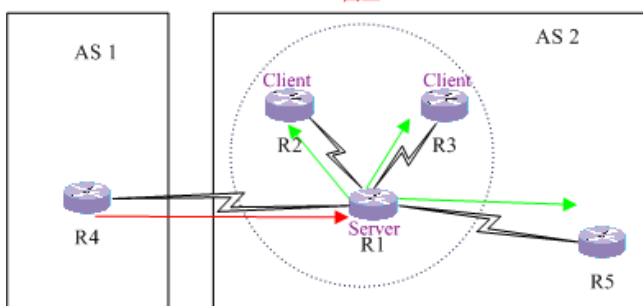
2 发给1,然后1再反射

图二



5发给1,然后1再反射

图三



4 发给1,然后1再反射

注意:拓扑的R2与R3之间是没有IBGP连接的

LAB7:Part-mesh IBGP+RR

step1:删除R2与R3之间的IBGP

IBGP的水平分割原则:

默认情况下, 对于一个BGP路由器来说, 从一个IBGP邻居那里学到的BGP路由, 是不会传递给另外的一个IBGP邻居的, 但是会传递给另外的EBGP邻居

对于RR:

从EBGP来的路由, 会传给客户, 非客户, EBGp  
从客户来的路由, 会传给客户, 非客户, EBGp  
从非客户来的路由, 会传给客户, EBGp

step2:解决办法:RR

在R1上, 定义R2/R3为自己的“路由反射器的客户端”

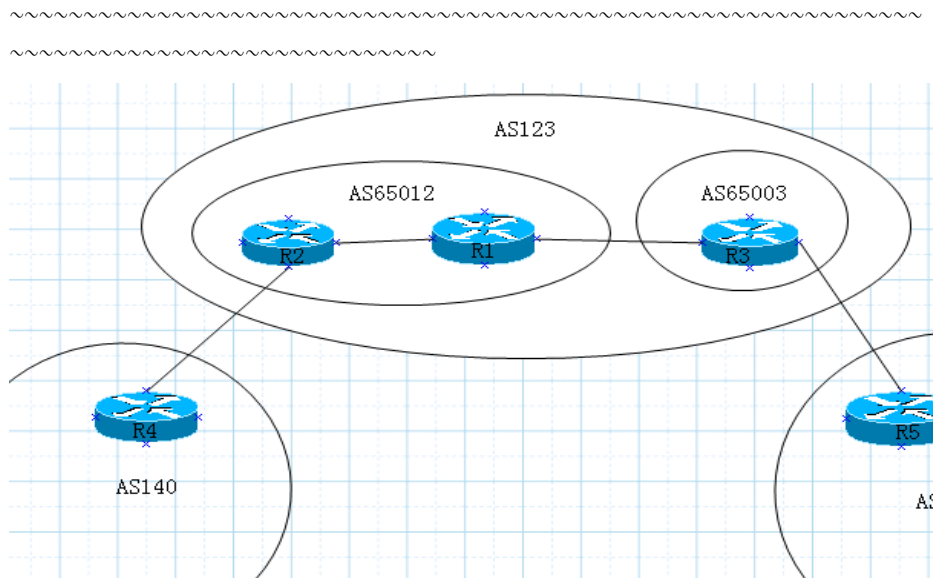
```
R1(config-router)#neighbor 2.2.2.2 route-reflector-client
R1(config-router)#neighbor 3.3.3.3 route-reflector-client
```

全网优!

step3:在R1上, 只定义R2为自己的“路由反射器的客户端”

```
R1(config-router)#neighbor 2.2.2.2 route-reflector-client
```

全网优!



LAB8:confederation/(community)

假定: 子AS65003/AS65012之间是不运行IGP的、AS65012内部运行的IGP是RIP.

STEP1:删除原来的AS123 子AS内运行RIP

```
no router bgp 123
```

```
router rip
version 2
network 1.0.0.0
```

```
network 1.0.0.0
network 12.0.0.0
no auto-summary
```

step2: 启动新的子AS:

```
R1/R2#
router bgp 65012
R3#
router bgp 65003
```

step3:

R1/R2/R3指定自己是属于AS123这个联邦  
(config-router) #bgp confederation identifier 123

step4: 在两个子AS相邻的边界路由器上, 互相指定对方的子AS号

```
R1(config-router)#bgp confederation peers 65003
R3(config-router)#bgp confederation peers 65012
```

step5: 在联邦中, 互相BGP邻居

构建R3-R5之间的EBGP:

```
R5(config-router)#neighbor 35.0.0.3 remote-as 123
R3(config-router)#neighbor 35.0.0.5 remote-as 150
```

构建R2-R4之间的EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

提醒: 在 联邦以外的EBGP邻居, 它们能查看到的是联邦的大AS号, 而不是子AS号。

R1-R3的联邦EBGP: (联邦子AS间有IGP)

R3:

```
nei 1.1.1.1 remote-as 65012
nei 1.1.1.1 upd lo 3
nei 1.1.1.1 ebgp-multihop
```

R1:

```
nei 3.3.3.3 remote-as 65012
nei 3.3.3.3 upd lo 1
nei 3.3.3.3 ebgp-multihop
```

R1-R2的联邦IBGP

R1#

```
nei 2.2.2.2 remote-as 65012
```

```
nei 2.2.2.2 upda lo 1
```

R2#

```
nei 1.1.1.1 remote-as 65012
```

```
nei 1.1.1.1 upda lo 2
```

stepx: 一下是观察BGP的路由的传递

R3上有优化路由

```
*> 105 5.0.0/16 35.0.0.5
```

但是由于R1无法通过IGP获得到达35.0.0.0这个网络的路由，所以R1到此路由的 下一跳不可达，从而无法优化。

R1

```
* 105 5.0.0/16 13.0.0.3
```

step6: 联邦EBGP和普通EBGP的异同点

下一跳:

在联邦的子AS中, 所有路由器看到的BGP下一跳, 都是相邻大AS的边缘节点, 而不是本联邦内子AS的下一跳, 这是区别与普通EBGP的.

同步:

联邦EBGP和普通EBGP一样, 无需考察同步问题.

step7: 联邦内的IBGP: (R1-R2)

R2#

```
* i105.1.0.0/16 3.3.3.3 (65003) 150 i
```

下一跳可达, 因为两个AS间运行了RIP

同步:

原因是R2从IBGP学到的路由, 默认要检查同步

但现在R2不可能通过RIP学到此BGP的路由(指这条路由:i105.1.0.0/16)

step8: 在两个AS间, 无IGP的情况

R3(config)#no router rip

R1(config)#router rip

R1(config-router)#no net 13.0.0.0

```
R1(config-router)#no net 13.0.0.0
```

手工建立到达对方环回口的静态路由;

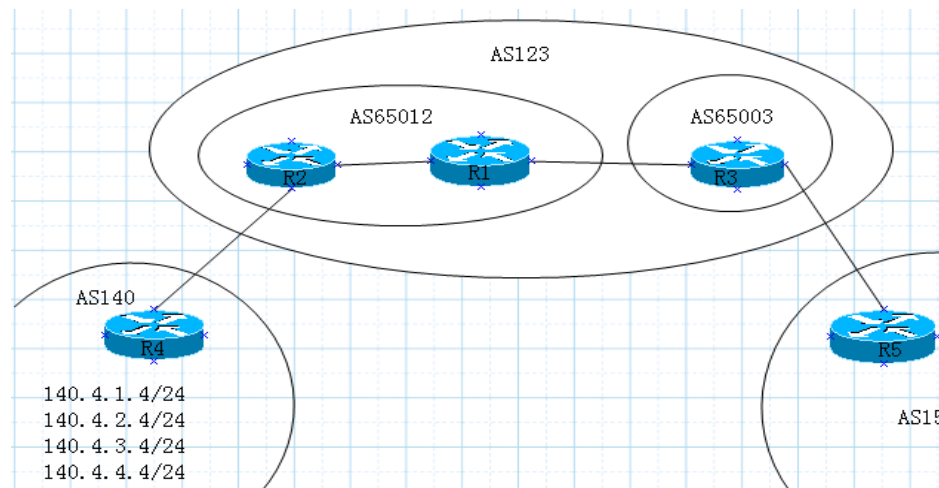
```
R1:ip route 3.3.3.0 255.255.255.0 13.0.0.3
```

```
R3:ip route 1.1.1.0 255.255.255.0 13.0.0.1
```

到此, R1-R3之间的联邦EBGP关系, 可以成功建立

但, R2无法通过IGP学到3.3.3.3的路由, 所以下一跳不可达. 需在R1指  
next-hop-self

~~~~~  
~~~~~



LAB9:community(团体)

相当于一种BGP路由的标识位, 用于标识这条BGP路由应该传播的范围。

R4#

step1:通过prefix, 定义出特定的BGP路由

R4:

```
ip prefix-list B-1 seq 5 permit 140.4.1.0/24
```

```
ip prefix-list B-2 seq 5 permit 140.4.2.0/24
```

```
ip prefix-list B-3 seq 5 permit 140.4.3.0/24
```

step2:通过route-map, 调用前缀列表设定每类路由的community种类

R4(config-route-map)#set community ?

<1-4294967295> community number

aa:nn community number in aa:nn format

additive Add to the existing community

internet Internet (well-known community)

local-AS Do not send outside local AS (well-known

community)

no-advertise Do not advertise to any peer (well-known

|              |                                                     |
|--------------|-----------------------------------------------------|
| no-advertise | Do not advertise to any peer (well-known community) |
| no-export    | Do not export to next AS (well-known community)     |
| none         | No community attribute                              |

```

route-map T-R2 permit 10
match ip add prefix B-1
set community no-advertise (R4通知R2, 不要发给任何BGP邻居)
!
route-map T-R2 permit 20
match ip add prefix B-2
set community local-as (联邦的子AS/小AS)
!
route-map T-R2 permit 30
match ip add prefix B-3
set community no-export (联邦的大AS)
!
route-map T-R2 permit 40 (match any, set nothing!)
!

```

step3:在R4上, 对R2的BGP路由策略发生, “出方向”的改变

```

R4(config)#router bgp 140
R4(config-router)#nei 24.0.0.2 route-map T-R2 out

```

step4:每个BGP路由器, 将community这些标签发送给下一个BGP路由器  
每向前走一个BGP Router, 就要“send-community”推一下。

在BGP进程中

```

R4#nei 24.0.0.2 send-community
R2#nei 1.1.1.1 send-community
R1#nei 3.3.3.3 send-community

```

```
clear ip bgp *
```

```

R2#sh ip bgp community
sh ip bgp community no-advertise
..... local-as
..... no-export
..... 40.3.0.0/16

```

step5:

如果route-map中, 没有最后的那句空的route-map“route-map T-R2 permit 40”,

R4向R2通告的bgp路由只有3条:

```
R4#sh ip bgp neigh 24.0.0.2 advertised-routes
```

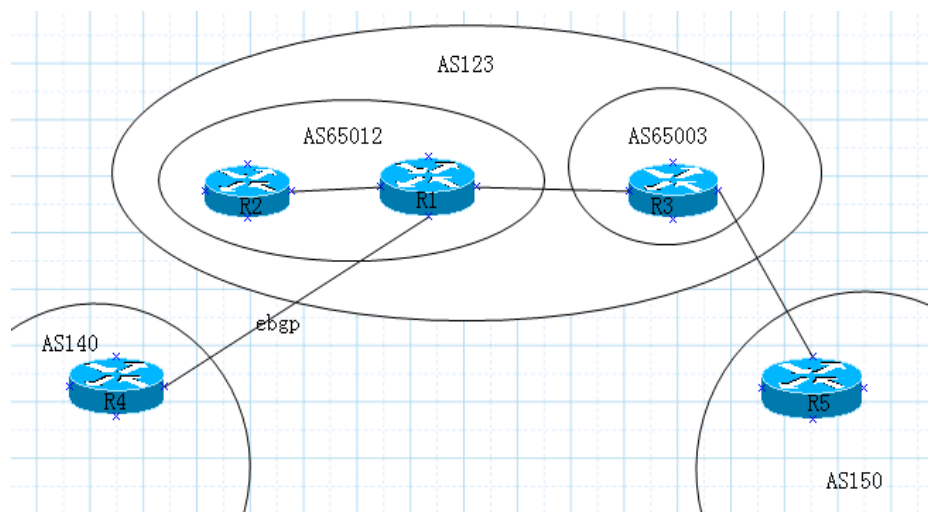
```

permit 40",
R4向R2通告的bgp路由只有3条:
R4#sh ip bgp neigh 24.0.0.2 advertised-routes
*>40.1.0.0/16
*>40.2.0.0/16
*>40.3.0.0/16

```

~~~~~  
~~~~~

## BGP-SUMMARIZATION



## LAB1:非专业汇总

step1:手工生成一条需要汇总的,静态的,空接口的路由:

```

R4指4条环回: 140.4.1.4/24
               140.4.2.4/24
               140.4.3.4/24
               140.4.0.4/24

```

R4(config)#ip route 140.4.0.0 255.255.252.0 <---指这条静态的原因是

是为了network的时候能在路由

表里发现这条指

向null 0 且22位的路由存在,才能network成功

step2:将上述路由,宣告到BGP进程里

```

R4(config)#router b 140

```

```
R4(config)#router b 140
R4(config-router)#net 140.4.0.0 ma 255.255.252.0
```

step3:删除原宣告到BGP的明细路由:  
(如果使用network命令做BGP汇总,是不需要宣告明细路由的)

```
no net 140.4.0.0 ma 255.255.255.0
no net 140.4.1.0 ma 255.255.255.0
no net 140.4.2.0 ma 255.255.255.0
no net 140.4.3.0 ma 255.255.255.0
```

删除后就只有一条22位的汇总路由!!

step5:  
R4(config)#no ip route 140.4.0.0 255.255.252.0 null 0  
R4在BGP表中都无法优化,更加不会传给别地BGP路由器了.

~~~~~  
~~~~~

LAB2:BGP的专业汇总(推荐的方法)

step1:准确地宣告每一条BGP明细路由  
net 140.4.0.0 ma 255.255.255.0  
net 140.4.1.0 ma 255.255.255.0  
net 140.4.2.0 ma 255.255.255.0  
net 140.4.3.0 ma 255.255.255.0

(不要用network宣告汇总路由)  
no net 140.4.0.0 ma 255.255.252.0

step2:使用aggregate命令,实现BGP路由的汇总(aggregate-address命令是不需要实现配置汇总路由的):  
R4(config-router-BGP)#aggregate-address 140.4.0.0  
255.255.252.0

此时,在BGP路由器上,都接收到了明细和汇总的路由.

step3:为了不让明细路由传播出去,启用summary-only参数.  
R4(config-router-BGP)#aggregate-address 140.4.0.0  
255.255.252.0 summary-only

此时的R4抑制了所有的明细路由,只发送了汇总路由给R3,实现了BGP路由的汇总

step4:R4的BGP进程,自动生成了一条用于汇总的空接口路由

## bgp-day4

BGP属性, 控制BGP的选路原则

1、在BGP路由器的BGP表中, 可能存在到达某个特定目标网络的, 都能满足“同步”和“下一跳可达”的多条路径。

2、BGP默认不执行负载均衡, 而是严格按照网管的策略/意志进行BGP选路。

网管通过BGP属性/Attribute, 去表达其策略/意志的, 实现BGP路由选择的控制。

而IGP是通过最小的Metric实现路由选择的。

网管可以通过“maximum-path(1~6)”命令, 实现BGP的负载均衡。

3、BGP是按照BGP属性自上而下, 依次提出不是最佳的路由, 直到优选出到达目标网络的最佳的那一条BGP路由。

4、BGP所提交给路由表选择的路由, 会和别的路由协议所生成的路由进行AD比较。

5、AD最小的那个路由协议所生成的路由, 将被注入/优选进路由表, 成为达到该目标网络的路由。

BGP的属性的种类:

1: Well-known attributes:

Well-known mandatory (公认, 强制的)

Well-known discretionary (公认, 自决定的)

2: Optional attributes :

Optional transitive attributes (可选, 可传递的, partial)

Optional nontransitive attributes (可选, 非传递的)

3. Well-known mandatory & transitive attributes:

as path

next-hop

origin (起源)

IGP(i) (RIP/IGRP/EIGRP/OSPF/IS-IS)

通过BGP中的network command, 宣告进BGP的。

```
R2#router bgp 12
```

```
network 120.1.0.0 mask 255.255.0.0
```

EGP(e)

```
redistributed(重分布) from EGP
```

Incomplete(?)

```
redistributed from IGP or static
```

redistributed from IGP or static

~~~~~  
~~~~~  
**Route Selection Decision Process (路由选择过程重点)**

- next Hop (下一跳)
- synchronized (同步)
- 1. highest weight (local to router)
- 2. highest local preference (global within AS)
- 3. route originated by the local router (next-hop =0.0.0.0)  
“起源” 以上值越大就匹配, 以下的是越小匹配
- 4. shortest AS path
- 5. lowest origin code (IGP < EGP < incomplete ) (只要是通过 network 命令 network 出来的都是 IGP)
- 6. lowest MED (from other AS ) **MED: 多出口区分**
- 7. EBGp path over IBGP path (EBGP AD:20/IBGP AD:200)
- 8. the path through the closest IGP neighbor
- 9. Oldest route for EBGp paths (选最早的EBGP路由)
- 10. the path with the lowest neighbor BGP router-ID

**BGP选路原则**

weight  
local-preference  
本地起源  
as-path  
起源代码  
med  
EBGP优于IBGP  
最近的IGP邻居  
最老的  
最低的router-id

~~~~~  
~~~~~  
**BGP路由优化的前提条件:**

- A: sync      B:next-hop
- 1、weight (LAB6)
- 2、L-P (LAB5)
- 3、As path (LAB4)
- 4、MED (LAB3)

5、EBGP Vs IBGP 的对比 (AD对比) (LAB2)

6、closet IBGPneighbor (LAB2)

BGP基本配置:

Step1: 确认L1/L2的连通性

Step2: 确认L3 IGP的路由的通达

Step3: 启动BGP, 用物理接口建EBGP, 用环回口建IBGP

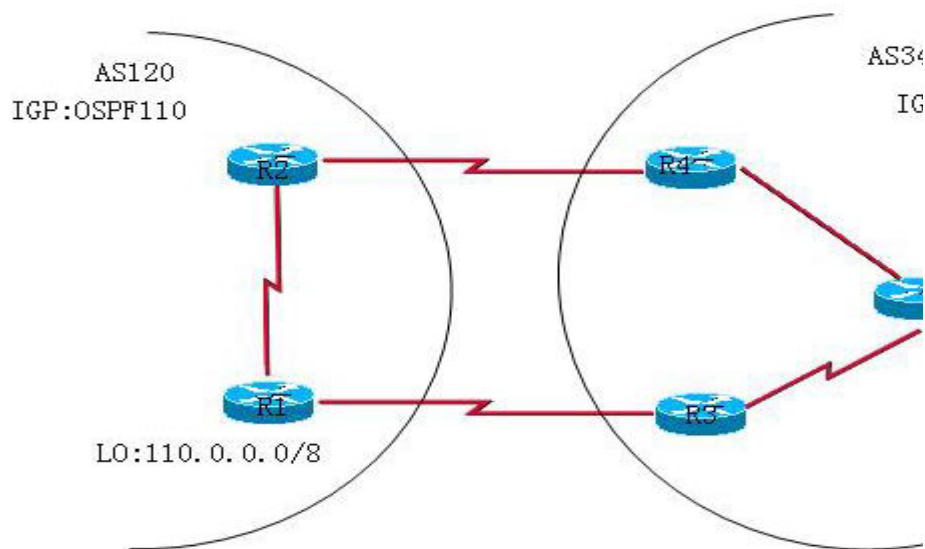
Step4: 宣告BGP路由

Step5: 对BGP路由进行控制.

(首先让R5满足同步/下一跳的基本优化条件)

(R3/R4: neighbor 5.5.5.5 next-hop-self )

(通过Shortest(最短) AS path , 选择了R3)



LAB1: Closet IBGP Neighbor

Step1: 将AS345中的IGP更改为:EIGRP

R5#show ip route

D 3.3.3.3 [90/2297856]

D 4.4.4.4 [90/409600]

R5#show ip BGP

\*>i110.0.0.0 4.4.4.4

\*i 3.3.3.3

R5#show ip route

B 101.0.0.0 [200/0] via 4.4.4.4

## LAB2:EBGP VS IBGP

在R4上观察:

Step1:如果从R2和R3的两个方向上,都收到BGP路由:

```
R4#show ip bgp
```

```
* i110.0.0.0      3.3.3.3 (来自IBGP)
```

```
*>              24.0.0.2(来自EBGP)
```

注:因为EBGP path over IBGP path (EBGP AD:20 VS. IBGP AD:200),所以优化了来自EBGP的路由.

Step2:切断R2/R4的链路:(测试没有选择的时候,当然只能选来自IBGP的路由了)

```
R4#
```

```
*>i110.0.0.0      3.3.3.3
```

## LAB3:MED(multi-exit discriminator)(also called the metric)

(整个AS345都从下面链路走)

特征:

1. MED的取值越小越好
2. 只发送给EBGP邻居,用于建议对方,如何离开对方的AS,来访问本AS的网络.
3. MED是一种**可选的,非传递**的属性.
4. MED的默认值为0
  - 关于“传递”的属性:
    1. 无论“可传递”还是“非可传递”,如果设备支持这种属性,那么都会传递.

如果设备不支持:

2-1:对于“可传递”,那么,,会被标识为“partial”,来进行继续传递.

2-2:对于“非可传递”,那么这种属性被丢弃,但BGP这条路由条目,还是正常传递.

Step1:通过前缀列表定义需要控制的BGP路由:

```
R1/R2(config)#ip prefix-list B-110 permit 110.0.0.0/8
```

Step2:在Route-map中,设定路由的MED值:

```
R1(config)#route-map T-AS345
```

```
R1(config-router-map)#match ip address prefix-list B-110
```

```
R1(config-router-map)#set metric 50
```

```
R1(config-router-map)#match ip address prefix-list B-110
R1(config-router-map)#set metric 50
```

```
R2(config)#route-map T-AS345
R2(config-route-map)#match ip address prefix-list B-110
R2(config-route-map)#set metric 100
```

Step3:将不同的route-map发送给不同的BGP邻居

```
R1(config)#router b 120
R1(config-router)#neighbor 13.0.0.3 route-map T-AS345 out
R1(config)#router b 120
R2(config-router)#neighbor 24.0.0.4 route-map T-AS345 out
```

```
clear ip bgp * soft out
```

Step4:

```
R4#show ip bgp
```

| network        | metric | next hop |     |
|----------------|--------|----------|-----|
| * 110.0.0.0/18 |        | 24.0.0.2 | 100 |
| *> i           |        | 3.3.3.3  | 50  |

```
R5#
```

```
*>i110.0.0.0      3.3.3.3      50
```

```
R3#
```

```
*>i110.0.0.0      13.0.0.1      50
```

```
~~~~~
~~~~~
```

#### LAB4:AS path

(整个AS345都从上面链路走)

实验需求:

整个AS345的所有路由器, 都从R2走

在R1与R3之间, 虚拟一个AS100

Step1:通过Prefix-list, 定义BGP路由:

```
R3(config)#ip prefix-list B-110 permit 110.0.0.0/8
```

Step2:在Route-map中, 为此路由添加一个虚拟的AS100

```
route-map T-AS345 permit 10
match ip address prefix-list B-110
set metric 150
```

```
match ip address prefix-list B-110
set metric 150
set as-path prepend 100
```

step3:在R3的BGP进程中,对来自R1的BGP路由,调用route-map T-AS345,发给R3

```
R3(config-router)#neighbor 13.0.0.1 route-map V-AS100 in
```

Step4:看效果

R3#

| Network         | Next-hop | metric | LocPrf | Weight |
|-----------------|----------|--------|--------|--------|
| Path            |          |        |        |        |
| *>i110.0.0.0/18 | 4.4.4.4  | 100    | 100    | 0      |
| 120 i           |          |        |        |        |
| *               | 13.0.0.1 | 50     |        | 0      |
| 100 120 i       |          |        |        |        |

R4#

|                 |          |     |  |   |
|-----------------|----------|-----|--|---|
| *> 110.0.0.0/18 | 24.0.0.2 | 100 |  | 0 |
| 120 i           |          |     |  |   |

R5#

|                 |         |     |     |   |
|-----------------|---------|-----|-----|---|
| *> 110.0.0.0/18 | 4.4.4.4 | 100 | 100 | 0 |
| 120 i           |         |     |     |   |

~~~~~  
~~~~~  
~~~~~  
~~~~~

#### LAB5:通过LOCAL-PREFERENCE, 操纵BGP路由选路

(整个AS345都从下面链路走)

特征:

1. LP的取值越大越好, 默认值是:100
2. LP只发给本AS中的IBGP邻居, 用于影响他们如何离开本AS, 去访问外AS.
3. LP的属性是公认的, 自决的, 只会在本AS传递的.

Step1:在本AS的边缘路由器上, 定义路由:

在R3/R4上:

```
R3/R4(config)#ip prefix-list B-110 permit 110.1.0.0/16
```

Step2:

R3#

```
route-map LP-3 permit 10
```

```

route-map LP-3 permit 10
  match ip address prefix-list B-110
  set as-path prepend 100
  set local-preference 130

```

Step3:在R3的BGP进程中,对来自R1对路由,调用route-map LP-3:(同LAB4):

```
R3(config-router)#neighbor 13.0.0.1 route-map LP-3 in
```

Step4:观察:

```

R3#
      Network      next Hop      Metric      LocPrf      Weight
Path

```

```
*>110.0.0.0
```

```
R4#
```

```
*>i110.0.0.0
```

```
R5#
```

```
*>i110.0.0.0
```

```

~~~~~
~~~~~

```

### LAB6:Weight

Weight的特征:

- 1:cisco私有的属性.
- 2:默认值为0,越大越好.
- 3:不发送给任何BGP邻居,只影响本路由器的选路。(我的注:这样即使对端是huawei的也没关系)

方法1:

```
R4:(config-router)#neighbor 24.0.0.2 weight 10
```

看效果:

```

R4#
      Network      Next-hop      metric      LocPrf      Weight
Path
* i110.0.0.0/18    3.3.3.3       50          130         0
100 120 i
* >                24.0.0.2      100         10

```

```
10      120 i
R3#
      Network          Next-hop      metric  LocPrf    Weight
Path
* i110.0.0.0/18        4.4.4.4          100     100        0
120 i
* >                    13.0.0.1          50     130        0
100 120 i
R5#
      Network          Next-hop      metric  LocPrf    Weight
Path
* i110.0.0.0/18        (未记录)          100     100        0
120 i
* >                    13.0.0.1          50     130        0
100 120 i
```

方法2:

3-1:定义路由

3-2:

```
route-map WEI permit 10
```

```
match ip address prefix-list B-110
```

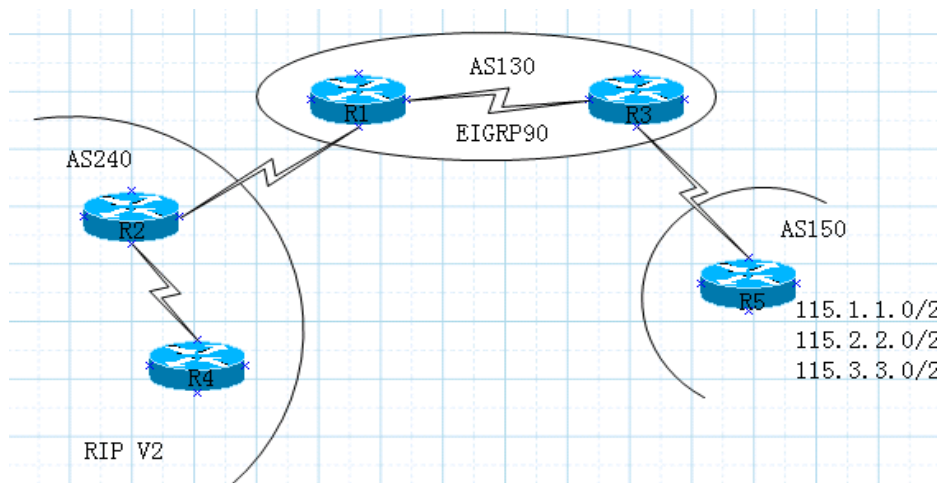
```
set weight 10
```

3-3:

```
R4#(config-router)#neighbor 24.0.0.2 route-map WEI in
```

## bgp-day5

BGP中路由的控制/过滤



### LAB1:distribute-list(+号表示调用)ACL(较落后)

step1:通过ACL定义BGP路由

(deny:不允许)

(permit:允许)

```
R5(config)#access-list 15 permit 115.2.2.0
```

```
R5(config)#access-list 15 permit 115.1.1.0
```

ACL的最后,有隐藏的deny any

step2:通过distribute-list/分布列表

```
R5(config)#router bgp 150
```

```
R5(config-router)#nei 35.0.0.3 distribute-list 15 out
```

软清除:

```
R5#clear ip b * so ou (使用在:出方向的策略发生改变时)
```

step3:观察R5对R3发送了哪些BGP路由

```
R5#sh ip b nei 35.0.0.3 advertised-routes
```

```
~~~~~
~~~~~
```

### LAB2:通过neighbor命令,直接调用prefix-list(推荐,功能简单)

step1:通过前缀列表/prefix-list.,定义需要控制的BGP路由:

(deny:不允许)

(permit:允许)

```
R5(config)#ip prefix-list T-AS130 permit 115.2.2.0/24
```

```
R5(config)#ip prefix-list T-AS130 permit 115.1.1.0/24
```

ip prefix-list的最后,有隐藏的deny any

step2:在R5的出方向,调用prefix-list T-AS130,进行路由过滤  
R5(config-router)#nei 35.0.0.3 prefix-list T-AS130 out

~~~~~  
~~~~~

### LAB3:neighbor+route-map+ACL

step1:通过ACL定义BGP路由

R5(config)#access-list 1 permit 115.2.2.0 (permit:匹配)  
R5(config)#access-list 2 permit 115.1.1.0

step2:通过route-map,调用ACL.让route-map决定是否允许这些定义好的路由穿过本AS:

route-map T-AS130 permit 10 (permit:允许)  
match ip address 1  
!  
route-map T-AS130 deny 20 (deny:不允许)  
match ip address 2  
!

route-map T-AS130 permit 30

(是否有这句空的route-map,决定了没有明确定义的路由能否通过,有表示可以通过;没有这条表示不可通过。)

step3:在BGP进程里,对AS130里的用户,调用路由控制策略

R5(config-router)#nei 35.0.0.3 route-map T-AS130 out

~~~~~  
~~~~~

### LAB3:neighbor+route-map+prefix-list (推荐 功能强大)

step1:通过前缀列表/prefix-list.,定义需要控制的BGP路由:(permit:匹配)

R5(config)#ip prefix-list R-1 permit 115.1.1.0/24  
R5(config)#ip prefix-list R-2 permit 115.2.2.0/24

step2:通过route-map,决定是否允许特定路由穿越:

route-map T-AS130 permit 10  
match ip address prefix-list R-1

```
!
route-map T-AS130 deny 20
match ip address prefix-list R-2
!
route-map T-AS130 permit 30
!
```

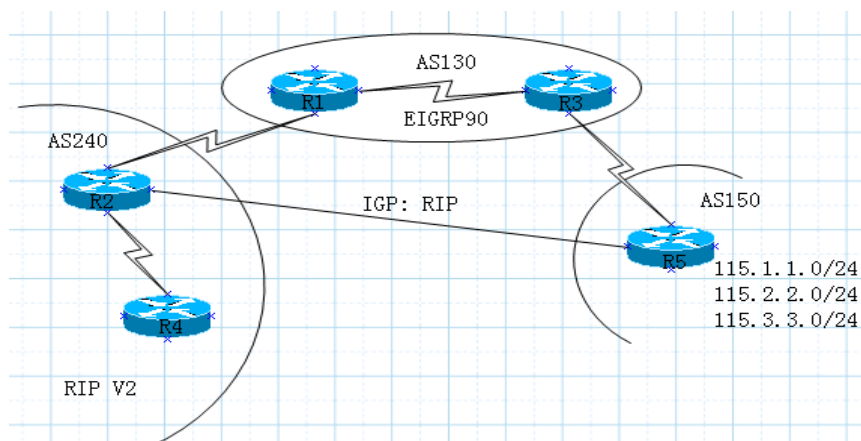
step3:R5对R3的BGP路由出口策略发生改变:

```
R5(config-router)#nei 35.0.0.3 route-map T-AS130 out
```

对于上述4个实验,如果在R5上做in方向的路由过滤,则需要硬清除。

```
clear ip bgp * (会reset TCP连接)
```

~~~~~  
~~~~~



#### LAB5:BGP的后门/backdoor

(用于EBGP与IGP之间的AD竞选时,人为的让IGP优先进入路由表的一种操作.)

20 120

R2做后门之前的路由

```
R 115.2.2.0 [120/1] via 25.0.0.5
B 115.3.3.0 [20/1] via 12.0.0.1
B 115.1.1.0 [20/1] via 12.0.0.1
```

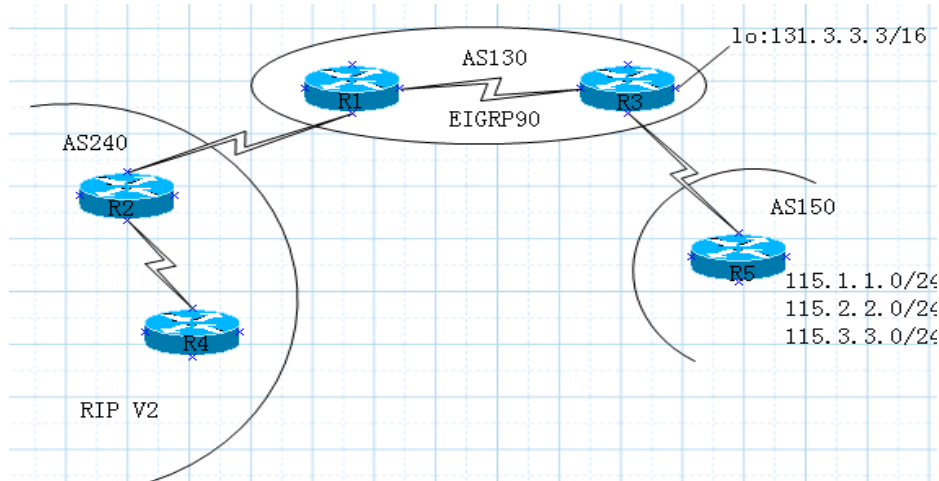
做后门:

```
R2(config)#router b 240
R2(config-router)#nei 115.2.2.0 ma 255.255.255.0 backdoor
R2(config-router)#nei 115.3.3.0 ma 255.255.255.0 backdoor
```

做完backdoor后:

```
R 115.2.2.0 [120/1] via 25.0.0.5
R 115.3.3.0 [120/1] via 25.0.0.5
B 115.1.1.0 [20/1] via 12.0.0.1
```

~~~~~  
~~~~~



LAB6:通过AS-Path access-list控制穿越特定AS的BGP路由

6-1:

开始标识符 ; \$ 结束标识符 \_ 表示任意字符

step1:只接收起源自AS130的BGP路由:

```
R4(config)#ip as-path access-list 130 permit _130$ (起源自AS130)
```

step2:通过filter-list,调用as-path access-list 130

```
R2(config)#router b 240
```

```
R2(config-router)#nei 12.0.0.1 filter-list 130 in
```

6-2:

```
ip as-path access-list 150 permit _150$ (起源自AS150)
```

step2:同上

6-3:只接收与本AS直接相连的那个AS130所发来的BGP路由:

```
ip as-path access-list 140 permit ^130$ (与本AS直接相连的是AS130)
```

6-4:只接收穿越AS130的BGP路由:

```
ip as-path access-list 130 permit _130_
```

(与access-list, prefix-list 一样, as-path-list在列表的最后都有隐含的deny any)

step2:同上

6-5:接收any路由:

```
ip as-path access-list 130 permit .*
```

step2:同上

6-6:在一个as-path-list中的多个匹配条件: (允许AS130和AS150)

```
ip as-path access-list 130 permit _130$
```

```
ip as-path access-list 130 permit ^150$
```

step2:同上

~~~~~  
~~~~~

#### LAB7:BGP Damping

15分钟 half-life 半衰期

750 reusing 重用门限

2000 maximum duration to suppress 最大门限

60分钟 最大抑制时间 (半衰期的四倍)

step1:通过前缀列表, 定义需要进行damp的BGP路由:

```
R3(config)#ip prefix-list R-115 permit 115.1.1.0/24
```

step2:创建route-map, 对特定路由进行damping, 并且可以设定damping的参数

```
route-map DAMP permit 10
```

```
match ip rrefix-list R-123
```

```
set dampening 15 750 2000 60
```

step3:在BGP进程中调用route-map, 进行BGP dampening

```
router bgp 110
```

```
bgp dampening route-map DAMP
```

```
R3#show ip bgp dampening flap-statistics
```

```
show ip bgp dampening dampended-paths
```

R3#sh ip bgp flap-statistics 查看BGP曾经抖动的路由

R3#sh ip bgp dampened-paths 查看BGP抑制的路由

## Feture qos

链路上有效的**带宽**就是链路所有的路径节点中，最少的带宽值。  
..... **延时**.....，所有延时之和。

网络出现拥塞的情况：

Lack of bandwidth（带宽不足）  
Too much Delay（延时）  
Variable Delay jitter（抖动）  
Dorp（丢包）

解决方法：

1：解决Lack of bandwidth方法：

- 1-1向ISP购买更多的链路带宽
- 1-2压缩不那么重要的数据流量，优先保证重要的数据流量通过。
- 1-3压缩2层的数据frame的净荷，压缩L3/L4的报头。

2：降低延时的方法：

同上

3:Prevent Packet loss的方法

3-1:同上

3-2:优先保证，“特别敏感的”数据包的带宽(资源预留)

3-3:在链路/接口临近拥塞出现之前，先随即丢一小部分不重要的数据. 来防止拥塞的出现.

QoS 实行的4大类方法

Best effort(no QoS,default behavior)

Integrated Services model

Integrated Services model  
Differentiated Services model  
MPLS (Multi-Protocol Labeled Switching)

Integrated Services model (RFC1633) IntServ.  
Resource reservation  
Admission control

Resource Reservation Protocol (RSVP) (RFCs 2205 to 2215).  
Common Open Policy Service (COPS) (RFCs 2748 to 2753).

3: Different Services Model

## Traffic Terminology

- **Flow**: a single instance of an application or application flow of packets which is identified by source address, source port, destination address, destination port and protocol id.
- **Traffic stream**: an administratively significant collection of one or more flows which traverse a particular path. A traffic stream may consist of a set of flows which are selected by a particular class of service.
- **Traffic profile**: a description of the temporal properties of a traffic stream such as average rate, peak rate and burst size.

### Blocks of IP QoS Mechanisms

QoS的工作流程:

#### Step1:Classifier

区分不同的数据包

access lists, route maps, class maps,

#### Step2:Meter/测量(速率)

Token bucket/令牌桶

rate limiting, shaping, scheduling,

#### Step3:Marker/标记

CAR, class-based marking,

通常为特定的数据包, 打上以下的一种标签:

IP Precedence

DSCP

QoS group

MPLS experimental bits

Frame Relay DE bit

ATM CLP bit

IEEE 802.1Q or ISL CoS

Step4:Conditioner/调节器-----主要分为2种技术:

##### 1 Dropping mechanisms

- **Committed Access Rate (CAR)** and **Class-based Policing** can drop packets that exceed the contractual rate
- **Weighted Random Early Detection (WRED)** can randomly drop packets when an interface is nearing congestion

##### 2 Shaping mechanisms:

- **Generic Traffic Shaping (GTS)**
- **Frame Relay Traffic Shaping (FRTS)**
- **Class-based Shaping**
- **Hardware shaping on ATM VC**

Step5:Forwarding/转发器

1:Process Forwarding/进程转发

2:Fast Forwarding/快速转发

3:Cisco Express Forwarding(CEF)

Step6:软件队列:

队列种类:

FIFO, priority queuing (PQ), custom queuing (CQ)  
WFQ, DWFQ, CoS-based DWFQ, QoS-group DWFQ  
Class-based WFQ, Class-based LLQ

#### Step7: 能否加入软件队列?

有多技术可以控制能否进对:

- 1: Tail drop (最常用的) (伪丢弃: 满了就排不进去)
- 2: WFQ has an improved tail-drop scheme.
- 3: WRED randomly drops packets when nearing congestion.  
(在链路/接口临近拥塞出现之前, 先随机丢弃一小部分不重要的数据包, 来防止拥塞出现)

#### Step8: Scheduling/调度器

调度器负责按照一定的算法, 从多个软件队列中, 调入数据包, 送进硬件队列.

#### Step9: 硬件队列:

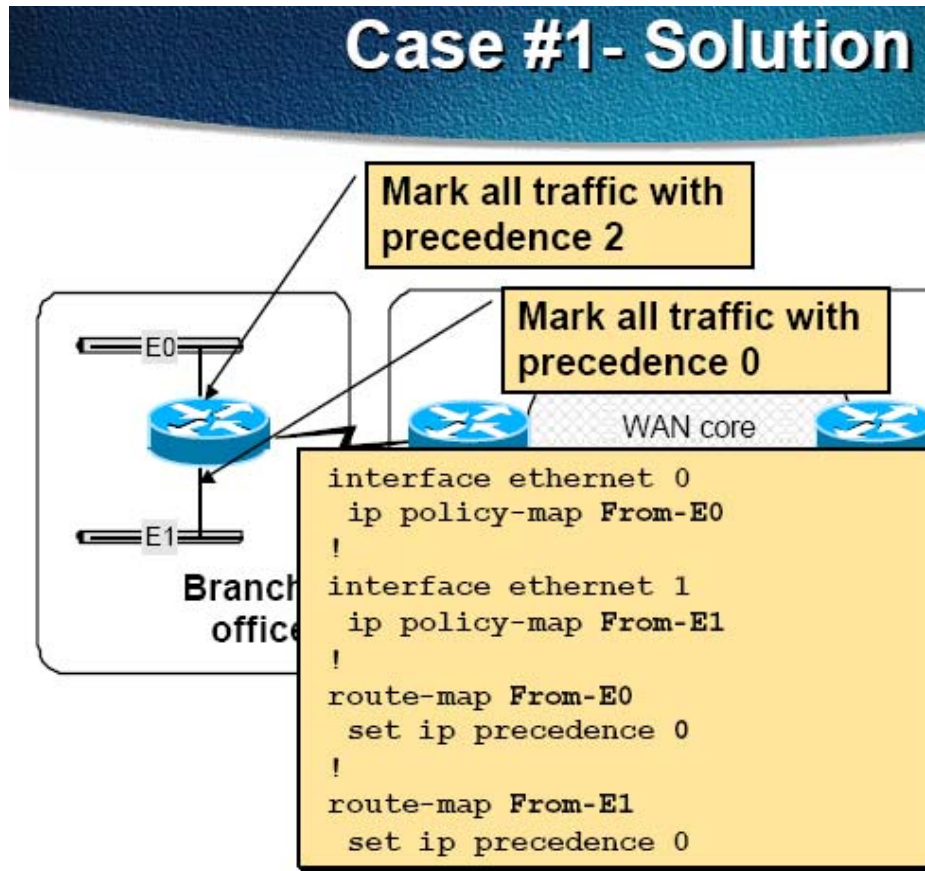
在硬件队列中, 按照“先进先出/FIFO”原则, 将数据送入接口, 进行转发.

Classification and Marking  
Policy-based routing (PBR)  
QoS Policy Propagation on BGP (QPPB)

## Traffic Classification and Marking

- This module describes the two mechanisms that are used for classification and marking only:
  - Policy-based Routing (PBR)
  - QoS Policy Propagation through BGP (QPPB)
- Other classification and/or marking mechanisms are described in other QoS modules

PBR: (只在入口生效)  
classification---分类器  
1. 大多数的QoS机制都包括了分类器  
2. 自动分类的有WFQ  
3. 手工分类的有PQ, CQ, CBWQ



LAB1

Step1:

```
route-map From-E0 permit 10
 set ip precedence routing
```

LAB2

~~~~~

Step1:通过ACL, 进行数据分类: (Classification) 表示名字
ip access-list standard From-BOSS
 permit 192.168.1.1
!
ip access-list extended Telnet
 permit tcp any any eq telnet(23)

Step2:通过Route-map, 进行数据标记(Marking)
route-map NET-CTL permit 10
 match ip address From-BOSS
 set ip precedence priority/1 (值越高越优先)
!
route-map NET-CTL, permit 20
 match ip address Telnet
 set ip precedence immediate/2
!
route-map NET-CTL, permit 30
 match ip address Telnet
 set ip precedence routine/0

Step3:在入口中调用Route-map
interface Ethernet0
 ip policy route-map NET-CTL

R2(config-route-map)#match length 1000 1500(最小1000 最大1500)
(配置L3的包长度)

~~~~~  
~~~~~

Queuing Mechanisms/队列机制(只对出口生效)

Hardware Queue/Transmit queue (TxQ)

FIFO Queue

跳过软件队列的前提条件:

硬件队列都没有满/Full, 没有发生拥塞.

Software Queue:

软件队列的3大组件:

classification/分类

insertion policy/入队

service policy/scheduling/调度机制

FIFO Queuing:

~~~~~  
~~~~~

硬件队列默认就是FIFO, 默认在队列中存储40Packets, 不建议更改.

在接口宽带高于2Mbps的接口上, 默认是FIFO.
在接口宽带低于或等于2Mbps的接口上, 默认是WFQ.

如果需要低于或等于2Mbps的接口上, 启动FIFO,
使用命令:

```
interface serial1
 no fair-queue (启动FIFO)
```

检测:

```
R2#Show int s1
Serial1 up , line protocol is up
    MTU 1500 bytes, BW 1544 bit, DLY 20000 usec,
    Queueing strategy: weighted fair
    Queueing strategy: fifo
R2#show queue serial 1
```

Priority Queuing/优先级队列/PQ

必须先传输完**高**优先级的队列的数据, 才能传输**低**优先级的队列.
可能导致低优先级的队列的数据的“饿死”

4个PQ队列的队列长度的比例:

```
R2(config)#priority-list 2 queue-limit 20 40 60 80
                                高 中 正常 低
```

LAB 3 Priority Queuing

~~~~~  
~~~~~

Step1:通过ACL区分不同数据:

```
access-list 20 permit 3.0.0.0 0.255.255.255
!
access-list 110 permit tcp any any eq telnet
(不支持命名ACL)
```

Step2:根据不同数据, PQ(PQ的编号的取值1~16)

```
priority-list 2 protocol ip high list 110
priority-list 2 interface serial1 medium
priority-list 2 protocol ip normal list 120
priority-list 2 default low
```

Step3:在数据**出口**调用PQ-2

```
interface serial0
 priority-group 2
```

检测:

```
R2#show queueing priority
```

Cusom Queuing:/CQ/用户队列:

最多支持16个队列

Step1:

```
access-list 110 permit tcp any any eq telnet
access-list 150 permit ip any any precedence flash-override/4
access-list 150 permit ip any any dscp af41
access-list 150 permit ip any any tos min-delay/8
```

Step2:

```
queue-list 5 protocol ip 1(几号队)list 150
```

```
queue-list 5 queue 1 byte-count 800 limit 30
(一个队列, 一次循环中, 最少能传的字节. 默认:1500)
                                Bytes      Packets
```

queue-list 5 lowest-custom 2 -----设队列2为最低的. 指定上面的队列1是优先级队列

```
queue-list 5 protocol ip 2 list 110(Telnet)
```

```
queue-list 5 interface Ethernet 0 3
```

```
queue-list 5 default 4(FTP)
```

Step3:在接口中调用CQ5:

```
interface serial 0
 custom-queue-list 5
```

组播

Multicast

1. 组播概念简介
2. 概念回顾
3. 组

组播的好处

- 游戏节约了带宽
- 减少了服务器的负荷
- 减少了网络的负荷

组播的不理因素

组播是基于UDP协议

最短努力传送：丢包的必然性

没有拥塞避免：缺少TCP的慢启动机制，导致了拥塞的出现。

副本的出现(信息包的负责)：一些多播处理技术导致网络中的大量副本的出现。

传送持续丢失：由于网络拓扑的改变会导致数据包在传送过程丢失。

失.

组播的不利因素

由于是基于UDP的协议, 导致以下两点

1. 可能性
2. 安全性

组播的应用

1. 多媒体会议 (mbone)
2. 数据分布 (允许采用PUSH的方式进行文件和数据库的更新)
3. 实时数据组播 (例如股票信息的实时更新)
4. 游戏和仿真

组播的类型

- One-to-many: 单一组播源连接多个客户端
Many-to-many: 源:客户端
Other: 其他类型, 比如说多个对一个.

Multicast Application

组播地址

IP group address

- 是d类地址, 前四位设置为1110
- 地址范围: 224. 0. 0. 0 (11100000. 00000000. 00000000. 00000000) 到 239. 255. 255. 255 (11101111. 11111111. 11111111. 11111111)

组播地址的分类:

- 已被IANA分配的组播地址: 224. 0. 0. 0 到 224. 0. 0. 255
224. 0. 0. 1---all multicast systems on subnet (所有的组播都监听这个地址)
224. 0. 0. 2---all routers on subnet (这个子网的所有路由器)
224. 0. 0. 4---all DVMRP routers ()
224. 0. 0. 13---all PIMv2 routers
224. 0. 0. 5, 6, 9, 10 used by unicast routing protocols
224. 0. 1. 39---RP 宣告
224. 0. 1. 40---RP 发现

RPF的检查决定RPF接口

检测是不是正确的接口进来, 如果正确则接受并发送. 不正确则丢弃

- RPF接口接近于源
- RPF接口取决于单播的或者专有的路由协议 (DVMRP, MBGP等)
- 周期性的RPF检查: cisco路由表是默认5秒.

组播的范围-TTL

1. 在组播的范围内使用TTL来定义需要传输的范围;

2. 所有路由器必须确保只有在信息包的TTL值大于或等于接口的TTL值时才允许将该信息包在接口上转发出去。

组播的分发树以及协议的类型

1. 有源树:基于源的树. 也叫最短路径树:只会生成一条到目的最短的路径(SPT) (在实际工程中用得比较少)
2. 共享树:指定的RP
3. Dense mode Protocols
4. Sparse Mode Protocols

SPT最短路径树(基于源的树, 即有源树)

对于接收者来说都是最近的.. 对单用户而言

共享树

组播分发树的旁认
(S, G)
-source

Dense mode Protocols

采用“PUSH”的模式发送组播信息

- 存在组播的“flooded”()与“Prune”(修剪)
- 最初的信息flooded给所有的存在于分发树的接收方
- 然后通过接收方的单方面的修建. 决定要与不要

Sparse Mode Protocols

采用“pull”模式发送组播信息
是一种“join”的模式

PIM-Dense Mode (PIM-DM)

与协议无关-支持所有低层的路由协议:
使用flood和prune机制(会定期flood)
适合于小的网络

PIM-dense mode (pim-dm) 扩散/修减, 每三分钟扩散修减一次

与协议无关-支持所有底层的路由协议:
同时支持static, rip.....
使用flood和prune机制
适合于小的网络(带宽大的)
是建立在基于源的树

PIM-DM的评价

优势:
配置简单

机制简单, flood和prune

潜在的问题:

flood与prune的效率比较低

声明的机制比较复杂

PIM Sparse Mode

与协议无关-支持所有底层的路由协议

同时支持有源树以及共享树

基于“pull”的工作方式

使用(RP)

PIM-SM 评价

优势

- 组播信息仅传送给加入的分支机构
- 动态选择树的结构
- 与底层的路由协议无关

只要用户知道了源在那里, 就不会在去往RP

IGMP

IGMPV1

成员加入: 加入成员发送224. 0. 0. 1, 并且立即被加入到组中

普通查询: 周期性的向224. 0. 0. 1发送查询包来探测成员

其中一个成员发送报告(不同的组, 不同的网段)

其他的成员抑制报告的发出.

路由器周期查询

其中的一个成员静静的离开

路由器仍旧发送周期查询

但是并没有离开的报告发送给路由器

知道查询超时.

IGMPv2

加入和版本一一样

最初所有的路由器发出查询信息

只有具有最低IP地址的路由器被选据成为“查询者”

其他路由器成为非查询者

组的维持

查询者发送周期查询

其中的一个成员发送查询报告

查询者发出细节查询

剩余的发出响应

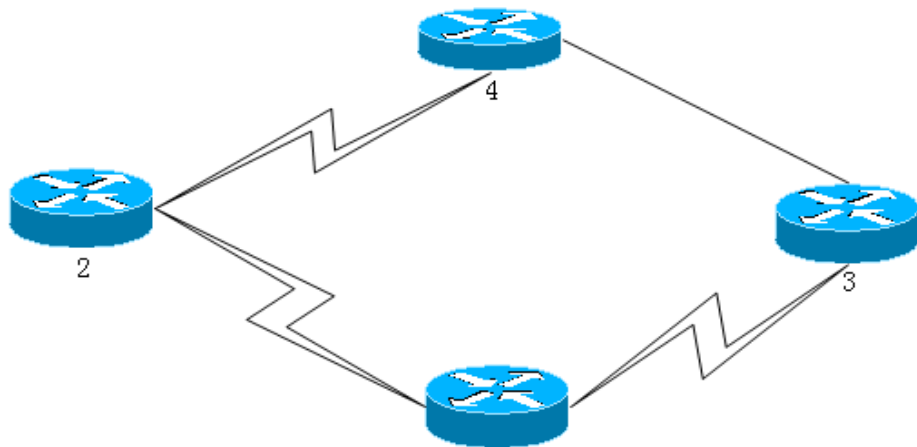
其中一个成员离开会发出离开信息, 发给224. 0. 0. 2, 所有路由器都会看到

仍旧存在的成员付出查询报告

仍旧存在的成员付出查询报告
单最后一个用户离开的时候, 也发出离开报告
查询者仍旧发出细节查询
此时每有相应
等待组的超时

版本一与版本二的比较
版本一无离开消息, 版本二有

LAB1:密集模式:(使用EIGRP)



用push模式:
流量在整个网络泛洪, 在不需要的地方进行修剪.
周期泛洪和修剪(3分钟)

Step1:R1/2/3/4/5允许EIGRP

Step2:每个需要运行组播/MC的路由器上
config#ip unlticast-routing

Step3:每个要进行组播的接口中配置, 接口的组播运行模式:
(if)#ip pim dense-mode

检查:
show ip pim interface(观察PIM的接口)
show ip pim neighbor(观看PIM的邻居)

Step4:

R5#
(int lo100) ip add 100.0.0.5 255.255.255.0

```
ip pim dense-mode
ip igmp join-group 225.0.0.5 (模拟R5加入了225.0.0.5
组)
```

Step5:

测试:

R2:ping 225.0.0.5

Reply to request 0 from 35.0.0.5 40ms

检查:

R3:show ip mroute

(*, 224.0.1.40), 00:10:11/00:00:00, rp 0.0.0.0, flags:DJCL

~~~~~  
~~~~~

稀疏模式:

用pull模式, 流量仅发到需要的地方

(explicit join behavior/显示加入机制)

基于共享分发树(从源到RP是SPT, 从RP到接收方是共享树)

LAB2:稀疏模式, 静态RP:

Step1:完成L3的单播的

Step2:每个需要运行组播/MC的路由器上, 启动组播路由:(同LAB1)

Step3:在所有运行稀疏模式的接口中, 都指定其运行模式为:

R4(config-if)

ip pim sparse-dense-mode

检查:

show ip pim

show ip pim rp

LAB3

稀疏模式, 自动RP, CISCO私有

~~~~~  
~~~~~

路由器自学到RP地址:

只需要配“候选RP”和“映射代理”, 利用组播来分发信息:

两个特定的IANA组播地址:

cisco-announce:224.0.1.39

cisco-discovery:224.0.1.40

使用dense模式转发这些组.

Step1:完成L3的单播路由吸引:(同LAB1)

Step2:每个需要运行组播/MC的路由器上,启动组播路由:(同LAB1)

Step3:

在参与组播的接口下,指定接口的组播运行模式:

Step4:删除LAB2中,所有路由器的手工指定的RP.

```
R1(config)#no ip pim rp-address 3.3.3.3
```

Step5:使用R1/R4成为RP的候选者:(每60秒发送一次RP-Announce的信息)

```
R1(config)ip pim send-rp-announce loopback1 scope 10
```

```
R4(config)ip pim send-rp-announce loopback 4 scope 10
```

Step6:R3成为RP的映射代理,负责寻找选择RP.

```
R3(in lo 3)#ip pim sparse-dense-mode
```

```
R3(config)ip pim send-rp-discovery loopback 3 scope 10
```

检查:

```
debug ip pim auto-rp(观察auto-rp的自动选择过程)
```

```
show ip PIM rp
```

```
show ip pim rp mapping(R4 down之前的)
```

LAB4:稀疏模式:

BSR(bootstrap Router) PIM v2 RFC 2117(开发性标准)

~~~~~  
~~~~~

BSR是一种给PIM稀疏 是的,

让PIM路由器自动学习“group到RP”映射需求的机制.

作用与Auto-RP类似.

Bootstrap:224.0.0.13(all PIM Router)

Step1:完成L3的单播路由协议:(同LAB1)

Step2:每个需要运行组播/MC的路由器上,

Step4:删除LAB3中的Auto-RP的操作

Step5:在准备成为RP的路由器上:

```
R1(config)#ip pim rp-candidate loopback1
```

```
R1(config)#ip pim rp-candidate loopback1  
R4(config)#ip pim rp-candidate loopback4
```

step6:在准备成为BSR的路由器上:

```
R3(config)#ip pim bsr-candidate loopback 3
```

检查:

```
R5#show ip pim bsr-router(察看当前的BSR路由器)
```

```
show ip pim rp mapping
```

```
R5#show ip pim rp(察看当前的RP路由器)
```

```
Group:228.0.0.
```

PIM ver 1 是cisco私有的, ver2是ietf的开发性标准.

PIM ver 2 才支持BSR.

注意:稀疏或者密集, 是组的属性, 而不是接口的属性.

IGP

IGP概括

	类别(Class)	更新方式 (Update)
RIPv1	有类 (classful)	定期广播(255.255.255.255) 可配置成单播; 带触发更新
RIPv2	无类 (classless)	定期组播(224.0.0.5) 可配置成单播; 带触发更新
IGRP	有类 (classful)	同 RIPv1
EIGRP	无类 (classless)	非周期的、部分有边界 靠组播”(224.0.0.5)
OSPF	无类 (classless)	组播, DRouters(224.0.0.5) AllDRouters(224.0.0.6)
IS-IS	无类 (classless)	定期组播 触发更新
	度量 (Metric)	管理距离 (Distance)
RIPv1	Hop count (1~16)	120
RIPv2	Hop count (1~16)	120
IGRP	$BW_{min} + DLY_{sum}$ Default=100 hop Max=255 hop	100
EIGRP	$IGRP_{metric} \times 256$	90 (内部) 170(外部) 5
OSPF	$10^8 / BW(1 \sim 65535)$	110
IS-IS	Default=10 (0~63) Max=1024	115
	进程 (Process)	域 (Domain)
RIPv1	仅单个	无
RIPv2	仅单个	无
IGRP	可多个	进程域 (Process Domain)
EIGRP	可多个 (1~65536)	同上
OSPF	可多个	同上
IS-IS	Max=3	路由选择域(Routing Domain)
	汇总 (Summary)	可变子网掩码 (VLSM)
RIPv1	不可关闭自动汇总 无手工汇总	√
RIPv2	可以关闭自动汇总 只能在网络边界手工汇总	√
IGRP	不可关闭自动汇总 无手工汇总	×
EIGRP	可关闭自动汇总 可在任何地方手工汇总	√
OSPF	无自动汇总 可在任何地方手工汇总	√
IS-IS	无自动汇总 可在任何地方手工汇总	√

OSPF-day1

OSPF基本理论

OSPF 三张表

1. neighbor table:列出了所有和本路由器直接相连的OSPF邻居
2. topology table (LSDB链路状态数据库)
列举了所有从自己的邻居那得到得LSA, 在同一个ospf区域中的路由器, 都有完全一致的database
一个ospf区域就对应一个ospf database
3. route table (从ospf这个路由协议, 学到的路由)
在ospf里, 通过spf算法得到的到达目的地最短的路径

区域划分, 及划分的目的

ospf two-level hierarchy:
transit area (backbone or area 0)
regular areas (nonbackbone areas)

划分的目的:

1. 提高路由的效率:缩减部分路由器的ospf路由条目, 对某些特定的lsa, 可以在区域边界上, 实现汇总/过滤/控制, 而实现全网互通
2. 提高网络的稳定性:当某个区域的某条路由出现抖动时, 可以减少受影响的波及面.

ospf的链路类型:

1. P2P (HDLC/PPP Serial/Point2Point sub if)
2. BMA (Ethernet/TR/FDDI) (局域网)
3. NBMA (FR/ATM/X. 25) (广域网)
(可通过sh ip ospf int 查看各接口的网络类型)

串口网络类型默认:POINT_TO_POINT

以太网网络类型默认:BROADCAST

环回口网络类型默认:LOOPBACK

关于MA网络的DR/BDR选举:

step1:根据ospf路由器的接口的优先级选举DR/BDR, 每个接口默认的优先级都是1

其中优先级最大的成为DR, 次大的成为BDR, 其它的都是DR-Other
如果有路由器的pri为0, 表示放弃DR/BDR的选举.

Other

如果有路由器的pri为0, 表示放弃DR/BDR的选举.

priority:1--255

(串口默认没有优先级, 因为它默认是POINT_TO_POINT, 不需要选举DR/BDR的, 可通过sh ip ospf int e0查看)

step2:如果接口的优先级相同, 将使用router-id来决定DR/BDR的选举:其中router-id最大的成为DR, 次大的成为BDR, 其他的都是DR-other

OSPF的算法:

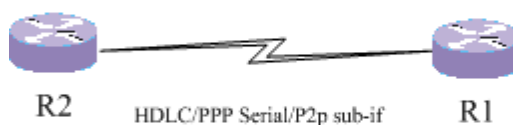
1. 每一个ospf区域就对应着一个独立的ospf database (LSDB)
2. 每一个ospf路由器, 都生成了以自己为根的一棵SPF树
3. 从本路由器出发, 到特定目标网络的整体开销最小的路由成为最佳路径
4. 那么这条最佳路径, 就成为ospf这个协议, 提交给路由表的, 到达这个目标网络的路由.

OSPF 对 不同链路 的处理分类:

OSPF Routing updates and topology information are only passed between FULL adjacent routers

1. P2P点对点 (HDLC/PPP Serial/Point2Point Sub-if) 一定FULL状态 (有DR/BDR的选举的)

点对点



S0

2. BMA多路访问 (Ethernet/TR/FDDI) (局域网)
Broadcast Multit-Access (有DR/BDR的选举的)
3. NBMA 非广播访问 (FR/ATM/X. 2. 5) (广域网) 图1
Non-broadcast Multi-access (有DR/BDR的选举的)

DR: 相当村长

Non-broadcast Multi-access (有DR/BDR的选举的)

DR: 相当村长

BDR: 相当付村长

DR-Other: 相当村民

MA: 相当是村

关于MA (Multi-access) 网络的DR/BDR的选举:

STEP1: 根据OSPF路由器的OSPF接口的优先级选举DR/BDR:

每个接口默认的优先级都是: 1

其中优先级最大的成为DR, 次大的成为BDR

如果有路由器的Priority=0, 放弃DR/BDR的选举

(OSPF Priority: 0~255)

图2

STEP2: 如果接口的优先级相同, 将使用Router-ID来进行DR/BDR的选举:

其中Router-ID最大的成为BD, 次大的成为BDR.

OSPF的SPF算法

1. 每一个的OSPF区域, 就对应着一个独立的OSPF Database (LSDB)
2. 每一个OSPF路由器, 都生成了以自己为根的, 一棵SPF树.
3. 从本路由器出发, 到特定目标网络, 的整体开销最小的那个路径, 成为最佳路径.
4. 那么这条最佳路径, 就成为OSPF这个协议, 提交给路由表的, 到达这个目标网络的路由.

最长匹配比较, AD比较, Metric比较:

LAB1: 最长匹配比较 图

LAB2: AD比较

LAB3: Metric比较

LSA (数据链路通告) 的传播更新规律 (OSPF是LS协议, 无需遵循水平分割)

STEP1: 如果本路由器从来没有收到过此LSA, 那么路由器就将其加入LSDB, 并且转发/泛洪此LSA, 同时进行SPF计算, 得出达到此目标的最佳路由.

STEP2: 如果本路由器, 曾经收到描述同一个网络的LSA:

STEP2: 如果本路由器, 曾经收到描述同一个网络的LSA:

2-1: 如果LSA的序号, 与自己已有的相同, 则丢弃此LSA.

2-2: 如果LSA的序号, 与自己的更新, 则同 “STEP1”, 去计算最佳路由.

2-3: 如果LSA的序号比自己的更旧, 就将自己较新的LSA, 发送给源.

OSPF 的五种数据包:

1. hello
2. database description (简称DBD) 数据库描述包
3. link-state request (简称LSR)
4. link-state update (LSA是包含在LSU中的) (简称LSU)
5. link-state acknowledgement (简称LSAck)

IP协议号

OSPF 的L3 protocol = 6 , 那么L4就是TCP模式

OSPF 的L3 protocol = 17 , 那么L4就是UDP模式

OSPF 的L3 protocol = 88 , 那么L4就是 EIGRP模式

OSPF 的L3 protocol = 89 , 那么L4就是 OSPF模式

OSPF的Protocol ID:89 (EIGRP:88)

在OSPF的HELLO包中, 影响建立邻居的4个关键因素:

1. HELLO/Dead intervals(间隔)
2. 链路所在的AREA ID
3. OSPF认证的密码
4. Stub(NSSA)标识位.

以上的4个因素, 必须一致, 否则无法建立OSPF邻居.

OSPF邻接建立过程

1. Down state
 2. Init state 初始化状态
 3. Two-Way state
 4. Exstart state (决定主从路由器, 从R-ID来区分, 然后DR优先发送DBD信息出来)
 5. Exchange state (大的R-ID 发送到小R-ID DBD信息, 然后小的R-ID把大的R-ID没有的DBD信息发送回大的R-ID)
 6. Loading state
 7. Full state
-

OSPF数据包的发送地址

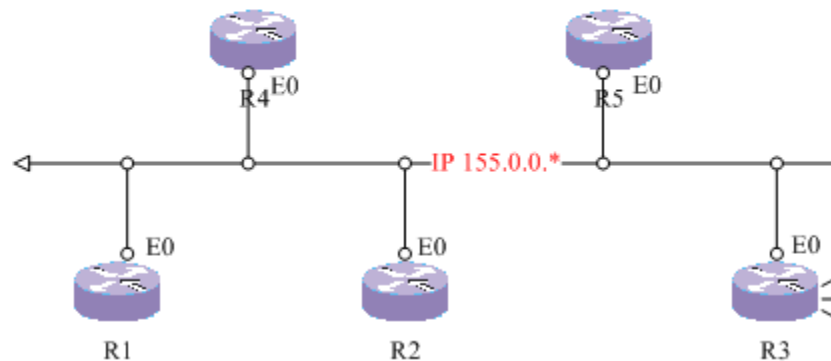
DR/BDR notifies LSU on 224.0.0.5

DR-Other notifies LSU TO OSPF DR on 224.0.0.6

LAB1: OSPF 立刻确定 Router-ID (要求全网唯一的, 相当身份证)

图

Router ID 100.0.0.*



R1# show ip ospf

在OSPF的路由器上, 确定Router-Id的3个优先级别:

STEP1: (建议使用route-id 命令来确定Router-ID)

通过Router-ID命令, 修改Router-ID, 其优先级别最高, 也是建议的

R1(config)#router ospf 110

R1(config-router)#router-id 100.0.0.1
(loopback1:100.0.0.1)

STEP2: 假如, 没通过Router-ID命令指定R-ID,

那么路由器会自动地将自己的环回口的IP, 作为R-ID.

如果存在多个环回口:

那么路由器会自动地选择一个IP地址最大的那个IP, 作为自己的R-ID:

LP5:20.0.0.3

LP8:10.0.0.3

STEP3:

如果路由器上, 连一个环回口都没有, 那么路由器会自动从当前是Active (激活状态下:UP/UP)的物理接口中, 选择IP地址最大的那个IP, 作为自己的R-ID.

这是不稳定的, 不建议的方法.

图2:

图2:

LAB2:通过反掩码, 控制有那些接口, 在运行OSPF协议.

STEP1: 启动OSPF, 并宣告网络:

```
R5(config-router)#network 35.0.0.0 0.0.0.0 area 0
```

表示特定一个接口, 在运行OSPF协议

```
R3(config-router)#network 0.0.0.0 255.255.255.255 area 0
```

表示本路由器上的所有接口, 都在运行OSPF协议.

```
100.0.0.0 0.255.255.255
```

反掩码的匹配原则:

0:表示准确匹配

1:表示忽略不计

结论:在OSPF/EIGRP 中, 不表示这个接口所在的网络的长度, 而表示运行路由协议的接口的范围(那有些接口运行OSPF)

默认情况下:

因为OSPF的环回口都是以LOOPBACK作为“其网络类型”(OSPF的运行模式)

所以其路由长度是32的主机路由

通过以下命令察看网络类型:

```
show ip ospf interface loopback 3
```

```
network type:LOOPBACK
```

若要显示准确的显示路由长度, 需要改为P2P,

```
R5(config)# Interface Loopback 3
```

```
R5(config)# ip OSPF network Point-in-point
```

LAB3: OSPF 必需察看的4个show

察看路由器, 当前有那些接口在运行OSPF:

```
Show ip ospf interface
```

察看路由器的OSPF邻居表, 当前有那些OSPF的邻居(第一张表)

```
Show ip OSPF neighbor
```

察看路由器的LSDB (第二张表)

```
Show ip ospf database
```

察看从OSPF学到的路由:(第三张表)

Show ip route ospf

图3:

LAB4:DR/BDR的选举:(只发生在多路访问网络/Mmulti-Access Network)

```
R1#show ip ospf interface ethernet 0
      Router ID 100.0.0.1
      State DR/BDR/Drother
```

```
R1(config)# in e 0(修改特定接口的优先级)
R1(config-if)# ip ospf priority 10
```

特别注明:OSPF的优先级是对某个的MA端口而言,而不是针对整个路由器的.

在HUB&spoke的NBMA网络中,中心点hub应该为DR.

结论:

1. 同一个路由器的不同MA接口,可能在不同的MA (Mmulti-Access Network)网络中,充当不同的DR/BDR/DR-other
2. 在一个MA (Mmulti-Access Network)网络中:
DR/BDR与所有的邻居都是FULL状态,DR-Other与DR/BDR都是full的,但与别的DR-Other是2way状态.

交换

day-1

VLAN/TRUNK/VTP

VLAN的主要目的:低成本的实现网络的分片/分段/segment

VLAN的主要目的:低成本的实现网络的分片/分段/segment

A vlan:a broadcast domain 一个vlan对应一个广播域(广播流量不能穿越VLAN的边界)

A logical network (subnet) (为了实现VLAN之间的数据通信,需要依靠不同VLAN的不同子网实现VLAN间的路由)

一个vlan就对应一个广播域,也对应一个逻辑子网

VLAN:限制了广播域的范围,
VLAN间路由:又允许了VLAN间的正常数据包的通信.

vlan的分类:

静态vlan(基于端口的vlan)

动态vlan(基于MAC的vlan) (网管记录网络上机器的MAC到vmps服务器上,当机器登录网络的时候,vmps分配不同vlan)

LAB1:vlan的创建,修改,删除

~~~~~  
~~~~~

默认情况下,所有的交换机接口,都在vlan1里面.

vlan1001---vlan1005,是默认fddi等用途,和vlan 1一样,是不能删除的.

VLAN的创建:

方法1:

sw1(config)#vlan 10(新方法,主流设备)

sw1(config-vlan)#exit

方法2:

sw1#vlan database (老式)

sw1(vlan)#vlan 20

sw1(vlan)#apply (应用,即保存了所做的配置)

sw1(vlan)#exit

sw1#sh vlan brief (查看vlan的摘要信息)

vlan的修改:

sw1(config)#vlan 10

sw1(config-vlan)#name ENG

sw1#vlan database

sw1(vlan)#vlan 20 name DRAGON

sw1(vlan)#apply

sw1(vlan)#exit

```
sw1(vlan)#exit
```

(注意:在使用老方法创建,修改vlan时,如果不使用apply,而直接使用ctrl+z推出,则所有配置都不保存)

vlan的删除:

```
sw1(config)#no vlan 10
```

```
sw1#vlan database
sw1(vlan)#no vlan 20
sw1(vlan)#apply
sw1(vlan)#exit
```

```
~~~~~
~~~~~
```

将特定的1号端口,放入vlan 10中:

```
sw2(config)#interface fastEthernet 0/1
sw2(config-if)#switchport access vlan 10
```

对多个端口同时进行操作命令:

关闭一些没有用到的接口:

```
in range f0/5-8
shut
```

```
sw2(config)#interface range fastEthernet 0/6 - 12 (注意:端口号
之间需要空格)
sw2(config-if-range)#switchport access vlan 10
```

```
(7,9)
sw2(config)#interface range fastEthernet 0/7 ,fastEthernet 0/9
sw2(config-if-range)#switchport access vlan 10
```

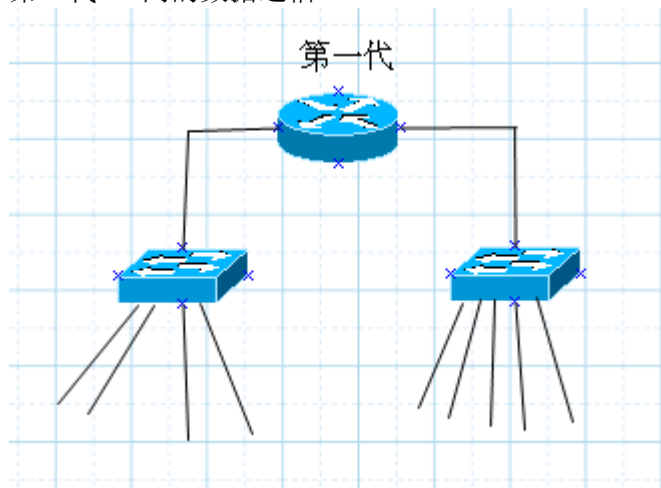
LAB:

```
~~~~~
~~~~~
```

Routing Between vlans

Routing Between vlans

第一代LAN间的数据通信:



不支持vlan的交换机:

一个路由器, 加上几个交换机,
每个交换机的所有端口, 都同属于一个LAN/网段.
在路由器上, 有几个网段, 就有几个与之对应的直链路由,
在那个时代, 还没有vlan这种技术.

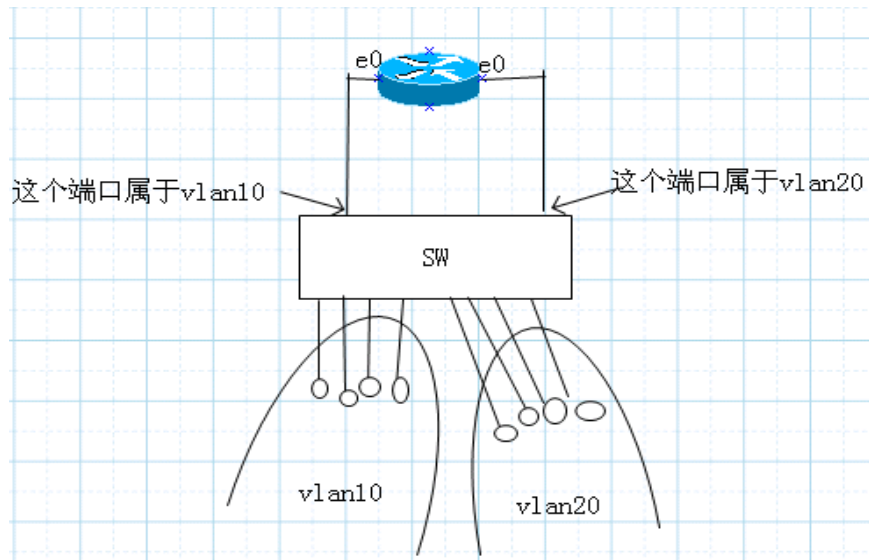
缺点:

交换机的所有端口都同属于一个LAN,
需要路由的端口, 与LAN的个数, 一一对应,
导致建网成本很高.

优点:

易于理解, 便于实施

第二代VLAN间的数据通信:



在这个时代, 出现了能够支持vlan的交换机

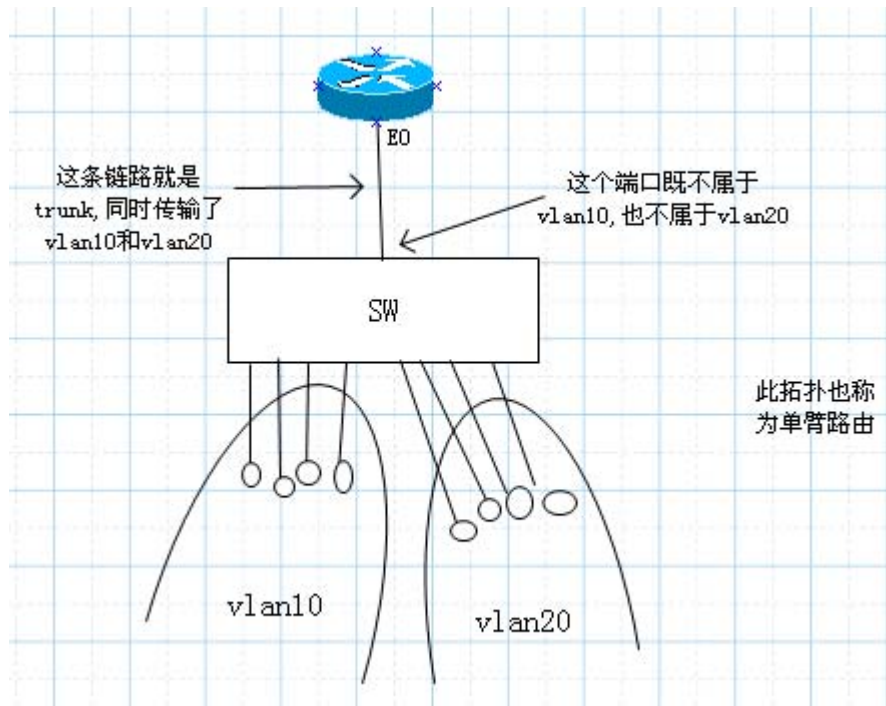
在路由器和交换机之间的链路, 的那个交换机端口, 是完全属于所对应的那个vlan的

优点:

可以将原属于每一个LAN的交换机, 减少到只需要一台交换机, 大大降低成本.

缺点: 路由器上, 还是一个端口对应一个vlan, 成本还是很高

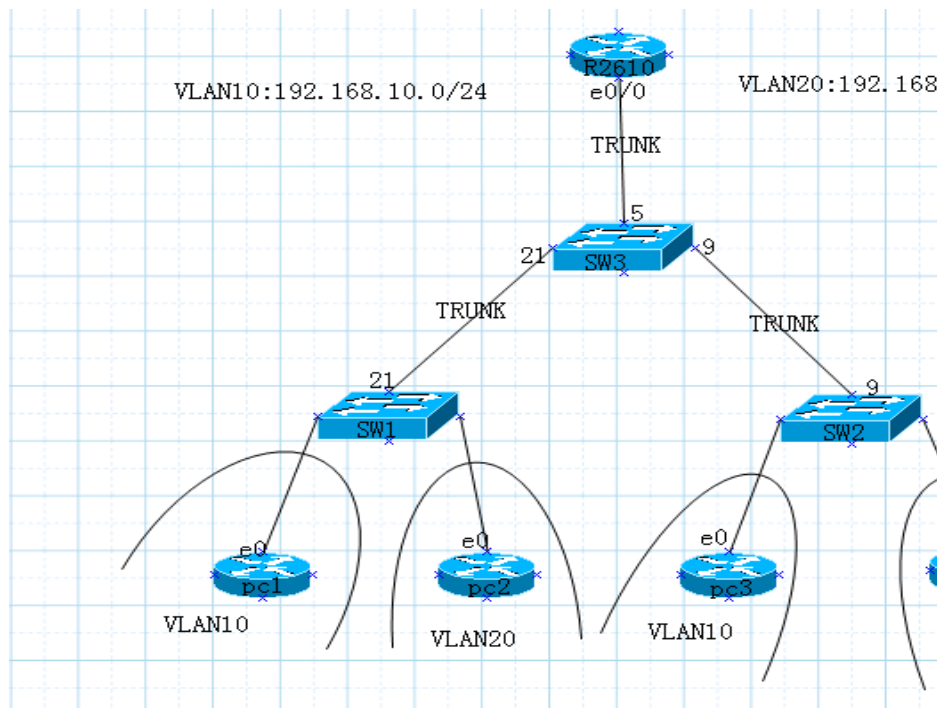
第三代vlan间的数据通信:



在这个时代, 出现了trunk技术:

trunk(cisco的称谓): 在一条物理介质上(网线/光纤), 传输多个vlan的信息

~~~~~  
~~~~~



单臂路由(router on a stick) 注:stick [stik] n. 棍, 棒, 手杖
step1:确认交换机之间的trunk链路:

Trunk:

在一条网络介质上, 同时传输多个vlan的数据, 这种技术, 在cisco称为Trunk.

CISCO交换机支持的两种trunk:

1. 802.1q

业界标准, 开放性的标准, 能够兼容老式设备(不能支持trunk的设备)

当交换机检测到数据包的目标地址不在本交换机时,

当检测到应当发到某条trunk链路时,

先进行trunk封装:

在MAC地址后面, 添加上4Byte的tag, 其中包含12bits的VLAN ID. (贾雷

注:所以可以通过4096个vlan)

VID是用于描述数据帧是属于哪个VLAN的.

在数据包完成添加tag后, 会重新计算其FCS校验.

完成封装后, 就可以从trunk链路上发给对方交换机了.

SW3是一个3550

SW3(config)#int fa0/21

SW3(config-if)#switchport trunk encapsulation dot1q

SW3(config-if)#switchport mode trunk

SW3另外的f0/9. f0/5都是同样配置.

SW1/SW2 (这两个是SW2950, 只支持802.1q, 默认就是)

```
SW1/SW2(config)#int fa0/21
```

```
SW1/SW2(config-if)#switchport mode trunk
```

SW3#sh int trunk 查看trunk链路的建立

step2:在路由器2610上, 构建以太口的子接口, 并在子接口中构建trunk.

```
int e0/0
```

```
no shut
```

```
int e0/0.10
```

```
encapsulation dot1q 10
```

```
ip add 192.168.10.100 255.255.255.0
```

```
int e0/0.20
```

```
encapsulation dot1q 20
```

```
ip add 192.168.20.100 255.255.255.0
```

step3:

```
R2610#sh ip route
```

会看到两条直连路由

step4:在3个交换机上, 配置vlan10/20

一共配置了6次.

step5:将连接用户的端口放入对应的vlan中:

```
SW1#
```

```
int f0/1
```

```
switchport access vlan 10
```

```
!
```

```
int f0/2
```

```
switchport access vlan 20
```

```
SW2#
```

```
int f0/3
```

```
switchport access vlan 10
```

```
!
```

```
int f0/4
```

```
switchport access vlan 20
```

step6:关闭R2610上的两个子接口的“代理ARP”功能:

```
int e0/0.10
no ip proxy-arp
!
int e0/0.20
no ip proxy-arp
```

step7:观察同一VLAN中的用户的通信

PC1-PC3!!!!!!!!!!

PC2-PC4!!!!!!!!!!

step8:观察不同VLAN之间的用户的通信

在没有指定默认网关之间,不同vlan间ping会显示封装失败.

整个vlan 10的所有用户,都以E0/0.10子接口为网关

PC1(config)#ip default-gateway 192.168.10.100

整个vlan 20的所有用户,都以E0/0.20子接口为网关

PC2(config)#ip default-gateway 192.168.20.100

~~~~~  
~~~~~

ISL:

是cisco私有的协议.

在原始数据帧外,生成一个26bytes的isl头,还有4个bytes的gsc尾,

在ISL头中包含了10 bits的VID. (应该是10位的,所以支持1024个vlan. CCIE考题)

贾雷的注:和802dot1q比较起来,改变了原来的帧的头部. 所以稳定性较差.

LAB2:使用isl构建trunk链路:

SW3(config)#int fa0/20

SW3(config-if)#switchport trunk encapsulation isl

SW3(config-if)#switchport mode trunk

在路由器上:

int e0/0.10 (为vlan 10工作的子接口)

encapsulation isl 10 (10指vlan 10)

ip add 192.168.10.100 255.255.255.0

关于native vlan(本征vlan)

1) 默认是vlan1. 一般情况下,不修改native vlan. 如果非要修改,那么网

么网络里面所有的交换机就要修改为相应的vlan, 比如vlan10.

2) native vlan是在trunk传输的时候, 不打tag标签.

3) native vlan的主要设计用途, 是连接那些, 老式的交换机.

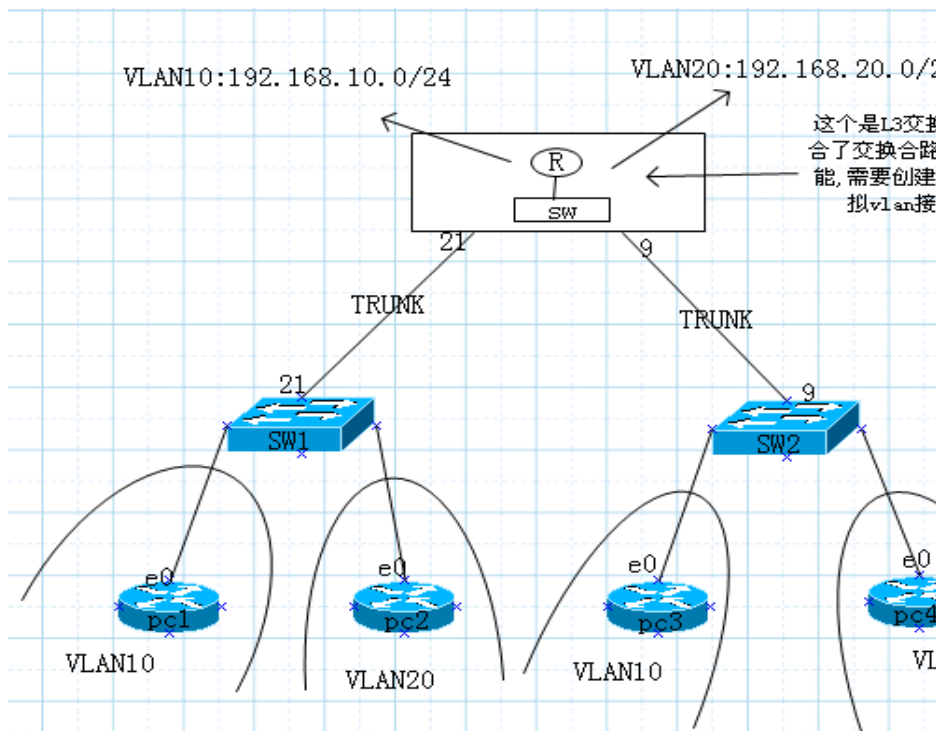
4) 做单臂路由的时候, 子接口上也可以封装native vlan. 命令如下:

```

int e0/0.10
encapsulation dot1q native
ip add 192.168.10.100 255.255.255.0

```

第四代VLAN间的数据通信: 将简化的路由器, 和交换机合二为一, 成为带有路由功能的交换机. (L3 SW)



step1: 在L3交换机上, 为每个vlan, 创建一个与之对应的vlan接口 (逻辑接口)

```

int vlan 10
ip add 192.168.10.100 255.255.255.0
no ip proxy-arp
!
int vlan 20
ip add 192.168.20.100 255.255.255.0
no ip proxy-arp

```

!

(贾雷的注:如果这里proxy-arp启用, 那么不需要启用路由功能, 不同vlan之间也可以联通.)

step2:所有vlan10/20的用户, 都指定本vlan接口作为网关

```
PC1(config)#ip default-gateway 192.168.10.100
```

```
PC2(config)#ip default-gateway 192.168.20.100
```

此时需要清下ARP表, 否则MAC会封装失败

```
sh ip arp
```

```
clear arp-cache
```

step3:需要在SW3上启动路由功能才能看到路由表.

```
SW3(config)#ip routing
```

```
c 192.168.10.0/24 ,vlan10
```

```
c 192.168.20.0/24 , vlan 20
```

step4:测试各vlan间的通信!!!!!!!

第四代的优点:从成本考虑, 不需要外置路由器, 可以降低成本.

缺点:受内置路由器的cpu查询路由表的能力限制, 多次路由, 多次交换

~~~~~  
~~~~~

第五代VLAN间的数据通信:

Multilayer switching 一次路由, 多次交换.

流掩码/Netflow mask

记录了:源IP/PORT, 目标IP/port, 协议

~~~~~  
~~~~~

第六代VLAN间的数据通信:

基于CEF的多层交换 cisco express forwarding

FIB(转发信息数据库)

邻接关系表

流掩码

没有路由, 直接多次交换

```
SW3550(config)#ip cef
```

```
SW3550#sh ip cef  
sh vlan brief
```

```
~~~~~  
~~~~~
```

switch ports: 交换端口, 是2层的物理端口

分为两种:

access port: (一般是连接主机/电脑)

只能属于一个VLAN, 在access port传输的数据包是**不携带tag标记**的帧, 是标准的以太网帧.

trunk port:

在两个交换机之间, 一定是trunk.

在做需要单臂路由的链路上, 路由器和交换机之间也是trunk

trunk可传输多个vlan的信息, 使用tag中的VID来标识数据帧属于哪个vlan

1) 贾雷的注: port mode有4种: desirable会主动协商. auto不会主动协商. on 一直开 off 一直关. 目前新的交换机默认状态是desirable的, 所以, 新的2台交换机之间对连, 默认情况, 可以自己学为trunk状态.

2) 贾雷的注: 路由器的接口划分子接口后, 本接口下还是可以正常使用?(万和证券, 焦彬案例)

```
~~~~~  
~~~~~
```

routed ports:

分为两类:

真实接口:A

A-1: 路由器的接口

A-2: L3 SW的原交换端口, 但关闭了交换能力, 变成了路由接口.

```
int f0/1
```

```
no switchport
```

```
ip add 10.0.0.35 0.0.0.0
```

虚拟路由接口:B

```
int vlan 10
```

```
ip add 192.168.10.100 255.255.255.0
```

```
~~~~~
```

~~~~~  
~~~~~

VTP:(vlan trunk protocol)

是cisco私有的协议 (业界用的应该是GVRP?)

VTP信息, 只能在trunk链路上传递.

目的: 提高vlan网络的配置效率, 减少网络配置的工作量:

VTP用于通告/同步vlan的配置信息, 简化关于vlan的配置

VTP不能用于将端口放入vlan的操作

(这需要在每个交换机上单独配置)

VTP信息是触发更新, 或者每5分钟周期性通告一次. 即使没有变化也要发一次. (贾雷疑问: 那触发更新是什么意思?)

VTP的三种模式: server/client/transparent

VTP的server/client的相同点:

都可以转发VTP通告信息, 而且都会向最新版本的VTP信息同步.

不同点:

VTP的Server:

可以创建/修改/删除/VLAN, 可以保存这些信息在NVRAM中.

VTP的Client

不能创建/修改/删除/VLAN, 不能保存这些信息在NVRAM中.

VTP的Transparent/透明模式:

拥有独立的一套VLAN数据库,

不会向Server端发出的VTP信息同步,

但会转发Server端发出的VTP信息

而且自己的独立数据库是可以保存的

LAB4:VTP

step0: 确认trunk链路已经成功建立:

SW2#sh interface trunk

step1: 确定一个VTP域名, 确认VTP server/client

一般以网络中, 最高端/中心的交换机做server.

SW1(config)#vtp domain CCNP

SW2(config)#vtp domain CCNP

SW3(config)#vtp domain CCNP

```
SW1(config)#vtp mode server (默认都是server)
```

```
SW2/3(config)#vtp mode client
```

```
sw2#sh vtp status
```

(查看vtp信息, 特别注意, sh run 是看不到vtp信息的)

(保存vlan.dat文件中 注意是小写)

```
sw2#delete flash:vlan.dat (删除vtp的配置信息)
```

```
sw2#delete flash:config.text (删除交换机的配置信息)
```

step2:通过在server端, 增加/修改/删除vlan, 观察client vlan信息的同步, 观察vtp修订版本号revision的变化

step3:vtp的安全性

需求:如果不配置密码, 有新来的恶意交换机, 它的vtp 修订版本号 revision比较高的话, 接入我们的网络来, 给我们的网络带来很大影响, 所有的vtp client都按照恶意交换机的vlan 信息修改了.

```
SW1(config)#vtp password 123
```

```
SW2(config)#vtp password 123
```

通过在server端, 增加vlan, 观察SW2, 发现已经有新vlan. 观察SW3, 发现没有新vlan. 所以说明没有配置VTP密码的交换机, 不能得到更新.

贾雷注: 更新vlan, 是看谁的revision更高的. 如果一个新的交换机接入现有的vtp domain, 不管它是vtp client, 还是vtp server, 只要它的vtp revision是更高的, 那么他就会网络产生影响, 更新 v l a n .

cisco solution:

quickly reconfigure all of the vlans on one of the vtp servers.

what to remember:

always make sure that the configuration revision of all switcher inserted into the vtp domain is lower than the configuration revision of the switches already in the VTP domain.

```
SW3(config)#vtp password 123
```

```
SW2#sh vtp password
```

```
VTP Password:123
```

step4:vtp的修剪(在server端做)

需求:如果某台client交换机的某个vlan没有用户, 他收到很多这个vlan的信息有丢弃, 浪费了很大资源, 为了减少资源浪费, 那么做vtp的修剪, 使

修剪,使得vtp server不传给他此vlan的信息.
SW3(config)#vtp pruning

day-2

STP 生成树

技术背景:

当两segment之间,当只有一个物理连接时,就可能出现“单点失效”.

segment的3种概念:

1. 在stp领域:

一段网络介质(网线/光纤)

2. 数据封装领域:

携带了4层包头的用户数据

3. 在路由领域:

被3层设备(路由器)所分割的,逻辑子网.

避免“单点失效”性的方法:构建冗余网络:

冗余交换网络避免了网络的“单点失效”/single point of failure

但是同时带来了3大致命问题/核心问题:网络环路(打环)

1. multiple frame copies/多帧复制(轻微) 某交换机收到多个相同的帧.

2. MAC database instability/MAC地址数据库的不稳定(中等)/端口漂移

3. broadcast storms /广播风暴(严重)

解决方案:STP== spanning-tree protocol

STP核心:provides a loop-redundant network topology,by placing certain port in the blocking state.

在冗余网络中,通过STP算法,将特定的端口置于阻塞状态,来实现既没有环路,也可以冗余的网络.

BPDU:(bridge protocol data unit)

BPDU:(bridge protocol data unit)

在交换网络中,由根桥,每2秒发一次,用于STP的计算和收敛.
(root bridge)

BPDU的作用:

选举根桥

确定本地是否形成环路

阻塞特定端口,防止环路形成

监控生成树的状态

STP 4大原则(越小越好) 贾雷注:类似于BGP的选路原则,从上到下匹配,一段比出不同,就结束.

- 1.lowest root BID
- 2.lowest path cost to root bridge
- 3.lowest sender BID
- 4.lowest port ID

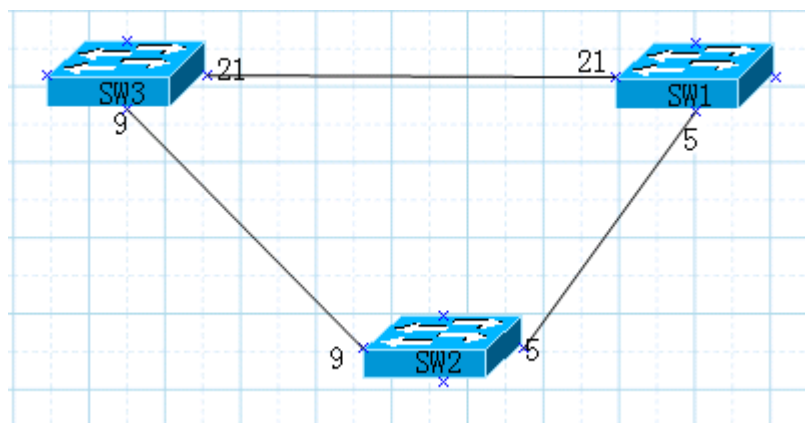
STP 4大结论:

- 1.one root bridge per network
- 2.one root port per nonroot bridge
- 3.one designated port per segment (这里的网段指第一种解释,即一段网络介质)
- 4.nondesignated ports are blocked

结论1:

one root bridge per network

每一个交换(STP)网络,都只有一个根桥



LAB1:根桥的选举

引用原则1:Lowest BID的那个交换机,就是root bridge

BID (bridge ID)

由本交换机的网管MAC地址(6字节), 和交换机的优先级(2字节)组成, 一共8个字节

(网管MAC地址: 可以理解为被telnet的那个地址)

step1: 察看交换机的网管IP 所对应的那个MAC地址

```
SW1#sh versions
```

```
Base ethernet MAC address : 00:0F:FE:2A:80:F6
```

交换机的STP的优先级, 默认都是0x8000 (32768)

step2: 查看STP相关信息: (查看本机BID)

```
sh spanning-tree
```

ROOT ID是指在本STP网络中, 根桥的BID

如果root ID=本交换机的bridge ID, 说明本交换机就是根桥.

step3: 人为控制, 根桥的选举(控制根桥/后备根桥的选举)

```
SW3(config)#spanning-tree vlan 1 root primary  
(24576=0x6000)
```

```
SW1(config)#spanning-tree vlan 1 root secondary  
(28672=0x7000)
```

step4: 通过直接控制交换机的优先级, 来控制根桥:

```
SW3(config)#spanning-tree vlan 1 priority 0 (0x0000)
```

```
SW3(config)#spanning-tree vlan 1 priority 4096 (0x1000)
```

```
~~~~~  
~~~~~
```

结论2:

每个非根桥都有一个根端口

```
one root port per nonroot bridge
```

(根端口) (除了根桥以外的所有交换机, 都是非根桥)

LAB2: 非根桥交换机的根端口, 的选择:

引用原则2:Lowest path cost to root bridge

根端口:在非根桥交换机上, 到达根桥所需的路径开销, 最小的那个端口

根桥上, 没有根端口, 所有端口都是“指定端口designated”

所有端口:参与STP运算的端口, 通常是交换机与交换机相连的端口

每个非根桥, 都只有一个根端口

link speed	cost
10Gbps	2
1Gbps	4
100Mbps	19
10Mbps	100

step1:

```
SW2#sh spanning-tree
```

当前的root port是9

cost 19 (当前去往根桥的cost值)

step2:

```
SW2(config0if-f0/9)#speed 10 (Mbps)
```

```
sh int status
```

```
sh spanning-tree detail
```

也可以通过直接调试cost值来控制:spanning-tree cost 50

~~~~~  
~~~~~

结论3:

每一个网段有一个指定端口

one designated port per segment

指定端口

LAB3:每条segment上的, designated port的选择

引用原则3:Lowest sender BID

```
SW1(config)#spanning-tree vlan 1 priority 36864
```

(0x9000)

~~~~~

~~~~~

LAB4:sender BID相同的情况下,designated port的选择:

如果sender BID相同,则引用原则4:Lowest port ID

step1:

```
SW3#sh spanning-tree int f0/22
```

step2:

修改接口的端口优先级:

```
SW3(config-if-f0/22)#spanning-tree port-priority 32 (这个值必须是16的整倍数)
```

再查看下:SW3#sh spanning-tree int f0/22

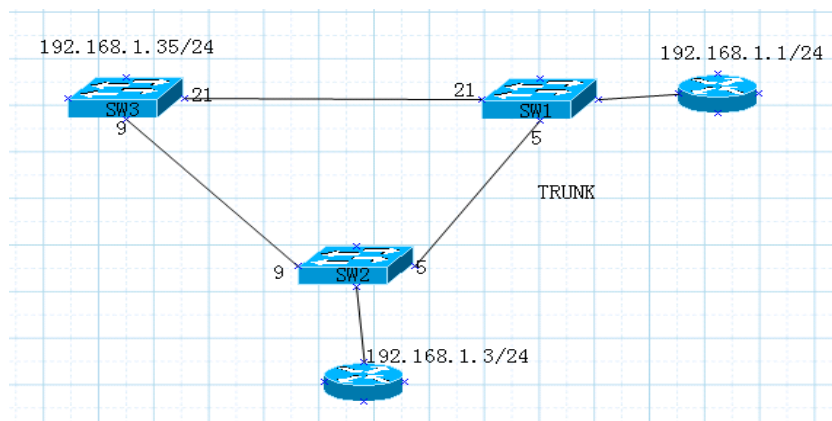
~~~~~  
~~~~~

LAB5:观察blocked端口

察看所有segment:观察除了指定端口,和根端口以外的端口,都是被堵塞的

~~~~~  
~~~~~

LAB6:观察STP的转发延迟



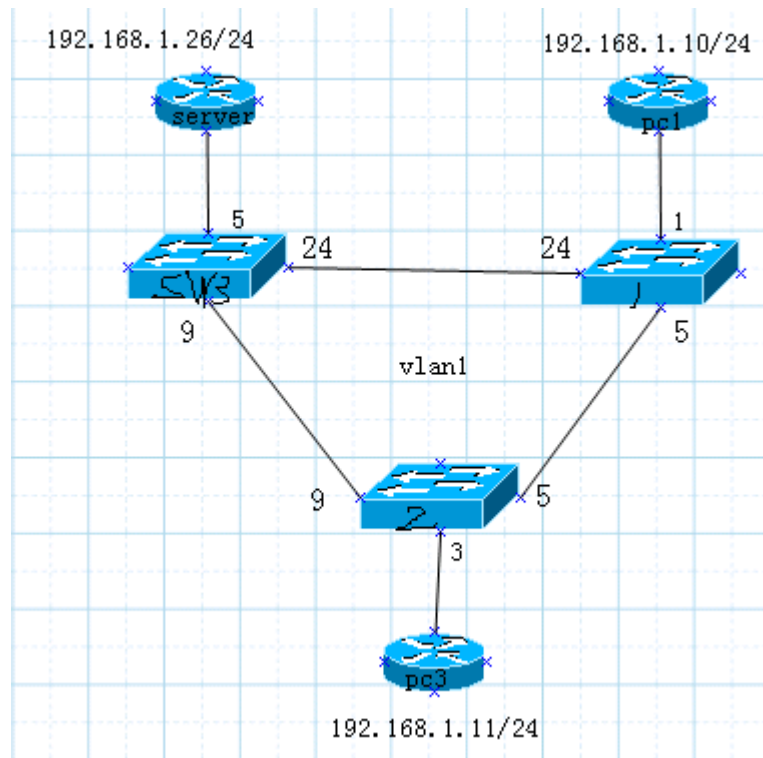
6-1:直链检测错误 (35秒) (贾雷注:在接入层交换机没有配置uplinkfast的情况下,是35s.如果配置了uplinkfast,那么就是接近0s)
(pc1 ping sw3 的网管IP, shutdown SW3 22口观察)

6-2:非直链检测错误(50秒)

(pc3 ping sw3 的网管IP, shutdown SW3 9口观察)

day-3

增强型STP:



1. portfast:

只适用于交换机与主机(电脑)相连的端口,
不应该在交换机与交换机, 路由器, hub互连的网络设备的端口使用.

int f0/1(在交换机上, 在连接主机的端口)

switchport mode access(接入链路)

spanning-tree portfast

对于portfast端口, 如果其接收到BPDU包,
意味这个本该连接主机的端口, 却不正确的连接到了交换机网络,
意味着很可能已经形成了STP环路(STP LOOP).

在STP算法判断出环路, 并且将特定端口堵塞之前, 已经立刻形成广播风

风暴,
交换网络立刻瘫痪.

2. uplink-fast:

在所有接入层交换机上,配置uplink-fast,用于加速因为直链故障/直链检测,所引起的STP网络变化的收敛速度.

direct link

failure

uplink-fast应该在所有接入层/非核心层交换机上配置, 否则很可能会引起STP网络的环路.

```
SW1/2(config)#spanning-tree uplinkfast
```

35 second-----接近0s (跳过blocking/listening 20s+15s=35s)

3. backbone-fast

在所有的交换机上,配置backbone-fast,以利于全网的交换机,当遇到非直链检测错误时,加速其收敛速度.

indirect link failure

```
sw1(config)#spanning-tree backbone-fast
```

```
sw2(config)#spanning-tree backbone-fast
```

```
sw3(config)#spanning-tree backbone-fast
```

能使收敛速度从50减少到35

50' s-----35' s

4. portfast bpduguard

交换机端口的"portfast bpduguard"是指:

在交换机的端口一旦接收到BPDU包时,立刻关闭端口,避免了更大范围的广播风暴.

在连接主机的端口上:(下面这条命令时基于接口的)

```
sw1(config-if)#spanning-tree bpduguard enable
```

全局命令:

```
sw1(config)#spanning-tree portfast bpduguard default
```

是指在所有已经启动了portfast端口中,启动bpduguard

注:

default:enable bpduguard by default on all portfast ports

在所有已经启动了port-fast端口中,启动bpduguard.

贾雷注:端口状态是err-disable状态,通常就是这种情况引起的.当故障排除,err-disable会自动enable?实践中是不会的应该需要手动打开,

set port enable 3/1-2(CCIE考题)

障排除, err-disable会自动enable?实践中是不会的应该需要手动打开,
set port enable 3/1-2(CCIE考题)

5. portfast-bpdufilter

在特定的portfast端口上配置:

```
sw3(config-if)#spanning-tree bpdufilter enable
```

bpdufilter: Don't send or receive BPDUs on the interface.

疑问:那么端口状态是什么?forwarding?不会形成环路吗?

全局命令:

```
sw1(config)#spanning-tree portfast bpdufilter default
```

注:default, portfast-bpdufilter 没有打开?(马老师语)

以上的5种特性, 都是cisco私有的.

802.1w Rapid STP (RSTP) (业界的开放性标准)

RSTP的4种端口角色:

root port : (forwarding)

designated port: (forwarding)

alternate port交替端口:收到别的交换机转发来的BPDU, 此端口是blocking

backup port:收到本交换机转发来的BPDU, 此端口是blocking

RSTP的3种端口状态:

discarding

learning

forwarding

RSTP链路分类:

全双工链路为:point to point 链路

半双工链路为:share link链路

启动RSTP:

```
SW1(config)#spanning-tree mode rapid-pvst
```

standard 802.1q是CST(单生成树)

缺点:

所有的vlan都是按照同一个STP来运行

所有的交换机都只维护一个STP实体(instance)

所有的vlan的数据流量, 都压向一个根桥交换机

无法做基于vlan的, L2的负载均衡

优点:

对交换网络的开销较小, 因为所有的vlan都共享一个STP实体

CISCO私有的STP MODE PVST (per vlan stp)

优点:

可以为每个vlan, 配置一个stp

不同交换机, 可以是不同vlan的根桥

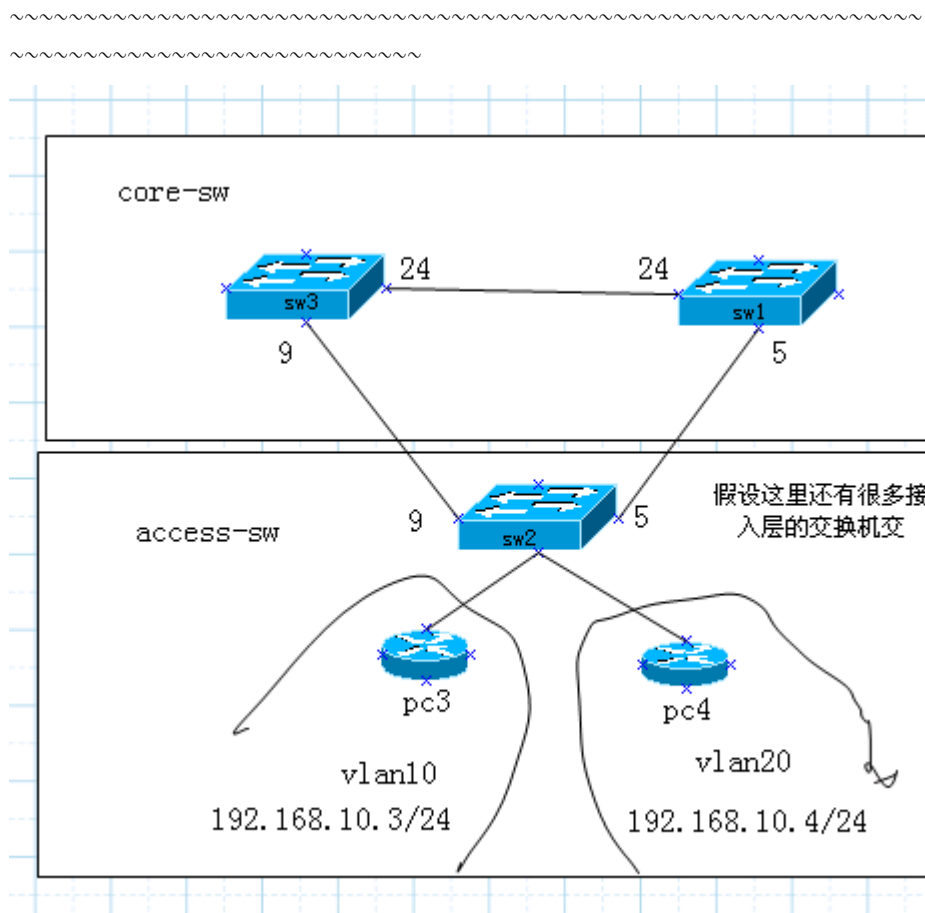
可以实现基于vlan的, L2的负载均衡

缺点:

交换机在维护很多的STP实体

对交换网络的开销比较大

一旦有任意一个VLAN的拓扑发生变化
都可能波及的交换机很多.



LAB3:通过PVST, 实现L2的vlan间的负载均衡:

step0:选定STP的运行模式:

```
sw3(config)#spanning-tree mode pvst
```

or:
sw3(config)#spanning-tree mode rapid-pvst

step1:配置VTP
sw1/2/3(config)#vtp domain CCNP

sw3(config)#vtp mode server
sw1/2(config)#vtp mode client

step2:在其中的一个VTP server上,增加vlan 10 ,vlan20

step3:
sw3成为vlan 10 的根桥,sw1成为vlan 20 的根桥
(同时两交换机要互为备份,互为对方的备份根)

确定两组vlan的根:
sw3(config)#spanning-tree vlan 10-15 root primary
sw1(config)#spanning-tree vlan 20-25 root primary

两交换机互为备份根:
sw3(config)#spanning-tree vlan 20-25 root secondary
sw1(config)#spanning-tree vlan 10-15 root secondary

step4:在sw2上,观察对于体定vlan,哪个是根端口:
sw2#sh spanning-tree vlan 10
sw2#sh spanning-tree vlan 20

~~~~~  
~~~~~

802.1s Multiple spanning treee (MST) (业界的开放性标准)

MST就是对standard 802.1q/PVST的折衷方案:

MST就是基于“组/instance”的stp

MST的主要配置步骤:

- 1:MST首先对VLAN进行分组(instance), 可以分16个组.
- 2:每个组都可以有独立的STP, 根桥, 实现了L2负载均衡, 冗余, 互为备份.

LAB4:通过MST, 实现L2的VLAN间的负载均衡

step1:选定交换机的stp模式:MST
sw1/2/3(config)#spanning-tree mode mst

step2:将vlan划分到不同的“instance/组”中:
sw1/2/3(config)#spanning-tree mst configuration
sw1/2/3(config-mst)#instance 1 vlan 10-15
sw1/2/3(config-mst)#instance 2 vlan 20-25

step3:在核心交换机上,为不同的组,配置不同的优先级,实现两个组的互为备份/负载均衡:

确定每个组的根桥:

```
sw3(config)#spanning-tree mst 1 root primary
sw1(config)#spanning-tree mst 2 root primary
```

互为备份:

```
sw3(config)#spanning-tree mst 2 root secondary
sw1(config)#spanning-tree mst 1 root secondary
```

step4:

```
sw1/2/3#sh spanning-tree mst configuration
instance      vlans mapped
0              1-9, 16-19, 26-4096
1              10-15
2              20-25
```

step5:

```
sw3#sh spanning-tree mst 2
```

day-4

First hop routers on the LAN

Redundancy Network /冗余网络

建立: Fault-tolerant /容错网络

避免: Single Points of Failure /单点失效

硬件冗余:

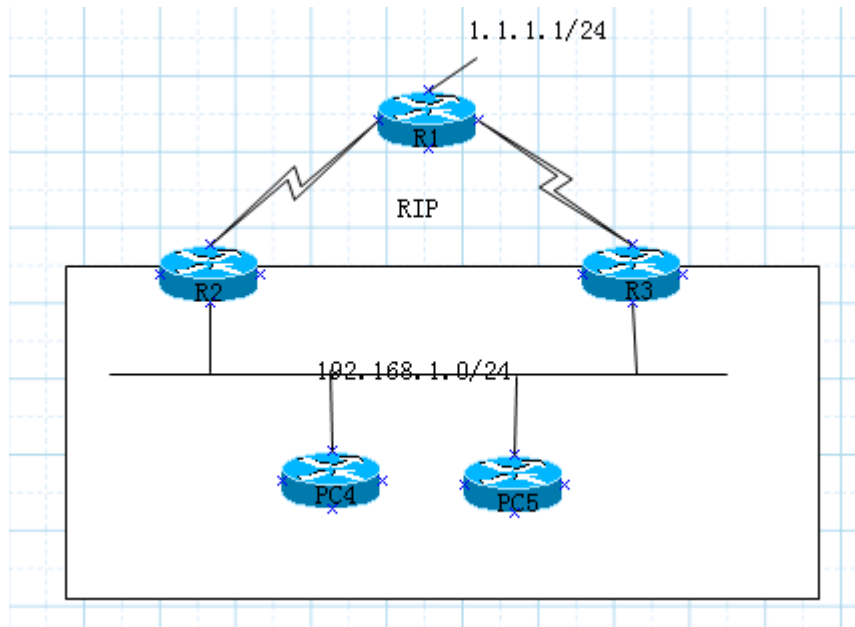
1. 拓扑冗余
2. 交换引擎的冗余, 电源冗余, 线卡冗余, 风扇冗余, 线路冗余

软件/协议的冗余:

HSRP (RFC2281)
VRRP (RFC2383)
IRDP (RFC1256)

IRDP(RFC1256)
GLBP(Gateway Load Balancing Protocol)
SRM(Single Router Mode)
SLB(Server Load Balancing)

~~~~~  
~~~~~



LAB1:default gateway(不运行代理arp)
特点:配置简单,主机上手工配置网关,单点失效

step1:
ho PC4
no ip routing(关闭其路由能力)

ho PC5
no ip routing(关闭其路由能力)

PC4(config)#ip default gateway 192.168.1.2
PC5(config)#ip default gateway 192.168.1.3

step2:在外网运行rip

step3:R2, R3上做NAT:(基于NAT路由器外口地址的端口复用)
3-1:定义内网的用户群

```
access-list 1 permit 192.168.1.0 0.0.0.255
```

3-2:定义NAT的内口,外口

```
int s0
ip nat outside
!
int e0
ip nat inside
!
```

3-3:进行基于NAT路由器外口的端口复用

```
ip nat inside source list 1 int s0 overload
```

PC4和PC5 PING R1, 在断开一边后, 另一边仍然是通的, 且断开的那边的PC不会走另一条链路

无法检测到设备/链路故障, 可能导致单点失效.

step4:在两个网关路由器R2/R3上, 都无法支持代理ARP的情况下:

```
int e0
no ip proxy-arp
```

~~~~~  
~~~~~

LAB2. Proxy ARP (代理ARP):

the client uses address resolution protocol (ARP)

To get the destination it wants to reach ,

and a router will respond to the ARP request with its own MAC address

step1:PC上, 无需配置网关

```
PC4/5#no ip default-gateway
```

step2:在网关路由器的以太口上, 启动代理ARP

```
int e0
ip proxy-arp
(在cisco口路由器上, 默认是启动的)
```

step3:

分别进行PC4, PC5 PING 1.1.1.1进行测试

~~~~~  
~~~~~

贾雷注:

在交换机和路由器里面清除ARP表的命令是:clear arp-cache. 取消proxy arp的命令是:no ip proxy-arp.

在主机里面清除ARP表的命令是:arp -d, 查看是arp -a.

~~~~~  
~~~~~  
3. Routing protocol (路由协议):

(贾雷注:主机上配置动

态路由协议实现备份.)

The client listens to dynamic routing protocol updates

(for example , from IGP (RIP/OSPF))

And forms its own routing table

IRDP (ICMP Router Discovery Protocol):

(贾雷注:很老式的协

议, 已经淘汰.)

Client-the client runs an internet control message protocol

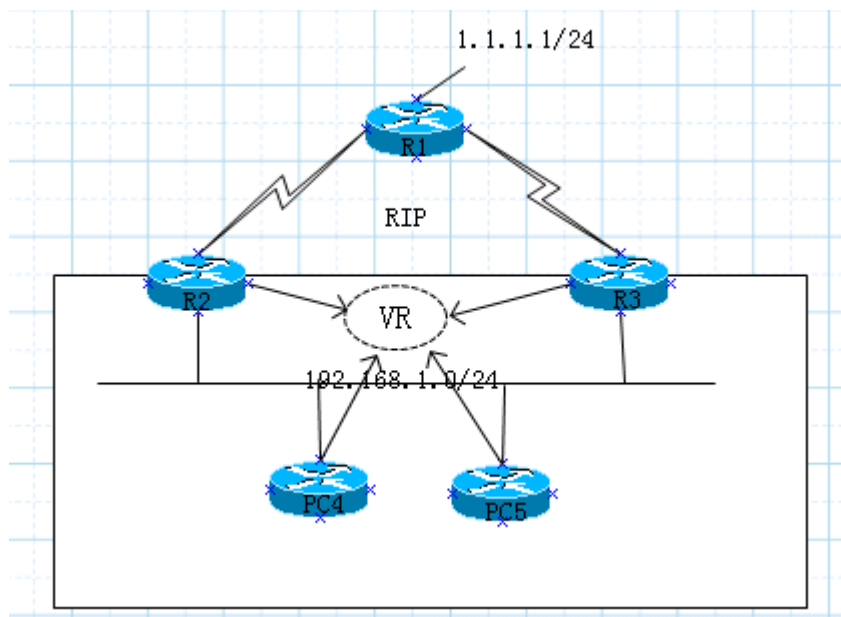
(IGMP) router discovery client.

有缺陷:网络收敛性较慢, 而且受限于主机的操作系统, 比较少有操作系统
win31/95/98支持, 以后的windows2000都需要安装单独的插件.

~~~~~  
~~~~~  
HSRP (Hot standby router protocol热备份路由协议) (CISCO私有)

The hot standby router protocol (HSRP) is a **first-hop**
redundancy protocol (FHRP)

Designed to allow for transparent fail-over of the first-hop
IProuter



LAB3: HSRP

step1:关闭R2/R3的代理ARP

```
int e0
```

```
no ip proxy-arp
```

no ip proxy-arp

step2:构建HSRP

R2/R3#int e0

standby 1 ip 192.168.1.100 (定义虚拟路由器/网关的IP)

R2:standby 1 priority 105 (定义HSRP优先级, 越高越可能成为active
路由器, 默认100)

R3:取默认值

R2/3(config)#standby 1 preempt (抢占, 哪个路由器优先级高谁就成为
active)

step3:内网指定VR(即虚拟路由器)为默认网关

ip default gateway 192.168.1.100

step4:查看HSRP的简要状态

R2/3#sh standby brief

PC4/5 ping 1.1.1.1

step5:debug standby

step6:跟踪HSRP路由器的外口:

R2/3(config-if)#standby 1 track s0 (30)

(跟踪外口, 使当外口断了后, 路由器能检测到后, 本机自动将自己的HSRP
优先级减少10(默认, 可修改), 从而使active router切换到另外一台路由
器上.)

step7:其他命令

R2(config-if)#standby 1 authentication 123 (HSRP的认证)

R2(config-if)#standby 1 mac-address 00c0.abcd.1234

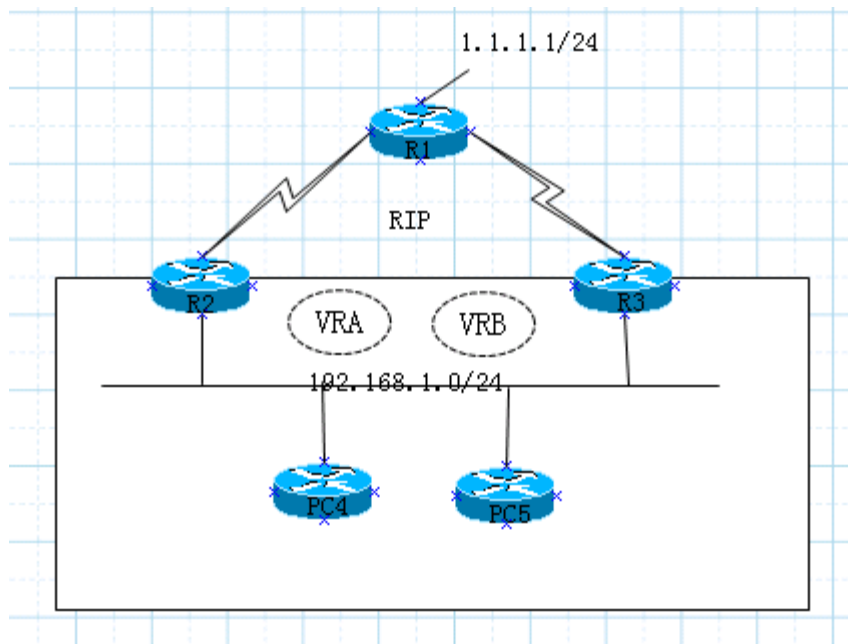
R2(config-if)#standby 1 name vlan-eng

R2(config-if)#standby 1 timers 3 10

~~~~~  
~~~~~

LAB4:使用HSRP, 在同一网段/VLAN, 实现的负载均衡: (R2500 ROUTER)

HSRP	R2	R3
VR-A	Active	100
VR-B	100	Active



step1:

```
R2#  
int e0  
standby 1 name VR-A (贾雷注:仅仅给standby起个名字,标识一下,好记忆.)  
standby 1 priority 105 preempt  
standby 1 ip 192.168.1.100  
standby 1 track s0
```

```
R3#  
standby 1 name VR-A  
standby 1 priority 100 preempt  
standby 1 ip 192.168.1.100  
standby 1 track s0
```

step2:

R2/3(config-if)#standby use-bia (启动多个组)

```
R2#  
standby 2 name VR-B  
standby 2 priority 100 preempt  
standby 2 ip 192.168.1.200  
standby 2 track s0
```

```
R3#  
standby 2 name VR-B  
standby 2 priority 105 preempt
```

```
standby 2 ip 192.168.1.200
standby 2 track s0
```

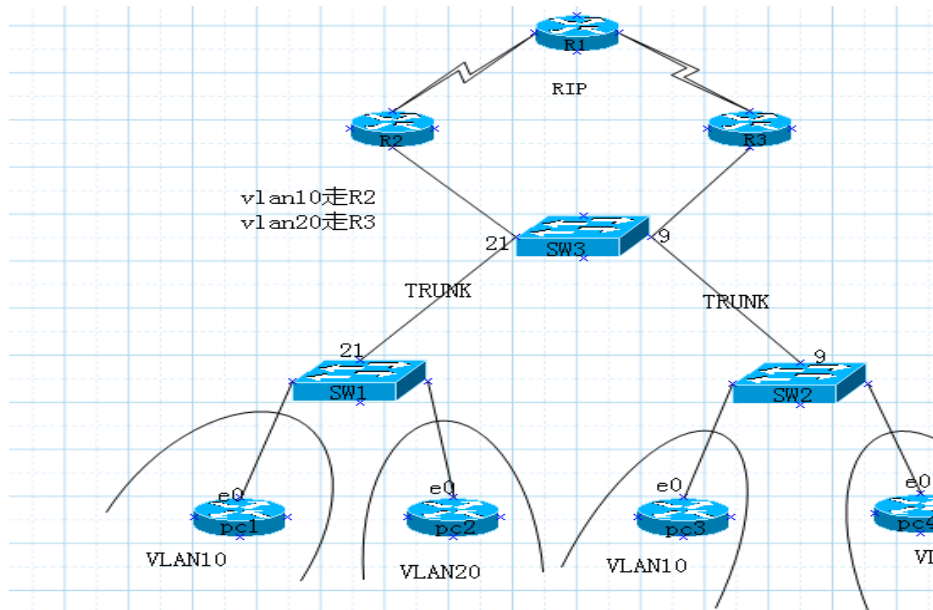
step3:在不同的分组用户中指定不同的VR作为网关:

```
PC4(config)#ip default-gateway 192.168.1.100
```

```
PC5(config)#ip default-gateway 192.168.1.200
```

step4:ping 测试

~~~~~  
~~~~~



LAB5:不同子网(vlan)间的负载均衡(R2600)

```
R2(config)#in e0/0
no shut
```

```
int e0/0.10
en dot1q 10
ip add 192.168.10.1 255.255.255.0
standby 10 ip 192.168.10.100
standby 10 priority 105
standby 10 preempt
standby 10 name VR-10
standby 10 track s0/0
```

```
int e0/0.20
```

```
en dot1q 20
ip add 192.168.10.1 255.255.255.0
standby 20 ip 192.168.20.100
standby 20 preempt
standby 20 name VR-20
standby 20 track s0/0
```

```
~~~~~
~~~~~VRRP (virtual router redundancy
protocol)
```

VRRP 是业界标准协议
组播地址: 224.0.0.18

```
Router 2#
Interface Ethernet 1/0
Ip address 192.168.1.2 255.0.0.0
Vrrp 1 description VL-1
Vrrp 1 priority 100
Vrrp 1 preempt
Vrrp 1 ip 192.168.2.100
```

```
Vrrp 1 authentication cisco
Vrrp 1 timers advertise 2
```

有跟踪接口吗? (贾雷注:有,待确认一下具体用法.)

```
Router 3#
Interface Ethernet 1/0
Ip address 192.168.1.3 255.0.0.0
Vrrp 1 description VL-1
Vrrp 1 priority 105
Vrrp 1 preempt
Vrrp 1 ip 192.168.1.100
```

```
Vrrp 1 authentication cisco
Vrrp 1 timers advertise 2
```

```
~~~~~
~~~~~
```

GLBP: (CISCO私有) (Gateway Load Balancing Protocol)

和它竞争的是HSRP和VRRP

The advantage of GLBP is that it additionally provides load
balancing over multiple routers (gateways) using a single

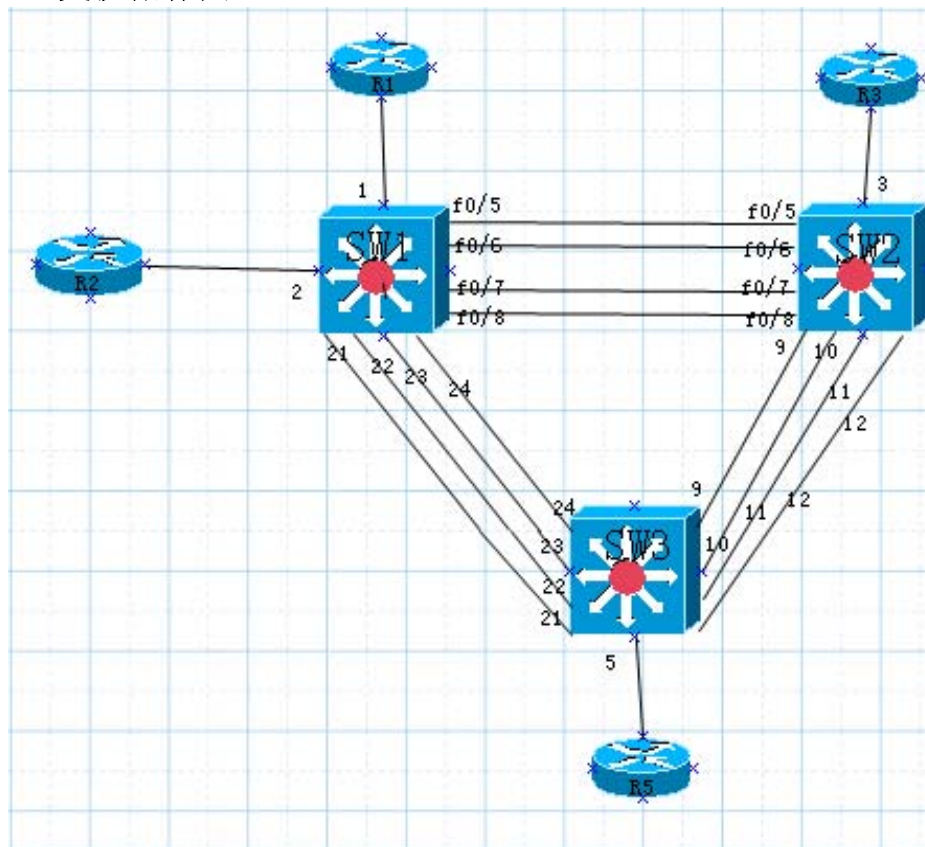
single virtual ip address and multiple virtual MAC addresses

(贾雷注:一个虚拟IP和多个虚拟MAC, 多个MAC同时转发包, 实现真正的负载均衡.)

组播地址: 224.0.0.12

```
Interface fastethernet 0/0
Ip address 10.21.8.32 255.255.255.0
Glbp 10 priority 100
Glbp 10 preempt
Glbp 10 ip 10.0.0.1
```

NP交换拓扑图



贾雷20060816补充

latency 延时。

1) vds1 = long reach ethernet长距离以太网。

1) vdsl =long reach ethernet长距离以太网。

交换机的命令方式:

catOS

configures layer 2 switching.

cisco iso (native mode)

注意datetime和uptime的区别: (默认是uptime)

datetime 日历时间

uptime 开机了多长时间

```
#service timestamps debug datetime msec
```

```
*Mar 13 22:56:29.702: datagramsize=100, ip 25: s=172.16.1.11  
(local), d=10.1.1.1  
(vlan), totlen 100, fragment 0, fo 0, sneding.
```

```
#service timestamps log datetime msec
```

```
*Mar 13 22:55:46.342: %sys-5-config_I: configured from console  
by console
```

LAB1:

ARP

pc1#show interfaces ethernet 0 (察看二层三层上的地址。)

```
pc1: L3 ip 192.168.1.1  
L2 MAC 0010.7b37.c66c
```

```
R2: L3 ip 192.168.1.2  
L2 MAC 0010.7b38.2bf4
```

pc1#debug arp

```
1: creating incomplete entry for ip address: 192.168.1.2  
interface ether0
```

```
2: set req src 192.168.1.1 0010.7b37.c66c,  
dst 192.168.1.2 0000.0000.0000 ethernet0 (这里的  
0000.0000.0000, 可理解为-. -. --)
```

```
3:rcvd rep src 192.168.1.2 0010.7b38.2bf4,  
dst 192.168.1.1 ethernet0
```

pc1#show arp

protocol	address	age (min)	hardware addr	type
interface				
internet	192.168.1.1 -		0010.7b37.c66c	ARPA
ehternet0				

```
internet 192.168.1.2 - 0010.7b38.2bf4 ARPA
ehternet0
```

清除arp表
#clear arp-cache

LAB2: 代理ARP/proxy ARP

```
R2#show ip interface ethernet0
proxy ary is enableed
```

step1:为pc1,指定网关
pc1(config)#ip default-gateway 192.168.1.2
pc1在访问外网时,无需再作ARP查询.

step2:取消默认网关,使用默认启动的代理ARP.
pc1(config)#no ip default-gateway 192.168.1.2

代理arp:
1:creating incomplete entry for ip address: 4.4.4.4 interface
ethernet0
2:sent req src 192.168.1.1 0010.7b37.c66c,
dst 4.4.4.4 --.--.--- ehternet0
3:rcvd rep srv 4.4.4.4 0010.7b37.c66c, (注:ARP欺骗)
dst 192.168.1.1 ethernet0

step3:

疑问:是R2主动发给个pc的信息,进行arp欺骗呢,还是pc主动..... ?

day-5贾雷

NAT (Network Address Translation)

NAT的 3 组关键概念:

1)

inside 我方的网段

outside 对方的网段

2)

global 在公网上能够路由的

local 私有保留地址 10.0.0.0/8 172.16.*.* 192.168.*.*

3)

ip nat outside 思科N A T路由器的外口, 和outside(对方)没有关系

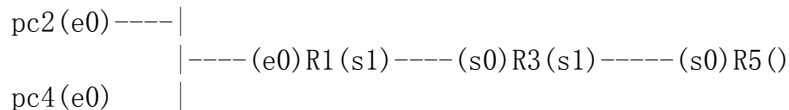
ip nat inside 思科N A T路由器的内口, 和inside(我方)没有关系

~~~~~

~~~~~

LAB:NAT的基本配置(原理性的N A T) (静态, 独立的对外服务器)

图:



s t e p 1: 按图配置 I P 网络, 测试链路, R 1 — R 3 — R 5 之间的链路运行 I G P: R I P V 2。

R1配置指向 I S P 的默认路由

R 1 (config) #ip route 0.0.0.0 0.0.0.0 13.0.0.3

s t e p 2: 在模拟 p c 2 / p c 4 的路由器上, 关闭 i p 路由, 并指定网关:

R4(config)#hostname pc4

pc4(config)#no ip routing

pc4(config)#ip default-gateway 192.168.1.1

并确认, pc2/pc4都能访问到自己的网关.

因为公网路由器上, 没有pc2/pc4所在的网段 / 路由.

并确认，pc2/pc4都能访问到自己的网关。
 因为公网路由器上，没有pc2/pc4所在的网段 / 路由。
 所以，导致对外访问的数据包，无法返回。

step2.5:指定NAT路由器的外口，内口：

```
R1(config)#interface e0
R1(config-if)#ip nat inside
R1(config)#interface s1
R1(config-if)#ip nat outside
```

step4: 静态N A T

```
R1(config)#ip nat inside source static 192.168.1.2 13.0.0.10
                                         私网 I P      公网 i
                                         p
```

step5:观察N A T转换的过程

```
R1(config)#show ip nat translations
```

pro	inside global	inside local	outside global
outside local			
----	13.0.0.1	192.168.1.2	-----

任意协议	我方公网	我方私网	对方公网
对方私网			

贾雷注：对方公网和对方私网是空的表示可以访问对方任何网络。

```
R 1 #debug ip nat
```

出方向的N A T转换

```
s =192.168.1.2-->13.0.0.10, d=5.5.5.5 [5]
```

回方向的N A T转换

```
s =5.5.5.5, d=13.0.0.10 ---> 192.168.1.2 [5]
```

贾雷注：从R 5访问192.168.1.2，可以用13.0.0.10这个地址来访问了。

LAB2:在中小型企业，通常使用基于接口的端口复用：
 (PAT/NAPT/OVERLOAD)

step1:定义N A T的外口 / 内口：（同L A B 1）

step2:通过ACL,定义准备进行N A T的内网用户群：

```
R 1 (config)#access-list 1 permit 192.168.1.0 0.0.0.255
                                         192.168.1.*
```

192.168.1.*

step3:进行基于公网接口s1的N A T端口复用

R1(config)#

```
ip nat inside source list 1 interface s1 overload
                        内网用户      NAT的外口      端口复用
```

测试;

~~~~~

L A B 3：使用route-map,调用内网用户,进行N A T转换,以解决远端端口的一些复杂的端口映射问题.

step1/2:定义N A T内口 / 外口,定义内网用户,(同L A B 2)

step3:在route-map,中调用ACL1

R1(config)#route-map T

R1(config-route-map)#match ip address 1

step4:

R1 (config) #在N A T转换中,调用route-map所定义的用户群:

ip nat inside source route-map T interface serial 1 overload

~~~~~

LAB4:大型企业,向ISP购买较多的公网IP后,

将其分别放入不同的NAT-pool,以供不同的部门进行N A T转

换:

step1:定义NAT的外口 / 内口:(同LAB1)

step2:在R 1 上模拟2个网段的网关:

R 1 #interface e0

ip address 192.168.2.1 255.255.255.0 secondary

ip address 192.168.1.1 255.255.255.0

step3:在2个网段.

pc2:ip ad 192.168.1.2 255.255.255.0

ip default-gateway 192.168.1.1

pc4:ip ad 192.168.2.4 255.255.255.0

ip default-gateway 192.168.2.1

确认每个内网用户,都能访问到自己的网关.

step4:通过ACL,定义准备进行NAT的用户群:

R 1 (config)#

access-list 1 permit 192.168.1.0 0.0.0.255

access-list 2 permit 192.168.2.0 0.0.0.255

step5:将从ISP处买到的公网IP,分别放入为不同VLAN准备的地址池中:

step5: 将从ISP处买到的公网IP, 分别放入为不同VLAN准备的地址池中:

```
R1 (config)#
ip nat pool PL-1 13.0.0.8 13.0.0.11 prefix-length 24
ip nat pool PL-2 13.0.0.12 13.0.0.15 prefix-length 24
=ip nat pool PL-2 13.0.0.12 13.0.0.15 netmask 255.255.255.0
```

step6: 为不同的部门, 进行基于不同NAT-P001的转换:

```
R1 (config)#
ip nat inside source list 1 pool PL-1 overload
ip nat inside source list 2 pool PL-2 overload
```

测试:

```
R1 (config) #show ip nat translations
```

~~~~~  
~~~~~

LAB5: 通过NAT, 实现镜像服务器的负载均衡: (工程中不常用, 一般用windows的cluster)

step1: 定义NAT外口 / 内口. (同LAB1)

step2: 通过ACL, 定义“我方的, 公网”地址, 座位来自外网访问的目标地址.

```
R1 (config)#access-list 30 permit host 13.0.0.100
```

step3: 定义内网的服务器群的地址池, 轮转类型.

```
ip nat pool SER 192.168.10.2 192.168.10.3 prefix-length 24
type ratary
```

(注意这 2 个地址要连续, 不然, 它指的一段地址)

step4:

```
R1 (config)#ip nat inside destination list 30 pool SER
```

step5:

```
Server2/4 (config)#line vty 04
```

```
server2/4 (config-line)#no login (不需要配置telnet密码也可以telnet)
```

测试:

```
ping 13.0.0.100 (不通, 因为没有准确的地址映射)
```

但是:

第一次telnet 13.0.0.100, 登录到server2了

第二次telnet 13.0.0.100, 登录到server3了

第三次telnet 13.0.0.100, 登录到server2了

第四次telnet 13.0.0.100, 登录到server3了

等等, 轮转了.

~~~~~

等等，轮转了。

~~~~~  
~~~~~

LAB6:使用一个接口的公网IP，对外网提供多个不同的服务器：  
(静态的TCP端口映射，静态的端口复用)

贾雷注:目的是达到效果：访问13.0.0.100 80,那么就到server2的www服务，访问13.0.0.100 21,那么就到server3的FTP服务，

step1:定义NAT的外口 / 内口：（同LAB1）

step2:

www server:

```
ip nat inside source static tcp 192.168.1.2 80 13.0.0.100 80
```

FTP server:

```
ip nat inside source static tcp 192.168.1.3 21 13.0.0.100 21  
                                (  私网      )      (  公网      )  
)
```

测试看效果:

因为没有服务器，所以看效果用以下命令

```
ip nat inside source static tcp 192.168.1.2 23 13.0.0.100 80  
ip nat inside source static tcp 192.168.1.3 23 13.0.0.100 21
```

测试：在R5上，telnet 13.0.0.100 80,就能telnet server2了。

在R5上，telnet 13.0.0.100 21,就能telnet server3了。

~~~~~  
~~~~~

远程

day-1

## RA概述： Remote access

广域网的远程连接, 按L1分类:

1:通过电路交换网络实现的专线(circuit switching)

1-1:通过真实的专用物理线路, 实现的专线:

一般是通过同步串行链路(V.35)连接的, 其支持的二层封装协议主要

1:通过电路交换网络实现的专线(circuit switching)

1-1:通过真实的专用物理线路,实现的专线:

一般是通过同步串行链路(V. 35)连接的,其支持的二层封装协议主要包括:HDLC/PPP/SLIP

注: HDLC (不推荐使用)

1-2:通过TDM网络(时分多路复用)实现的专线:

典型有:E1

其支持的二层封装协议与物理专线一样.

2. 按需拨号的电路交换网络(On-Demand Circuit Switched)

一般用于网络的链路备份.

常见业务包括:ISDN/PSTN (其中PSTN通常是以异步串行连接)  
(ISDN也可以通过异步串行连接)

ISDN 分为两种业务 (在中国ISDN交换机类型:Basic-net3)

BRI:2B+D

PRI:30B+D

B信道是用户用于传输数据的信道,其带宽为64Kbps

D信道是信令信道,不传输用户数据

BRI的D信道带宽为16kbps

PRI的D信道带宽为64kbps;

PSTN:56kbps (48~52Kbps) 模拟信道

注: 中国使用的是30B+D.

3. 包交换/分组交换(Packet Switching)

在开始传输数据之前,必需事先建立一条VC(虚电路),用于数据包的传输.

通过同步串行链路连接ISP

常见业务:X. 25/FR/ATM

注1: X. 25/ATM已经淘汰. FR目前常用.

注2: 家里上网的端口,上行的时候是独立的.

4. 宽带接入/Broadband Access

xDSL:主要是传统的电信运营商所经营的网络

主要使用传统的固话网(双绞线),作为最后一公里的末端接入  
(语音+Data)

包括: ADSL, GDSL, 等等.

Cable:主要是传统的有线电视运营商所经营的网络

主要使用传统的CATV网络,经过线路的双向改造,作为最后一公里的末端接入.

(视频+Data)

注: 家里上网的端口,上行的时候是并不是独立的. 整个小区上网,可能都是一条上行线路.

无线宽带:CDMA/GPRS/PHS/3G

小灵通

深圳市的市话通是用的CDMA,并不是PHS.

Powerline Modem: (电能+Data)  
注意: powerline西门子大力支持运作.

被板模块:  
NM: 网络模块            WIC: 广域网接口卡 WAN interface card  
注: cisco的模块槽号定义顺序是: 从下到上, 从右到左. 分别:  
slot0, slot1, slot2, slot3.

ISDN: u---NT1---S/T  
AUI接口: 可以用作以太网口.  
E1接口外观和AUI接口外观完全一样, 是 15 针的.  
FLASH CF:

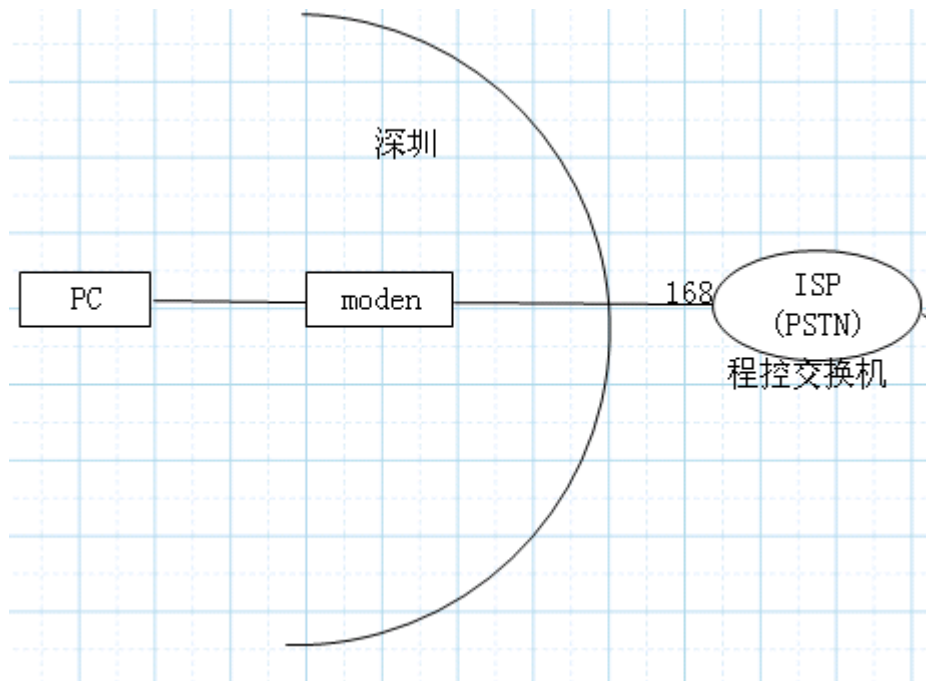
工程故事: ISDN口和以太网口相连接, 正好ISDN拨号来了, 电压达到 90V 以上, 烧了以太网口.

~~~~~  
~~~~~

## 带外网管

modem connections:

LAB1: 通过异步MODEM/ (PSTN程控交换机 / 语言交换网 / 7 号信令网 / SS7), 远程控制路由器的AUX口:  
(带外网管)



LAB1:通过异步MODEM/(PSTN 程控交换机/语言交换网/7号信令网/SS7),  
远程控制路由器的AUX口:

step1:当人还在北京的机房中时,在路由器的AUX进行基本的配置:  
(使用Console线控制Console口)  
(目的:用于控制路由器的AUX口和MODEM之间的通信)

1-1:(察看路由器的AUX口的绝对线路号)

```
BJ-2610#sh line
```

```
  Tty  Typ
    0   CTY(Console口)
```

64 (2个4\*8异步串口模块/32AS) (26的路由器有两个slot,一个  
slot 串口,即八爪鱼线)

```
65  AUX
```

```
66  VTY0
```

```
67  VTY1
```

```
68  VTY2
```

```
69  VTY3
```

```
70  VTY4
```

(绝对链路号)(相对链路号)

```
BJ-2509#sh line
```

```
  Tty  Typ
```

0 CTY(Console口)

1--8 TTY(1个8异步串口)

9 AUX

10 VTY0

11 VTY1

12 VTY2

13 VTY3

14 VTY4

(绝对线路号)(相对线路号)

1-2:进入AUX口

BJ-2610(config)#line 65 (65是绝对线路号)

BJ-2610(config)#line aux 0(相对线路号)

1-3:配置AUX口:

BJ-2610(config)#line 65

or BJ-2610(config)#line aux 0

BJ-2610(config-line)#flowcontrol hardware (硬件流控)

BJ-2610(config-line)#transport input all (在接口中传输所有协议)

BJ-2610(config-line)#modem inout (可以进行入/出方向的呼叫)

BJ-2610(config-line)#password 123 (从AUX口进入路由器的密码)

BJ-2610(config-line)#login (登陆,默认就有)

1-4:配置进入特权模式的密码:

BJ-2610(config)#enable password wolf

step2:使用AT命令集,配置MODEM:

(笔记本电脑的COM口,通过console线,RJ-45 转 DB-25 (or DB-9) 转换器,连接到MODEM的DB-25 (or DB-9) 接口)

每个的MODEM厂商的AT命令集都可能不一样,需要查询文档.)

然后通过虚拟终端软件(超级终端/Secu-CRT)

1:

at (输入) (贾雷注: a t 大意应该是:attention.

注意,香港电影警察喊 " 立正 ")

OK (输出)

2:

at&v (查看MODEM的当前配置)

3:

at&f (将MODEM恢复出厂的默认配置)

at&f1 (每个MODEM厂商的AT命令都有可能不一样)

OK

4: 自动应答(autoanswer/AA)

(S0寄存器=0:表示不应答,是默认值)

ats0=3(将MODEM的自动应答的响铃次数,从0设置为3) 贾雷注:cisco建议是1 工程实践不要用1,用2or3,接的太快会断.

OK

5:

at&w (保存配置)

OK

6:

ate0 (关闭字符回显)(默认)

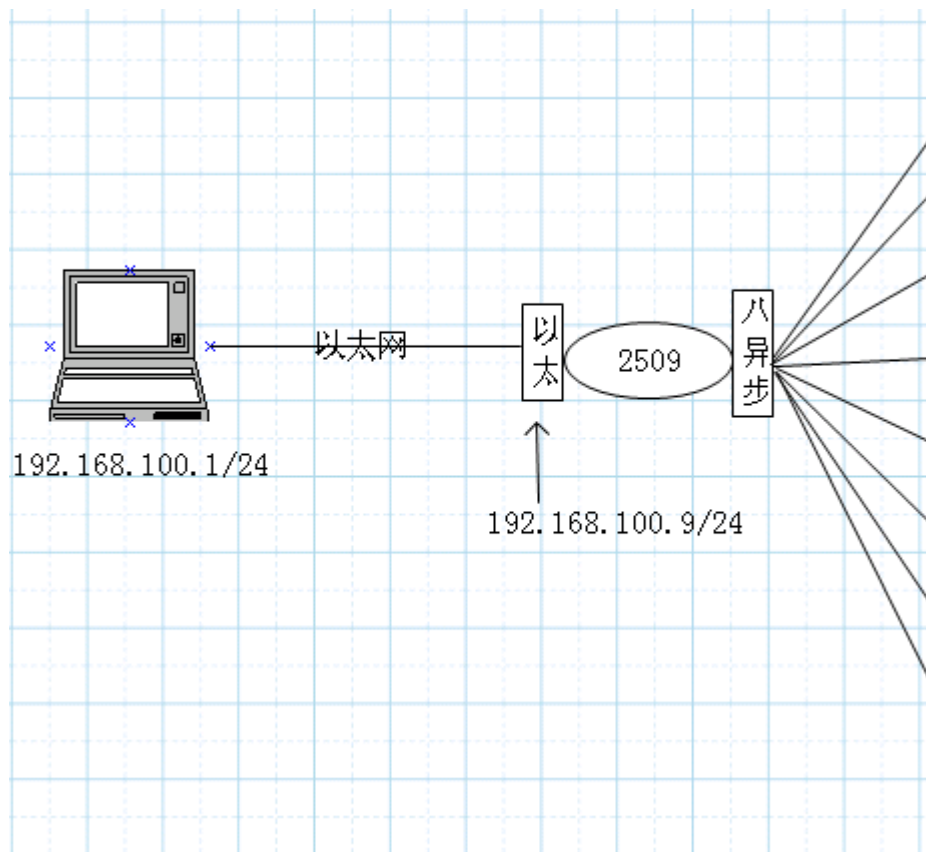
atel (打开字符回显)

step3:按图连接好,路由器AUX-反转线-MODEM-电话线-PSTN网

step4:回到深圳,使用深圳本机的MODEM,拨北京的MODEM  
(010-81012345) (810)

建议使用超级终端,通过长途电话,拨北京的电话号号码,控制路由器,获得和Console相同的权限.

~~~~~  
~~~~~



LAB2:终端服务器

通过telnet到终端服务器的8异步口, 反向telnet到多个网络设备的console口

固定异步串行线路:2509:8个异步口;2511:16个异步口.

模块化的AS Network Module:26\*\*/36\*\*系列.

step1:使用console线, 配置2509的的以太网口, 并且配置号密码, 便于telnet

```
hostname R2509
!
enable password wolf
!
interface e0
 ip add 192.168.100.9 255.255.255.0
 no sh
!
line vty 0 4
 password 123
 login
!
```

(此时就可以把console线拿掉了, 不需要再用)

step2:在主机上, telnet 192.168.100.9

step3:

```
R2509(config)#line 1 8??????
```

```
R2509(config-line)#transport input all    (传输所有协议)
```

step4:配置用于反向telnet的环回口

```
R2509(config)#int lo 1
```

```
R2509(config-if)#ip add 1.1.1.1 255.255.255.255
```

step5:把八爪鱼线的第8根线, 连到rack01r1的console口, 并为此创建“快捷方式”

```
R2509(config)#ip host rack01r1 2008 1.1.1.1
```

(2008表示第八根线:2000+8, 2000是cisco定义的, 8是第八根线的绝对线路号, 2008就是表示第八根线)

个人理解:此命令即是在键入rack01r1之后, 通过telnet本地环回接口地址1.1.1.1的2008端口, 将向此端口连接的路由器发送反向telnet数据.

step6:

键入“rack01r1”, 就可以进行对本地的1.1.1.1这个设备的2008端口的反向telnet.

另外一种反向telnet的方法:R2509#telnet 1.1.1.1 2008

```
~~~~~
~~~~~
```

欢迎界面:

```
R2509(config)#banner motd &
```

```
Enter text message. end with the character '&'.
```

```
Welcome to W01F CCIE LAB to do ***!
```

```
&
```

```
!
```

&是自己定义的一个不会用到的字符, 用于表示欢迎界面的开始和结束.

```
~~~~~
~~~~~
```

day-2

# HDLC

# PPP

HDLC 一般不推荐的, 原因是有两个:

1. CISCO 的HDLC帧头格式, 携带了一个CISCO的私有位:

其好处: 实现在HDLC的环境中, 支持多协议: IP/IPX/AT (AppleTalk)

其缺点: 只能跟CISCO的设备互通, 不能兼容各厂商设备

(原因: 标准的HDLC只支持单协议:IP, cisco加了私有位后, 可以支持多协议了. 但又不兼容其他厂商了.)

CISCO默认在串口中, 以HDLC为2层封装协议.

2. HDLC协议, 本身不支持认证, 无法保证安全性

建议使用PPP ,

PPP有多种可选模块, 可以提高网络安全性, 提升性能  
(PPP可支持认证)

SLIP, 相当于是PPP前身, 功能单一, 趋向淘汰

在CISCO的设备上, 串行链路默认使用HDLC,

在华为的设备上, 默认使用PPP

~~~~~  
~~~~~

PPP (point-to-point protocol)

PPP是业界开放性的标准, 支持多协议环境, 所有的厂商都可以支持.

HDLC/PPP 的对比:

HDLC 不支持多协议, PPP支持多协议

HDLC 不支持认证, PPP可以支持认证

LCP(link control protocol)

LCP(link control protocol)

负责**对L1**的物理层链路, 进行链路的建立, 控制, 维护,

NCP(network control protocol)

负责对L3的网络层, 向下提供无差别的接口( \*CP, 比如ipcp, ipxcp...)

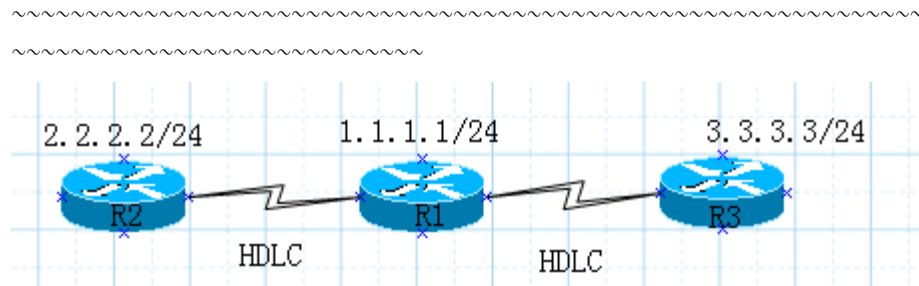
LCP包含了4大模块:

1 authentication (PAP/CHAP) 认证

2 callback 回拨 贾雷注: 便于对方付费的计费

3 compression 压缩

4 multilink 多链路捆绑



默认情况下的接口封装格式:

```
R1#sh int s0
```

```
Serial0 is up, line protocol is up
```

```
Hardware is HD64570
```

```
Internet address is 12.0.0.1/24
```

```
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
```

```
reliability 255/255, txload 1/255, rxload 1/255
```

```
Encapsulation HDLC, loopback not set
```

```
R1#sh int e0
```

```
Ethernet0 is administratively down, line protocol is down
```

```
Hardware is Lance, address is 00e0.1e60.5385 (bia
```

```
00e0.1e60.5385)
```

```
MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
```

```
reliability 255/255, txload 1/255, rxload 1/255
```

```
Encapsulation ARPA, loopback not set
```

```
R1#sh int lo 0
```

```
Loopback0 is up, line protocol is up
```

```
Hardware is Loopback
```

```
Internet address is 1.1.1.1/24
```

```
MTU 1514 bytes, BW 8000000 Kbit, DLY 5000 usec,
```

```
reliability 255/255, txload 1/255, rxload 1/255
```

```
Encapsulation LOOPBACK, loopback not set
```

Encapsulation LOOPBACK, loopback not set

LAB1:encapsulation PPP(从HDLC到ppp的迁移)

Step1: 确认L1/L2/L3通达: 全网配置RIPv2

(L2:HDLC) (L3:网络协议/被路由协议 (routed):IP, 路由协议 (routing):RIP)

L1: V.35 的同步串行链路 (一层)

L2: HDLC (二层)

L3: IP/RIP (三层)

R1#show interface serial 1

Serial 1 is up, line protocol is up  
L1 L2

Encapsulation HDLC,

确认 routed 协议:

R2#show ip route rip

3.0.0.0 .....

确认三层联通.

Step2: 将R1-R3之间的链路更改为PPP:

要在R1和R3的接口中:

R1/R3(config-S1/0)#Encapsulation PPP

观察:

R1#debug ppp negotiation (PPP的协商)

1. Interface serial 1, changed state to up (L1 up)

2. LCP: state is open

3. PPP的认证: (这是可选项目, 如果进行认证, 就必需成功, 才有NCP的工作)

4-1: sel IPCP: state is open (IP)

4-2: sel CDPCP: state is open (CDP) (show cdp neighbors)

5: line protocol on interface serial 1, changed state to up (L2 up)

封装过程:

R3#debug ppp negotiation

```
R3#debug ppp negotiation
PPP protocol negotiation debugging is on

R3#
03:42:05: Se0 PPP: Treating connection as a dedicated line
03:42:05: Se0 PPP: Phase is ESTABLISHING, Active Open [0 sess,
0 load]
03:42:05: Se0 LCP: 0 CONFREQ [Closed] id 3 len 10
03:42:05: Se0 LCP: MagicNumber 0xE17BB0DB (0x0506E17BB0DB)
03:42:05: Se0 LCP: I CONFREQ [REQsent] id 13 len 10
03:42:05: Se0 LCP: MagicNumber 0xE0EC0D9A (0x0506E0EC0D9A)
03:42:05: Se0 LCP: 0 CONFACK [REQsent] id 13 len 10
03:42:05: Se0 LCP: MagicNumber 0xE0EC0D9A (0x0506E0EC0D9A)
R3#
03:42:07: Se0 LCP: TIMEout: State ACKsent
03:42:07: Se0 LCP: 0 CONFREQ [ACKsent] id 4 len 10
03:42:07: Se0 LCP: MagicNumber 0xE17BB0DB (0x0506E17BB0DB)
03:42:07: Se0 LCP: I CONFREQ [ACKsent] id 14 len 10
03:42:07: Se0 LCP: MagicNumber 0xE0EC0D9A (0x0506E0EC0D9A)
03:42:07: Se0 LCP: 0 CONFACK [ACKsent] id 14 len 10
03:42:07: Se0 LCP: MagicNumber 0xE0EC0D9A (0x0506E0EC0D9A)
03:42:07: Se0 LCP: I CONFACK [ACKsent] id 4 len 10
03:42:07: Se0 LCP: MagicNumber 0xE17BB0DB (0x0506E17BB0DB)
03:42:07: Se0 LCP: State is Open
03:42:07: Se0 PPP: Phase is UP [0 sess, 0 load]
03:42:07: Se0 IPCP: 0 CONFREQ [Closed] id 2 len 10
03:42:07: Se0 IPCP: Address 13.0.0.3 (0x03060D000003)
03:42:07: Se0 CDPCP: 0 CONFREQ [Closed] id 2 len 4
03:42:07: Se0 IPCP: I CONFREQ [REQsent] id 1 len 10
03:42:07: Se0 IPCP: Address 13.0.0.1 (0x03060D000001)
03:42:07: Se0 IPCP: 0 CONFACK [REQsent] id 1 len 10
03:42:07: Se0 IPCP: Address 13.0.0.1 (0x03060D000001)
03:42:07: Se0 CDPCP: I CONFREQ [REQsent] id 1 len 4
03:42:07: Se0 CDPCP: 0 CONFACK [REQsent] id 1 len 4
03:42:07: Se0 IPCP: I CONFACK [ACKsent] id 2 len 10
03:42:07: Se0 IPCP: Address 13.0.0.3 (0x03060D000003)
03:42:07: Se0 IPCP: State is Open
03:42:07: Se0 CDPCP: I CONFACK [ACKsent] id 2 len 4
03:42:07: Se0 CDPCP: State is Open
03:42:07: Se0 IPCP: Install route to 13.0.0.1
R3#
03:42:08: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial0, changed state to up
```

```
03:42:07: Se0 CDPCP: I CONFACK [ACKsent] id 2 len 4
03:42:07: Se0 CDPCP: State is Open
03:42:07: Se0 IPCP: Install route to 13.0.0.1
R3#
03:42:08: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial0, changed state to up
```

只要二层PPP封装成功, 两台路由器在路由表里就会自动生成一条32位的主机路由

```
R1#sh ip rou
      13.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C      13.0.0.0/24 is directly connected, Serial1
C      13.0.0.3/32 is directly connected, Serial1
```

32位的主机路由解决了很多二层的封装的问题, 它确定的指定了网络中的某一个点, 当二层封装出问题, 仍然可以通过32位的主机路由到达对方路由器.

检查:

```
R1#sh int s1
Serial1 is up, line protocol is up
  Hardware is HD64570
  Internet address is 13.0.0.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Open
  Open: CDPCP, IPCP, loopback not set
```

```
int s1
no peer neighbor-route
通过此命令不显示32位主机路由;
```

```
~~~~~
~~~~~
```

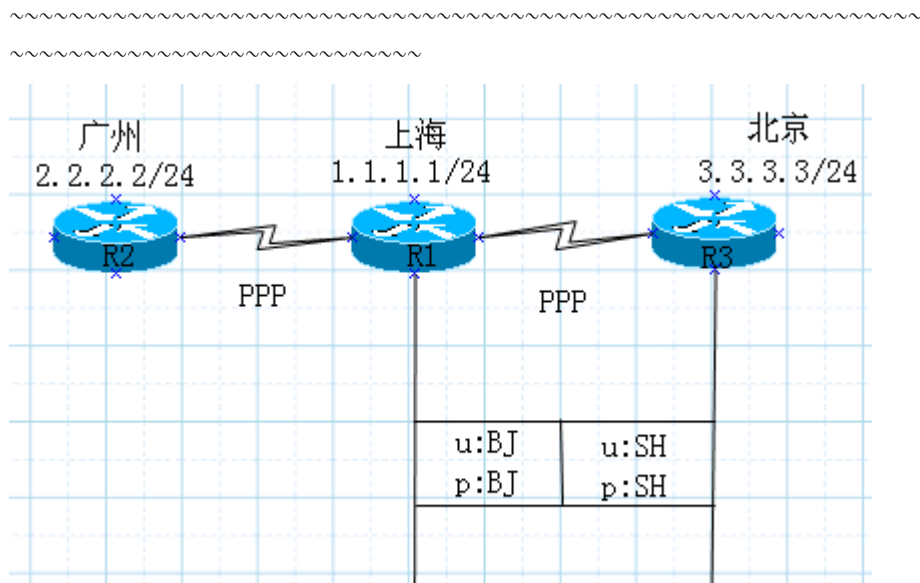
PAP/CHAP (PPP的认证, 是链路的认证)

**PAP** (Password Authentication Protocol)  
两次握手, 建议在网络工程中都使用双向认证

两次握手:

1. 被认证方, 将对方所定义的帐号/密码, 以明文方式, 发送给主认证方
2. 主认证方, 把收到的帐号/密码, 与自己数据库进行核对后, 发回认证成功与否的信息

PAP缺点: 帐号/密码以明文方式在链路上传输, 不安全



LAB2: PAP 认证(R1-R3)全网RIP V2

Step1: 确认链路已经是封装为PPP链路

Step2: 在本路由器的数据库中, 为对方构建帐号/密码:

(贾雷注: 为对方构建帐号, 相当于给你的朋友开一个FTP下载, 要给他建账号.)

R1(config)#username BJ password BJ

R3(config)#username SH password SH

Step3: 选定PPP的认证方式为: PAP

在R1/R3接口上: (建议之前先把接口shutdown, 修改后再no shutdown)

R1/R3(config-if)#PPP authentication PAP

Step4: 将 "自己在对方数据库中的" 帐号/密码, 发送给对方, 供对方进行校验

在R1的S1接口上:

R1(config-if)#ppp pap sent-username SH password SH

在R3的S0接口上:

R3(config-if)#ppp pap sent-username BJ password BJ

观察:

R3#debug ppp authentication (ppp的认证)

R1#debug ppp authentication

PPP authentication debugging is on

R1

\*Mar 1 04:28:22.442: Se1 PPP: Using default call direction

\*Mar 1 04:28:22.442: Se1 PPP: Treating connection as a  
dedicated line

\*Mar 1 04:28:22.446: Se1 PPP: Authorization required

\*Mar 1 04:28:22.454: %LINK-3-UPDOWN: Interface Serial1,  
changed state to up

R1#

\*Mar 1 04:28:22.462: Se1 PAP: Using hostname from interface  
PAP

\*Mar 1 04:28:22.466: Se1 PAP: Using password from interface  
PAP

\*Mar 1 04:28:22.466: Se1 PAP: 0 AUTH-REQ id 2 len 10 from  
"sh"

\*Mar 1 04:28:22.470: Se1 PAP: I AUTH-REQ id 2 len 10 from  
"bj"

\*Mar 1 04:28:22.474: Se1 PAP: Authenticating peer bj

\*Mar 1 04:28:22.490: Se1 PAP: I AUTH-ACK id 2 len 5

\*Mar 1 04:28:22.502: Se1 PPP: Sent PAP LOGIN Request

\*Mar 1 04:28:22.510: Se1 PPP: Received LOGIN Response PASS

\*Mar 1 04:28:22.522: Se1 PPP: Sent LCP AUTHOR Request

\*Mar 1 04:28:22.530: Se1 LCP: Received AAA AUTHOR Response  
PASS

\*Mar 1 04:28:22.534: Se1 PAP: 0 AUTH-ACK id 2 len 5

R1#

\*Mar 1 04:28:23.534: %LINEPROTO-5-UPDOWN: Line protocol on  
Interface Serial1, changed state to up

R1#

R3#

R3#

04:25:50: %SYS-5-CONFIG\_I: Configured from console by console

04:25:51: %LINK-3-UPDOWN: Interface Serial0, changed state to  
up

R3#

04:25:51: Se0 PPP: Treating connection as a dedicated line

```
state to up
R3#
04:25:51: Se0 PPP: Treating connection as a dedicated line
04:25:51: Se0 PAP: O AUTH-REQ id 2 len 10 from "bj"
04:25:51: Se0 PAP: I AUTH-REQ id 2 len 10 from "sh"
04:25:51: Se0 PAP: Authenticating peer sh
04:25:51: Se0 PAP: O AUTH-ACK id 2 len 5
04:25:51: Se0 PAP: I AUTH-ACK id 2 len 5
R3#
```

~~~~~  
~~~~~

LAB3:使用“主机名”作为“用户名”的PAP认证

先shutdown接口,以求稳定.

```
step1:
R1(config)#username R3 password R3
R3(config)#username R1 password R1
```

```
step2:
在R1的S1接口上:
R1(config-if)#ppp pap sent-username R1 password R1
```

```
在R3的S0接口上:
R3(config-if)#ppp pap sent-username R3 password R3
```

~~~~~  
~~~~~

PPP CHAP(challenge handshake authentication protocol)

3 次握手:(发起挑战的是主认证方,回应的是被认证方)

1. 主认证方的路由器,发出随机数(X)
2. 被认证方的路由器,将接收的随机数,和事先定义好的密码,一起放入MD5加密器,进行HASH 算法加密,把得到的数值Y=49,以response 的形式,发送给主认证方
3. 主认证方,同样进行与第2步相同的操作,将得到的数值Y,与从被认证方发来的Y,进行比较

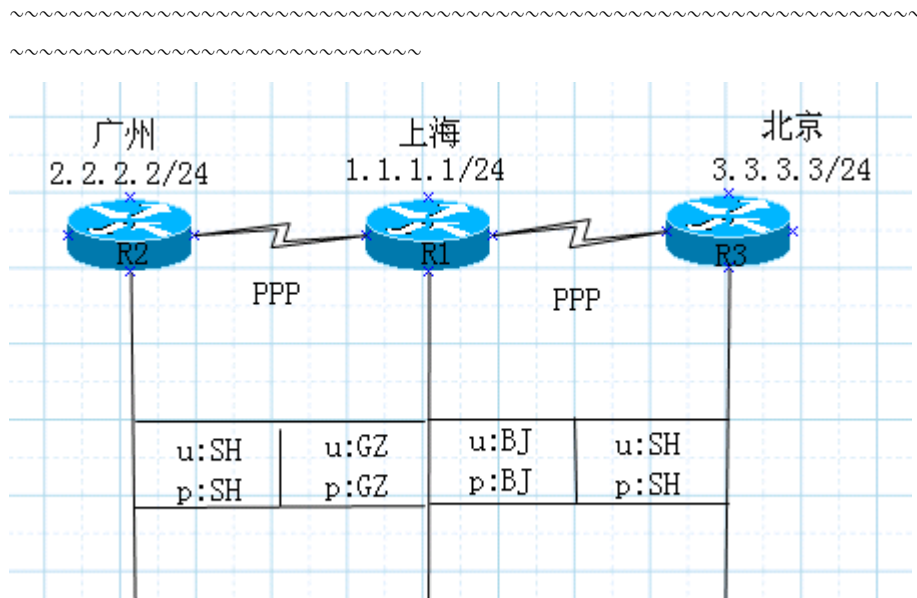
证方发来的Y, 进行比较  
 如果一致, 发出认证成功信息  
 如果不一致, 发送认证失败信息.

贾雷注: 马老师举生动例子说明: 2 个朋友 Q Q 聊天, 为避免对方的老婆冒名顶替, 双方约定一个密码pwd和算法 (比如  $(\text{pwd}+x)$  乘以 2 ).  
 当要聊天的时候开始认证:

1. 主认证方发出随机数 x
2.  $(\text{pwd}+x) * 2 = Y_1$ , 把 Y 1 传给主认证方.
3. 主认证方, 同样用  $(\text{pwd}+x) * 2 = Y_2$ , 然后比较 Y1和Y2是否一致.

CHAP的优点:

从不在链路传送密码, challenge(X) 和Response(Y) 都是随机数, 这两者间是不可逆运算, 可以确保密码不被破译, 保证网络的安全性



LAB3: CHAP 认证:

Step1: 确认链路已经是封装PPP链路

Step2: 为对方建帐号/密码:

R1(config)#username GZ password G-s B- (此处密码不一致, 将导致链路无法正常建立)

R2(config)#username SH password G-s B- (此处密码不一致, 将导致链路无法正常建立)

Step3: 选定认证方式是CHAP:

在R1/R2接口上:

```
R1/2(config-if)#ppp authentication chap
```

Step4: 选定某组帐号密码, 进行CHAP认证:

```
R2(config-if)#ppp chap hostname GZ
```

```
R2(config-if)#ppp chap password GZ
```

```
R1(config-if)#ppp chap hostname SH
```

```
R1(config-if)#ppp chap password SH
```

这样子做, 在debug ppp authentication会观察到认证失败, 原因是密码不一致!!!

Step5: 在CHAP中, 密码必须一致:

```
R1(config)#username GZ password SS
```

```
R2(config)#username SH password SS
```

Step6: 不使用特定的帐号, 而直接使用路由器的主机名, 进行CHAP认证:

```
R1(config)#username R2 password SS
```

```
R2(config)#username R1 password SS
```

在PPP接口中, 只需要以下命令:

```
int s0
```

```
encapsulation ppp
```

```
ppp authentication chap
```

在R1/R2上: show running-config

可以查看到以下信息:

```
Interface serial 0
```

```
Encapsulation ppp
```

```
Ppp authentication chap
```

默认情况下, 在没有设置其他帐户和密码时, chap协议时默认发主机名

LAB3: 使用“主机名”作为“用户名”的PAP认证

先shutdown接口, 以求稳定.

step1: 直接使用路由器的主机名, 进行c h a p 认证:

```
R1(config)#username R2 password R12
```

```
R2(config)#username R1 password R12
```

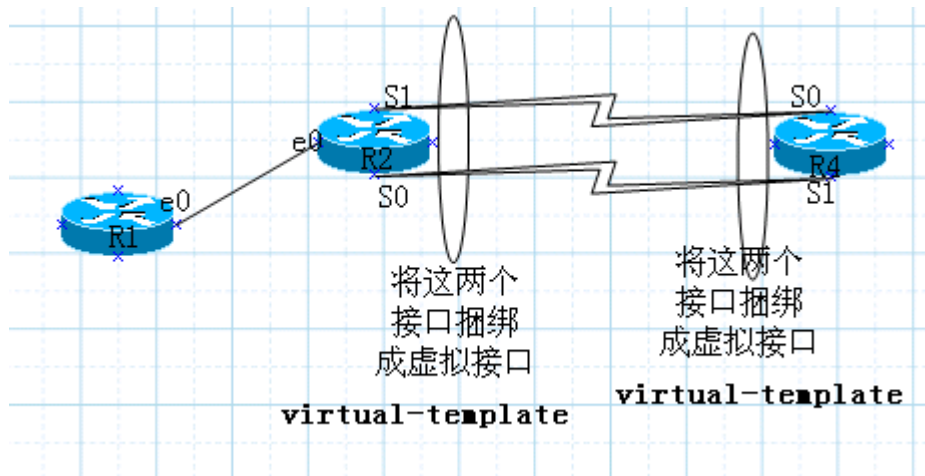
step2: 在ppp接口中, 只需要以下命令.

```
Interface serial 0
```

```
Encapsulation ppp
```

```
ppp authentication chap
```

## PPP multilink



PPP/MLP(multilink Protocol) (2层冗余)

对比实验(3层冗余) (RIP)

(收敛速度受路由协议的收敛速率影响, 通常收敛较慢)

通过 Multilink protocol 实现2层冗余:

Step1:

将冗余的物理链路上的接口, 原有配置都删除,  
但注意要在R2的两个接口都是DCE端, 需要配置同步时钟

因为:

```
R2#sh controllers serial
```

```
HD unit 0, idb = 0x939294, driver structure at 0x940860
```

```
buffer size 1524 HD unit 0, V.35 DCE cable, clockrate 64000
```

```
cpb = 0xE1, eda = 0x5078, cda = 0x508C
```

```
RX ring with 16 entries at 0xE15000
```

Step2:

(R2/R4同时做) 4个物理接口都封装PPP, 并且运行Multilink (无需配置IP地址)

```
Interface S0/S1:
```

```
Encapsulation ppp
```

```
Encapsulation ppp
Ppp multilink
```

Step3: 在双方路由器上, 创建虚拟模板接口, 配置地址, 指定MLP

```
R2(config)#
Interface virtual-template 1
 ip add 24.0.0.1 255.255.255.252
 ppp multilink
```

```
R4(config)#
Interface virtual-template 1
 Ip add 24.0.0.2 255.255.255.252
 ppp multilink
```

Step4: 在MLP中, 调用虚拟模板:

```
R2/R4(config)#multilink virtual-template 1
```

设置虚拟接口后, 带宽加倍:

```
R2#show interfaces virtual-access 1
 Interface address is 24.0.0.1/30
           BW 3088 Kbit,
R3#show interfaces virtual-access 1
 Interface address is 24.0.0.2/30
           BW 3088 Kbit,
```

贾雷注: 2条链路带宽加宽, 互为冗余.

~~~~~  
~~

EC, FEC, GEC

L A B: 交换机之间的以太网链路二层冗余:

EtherChannel:

logical aggregation viewed as one logic port
switch-level load balancing & redundancy.

```
interface fastethernet0/21/22
duplex full
speed 100
channel-group 1 mode on
```

交换机自动生成:

```
interface port-channel 1
```

```
SW2(config)#show interface port-channel 1
BW 2000000 kbit,
show spanning-tree
po1      root FWD 12      128.65 p2p
```

~~~~~  
~~~~~

可以在接口中, 关闭PPP的主机路由(host route)

```
R2# C      12.0.0.1/32 is directly connected , Serial 0
```

```
Interface serial 0
No peer neighbor-route
```

~~~~~  
~~~~~

ISDN(不考)

ISDN

~~~~~  
~~~~~

1. ISDN主要有两种接口类型:BRI & PRI
2. BRI:是2B+D (B信道:64kbps, D信道:16kbps)
3. PRI:中国的PRI是基于E1, 是30B+D (B信道:64kbps, D信道:64kbps)

ISDN的交换机类型:

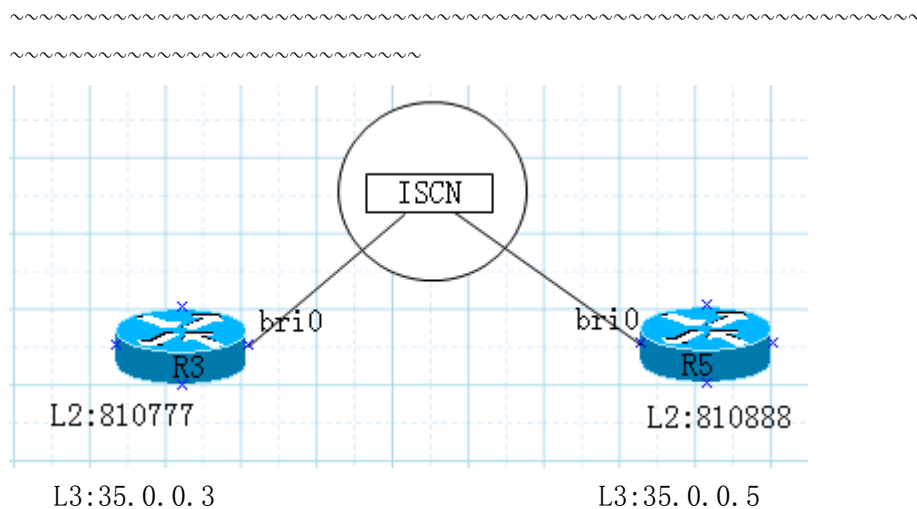
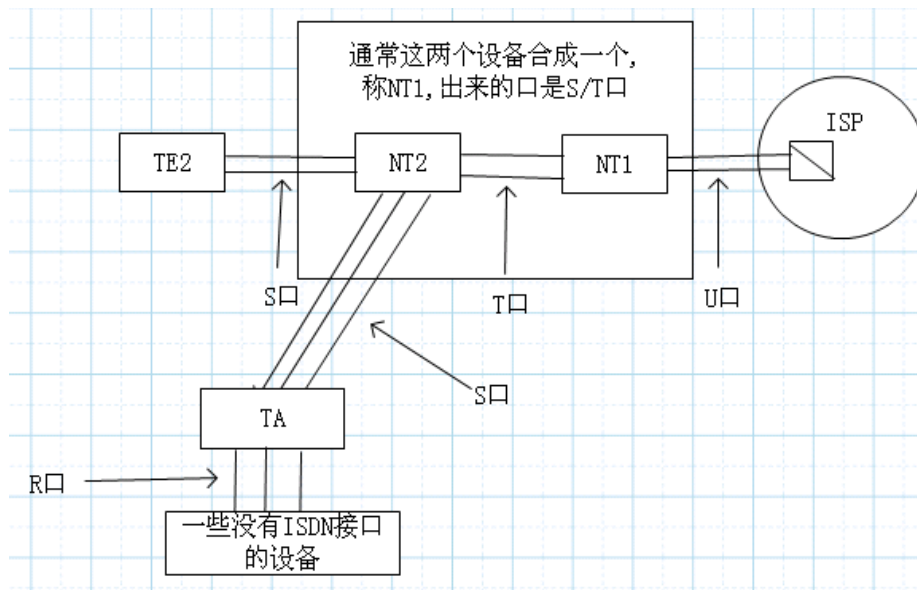
isdn switch-type basic-net3 (中国标准, 欧洲标准)

isdn switch-type basic-ni (wolf实验室, 北美标准)

ISDN参考点:

R-->S--->T--->U--->V

ISDN协议栈	D	B
L3	Q.931	2P
L2	Q.921	HDLC/PPP
L1	I.430/I431	



LAB1:

Step1: L1/L2通:

R3(config)#isdn switch-type basic-ni (此命令在全局配, 在接口会自动继承此命令, 即也有此命令)

R3(config)#iin bri0

R3(config-if)#no shutdown

检查:

R3#show isdn status

Layer 1 status:

ACTIVE

Layer 2 status:

State = MULTIPLE_FRAME_ESTABLISHED 注:如果没有这个标识位,

说明此接口和ISDN交换机没有协商成功.

多 帧

已建立

L2层测试: (拨通对方路由器的电话号码(L2地址))

R3#isdn (test) call interface bri 0 810888 (拨号测试)

R5#isdn (test) disconnect interface bri 0 all (断开ISDN的连接)

R3#show isdn active (察看ISDN当前连接状态)

会看到在 R 3 有out 的电话号码810888显示, R 5 有 i n 的电话号码810777显示. 当然如果断开ISDN, 则没有显示了.

step2:L3通(L3网络协议 IP)

所有的配置命令, 都在ISDN接口中:

R3#

物理命令:

2-1:isdn switch-type basic-ni(接口会自动继承全局的配置)

2-2:encapsulation hdlc(默认的二层封装是HDLC)

2-3:SPID号, 相对于是ISP提供一种服务的标识位(中国不需要)
(if#isdn spid1 81077701)

逻辑命令:

2-4 Ip address 35.0.0.3 255.0.0.0

2-5 dialer map ip 35.0.0.5 (broadcast) 810888
对方的L3地址 与 对方的L2地址 的映射

2-6: 定义” 能够触发ISDN起拨的” 数据流:

2-6-1:使用 A C L, 定义可以触发 I S D N起拨的数据包:

R 3 (config) # access-list 1 permit any (定义任何数据包)

2-6-2:在dialer-list 3中, 调用 A C L 1:

R3(config)#dialer-list 3 protocol ip list 1

凡是IP数据包, 都能够触发ISDN起拨

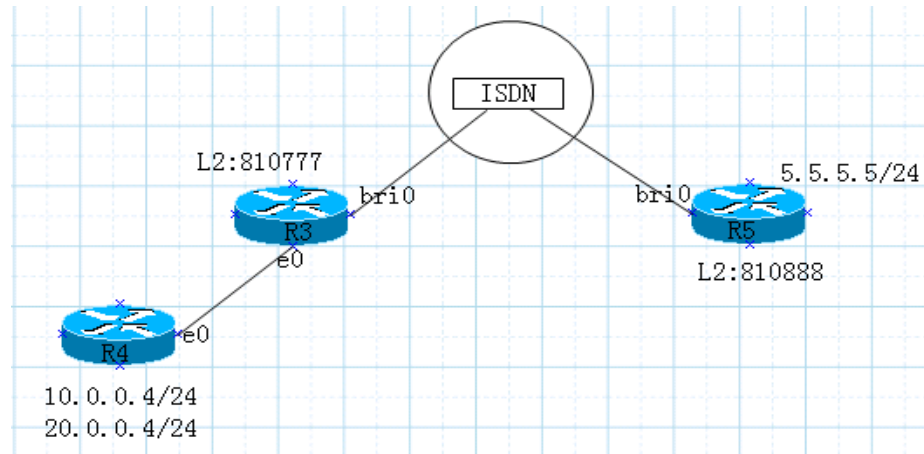
2-6-3:在 I S D N接口中, 调用dialer-list 3

R3(config)#int bri0 (在接口中调用 dialer-list 3)

2-6-3:在 I S D N接口中, 调用dialer-list 3
R3(config)#int bri0 (在接口中调用 dialer-list 3)
R3(config-if)#dialer-group 3

L3测试

Ping 35.0.0.5 !!!!!!!!!!!!!!!



Step3: DDR/按需拨号 (dial on demand route)

3-1: 确保两端路由器, 来去都有正确路由:

R4(config)#ip route 5.5.5.0 255.255.255.0 34.0.0.3

R3(config)#ip route 5.5.5.0 255.255.255.0 35.0.0.5

R3(config)#ip route 10.0.0.0 255.255.255.0 34.0.0.4

R3(config)#ip route 20.0.0.0 255.255.255.0 34.0.0.4

R5(config)#ip route 10.0.0.0 255.255.255.0 35.0.0.3

R5(config)#ip route 20.0.0.0 255.255.255.0 35.0.0.3

3-2: 确认上述路由的下一跳的可达性:

(确认有到达”路由的下一跳”的映射)

R3#dialer map ip 35.0.0.5 810888

R5#dialer map ip 35.0.0.3 810777

Show dialer maps (察看”路由的下一跳”的映射)

附加:

1察看 Frame-Relay的的映射的命令: show frame-relay map

2察看ether的的映射的命令:show arp

3-3: 修改”能够触发ISDN起拨的”数据流:(定义”感兴趣流”)
收窄”能够触发ISDN起拨的数据包的”范围

以下命令都在R3上执行:

3-3-1: 通过ACL, 定义能够触发ISDN起拨的用户/数据包:

Access-list 10 permit 10.0.0.0 0.255.255.255

(ACL列表后的如果只写:access-list 10 permit 10.0.0.0

等价于:access-list 10 permit 10.0.0.0 0.0.0.0 即精确匹配)

3-3-2: 通过dialer-list3, 调用ACL10

Dialer-list 3 protocol ip list 10

3-3-3: 在接口中, 调用dialer-list 3

Interface bri 0

Dialer-group 3

ISDN的idle time 是只能让”能够触发ISDN起拨的”数据流, 去复位的
interesting traffic

即使是在ISDN上通信的, 非”能够触发ISDN起拨的”数据流, 是不能让
idle time 复位

DDR行为规则		之前	之后	
能自动起拨的	10.0.0.0	断	通	第2次实验
		通	通(此时会reset idle time)	第3次实验
不能自动起拨的	20.0.0.0	断	断	第1次实验
		通	通(即使通,idle time没复位)	第4次实验 idle time一直增长到120s, 就不通了

```
R3/r5(config-if)#encapsulation ppp
```

4-2: 进行CHAP认证

R5#

```
Username R3 password 0 R35
```

```
Interface BRI 0
```

```
 Ppp authentication chap
```

step5:根据实际需求, 启用PPP的增强功能

5-1:PPP multilink

(可以将二个信道捆绑在一起, 成为一个128kbps的信道)

```
R3(config-if)#ppp multilink load-threshold 165 (255*0.65=165, 255是最大值)
```

(链路负荷/利用一个B信道的65%时, 启动第二B信道)

贾雷注:

1) 察看 D 信道

```
show interface bri 0
```

```
bri 0 is up, line protocol is up (spoofing) D信道一直是 u p
```

的, 因为要和 I S D N 交换机协商

```
interface bri0 D信道 (interface b)
```

2) 察看第一 B 信道

```
show interface bri 0:1
```

```
interface bri0:1 第一个 B 信道
```

3) 察看第二 B 信道

```
show interface bri 0:2
```

```
interface bri0:2 第一个 B 信道
```

5-2:PPP的压缩

在R3-R5的PPP链路, 的接口中, 启动PPP压缩(3选1)

```
(config-if-b0)#compress mppc
```

```
(config-if-b0)#compress predictor
```

```
(config-if-b0)#compress stac
```

TCP头压缩: (语音的数据包, 其数据净荷很小, 头大身小情况)

```
(config-if-b0)#ip tcp header-compression
```

5-3:PPP Call-back/回拨 (回拨主要的目的是, 决定那乙方付拨号费用, 类似于挂调电话打回去的例子)

5-3-1:

```
R3(config-if)#ppp callback request (R3请求) 在callback client  
端只需要配置这一句.
```

5-3-2:

```
R5(config-if)#ppp callback accept (R5接受)
```

5-3-3:

```
R5(config-if)#dialer callback-secure
```

5-3-4:

```
R5(config)#map-class dialer CB
```

```
R5(config-map-class)#dialer callback-server username
```

5-3-5:

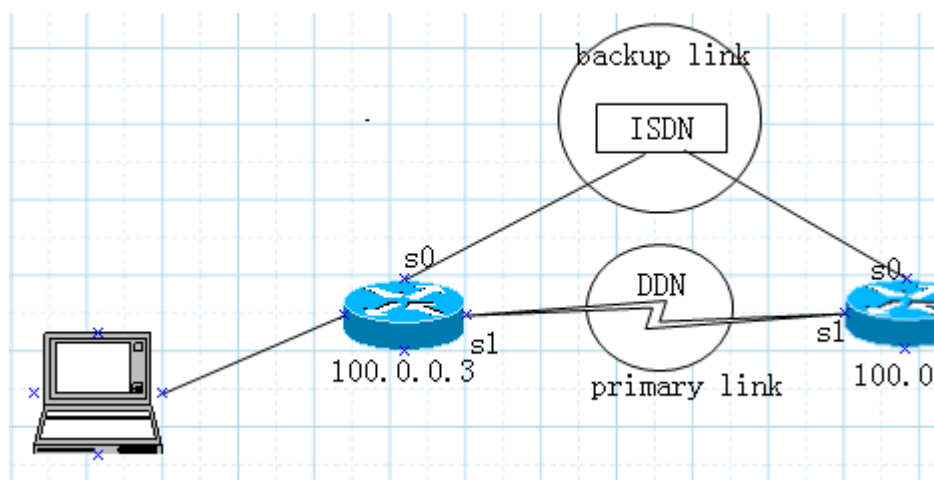
```
R5(config-if)#dialer map ip 35.0.0.3 name R3 class CB  
broadcast 810777
```

```
R3(config-if)#dialer map ip 35.0.0.5 name R5 broadcast 810888
```

测试:

debug isdn Q931(观察ISDN连接的过程)

贾雷注: 工程实践中, 可以用一种更简单的方法实现: 从一个固定单方向起拨, 可以只让一放掏钱. 呵呵. 方法就是只在server端建立映射, 不在client端建立映射. 例如: R5#dialer map ip 35.0.0.3 810777



step6: 使用ISDN做备份:

6-1:

```
router eigrp 90  
net 0.0.0.0
```

6-2:

```
R3/R5(config)#dialer-list 5 protocol ip permit
```

6-3:

```
R3(config-if-s1)#backup interface bri 0
```

定义ISDN是: 主链路S1口的备份接口

```
R3(config-if-s1)#backup interface bri 0
定义ISDN是:主链路S1口的备份接口
```

ISDN接口成为主链路的备份接口后,接口立刻down
(standby mode).

```
R3#sh isdn static
L1 DEACTIVE.
```

.....

```
R3#sh int bri 0
interface is standby mode ,line protocol is down.
```

step7:使用虚拟的“dialer”接口,做备份:(dial-profile)

用虚拟接口的目的: 一个bri0,想同时给多条D D N作备份. 如果用物理接口只能给一个物理接口作备份.

7-1:物理接口:

```
int bri0
encapsulation ppp
dialer pool-member 10 (将物理接口放入dial-pool 10中)
isdn switch-type basic-ni
```

7-2:“Dial”接口:(逻辑接口)

```
int dialer 5
ip add 35.0.0.3 255.255.255.0
en ppp
dialer pool 10 (dial 5口在dial-pool 10中调用物理口)
```

dialer string 810888 (原来接口的命令是dialer map ip 35.0.0.5 810888,这里形式要变化)

```
dialer-group 3
ppp authentication chap
```

附加:

Dial”接口:(另一个逻辑接口)

```
int dialer 9
ip add 200.0.0.1 255.255.255.0
en ppp
dialer pool 10 (dial 5口在dial-pool 10中调用物理口)
```

dialer string 029 2222 (原来接口的命令是dialer map ip 35.0.0.5 810888,这里形式要变化)

```
dialer-group 3
ppp authentication chap
```

7-3:

```
R3(config-if-s1)#backup int dialer 5
```

```
附加: R3(config-if-s2)#backup int dialer 9
```

```
R3(config-if)#backup delay 5 15
```

当主链路断开5秒后,才开始拨号;

当主链路恢复15秒后,才断开(shutdown) ISDN

```
R3(config-if)#backup load 50 20
```

当主链路的带宽利用率高于50%,启动ISDN进行负载均衡;

当主链路+备份链路的总带宽利用率低于20%,断开ISDN.

注意: 在真实情况下,应该备份链路的两端都” backup-if”

~~~~~  
~~~~~

day-4未完整听课

FR(帧中继)(帧中继是纯2层协议)

帧中继特征:

1:面向连接的服务:

 必须在通信开始之前,通过VC(虚电路)构建好
 (VC:通信双方之间的虚拟通道)

 PVC:永久虚电路,长期连接不中断的

 SVC:交换式虚电路,在用户需要连接的时候,才临时构建起来的.

2. FR的用户, 是充当FR的DTE端

而ISP的连接用户的边缘设备的接口充当FR的DCE端 (FR cloud)

帧中继交换机之间使用的是NNI接口 (network network interface)

3. FR的信令 / frame relay signaling:

FR用户通过LMI (local management interface)

本地管理接口

跟FR交换机进行协商, 获得PVC的相关信息.

LMI的三种标准: (在IOS12.0以后的版本中, CISCO路由器可以自动检测到ISP所使用的LMI类型)

1. ANSI

2. Q933A

3. CISCO兼容的

(LMI就是工作在帧中继交换机和用户之间的那段网络)

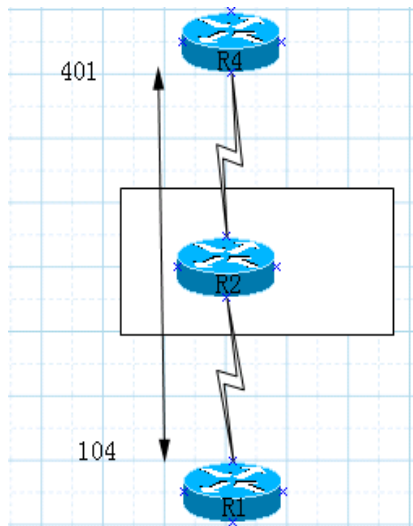
4. 通过LMI的状态, 判断FR PVC的故障点

LMI的状态	Local	Remote
Active	√	√
Inactive	√	X
Delete	X	?

5. FR的L3 IP地址, 与L2的DLCI地址的映射关系:

对方的L3 IP 与本地的 L2的DLCI进行映射

~~~~~  
~~~~~



LAB1:基本的帧中继交换机(基于每个物理连接,一条PVC的HUB&SPOKE架构)

step1:

R29 (config)

no ip routing

frame-relay switching

step2:为V.35 dce接口配置同步时钟

cl ra 2000000

step3:FR接口的基本配置:

interface s0/s1

encapsulation frame-relay(封装FR)

frame-relay lmi-type cisco

frame-relay intf-type dce(FR DCE)

step4:配置FR路由:

FR-SW2 (config-if-s0) #在PVC的入口S0

frame-relay route 104	interface serial 1	401
进入的PVC	出口	出去的

PVC

FR-SW2 (config-if-s1) #在PVC的入口: S1

frame-relay route 401 interface serial 0 104

step5:在用户端封装FR，然后将ISP和用户的端口都打开（L2测试）

```
R4/R1#  
interface serial 0  
encapsulation frame-relay  
no shutdown  
R1/4#show ip int brief  
serial0      L1:up          L2:up
```

```
R1#show frame-relay pvc  
DLCI=104,.....PVC STATUS=ACTIVE, INTERFACE=Serial0
```

step6:L3测试:

在PVC上，用户的接口中，配置同一个网段的IP地址

```
R1#show frame-relay map
```

FR的自动反向ARP:

```
R1#show frame-relay map
```

```
serial0(up):ip 10.0.0.4 dlci 104,
```

对方的IP 本地的DLCI的映射

dynamic:通过FR的自动反向ARP学到的

broadcast:L2的FR PVC允许广播/组播的流量

通过

active : PVC工作状态是OK的

```
R1#ping 100.0.0.4!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

step7:静态映射

先在用户端关闭ARP:

```
R1(config-if)#no frame-relay inverse-arp
```

```
R1#clear frame-relay inarp
```

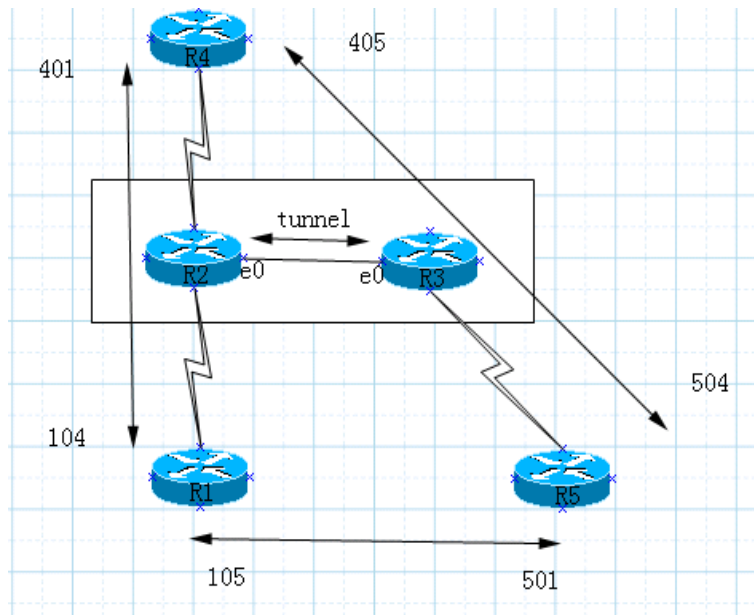
```
R1(config-if)#frame-relay map ip 100.0.0.4 104
```

```
R4(config-if)#frame-relay map ip 100.0.0.1 401
```

```
R4#sh frame-relay map
Serial0 (up): ip 100.0.0.1 dlci 401(0x191,0x6410),
static,
broadcast,
CISCO, status defined, active
```

~~~~~

~~~~~



LAB2:3端口FR

step1--step3同 LAB1;

step4:构建tunnel:

4-1:

确认FR-SW2, FR-SW3的以太网接口:
在以太网接口配置IP地址。

4-2:

```
FR-SW2 (config) #
interface tunnel 1
tunnel source 23.0.0.2
tunnel destination 23.0.0.3
```

```
FR-SW3(config)#
```

```
FR-SW3(config)#  
interface tunnel 1  
tunnel source 23.0.0.3  
tunnel destination 23.0.0.2
```

step5:R4~R5之间的PVC（405/504）的FR路由:

```
FR-SW2 (config-is-s1) #  
frame route 405 int tunnel 1 1000
```

```
FR-SW3 (config-is-s1) #  
frame route 504 int tunnel 1 1000
```

step6:R1~R5之间的PVC（105/501）的FR路由:

```
FR-SW2 (config-is-s0) #  
frame route 105 int tunnel 1 1001
```

```
FR-SW3 (config-is-s1) #  
frame route 501 int tunnel 1 1001
```

FR-SW3#show frame-relay route (在帧中继交换机的路由状态)

~~~~~  
~~~~~

IGP在FR网络的运行:

LAB3:通过FR的自动反向ARP, 构建fullmesh的PVC, IGP网络, (RIP/EIGRP/OSPF)

FULLmesh: 在N个节点的网络中, 任何两个节点间, 都有直接相连的连接.

fullmesh连接, 所需要的PVC条数的计算公式: $N * (N - 1) / 2$

step1:在FR的用户端(CPE), 封装FR, 打开接口:

step2:在CPE端配置IP地址, 观察FR静态映射 / 自动反向ARP, 的L2/L3的映射:

R1#show frame-relay map

step3: 启动IGP (RIP/EIGRP/OSPF)

3-1: R1 over fullmesh FR PVC 网络

3-2: EIGRP over fullmesh FR PVC 网络

R1#router eigrp 100

network 0.0.0.0 (所有接口, 都运行EIGRP)

no auto-summary

3-3: OSPF over fullmesh FR PVC 网络

R1#

router ospf 100

router-id 100.0.0.x

network 0.0.0.0 255.255.255.255 area 0 (所有接口都运行OSPF)

观察:

1: show ip ospf interface

(观察当前运行OSPF的接口有哪些, 并且特别注意: 接口的OSPF运行模式/network type是哪种)

FR主接口默认是NBMA, 默认不主动发送组播的hello包的)

(默认情况下, FR的主接口/multipoint subinterface, 其OSPF的运行模式都是NBMA;

point to point subinterface: 的运行模式都是P2P)

2: show ip ospf neighbor

因为OSPF的运行模式是non_broadcast (nbma), 所以OSPF无法建立邻居

解决方案:

将OSPF的接口运行模式, 从NBMA改为BROADCAST:

R1/R4/R5#

conf t

interface serial0

ip ospf network broadcast

OSPF邻居建立成功

```
ip ospf network point-to-point
```

```
~~~~~  
~~~~~
```

LAB4: 使用物理接口, 通过静态映射构建IGP网络 (RIP/EIGRP/OSPF)

Step1: 关闭FR的自动反向映射: (在CCIE考试中, 必须关闭)
在较早的IOS版本中, 只在主接口(物理接口)中, 有自动反向映射功能
在较新的IOS版本中, 主接口和子接口, 都有自动反向映射功能

```
R1(config-if-S0)#no frame-relay inverse-arp
```

```
R1#clear frame-relay inarp (清除FR map 的Cache)
```

Step2: 在接口中, 根据题目中, 明确允许使用的PVC, 建立FR静态/手工映射:

```
R1#
```

```
Interface serial 0
```

```
Frame-relay map ip 100.0.0.4 104 broadcast ---- (L2  
的概念)
```

注: 这里的broadcast, 决定了广播 / 组播流量, 是否能够穿越pvc

```
Frame-relay map ip 100.0.0.5 105 broadcast
```

Step3: IGP网络(RIP/EIGRP/OSPF)的运行, 与LAB3一致.

```
~~~~~  
~~~~~
```

FR子接口:

子接口判断原则:

1. 判断创建几个子接口: (一个子接口, 对应一个子网)

在路由器的一个物理的FR接口中,
如果对应着几个子网, 就应该创建几个子接口.

2. 判断子接口的类型: (P2P/MP)

如果一个子接口中,对方是多于一个点,那么接口类型是MP;
建议:多点子接口,只适用于full mesh的网络拓扑
(多条PVC,对应一个IP子网)
(相当于一个Mult-Access接口)

如果一个子接口中,对方是只有一个点,那么接口类型是P2P.
建议: (一条PVC,对应一个IP子网)

HUB & Spoke/星型 网络拓扑,有多少个分支点/分公司,就应该创建多少个点对点子接口
相当于一个点对点的串行接口.

~~~~~  
~~~~~  
LAB5:使用多点子接口,替代物理接口,构建IGP网络: 与

LAB4相比工程中推荐使用LAB5

(考虑到网络将来的发展,扩展性) (full-mesh)

多点子接口和物理接口的配置方法完全一致(同LAB4)

主接口配置:

```
int s0  
en fra  
no shut
```

子接口配置:

```
int s0.100 multipoint(子接口)  
ip add 100.0.0.1 255.255.255.0  
fra map ip 100.0.0.4 104 b  
fra map ip 100.0.0.5 105 b
```

~~~~~  
~~~~~  
LAB6: 通过将 HUB & Spoke 的FR网络 使用中推荐使用

每个分支点的PVC作为独立的一个子网/30 (网络位为30位),构建P2P的子接口

(作为HUB & Spoke拓扑的, 最佳解决方案)

Step1: 配置主接口:

R1#

No ip address

Encapsulation frame-relay (无需配置IP)

No frame-relay inverse-arp (关闭FR的自动反向映射)

No shutdown

R5#interface serial0.504

R5(config-subinf)#no frame-relay inverser-arp

Step2: 配置FR的P2P子接口:

R4

Interface serial0.405 point-to-point

Ip address 100.0.0.9 255.255.255.252

no frame-relay inverser-arp

Frame-relay interface-dlci 405

Interface serial0.401 point-to-point

Ip address 100.0.0.5 255.255.255.252

no frame-relay inverser-arp

Frame-relay interface-dlci 401

R1:

int s0.104 point-to-point

ip add 100.0.0.6 255.25.255.252

frame-relay interface-dlci 104

R5:

int s0.504 point-to-point

ip add 100.0.0.10 255.255.255.252

no frame-relay inverser-arp

frame-relay interface-dlci 504

(子接口也需要配置关闭反向ARP的)

Step3: 察看FR的映射:

R1#show fram-relay map

Serial0.104 (up)

(P2P子接口默认是允许广播通过) broadcast

P2P子接口中, ping该子接口所在网段中的所有节点, 都能通
(包括本机节点)

MP子接口中, ping该子接口所在网段中的所有节点, 都必须手
工做映射(FrameRelay map) (包括本机节点)

Step4: IGP over HUB&spoke

4-1: RIP over HUB&spoke点对点子接口, FR网络 (R1/R4/R5
都运行EIGRP协议)

有全路由(Full Roter), 能够全面通达

R4#

router rip

ver 2

net 4.0.0.0

net 100.0.0.0

no auto-summary

4-2: EIGRP over HUB&spoke 点对点网络 (R1/R4/R5都运行
EIGRP协议)

no router rip

router eigrp 100

net 0.0.0.0

no auto-summary

4-3: OSPF over HUB&spoke 点对点网络 (R1/R4/R5都运行
OSPF协议)

no router eigrp 100

router ospf 100

router-id 100.0.0.X

net 0.0.0.0 255.255.255.255 area 0

```
R1#show ip ospf interface serial 0.102/103
      Network type POINT_TO_POINT
```

FR P2P子接口的网络类型, 默认是点对点运行模式

Step5: 关于本实验中的所有的分支点的下一跳:

```
R4#    any Router via 100.0.0.1
```

```
R5#    any Router via 100.0.0.5
```

所以 所有的分支点之间的, 路由的下一跳, 都是可达的

~~~~~  
~~~~~

使用多点子接口 (MutltiPiont Sub-if), 构建HUB&SPOKE FR网络, 之

LAB7: 通过静态映射构建IGP网络: (RIP/EIGRP/OSPF)

贾雷注: 如果工程中用, 用LAB6, 考试一般考LAB7.

Step1: 配置主接口:

```
R4(config)#interface serial 0
      Encapsulation frame-relay
      No frame-relay inverse-arp
      No shutdown
```

Step2: R1/R4/R5创建子接口:

R1

```
Interface serial 0.100 multipoint
  Ip address 100.0.0.1 255.255.255.0
  No frame-relay inverse-arp
  Frame-relay map ip 100.0.0.4 104 broadcast
```

R4

```
Interface serial 0.100 multipoint
  Ip address 100.0.0.4 255.255.255.0
  No frame-relay inverse-arp
  Frame-relay map ip 100.0.0.1 401 broadcast
```

```
Frame-relay map ip 100.0.0.5 405 broadcast
Frame-relay map ip 100.0.0.4 405 broadcast 注：这样
R 4 才能ping通自己的接口100.0.0.4
R5
Interface serial 0.100 multipoint
Ip address 100.0.0.5 255.255.255.0
No frame-relay inverse-arp
Frame-relay map ip 100.0.0.4 504 broadcast
```

Step3: RIP Over FR: (R1/R4/R5运行RIP v2协议)
关于DV协议的水平分割问题: (RIP/IGRP)
现象: 分支点的路由器, 只有中心点的路由, 没有别的分支点路由。

在FR的子接口中, 不论是点对点, 还是多点子接口, 其水平分割都是启动/Enable的

```
R1#show ip interface serial 0.100
Split horizon is enabled
```

1 / 贾雷注: 这里的水平分割的理解是: **R 4 接口 (100.0.0.4)** 从 R 5 收到路由信息, 不会从**同一个接口 (100.0.0.4)** 发出给 R 1.

2 / 贾雷注: D V 协议的水平分割仅仅不会从同一个接口收后再发该路由协议的路由信息, 对于其他数据的收发, 不收水平分割的影响.

3 / 贾雷注: 水平分割设计的目的是防止 D V 协议产生环路, 因为DV协议发送的路由信息是路由条目, 而 L S 协议, 发送的路由信息是泛洪SLA or SLP的, 不担心环路.

4 / 贾雷注: R I P 和 I G R P 的水平是上面说的情形, 对于 E I G R P, 有它专有的水平分割. 后面会讲到.

5 / 贾雷附加: 路由器ping的时候, 选取源地址的原则是: 察看路由表, 递归查找, 出口接口的地址作为源地址.

(而FR物理接口/主接口的水平分割, 默认是关闭的)
(只有FR的主接口默认是关闭水平分割, 其他所有接口都是默认开启的)

现象: 每个分支点的路由器, 只有中心点的路由, 没有别的分支点路由。

解决方案: 关闭中心点路由器接口的水平分割

```
r4#int s0.100
    no ip split-horizon
```

路由下一跳存在不可达问题

```
R5#          1.0.0.0/8 [120/2] via 100.0.0.1
R1#          5.5.5.0 [120/2] via 100.0.0.5
```

Step4: 为了解决路由的下一跳问题, 在所有分支点上做别的分支点的映射

```
R1(config-subif)#frame-relay map ip 100.0.0.5 104
(也可以后面加broadcast参数)
```

```
R5(config-subif)#frame-relay map ip 100.0.0.1 504
(也可以后面加broadcast参数)
```

~~~~~  
~~~~~

使用多点子接口 (MultiPoint Sub-if), 构建HUB&SPOKE FR网络, 之

LAB 8: 通过静态映射构建EIGRP网络

Step1:

EIGRP的水平分割, 与普通的DV 协议是不同的:

```
R4(config-subif)#no ip split-horizon eigrp 90 (注:
这里多了参数, 这是和RIP/IGRP不同的.
```

Step2:

R1/R5两分支点间的用户间的通讯，无需(象LAB 7 最后一步)做映射:

```
R5# 1.0.0.0 [90/2809856] via 100.0.0.4
    (中心点，自动下一跳)
R1# 5.5.5.0 [90/2809856] via 100.0.0.4
```

```
R1 #Ping 5.5.5.5 source 1.1.1.1 !!!!!!!!
R1 #Ping 5.5.5.5 ..... (这种直接ping不通，是因为源
地址是100.0.0.1, 这样要的话包能送到
5.5.5.5, 但是不能回包回来 (回来的时候，R5上100.0.0.1
的没有封装)，所以
ping不通.)
```

Step3: 访问分支点的外口, 还是需要做相互映射:

```
R1#ping 5.5.5.5 .....
R1(config-subif)#Frame-relay map ip 100.0.0.5 104
R5(config-subif)#Frame-relay map ip 100.0.0.1 504
R1#ping 5.5.5.5 !!!!!!!!!!!!!!!!
```

(EIGRP运行在Hub&Spoke时, 下一跳是指向中心点, 而不是指向另一个分支点, 所以不需要做映射, 和RIP不同, 对rip 来说, 下一跳是另一个分支点)

~~~~~  
~~~~~

使用多点子接口 (MutltiPiont Sub-if), 构建HUB&SPOKE FR网络, 之
LAB 9: 构建NBMA模式, 构建OSPF网络。

贾雷附加: FR的接口类型-----ospf的模式
FR physical-----NBMA
FR p2p subif-----P2P
FR mp subif-----NBMA

确认OSPF L3的接口的运行模式 (network type)

```
#sh ip ospf interface
```

Network type NON_BROADCAST (NBMA)

确认L2的FR的PVC是否允许广播/组播流量通过。

```
#sh frame-relay map
```

broadcast (是否有broadcast对本

实验无影响)

现象：所以无法建立邻居。

Step1: 邻居问题：(通过OSPF的单播更新)

在中心点R4上：

```
R4(config)#
```

```
Router ospf 110
```

```
Neighbor 100.0.0.1
```

```
Neighbor 100.0.0.5
```

现象：DR/BDR/DR-Other混乱，影响OSPF数据库不正常。

Step2: DR问题：(通过OSPF优先级, 控制中心点一定成为DR, 无BDR)

```
R4#int s0.100
```

```
ip ospf priority 100
```

贾雷注：默认是1，值越

大越优先成为DR，值为0，放弃DR选举。

```
R1/5(config)#int s0.100
```

```
ip ospf priority 0 (放弃DR选举)
```

现象：

因为OSPF是LS协议，所以无水平分割问题，分支点之间都有对方路由。

如果R1/R5之间没有对方的下一跳映射是无法互通的

Step3: 下一跳问题：(手工配置FR的下一跳映射：)

```
R1#o 5.5.5.5 [110/65] via 100.0.0.5
```

```
R5#o 1.1.1.1 [110/65] via 100.0.0.1
```

```
R1(config-subif)#frame-relay map ip 100.0.0.5 104
R5(config-subif)#frame-relay map ip 100.0.0.4 504
```

~~~~~  
~~~~~

使用多点子接口 (MutltiPiont Sub-if), 构建HUB&SPOKE FR
网络, 之
LAB 10: 通过P2MP NON_BROADCAST 模式, 构建OSPF网络

Step1: (R1/R4/R5) 选定运行模式
in s0.100
ip ospf network point-to-Multipoint NON-BROADCAST

拓扑与LAB9完全一致

Step2: 邻居问题: (与NBMA一样, 同LAB 9)

Step3: DR问题:
(在点对多点中/P2MP中: 无DR的概念, 故无DR问题)

Step4: 下一跳问题: (无下一跳问题)
所有分支点的路由, 的下一跳都集中在中心点, 无下一跳问题
R1# o 5.5.5.5 [110/129] via 100.0.0.4
R5# o 1.1.1.1 [110/129] via 100.0.0.4

特别注意:
P2MP模式会自动产生所有该FR网络的接口, 的/32主机路由, 其
下一跳也在中心点,
所以所有分支点的路由器, 都只需要有到中心点的映射即可.
(分支点之间无需映射)

~~~~~  
~~~~~

使用多点子接口
(MutltiPiont Sub-if), 构建HUB&SPOKE FR网络, 之
LAB 11 : 通过P2MP(BROADCAST) 模式, 构建OSPF网络

网络环境:

L2 FR必须允许广播流量通过!

step1: R1/R4/R5选定运行模式:

```
conf t
in s0.100
ip ospf network point-to-multipoint
```

Step2: 邻居问题: (与broadcast 一样)

ospf邻居能够自动建立, 不存在问题.

不需要单播更新

```
no Neighbor 100.0.0.1
no Neighbor 100.0.0.5
```

Step3: DR问题:

(P2MP中, 根本没有DR的概念, 无DR问题)

Step4: 下一跳问题(无下一跳问题:)

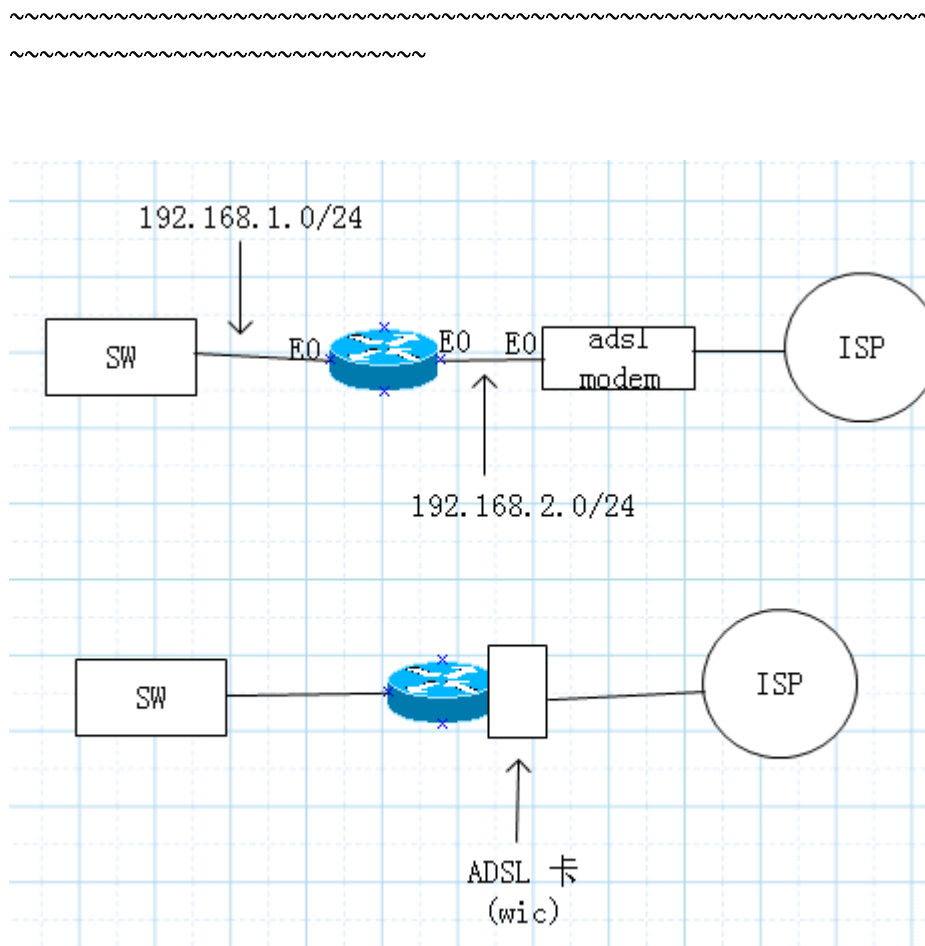
所有分支点的路由, 的下一跳都集中在中心点. (自动下一跳)

~~~~~  
~~~~~

贾雷注: LAB11相当于自动照相机, 而用NBMA的话, 相当手动照相机, 需要手动调整光圈 / 快门 / 焦距. 考试肯定会考手动照相机, 但是实践应用都用自动照相机. 也就是说工程实践中, 用p2mp模式最简单. 但考试需要学习其他的.

day-5

PPPoE (ADSL)



step1:VPDN(虚拟拨号专网):

(these vpdn commands are not needed with cisco ios software
release 12.2(13))

```
vpdn enable
no vpdn logging
!
vpdn-group pppoe
request-dialin
protocol pppoe
```

step2:internal ethernet network

```
int e0
ip add 192.168.1.1 255.255.255.0
ip nat inside
```

step3:xDSL inteface (ADSL卡上的接口)

```
int ATM0
no ip address
no atm ilmi-keepalive
dsl operation mode auto
bundle-enable
hold-queue 224 in
(all default)
```

step4:ATM sub-interface

```
int ATM0.1 point-to-point
pvc 1/1 (VPI/VCI)
pppoe-client dial-pool-number 1
(将这个虚拟的ATM子接口放入pool 1 中)
```

pvc 1/1 is an example value, that must be changed
to match the value used by the isp

step5:IF Dial
(the pppoe client code ties into a dialer interface upon
which a virtual-access interface is cloud.)

```
int dialer 1
ip address negotiated(与ISP协商IP地址)
ip mtu 1492
```

```
!---ethernet mtu defaule=1500
(1492+PPPOE headers=1500)
```

```
ip nat outside
encapsulation ppp
dialer pool 1 (去dialer pool 1 中, 调用刚放入的ATM的子接口)
```

CHAP OR PAP:

CHAP:

```
ppp authentication chap callin
ppp chap hostname <username>
ppp chap password <password>
```

PAP:

```
ppp authentication ppp callin
ppp pap send-username <username> password <password>
```

the isp instruets you about the type of authentication to use.

<password>

the isp instructs you about the type of authentication to use.

step6:NAT

```
access-list 1 permit 192.168.1.* 0.0.0.255
```

```
ip nat inside source list 1 interface dialer1 overload
```

```
ip route 0.0.0.0 0.0.0.0 dialer1
```

```
end
```

~~~~~  
~~~~~

E1线路知识要点:

在中国/欧洲使用E1 (欧标)

在北美/日本使用T1 (美标)

1. 一条E1是带宽为:2.048M的链路,用PCM编码.
2. 一个E1的帧长为256bit:分为32个时隙,一个时隙为8bit
(TimeSlots:时隙)
3. 每秒有8K个E1的帧通过E1接口,即 $8k * 256 = 2048k\text{bps}$
4. 因为每个时隙在E1帧中占8bit
所以一个时隙的带宽是 $8\text{bit} * 8k = 64k\text{bit}$,
即一条E1中含有32个64k信道/时隙

E1帧结构:

E1有不成帧,成帧,成复帧三种方式:

1. 在不成帧的E1中(透明模式):
所有32个时隙都可用于传输有效数据
2. 在成帧的E1中,第0时隙用于传输帧同步数据,其余31个时隙可以用于传输有效数据
3. 在成复帧的E1中,第0时隙用于传输帧同步数据,第16时隙是用于传输信令.
只有1到15,17到31,共30个时隙可用于传输有效数据.

于传输信令.

只有1到15, 17到31, 共30个时隙可用于传输有效数据.

Trunk: (中继线)

E1的原始的使用方法/场合:

第0时隙用于传输帧同步数据

第16时隙用于传输信令.

CE1:channel E1/信道化E1

就是把2M的传输分成了30个64K的时隙, 一般写成 $N \times 64$,

你可以利用其中的即个时隙, 也就是只利用 n 个64k, 必须接在CE1/PRI的接口上

CE1:最多可有31个信道承载数据

E1的2种接口 (G. 703):

同轴电缆:BNC头, 园头, 阻抗:非平衡的 75ohm

双绞线:RJ-45 阻抗:平衡的120 ohm

使用E1有三种方法:

1. 透明传输的专线:将整个2M用作一条链路, 如DDN 2M;

2. CE1:将2M用作 $N \times 64k$ 及其组合, 如128k, 256k等

3. E1最本来的用法:

在用作语音交换机/程控交换机 (PSTN网络中) 的数字中继线(trunk)时, 这也是E1最本来的用法.

工程中常见的E1连接方法:

PRI就是期中的最常用的一种接入方式, 标准较PRA信令.

相对便宜:用2611等的广域网接口卡 (WAN WIC DB-60/2T是SS的, 即 smart serial), 经V. 35--G. 703协议转换器接E1线. 例如: WIC-1T 大约RMB3000

相对较贵:E1卡:目前DDN的2M速率线路通常是经HDSI线路拉至用户侧.

例如: VWIC-1MFT-G703大约RMB15000

E1可由传输设备出的光纤拉至用户侧的光端机提供E1服务.

光端机用法:

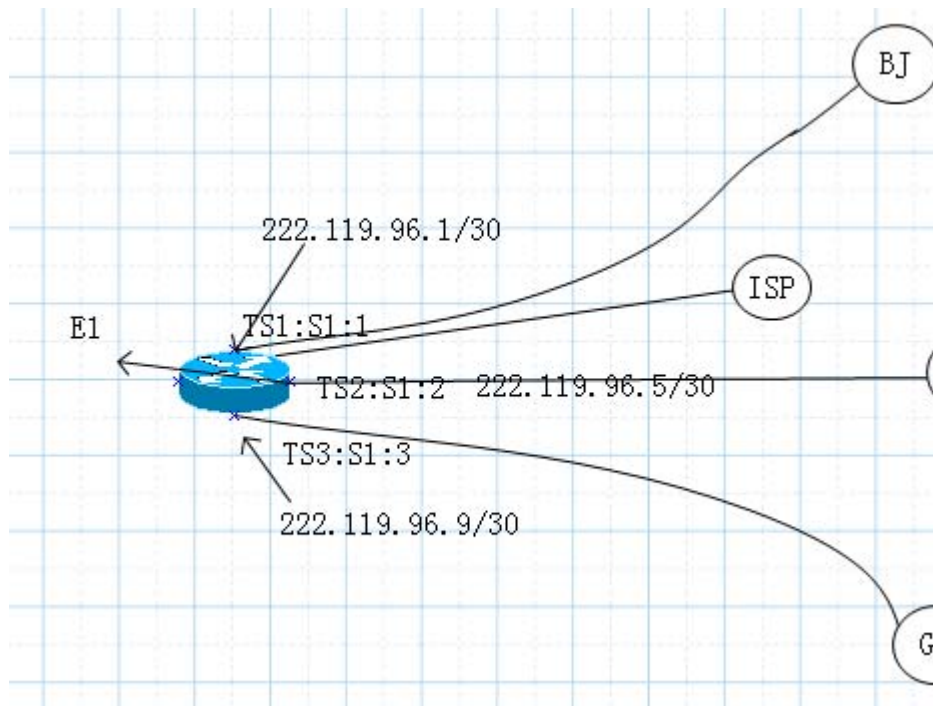
ISP的光传输网络中的传输设备---光纤---光端机---同轴线BNC (2根)---G703转V35转换器---

同步串行链路---路由器的同步串口

光端机--2*BNC--DB15--G703转V35转换器

阻抗转换cable:

BNC---DB15 , BNC---RJ48



贾雷注：图划的不准确.

使用E1线路实现多个64K专线连接:

相关命令:

0. 贾雷注:

card type e1 5 0

card type e1 5 1 没有这个命令，路由器根本没有controller模式可以进入.

1. 进入controller配置模式:

controller {t1 | e1} number

2. 选择帧类型

framing {crc4 | no-crc4}

3. 选择line-code类型:

linecode {ami | b8zs | hdb3}

E1连接3条64K专线, 帧类型为NO-CRC4, 非平衡链路, 路由器具体设置如下:

```
hostname shanxi
```

```
!  
step1:controller E1 0/1 （进入E1控制器）  
  
step2:framing NO-CRC4  
       linecode hdb3  
  
step3:channel-group 1 timeslots 1 （timeslots 1 对应下面的  
s1:1）  
       channel-group 2 timeslots 2  
       channel-group 3 timeslots 3  
       exit  
  
step4:  
int s0/1:1  
ip add 222.119.96.1 255.255.255.252  
!  
int s0/1:2  
ip add 222.119.96.5 255.255.255.252  
!  
int s0/1:3  
ip add 222.119.96.9 255.255.255.252  
  
step5:  
ip route 139.20.40.0 255.255.255.0 s0/1:1  
ip route 139.20.41.0 255.255.255.0 s0/1:2  
ip route 139.20.42.0 255.255.255.0 s0/1:3
```

~~~~~  
~~~~~

day-6

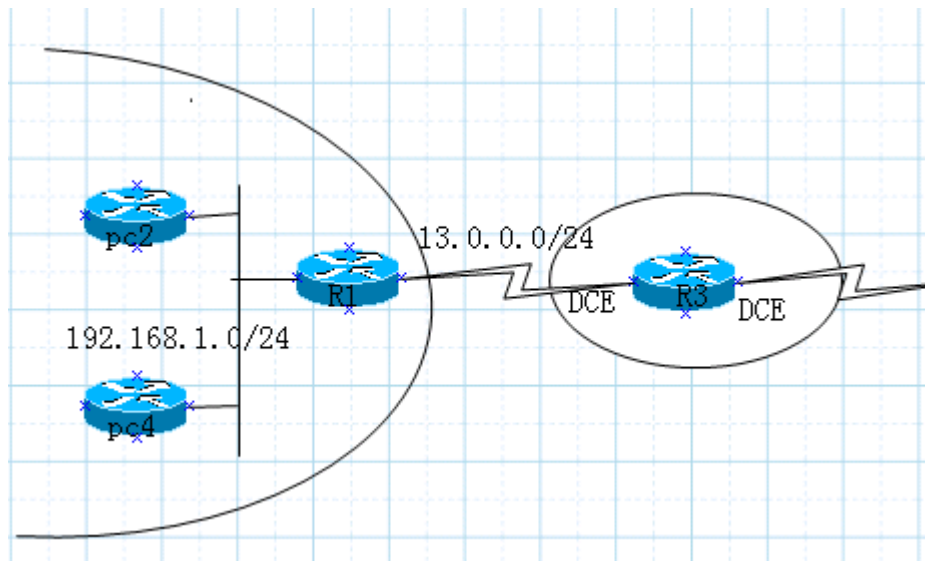
NAT (network address translation)

~~~~~  
~~~~~

NAT的3组概念:

1. inside:我方的网络
 outside: 对方网络
2. Global:公网的
 local:是指一些私网的
3. ip nat outside:(外口)是指一些指向公网的接口
 ip nat inside:(内口)是指一些指向内网的接口
 (外口和内口都在我方的路由器上)

LAB1:NAT的基本配置（原理性的NAT）



step1

按图配置网络, 在R1-R3-R5间运行IGP:RIP v2

RIP宣告网络时不宣告192.168.1.0/24网段. 因为实际工作中私网地址是不会被宣告到公网中的.

```
R1 (config)#ip route 0.0.0.0 0.0.0.0 serial 1
```

R3/R5:

```
router rip
v 2
net 5.0.0.0
net 35.0.0.0
```

step2:

在模拟PC4/PC2的路由器上关闭IP 路由, 并指定网关.

```
R4 (CONFIG)#hostname pc4
no ip routing
ip default-gateway 192.168.1.1
```

确认pc2/pc4都能访问到自己的网关.

~~~~~

这个时候, pc2/pc4能够将包发往R5, 但R5上没有PC4/PC2的路由(因为没有宣告), 所以无法回包.

现实中, ISP会在路由器上做ACL, 将私网地址过滤掉, 所以pc2/pc4发出的包也不能发到R5.

step0:网络背景

在ISP的接入路由器进行对源地址是私网的IP的数据包，进行过滤：

0-1：首先通过ACL定义源地址是

B类私网地址：

|     |         |     |     |
|-----|---------|-----|-----|
| 172 | 16      | *   | *   |
| 0   | 15      | 255 | 255 |
| 172 | (16-31) | *   | *   |

```
access-list 1 deny 10.0.0.0 0.255.255.255
access-list 1 deny 192.168.0.0 0.0.255.255
access-list 1 deny 172.16.0.0 0.15.255.255
access-list 1 permit any any
```

0-2:

R3(config)#int s0(在ISP连接用户的接入路由器上，调用数据包过滤)

R3(config-if)#ip access-group 1 in

0-3-3:ISP将连接到用户的路由重分布到本网络中

```
access-list 13 permit 13.0.0.0
```

```
router-map R-13 permit 10
  match ip add 13
  set metric 1
```

```
router rip
  redistribute connected route-map R-13
```

在R5上, sh ip route rip 则出现了13网段

~~~~~  
~~~~~

0-4:在NAT路由器的内口关闭代理ARP

```
R1(config) # int e0
              no ip proxy-arp (关闭代理ip)
```

step1:定义NAT的外口/内口

```
R1:  int e0
      ip nat inside

      int s1
      ip nat outside
```

step2:静态的NAT转换

```
R1 (config) #ip nat inside source static 192.168.1.2 13.0.0.10
```

我方 所发起的 私网地址 公网地址

R1#show ip nat translations(察看NAT的转换表)

R1#debug ip nat (察看NAT的转换过程)

R2#

s=192.168.1.2(local).d=5.5.5.5(ethernet 0), sending

R1#

NAT\*:s=192.168.1.2 ->13.0.0.10,d=5.5.5.5[15]

s=192.168.1.2 d=5.5.5.5

s=13.0.0.10 d=5.5.5.5

R5#

IP: s=13.0.0.10(serial10),d=5.5.5.5, len 100, recd 4

R5#

IP: s=5.5.5.5(local),d=13.0.0.10 (serial0), sending

R1#

NAT\*:s=5.5.5.5,d=13.0.0.10 --> 192.168.1.2 [15]

s=5.5.5.5 d=13.0.0.10

s=5.5.5.5 d=192.168.1.2

PC2#

IP: s=5.5.5.5(ethernet0),d=192.168.1.2, len 100, rcvd 1

如果此时想再做R4的, 会出现这样的情况:

```
R1(config)#ip nat inside source static 192.168.1.4 13.0.0.1
% 13.0.0.1 already mapped (192.168.1.2 -> 13.0.0.1)
```

说明一个接口只能做一个内网地址的映射

~~~~~  
~~~~~

LAB2:在中小型企业, 通常使用基于接口的端口复用:  
(PAT/NAPT/Overload)

Step1:定义NAT的外口/内口:(同LAB1)

Step2:通过ACL, 定义准备进行NAT的内网的用户群:

Step1:定义NAT的外口/内口:(同LAB1)

Step2:通过ACL, 定义准备进行NAT的内网的用户群:

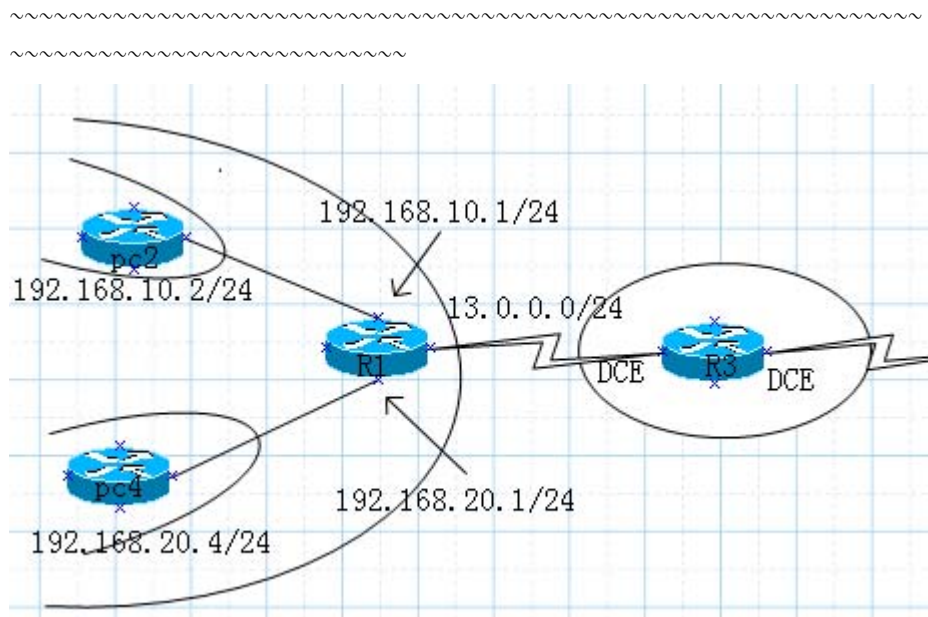
```
R1(config)#access-list 1 permit 192.168.1.0 0.0.0.255
                        (192.168.1.*)
```

Step3:进行基于接口的NAT端口复用:

```
R1(config)#
ip nat inside source list 1 interface serial 1 overload(新IOS
版本里面默认启用overload)
                        (内网用户) (NAT的外口)      端口复用
```

测试:

实现了192.168.1.\*整个网段的用户, 共享一个IP地址, 连接到Internet.



LAB3:大型企业, 向ISP购买较多的公网IP后, 将其分别放入不同NAT-POOL, 以供不同的部门进行NAT转换

step1:定义NAT的外口/内口: (同LAB1)

step1:定义NAT的外口/内口: (同LAB1)

step2:通过ACL, 定义准备进行NAT的用户群:

```
R1 (config) #access-list 10 permit 192.168.10.0 0.0.0.255
```

```
R1 (config) #access-list 20 permit 192.168.20.0 0.0.0.255
```

step3:在R1上模拟两个网段的网关: (实际工作中不用做这步)

```
R1#
```

```
int e0
```

```
ip add 192.168.20.1 255.255.255.0 secondary
```

```
ip add 192.168.10.1 255.255.255.0
```

step4:在两个网段/VLAN中的用户, 分布设定自己的网关

```
PC4 (config-if-e0) #ip add 192.168.20.4 255.255.255.0
```

```
PC4 (config)#ip default-gateway 192.168.20.1
```

```
PC2(config-if-e0)#ip add 192.168.10.2 255.255.255.0
```

```
PC2(config)#ip default-gateway 192.168.10.1
```

PC4/PC2都要测试, 是否能够ping通自己的网关!!!!!!!

step5:将从ISP处买到的公网IP, 放入地址池中:

(在R1上定义两个pool)

```
R1(config)#ip nat pool PL-A 100.0.0.8 100.0.0.11 prefix-length 24
```

```
R1(config)#ip nat pool PL-B 100.0.0.12 100.0.0.15 prefix-length 24
```

起始IP

终止IP

IP长

度(这个长度由ISP给定)

step6:为不同的部门, 进行基于不同NAT-POOL的转换:

```
R1(config)#ip nat inside source list 10 pool PL-A overload
```

```
R1(config)#ip nat inside source list 20 pool PL-B overload
```

```
R1#sh ip nat translation
```

step7:ISP/R3

为了让R5有回包路由, 在R3上重分布静态路由到RIP中:

```
7-1:ip route 100.0.0.0 255.255.255.248 13.0.0.1
```

```
7-2:ip prefix-list 100 permit 100.0.0.8/29
```

```
7-3:route-map R-100
```

```
match ip add prefix-list 100
```

```
7-3:route-map R-100
      match ip add prefix-list 100
      set metric 1
```

```
7-4:route rip
      redistribute static route-map R-100
```

```
7-5:
R5#sh ip route rip可读到100.0.0.8/29
```

```
~~~~~
~~~~~
```

LAB4:使用一个接口的公网IP，对外网提供多个不同的服务器：  
（静态的TCP端口映射，静态的端口复用）

step1:定义NAT的外口/内口：（同LAB1）

```
step2:
R1(config)#
FTP server:
ip nat inside source static tcp 192.168.10.2 21 13.0.0.1 21
                                   (内网) (公网)
```

```
WWW server
ip nat inside source static tcp 192.168.10.3 80 13.0.0.1 80
                                   (内网服务器的IP和端口)
```

为了演示实验实验结果,把命令改为以下:

```
ip nat inside source static tcp 192.168.10.2 23 13.0.0.1 21
ip nat inside source static tcp 192.168.10.3 23 13.0.0.1 80
```

```
step3:密码:
R1/2/4
R1 (config) #line vty 0 4
R1 (config-line)#no login
```

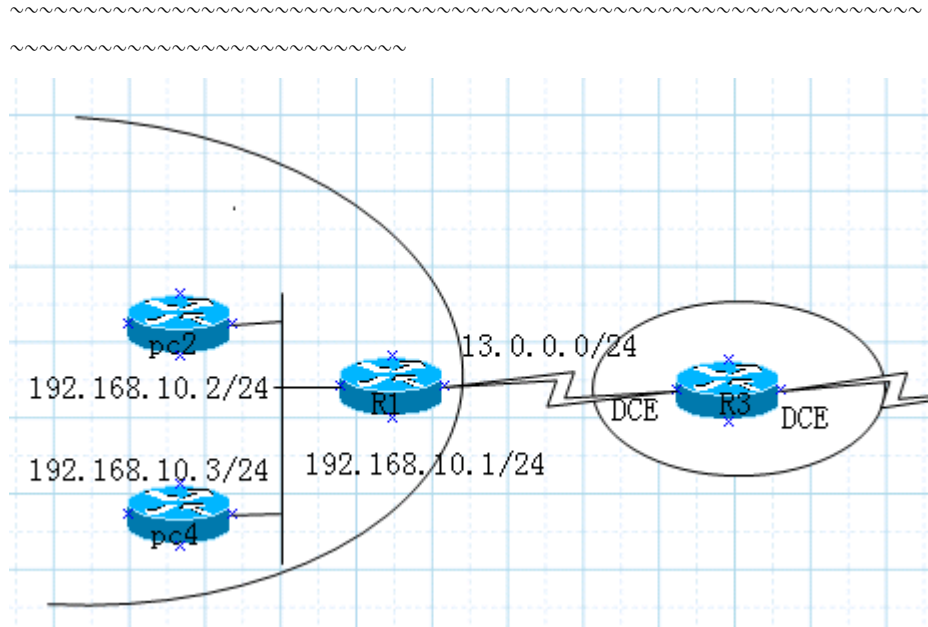
```
R5#telnet 13.0.0.1 >>>>R1 （默认telnet 23 端口）
```

```
R5#telnet 13.0.0.1 21 >>>>FTP
```

```
R5#telnet 13.0.0.1 >>>>R1 （默认telnet 23 端口）
```

```
R5#telnet 13.0.0.1 21 >>>>FTP
```

```
R5#telnet 13.0.0.1 80 >>>>WEB
```



LAB5: 通过NAT ,实现镜像服务器的负载均衡: (实际工程中很少用, 仅作参考)

pc2/pc4是两台镜像服务器, 要求将外网对服务器的访问流量进行负载均衡.

Step1: 定义NAT外口/内口(同LAB 1一样)

Step2: 通过ACL, 定义” 我方的, 公网” 地址,  
作为来自外网访问的目标地址:

```
R1(config)#access-list 9 permit host 100.0.0.9
```

或者

```
R1(config)#access-list 9 permit 100.0.0.9 (0.0.0.0)  
可不写, 意思是全匹配  
13.0.0.9 IP
```

Step3: 定义内网的服务器群的地址池, 轮转类型:

```
R1(config)#
```

```
Ip nat pool PL 192.168.10.2 192.168.10.3 prefix-length 24 type rotary
```

Step4:

```
R1(config)#ip nat inside destination list 9 pool PL
```

Step5:

```
Server-2/4(config)#line vty 0 4 (进入telnet的模式)
```

```
Server-2/4(config-line)#no login (表示telnet 上不需要密码, 就可以被telnet)
```

R5不断telnet13.0.0.9

(以下没讲)

~~~~~  
~~~~~

代理ARP:

step1

在关闭代理ARP的情况下:

```
int e0
```

```
no ip proxy-arp
```

PC2#不设置默认网关

```
clear arp-cache
```

PC2#debug arp

step2:默认网关的方法

```
PC2 (config) #ip default-gateway 192.168.1.1
```

step3:代理ARP(不使用默认网关)

```
PC2 (config) #no ip default-gateway
```

```
R1(config-if-e0)#ip proxy-arp
```

~~~~~  
~~~~~

动态的NAT: (需要先作PING操作触发)

LAB2: 在中小型企业, 通常基于接口的端口复用:

(PAT/NAPT/Overload)

step1: 定义NAT的外口/内口: (同LAB1)

step: 通过ACL, 定义准备进行NAT的用户群

```
R1 (config) #access-list 1 permit 192.168.1.0 0.0.0.255
```

step3: 进行基于接口的NAT端口复用:

```
R1 (config) #ip nat inside source list 1 int s1 overload
```

(内网用户) (NAT的外口) (进行端口复

用)

```
R1#sh ip nat translation
```

```
~~~~~  
~~~~~
```

LAB3: 使用route-map, 调用内网用户, 进行NAT转换, 以解决远端端口的一些复杂映射问题:

step1/step2: 通LAB2

step3: 在route-map, 中调用ACL1

```
R1 (config) #route-map T permit 10
```

```
R1 (config-map) #match ip add 1
```

step4:

R1 (config) #在NAT转换中, 调用route-map所定义的用户群

```
ip nat source route-map T int s1 overload
```

## 工程案例: 巨田

----- show running-config -----

Building configuration...

Current configuration : 11463 bytes

!

version 12.3

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

service multiple-config-sessions

no service single-slot-reload-enable

```
no service single-slot-reload-enable  
!  
hostname center7507  
!  
boot-start-marker  
boot-end-marker  
!  
card type e1 0 0  
card type e1 5 0  
card type e1 5 1  
!  
redundancy  
mode hsa  
enable secret 5 <removed>  
!  
clock timezone cst 8  
no aaa new-model  
ip subnet-zero  
!  
!  
!  
!  
ip cef  
ipx routing 0012.7f35.ad28  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
controller E1 0/0/0  
framing NO-CRC4  
channel-group 4 timeslots 7-8 speed 64  
channel-group 5 timeslots 9-10 speed 64  
channel-group 6 timeslots 11-12 speed 64  
channel-group 11 timeslots 22-23 speed 64  
channel-group 12 timeslots 24-25 speed 64  
channel-group 13 timeslots 26-27 speed 64  
!  
controller E1 0/0/1  
!  
controller E1 0/0/2  
!  
controller E1 0/0/3
```

```
!  
controller E1 0/0/3  
!  
controller E1 0/0/4  
!  
controller E1 0/0/5  
!  
controller E1 0/0/6  
!  
controller E1 0/0/7  
!  
controller E1 5/0/0  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/1  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/2  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/3  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/4  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/5  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/6  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/0/7  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/0  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/1  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/2
```

```
!  
controller E1 5/1/2  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/3  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/4  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/5  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/6  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
controller E1 5/1/7  
framing NO-CRC4  
channel-group 0 timeslots 1-31 speed 64  
!  
!  
interface Serial0/0/0:4  
description yuanling  
bandwidth 2000  
ip address 10.1.26.2 255.255.255.252  
ipx delay 20  
ipx network 26F  
ipx output-network-filter 802  
ipx output-sap-filter 1002  
!  
interface Serial0/0/0:5  
description xinzhou  
bandwidth 2000  
ip address 10.1.31.2 255.255.255.252  
ipx delay 20  
ipx network 31F  
ipx output-network-filter 801  
ipx output-sap-filter 1001  
!  
interface Serial0/0/0:6  
description guomao  
bandwidth 2000  
ip address 10.1.36.2 255.255.255.252  
ipx delay 20  
ipx network 36F  
ipx output-network-filter 801  
ipx output-sap-filter 1001
```

```
ipx output-network-filter 801
ipx output-sap-filter 1001
ipx output-gns-filter 1010
!
interface Serial0/0/0:11
description beijing
ip address 10.1.106.2 255.255.255.252
fair-queue 64 256 0
h323-gateway voip interface
!
interface Serial0/0/0:12
description zhuhai
bandwidth 2000
ip address 10.1.111.2 255.255.255.252
ipx delay 20
ipx network 111F
ipx output-network-filter 801
ipx output-sap-filter 1001
!
interface Serial0/0/0:13
description wuhan
bandwidth 2000
ip address 10.1.116.2 255.255.255.252
ipx delay 20
ipx network 116F
ipx output-network-filter 801
ipx output-sap-filter 1001
!
interface FastEthernet0/1/0
no ip address
duplex auto
speed auto
!
interface FastEthernet0/1/0.1
bandwidth 1000
encapsulation isl 1
ip address 10.66.0.252 255.255.0.0
no ip redirects
ip ospf cost 50
ipx encapsulation NOVELL-ETHER
ipx network 66
ipx output-gns-filter 1010
standby 1 ip 10.66.0.254
standby 1 priority 110
standby 1 preempt
!
interface FastEthernet0/1/0.2
bandwidth 1000
encapsulation isl 2
ip address 10.67.0.252 255.255.0.0
no ip redirects
ip ospf cost 65
```

```
no ip redirects
ip ospf cost 65
ipx encapsulation NOVELL-ETHER
ipx network 31AA
ipx output-gns-filter 1010
standby 2 ip 10.67.0.254
standby 2 priority 110
standby 2 preempt
!
interface FastEthernet0/1/0.3
bandwidth 1000
encapsulation isl 3
ip address 10.55.0.252 255.255.0.0
no ip redirects
ip ospf cost 50
ipx encapsulation NOVELL-ETHER
ipx network 55
ipx output-network-filter 803
ipx output-sap-filter 1003
ipx output-gns-filter 1010
standby 3 ip 10.55.0.254
standby 3 priority 110
standby 3 preempt
!
interface FastEthernet0/1/0.4
bandwidth 1000
encapsulation isl 4
ip address 10.78.0.252 255.255.0.0
no ip redirects
ip ospf cost 65
ipx encapsulation NOVELL-ETHER
ipx network 788
ipx output-network-filter 803
ipx output-sap-filter 1003
ipx output-gns-filter 1010
standby 4 ip 10.78.0.254
standby 4 priority 110
standby 4 preempt
!
interface FastEthernet0/1/0.5
description jijing
bandwidth 1000
encapsulation isl 5
ip address 10.60.0.252 255.255.0.0
no ip redirects
ip ospf cost 50
standby 5 ip 10.60.0.254
standby 5 priority 110
standby 5 preempt
!
interface FastEthernet0/1/1
no ip address
```

```
interface FastEthernet0/1/1
no ip address
duplex auto
speed auto
!
interface Serial5/0/0:0
description shekou
bandwidth 2000
ip address 10.1.10.2 255.255.255.252
ipx network 10F
ipx output-network-filter 805
ipx output-sap-filter 1005
!
interface Serial5/0/1:0
description zhongdian
bandwidth 2000
ip address 10.1.15.2 255.255.255.252
ipx network 15F
ipx output-network-filter 805
ipx output-sap-filter 1005
!
interface Serial5/0/2:0
description buji
bandwidth 2000
ip address 10.1.21.2 255.255.255.252
ipx delay 20
ipx network 21F
ipx output-network-filter 803
ipx output-sap-filter 1003
!
interface Serial5/0/3:0
no ip address
!
interface Serial5/0/4:0
no ip address
!
interface Serial5/0/5:0
no ip address
!
interface Serial5/0/6:0
description dongle
bandwidth 2000
ip address 10.1.46.2 255.255.255.252
ipx delay 20
ipx network 46F
ipx output-network-filter 802
ipx output-sap-filter 1002
!
interface Serial5/0/7:0
no ip address
!
interface Serial5/1/0:0
```

```
!  
interface Serial5/1/0:0  
  no ip address  
!  
interface Serial5/1/1:0  
  no ip address  
!  
interface Serial5/1/2:0  
  no ip address  
!  
interface Serial5/1/3:0  
  no ip address  
!  
interface Serial5/1/4:0  
  no ip address  
!  
interface Serial5/1/5:0  
  description wuxi  
  bandwidth 2000  
  ip address 10.1.130.2 255.255.255.252  
  ipx network 130F  
  ipx output-network-filter 801  
  ipx output-sap-filter 1001  
!  
interface Serial5/1/6:0  
  description hangzhou  
  bandwidth 2000  
  ip address 10.1.135.2 255.255.255.252  
  ipx network 135F  
  ipx output-network-filter 801  
  ipx output-sap-filter 1001  
!  
interface Serial5/1/7:0  
  description beijing2  
  bandwidth 2000  
  ip address 10.1.140.2 255.255.255.252  
  ipx network 140F  
  ipx output-network-filter 801  
  ipx output-sap-filter 1001  
!  
router ospf 180  
  log-adjacency-changes  
  redistribute static metric 20 metric-type 1 subnets  
  network 0.0.0.0 255.255.255.255 area 0  
!  
ip classless  
ip route 0.0.0.0 0.0.0.0 10.78.0.111  
ip route 10.106.0.0 255.255.0.0 10.1.106.1  
ip route 10.125.0.0 255.255.0.0 10.1.125.1  
ip route 10.143.0.0 255.255.0.0 10.1.140.1  
ip route 10.144.0.0 255.255.0.0 10.1.140.1  
ip route 192.10.5.0 255.255.255.0 10.66.0.29
```

```
ip route 10.144.0.0 255.255.0.0 10.1.140.1
ip route 192.10.5.0 255.255.255.0 10.66.0.29
ip route 192.202.1.0 255.255.255.0 10.66.0.29
ip route 196.1.28.0 255.255.255.0 10.66.0.29
no ip http server
!
!
!
access-list 801 permit 66
access-list 801 permit 352BF9A7
access-list 801 permit 7880101
access-list 801 permit 750001
access-list 801 permit 788
access-list 801 permit 61
access-list 801 permit 65
access-list 801 permit 71
access-list 801 permit 670001
access-list 801 permit 670002
access-list 801 permit 31AA
access-list 801 permit 7556688
access-list 801 permit 156
access-list 802 permit 45
access-list 802 permit 755701
access-list 802 permit 755702
access-list 802 permit 25
access-list 802 permit 7550601
access-list 802 permit 7550602
access-list 802 permit 66
access-list 802 permit 352BF9A7
access-list 802 permit 7880101
access-list 802 permit 750001
access-list 802 permit 788
access-list 802 permit 61
access-list 802 permit 65
access-list 802 permit 71
access-list 802 permit 31AA
access-list 802 permit 7556688
access-list 802 permit 156
access-list 803 permit 66
access-list 803 permit 352BF9A7
access-list 803 permit 31AA
access-list 803 permit 7880101
access-list 803 permit 750001
access-list 803 permit 788
access-list 803 permit 61
access-list 803 permit 65
access-list 803 permit 71
access-list 803 permit 20
access-list 803 permit 755001
access-list 803 permit 755002
access-list 803 permit 55
access-list 803 permit 19990711
```

```
access-list 803 permit 55
access-list 803 permit 19990711
access-list 803 permit 755101
access-list 803 permit 755102
access-list 803 permit 670001
access-list 803 permit 670002
access-list 803 permit 7556688
access-list 803 permit 156
access-list 805 permit 15
access-list 805 permit 755501
access-list 805 permit 755502
access-list 805 permit 66
access-list 805 permit 352BF9A7
access-list 805 permit 7880101
access-list 805 permit 750001
access-list 805 permit 788
access-list 805 permit 61
access-list 805 permit 65
access-list 805 permit 71
access-list 805 permit 755303
access-list 805 permit 755523
access-list 805 permit 10
access-list 805 permit 7550201
access-list 805 permit 7550202
access-list 805 permit 755503
access-list 805 permit 755504
access-list 805 permit 755525
access-list 805 permit 31AA
access-list 805 permit 7556688
access-list 805 permit 156
access-list 1001 permit 66
access-list 1001 permit 352BF9A7
access-list 1001 permit 7880101
access-list 1001 permit 750001
access-list 1001 permit 788
access-list 1001 permit 61
access-list 1001 permit 65
access-list 1001 permit 71
access-list 1001 permit 670001
access-list 1001 permit 670002
access-list 1001 permit 31AA
access-list 1001 permit 7556688
access-list 1001 permit 156
access-list 1002 permit 45
access-list 1002 permit 755701
access-list 1002 permit 755702
access-list 1002 permit 25
access-list 1002 permit 7550601
access-list 1002 permit 7550602
access-list 1002 permit 66
access-list 1002 permit 352BF9A7
access-list 1002 permit 7880101
```

```
access-list 1002 permit 352BF9A7
access-list 1002 permit 7880101
access-list 1002 permit 750001
access-list 1002 permit 788
access-list 1002 permit 61
access-list 1002 permit 65
access-list 1002 permit 71
access-list 1002 permit 31AA
access-list 1002 permit 7556688
access-list 1002 permit 156
access-list 1003 permit 66
access-list 1003 permit 352BF9A7
access-list 1003 permit 31AA
access-list 1003 permit 7880101
access-list 1003 permit 750001
access-list 1003 permit 788
access-list 1003 permit 61
access-list 1003 permit 65
access-list 1003 permit 71
access-list 1003 permit 20
access-list 1003 permit 755001
access-list 1003 permit 755002
access-list 1003 permit 55
access-list 1003 permit 19990711
access-list 1003 permit 755101
access-list 1003 permit 755102
access-list 1003 permit 670001
access-list 1003 permit 670002
access-list 1003 permit 7556688
access-list 1003 permit 156
access-list 1005 permit 15
access-list 1005 permit 755501
access-list 1005 permit 755502
access-list 1005 permit 66
access-list 1005 permit 352BF9A7
access-list 1005 permit 7880101
access-list 1005 permit 750001
access-list 1005 permit 788
access-list 1005 permit 61
access-list 1005 permit 65
access-list 1005 permit 71
access-list 1005 permit 755303
access-list 1005 permit 755523
access-list 1005 permit 10
access-list 1005 permit 7550201
access-list 1005 permit 7550202
access-list 1005 permit 755503
access-list 1005 permit 755504
access-list 1005 permit 755525
access-list 1005 permit 31AA
access-list 1005 permit 7556688
access-list 1005 permit 156
```

```
access-list 1005 permit 7556688
access-list 1005 permit 156
access-list 1010 permit FFFFFFFF
!
!
!
!
!
!
!
control-plane
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
password <removed>
login
!
!
!
end
```

## 路由

### 01晚-路由原理与静态路由

(已完成实验)

CCNP BSCI

CCNP routing 课程时间规划：（15—17）

路由原理 / 静态路由： 1

RIP： 1

EIGRP： 2

网络分层：

Access layer:终端用户的接入

Distribution Layer: 实现基本网络策略控制，和网络服务。

Core Layer:高速地传送数据包。

网络划分的种类：

1: 可以按照网络的不同的功能部门进行划分。

- 1: 可以按照网络的不同功能部门进行划分。
- 2: 可以按照网络所分布的地域范围进行划分。

网络拓扑:

Full Meshed / 全互连

任意两点（节点）之间，都有直接相连的连接。

Full Meshed连接，所需连接个数的公式:

公式:  $C_2^n$  (上标) n (下标)

∴: 在核心网中，构建双冗余的Full Mesh是较为昂贵， $N*(N-1)$

∴: 可以考虑构建双Hub&Spoke。  $2*(N-1)$

既可以保持冗余性，也可以降低建网成本。

一个规划良好的网络中，通常考虑到以下3点:

Scalability / 可扩展性

Predictability / 可预测性

Flexibility/灵活性

Benefits of Hierarchical Addressing/网络地址层次化的划分:

- 1: 在路由器上，有更高的路由转发效率，减少路由表的路由条目。
- 2: 提高地址的利用率。

VLSM: Variable-Length Subnet Mask

### VLSM

可以实现根据不同网络需求，拆分 / 划分不同大小的子网，  
特别注意:

VLSM不能凭空额外的创造出更多的IP地址来。

Step1:将所需要划分的网络，自大而小地排列出来。

195. 15. 20. 0 / 22 （有效IP:  $1024-2=1022$ ）主机位有10位  $2$ 的10次方=1024

500 / 200 / 50 / 10 / 2 (Serial Link)

图01—01-VLSM

500IP: 9个主机位 / Host Bits

200IP: 8个主机位

50IP: 6个主机位

10IP: 4个主机位

Serial Link: 2个主机位

step3: 根据每个子网的主机位, 算出子网的网络位:

建议: 在选择网络位时, 先取0, 再取1.

500IP:  $\therefore$  有23个网络位 ( $32 - 9 = 23$ )

200IP:  $\therefore$  有24

Step4: 计算每个子网的:

网络号, 广播地址, 首个有效地址, 最后一个的有效地址

原则:

如果主机位为全0, 那么这个地址为这个子网的网络号。

如果主机位为全1, 那么这个地址为这个子网的广播地址。

能够容纳500个主机的子网:

网络号: 195. 15. 20. 0 / 23

第一个有效IP: 195. 15. 20. 1 / 23

最后一个有效IP: 195. 15. 21. 254 / 23

广播地址: 195. 15. 21. 255 / 23

能够容纳200个主机的子网:

网络号: 195. 15. 22. 0 / 24

第一个有效IP: 195. 15. 22. 1 / 24

最后一个有效的IP: 195. 15. 22. 254 / 24

子网的广播地址: 195. 15. 22. 255 / 24

能够容纳50个主机的子网:

网络号: 195. 15. 23. 0 / 26

第一有效IP: 195.15.23.1 / 26  
最后一个有效IP: 195.15.23.62 / 26  
子网的广播地址: 195.15.23.63 / 26

能够容纳10个主机的子网:

网络号: 195.15.23.64 / 28  
第一个有效的IP: 195.15.23.65 / 28  
最后一个有效的IP: 195.15.23.78 / 28  
子网的广播地址: 195.15.23.79 / 28

NO.1 Serial Link:

网络号: 195.15.23.80 / 30 有效IP: 81 / 82

NO.2 Serial Link

网络号: 195.15.23.84 / 30 有效IP: 85 / 86

NO.3 Serial Link

网络号: 195.15.23.88 / 30 有效IP: 89 / 90

#### Route Summarization/路由汇总:

在需要进行汇总的明细路由中,  
寻找前部相同的位数, 以这一点为分界点, 将明细路由汇总到这一位。

Step1: 按照IP路由的4个字段, 寻找相同 / 不同点的分界。

Step2: 将第一个不同的字段, 展开为2进制数, 进一步寻找相同 / 不同点的分界点。

Setp3: 将不同的位数, 置为全0  
以相同的位数作为汇总路由的路由长度。

172.16.12.0 / 22

CIDR: (Classless Interdomain Routing) / 无类域间路由

可以突破网络的边界。

CIDR是可以突破网络主类边界，实现网络汇总。

CIDR:

Block addresses can be summarized into single entries without regard to the classful boundary.

Routing Principles/路由原理

IOS语法:

正体字: 表示需要准确的IOS命令 (不变)

斜体字: 根据实际情况, 输入的参数 (变化)

{        }: 必选项

[        ]: 可选项

| :        OR, 或关系, 多个元素中, 选择一个

LAB1: 静态路由的原理性实验:

图01-02-static route

NO. 1: 关于直链路由:

如果物理接口的链路状态: L1 / L2都能够UP / UP

那么接口所在的网段, 就以直链的形式出现在路由表中。

Step1:按图配置IP, 进行链路测试:

R1#show ip int brief(察看接口简要IP信息。)

R2#debug ip packet (察看IP数据包)

特别注意:

路由器不是基于接口考虑数据包的路由, 而是基于整个路由的路由表, 进行数据包的路由。

NO. 2: IP路由的核心原理 (路由对联): (路由的单向性)

上联: “去往目标网络的路径上的”所有路由器, 都要有, 去往目标网络的路由。(保证数据包能够送到目标网络)

下联: : “在返回源网络的路径上的”所有路由器都要有, 返回源网络的路由。(保证数据包能够从目标网络返回源地址)

横批: 有去有回, 肯定能通!

NO. 3: 路由的下一跳的可达性问题:

每条路由, 都必须检查其路由的下一是否可达, 如果下一跳不可达, 那么这条路由会被路由器从路由表中删除。

R2#clear ip route \* (reset/复位路由表)

路由表的递归查询:

NO. 4: 路由的单向性:

IP路由是单向的。

IP数据包所经过的路径，有可能往返是重叠的，也有可能是不同路径的，取决于路由的指向。（图）01—03-单向性

## 02晚-动态路由协议-RIP

（已完成实验）

Dynamic Routing:动态路由协议:

现代IP网络中，主要的动态路由协议:

AD/管理距离:

- 1: DV/距离向量协议: RIP (120) / IGRP (100)
- 2: LS/链路状态协议: OSPF (110) / IS-IS (115)
- 3: DV-LS/混合协议: EIGRP (90)
- 4: ODR (160)

贾雷的补充:

- 1) connet 0
- 2) static 1
- 3) EIGRP 20
- 4) IBGP 200
- 5) I EIGRP 90
- 6) E EIGRP 170

### LAB1:

ODR (On-Demand Routing)

CISCO私有的协议，基于CDP协议的运行。

（CDP只能发现直接相连的Cisco设备）

适用于：

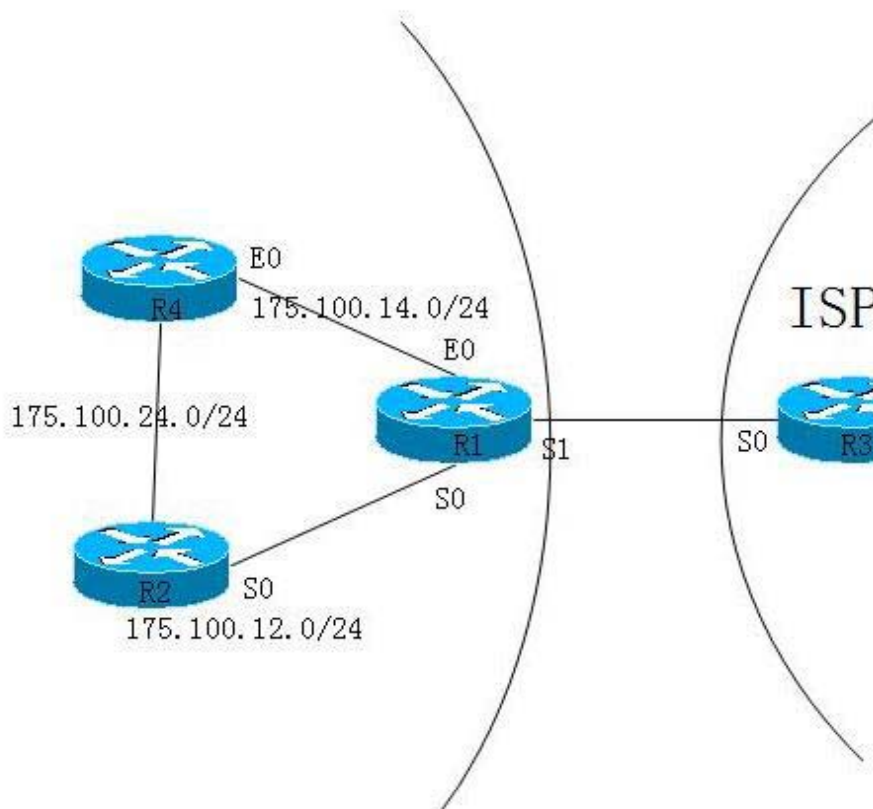
在全网都是CISCO设备的网络环境中，

而且网络拓扑是HUB&SPOKE架构的简单网络中，

（所有的分支点都是直接与Hub相连）

ODR用于使网络配置最简化的操作。

CDP的Hello周期是60S, Hold-time: 3\*60=180.



Step1:确认设备的连接(察看CDP的运行状态)

1-1:

show cdp interface

(察看正在运行CDP的接口, 默认所有接口都运行CDP)

1-2:控制CDP的运行范围:

R5(config)#cdp run

(在全局启动CDP, 默认已经启动, 所有接口都运行CDP)

```
R3(config)#in serial 1
R3(config-if)#cdp enable(对特定接口, 启动CDP)
```

1-3:

```
R5#show cdp neighbors
```

| Device ID | Local Intrfce | Holdtme | capability | Platform |
|-----------|---------------|---------|------------|----------|
| Port ID   |               |         |            |          |
| R3        | ser 0         | 120~179 | R          | 2500     |
| ser 1     |               |         |            |          |

对方接口

注意:

- 1:在ISP的角度考虑, ISP的PE是不会与用户的CE运行CDP的.
- 2:注意以太网交换机的对CDP的影响. (如果两个路由器上中间有个CISCO的交换机, 那么这台交换机将会阻止ODR的运行)

step2:在中心路由(HUB)R1上, 启动ODR:

```
R1(config)#router odr
```

实现了内网的全网通信, 但与HUB(中心点)是非直链的R2, 没有路由.

step3:外网的互通:

CE指向ISP的默认路由:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 13.0.0.3
                                对方IP
```

在ISP PE 指向 用户的静态路由:

```
R3(config)#ip route 175.100.0.0 255.255.0.0 serial 0
                                                本机的出接口
```

Step4: ODR 的路由观察:(在R2建立一个环回口)

每个分支点, 通过ODR HUB 发来的ODR默认路由, 访问外网:

```
R2#0* 0.0.0.0/0[160/1]via 175.100.12.1,
```

在中心点R1#上, 也会自动获得每个分支点下面所直链的明细路由

```
0 175.100.24.0 [160/1] via 175.100.12.2,
```

```
0 175.100.22.0 [160/1] via 175.100.12.2,
```

Step5:ODR路由的局限性:

- 1:全网设备必须都是CISCO设备.
- 2:分支点的路由器,必须跟中心点直接相连.

Classful Routing &Classless Routing

落后的淘汰的          先进的,正在运行的

~~~~~

Classful Routing/有类路由协议:

IGRP /RIP V1

1. 在发送路由更新信息时,不携带子网掩码,无法描述路由条目的路由长度.
2. 在主类的网络边界上,自动发生路由汇总,汇总到主类网络的默认的路由长度(不支持VLSM,只能汇总为A/B/C类).
(自动汇总是无法关闭的)
3. 由于上述原因,有类路由协议会产生“不连续子网”的路由通达性问题.

Classless Routing/无类路由协议

~~~~~

RIP V2 的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后,再通告出去.
3. 把收到的明细路由放进路由表中,但会对明细路由进行汇总后再通告出去!

EIGRP的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后,再通告出去
3. 把收到的明细路由放进路由表中,并且把收到的明细路由通告出去!

3. 把收到的明细路由放进路由表中, 并且把收到的明细路由通告出去!

IP Classless (在IOS为12.0以后的版本中, 默认启动无类路由.)

RIPV2/EIGRP/OSPF/IS-IS/BGPv4

1. 在发送路由更新信息时, 已经携带子网掩码.
2. 支持VLSM, 路由的手工/自动汇总, (可以关闭自动汇总)
3. 在部分先进的路由协议中, 支持CIDR(超网)

在网络边界:

DV协议默认会执行自动汇总, (RIP/IGRP/EIGRP)

LS协议默认不执行自动汇总. (OSPF/ISIS)

#### LAB2:RIP的基本配置:

~~~~~

R3#

router rip

network 176.16.0.0 (宣告与本路由直接相连的网络, 只需要宣告其主类网络)

show ip route rip

debug ip rip

un all

LAB3:RIP的版本控制:

~~~~~

show ip protocols (察看RIP的版本控制)

在未指定版本时, RIP的默认版本是: 发1, 收1/2

指定V1: 发1收1

指定V2: 发2收2.

在全局配置, 对所有接口生效:

R1(config)#router rip

R1(config-router)#version 1

R1(config-router)#version 2

对特定接口, 指定RIP版本, (接口的优先级一般比全局要高)

```
interface serial 0
    ip rip send version 1/2
    ip rip receive version 1/2
```

V1与V2不兼容~!

V1: 不携带子网掩码, 不携带下一跳信息, 向广播地址发送路由更新.

V2: 携带子网掩码, 携带下一跳信息, 向组播地址 (224. 0. 0. 9) 发送路由更新.

#### LAB4:RIP的单播更新: (Unicast-Update)/被动接口

(适用于物理链路上, 无法支持广播/组播流量的情况)

Step1: 让需要进行单播更新的接口, 不再发送广播/组播更新.

1-1: 需要PASS的接口很少时:

```
R5(config-router)#passive-interface serial 0
```

(让特定一个接口不再发送广播/组播的路由更新.)

1-2: 需要pass的接口很多时:

```
R2(config-router)#passive-interface default
```

(让所有接口都不发送广播/组播的路由更新. =(passive 所有的接口)

```
R3(config-router)#no passive-interface serial 0
```

(单独让一个接口可以发送广播/组播更新)

Step2: 让已经Pass的接口, 发送单播的路由更新.

```
R3(config-router)#neighbor 35.0.0.5
```

```
R5(config-router)#neighbor 35.0.0.3 (对方的IP)
```

```
debug ip rip (察看RIP路由的收发情况)
```

#### LAB5:RIPv2的手工汇总

step1:关闭自动汇总

```
router rip
  no auto-summary
```

手工汇总

在需要进行路由汇总的出接口:

```
R3(config-if-S1)#IP summary-address rip 175.100.0.0
255.255.192.0
```

在运行RIP的接口中进行手工汇总,把汇总后的结果从接口通告出去

```
R1(config-if)#ip summary-address eigrp 90 172.16.0.0
255.255.248.0
```

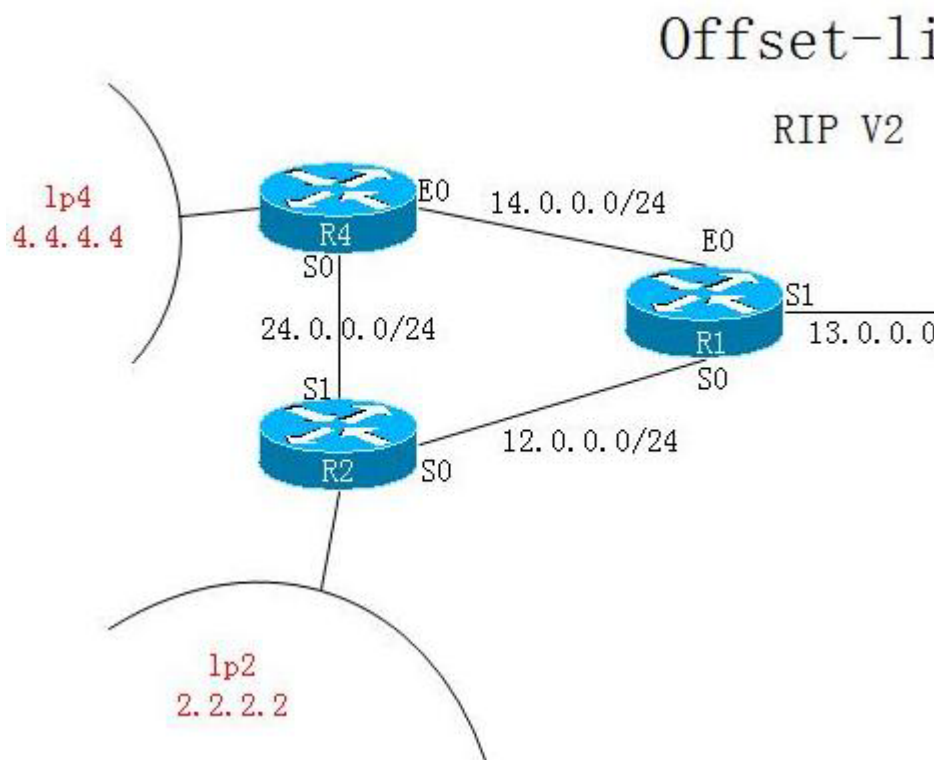
在运行EIGRP的接口中进行手工汇总,把汇总后的结果从接口通告出去

```
R5#
R 175.100.0.0/18
```

### 03晚-动态路由协议-RIP-EIGRP

LAB6:Offset-list/偏置列表,实现RIP路由的控制:

~~~~~  
~~~~~



step0:观察RIP的等价的负载均衡:

R4:  
R 12.0.0.0/8 [120/1] Via

step1:通过offset-list, 实现RIP中的不等价网络拓扑中的, 等价负载均衡:

(R4观察2.2.2.0/24)

出方向的控制(R2当前是汇总的):

Step2:通过ACL定义需要控制的路由:

R2(config):#access-list 2 permit 2.0.0.0 0.0.0.0

Step3:在R2的S1口做路由偏置(+1):

因为对R4来说, 到达R2有两条路, 其中一条是一跳, 另一条是两跳. 如果将原本是一跳的路由人为的+1, 使得它的跳数变成二, 那么R4到R2就能实现负载均衡.

R2#router rip

```
R2(config-router)#offset-list 2 out 1 serial 1
(出口的偏置, 只影响下游路由器, 不影响本机)
```

```
R4#
R 2.0.0.0/8 [120/2]VIA 14.0.0.1, e0
[120/2]VIA 24.0.0.2, Serial0
```

入方向的控制:(no auto-summary)

Step4:通过ACL定义需要控制的路由:

```
R4(config)#access-list 20 permit 2.2.2.0 0.0.0.0
```

step5:

```
R4(config-router)#offset-list 20 in 1 serial 0
(入口的偏置, 本机和下游路由器都受影响)
```

```
R 2.2.2.0/24 [120/2]via 14.0.0.1, e0
[120/2]via 24.0.0.2, s0
```

#### LAB7:RIP authentication/RIP 认证:(保证RIP的网络安全性)

Step1:在全局模式, 配置KEY-CHAIN:

```
R4/R3#
key chain ccnp
key 1
key-string cisco
```

step2:在接口中, 调用key chain:

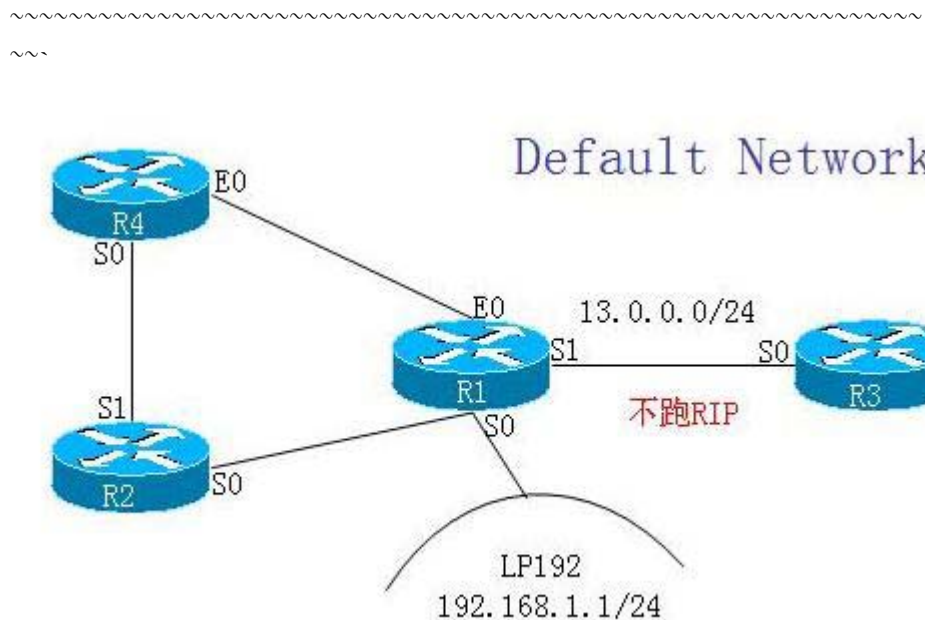
```
R5/R3(config-if)#ip rip authentication key-chain CCNP
```

Step3:在接口中, 选择认证类型:(明文/密文)

```
R1/3 (config-if)#ip rip authentication mode text (明文)(默认的,
可以不打这条命令)
```

```
R1/3 (config-if)#ip rip authentication mode md5 (密文)
```

#### LAB8:default-network:



Step0:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial 13.0.0.3
```

step1:

```
R1(config)#interface loopback 192  
R1:ip add 192.168.1.1 255.255.255.0
```

step2:

```
R1(config)#router rip  
R1(config-router)#network 192.168.1.0
```

Step3:

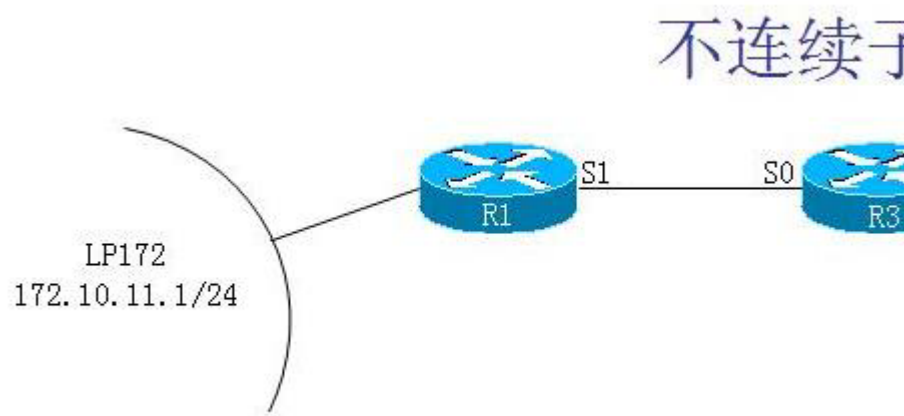
```
R1(config)#IP default-network 192.168.1.0
```

```
R* 0.0.0.0/0
```

```
R*192.168.1.0
```

~~~~~  
~~~~~

LAB10:不连续子网 (Discontinued Network)



V1:

第二地址的虚拟管道,

要求一:与不连续子网同在一个主类网络(172.10.\*.\*)

要求二:与不连续子网的网络长度相同(/26)

Step1:配置第二地址:

Step2:

在R3增加172.10.0.0的网络宣告

```
R3(config)#router rip
```

```
R3(config-router)#network 172.10.0.0
```

V2:不需要第二地址:

Step1:

```
Router rip
```

```
Version 2
```

```
auto-summary(默认)
```

```
R3(config-router)#no network 172.10.0.0
```

Step2:在全网的RIP路由器上,自动关闭汇总

```
~~~~~
~~~~~
```

## EIGRP (Enhanced IGRP)

EIGRP的特:

IGRP/EIGRP都是CISCO的私有协议.

1:是唯一的一种LS/DV的混合协议.

2:Rapid convergence

EIGRP拥有目前最快的网络路由收敛性. (依靠后备路由器/FS)

3. 配置简单, 能够支持中型到大型网络.

4:Incremental updates

增量/触发更新.

5:EIGRP可以支持等价/不等价的负载均衡, 默认是支持等价负载均衡,  
可以通过调整Variance, 来实现不等价的负载均衡.

EIGRP到达同一个目标网络, 可以同时有4条路径, 最多6条.

6:EIGRP默认使用组播 (224. 0. 0. 10) 进行路由更新.

也可以支持单播更新.

IGRP:广播:225. 255. 255. 255

7:EIGRP可以支持VLSM,

支持汇总:默认有自动汇总/手工汇总.

EIGRP可以汇总到超网, CIDR

8:EIGRP可以支持多种网络协议:

IP/IPX/AT (AppleTalk)

EIGRP的3张表:

~~~~~

NO. 1:EIGRP的邻居表:

本路由器的接口, 所直接相连的EIGRP邻居的信息.

NO. 2:EIGRP的拓扑表: (详细的拓扑表)

本路由器, 从自己的邻居那里, 得到去往特定目标网络的(一切)可能的路

本路由器,从自己的邻居那里,得到去往特定目标网络的(一切)可能的路径,都承载/存在于拓扑表中.

NO. 3:从EIGRP形成的路由表:

是EIGRP路由器,从拓扑表中,择优将"去往特定目标网络开销最小的"路由,放入了路由表.

EIGRP和IGRP在AS号相同的情况下,可以实现自动重分布.

EIGRP的Metric值是IGRP的256倍.

EIGRP Packets

~~~~~

1:Hello:用于建立/维护EIGRP邻居关系.

2:Update:更新包:发送路由更新信息.

3:Query:查询包:当路由器丢失了原有的路由后,会向邻居发送"查询请求".

4:Reply:当被查询路由器,收到"查询请求"后,将自己知道的路由信息回应给发起查询路由器.

5.Ack:用于对EIGRP的可靠传输报文的进行确认.(相当于收条)

## 04晚-动态路由协议-EIGRP

### EIGRP的Hello包:

1:EIGRP路由器,向224.0.0.10发送Hello包,同时也监听这个组播地址.

2:Hello包中,包含了EIGRP的K值,两个路由器的K值必需匹配,如果不匹配无法建立邻居.

2:Hello包中, 包含了EIGRP的K值, 两个路由器的K值必需匹配, 如果不匹配无法建立邻居.

K值的默认值:  $K1=K3=1, K2=K4=K5=0$

3. Hello包中, 包含了AS号, 两个路由器的AS号必需相同, 如果不相同, 无法建立邻居

Autonomous-System/自治系统.

4: 两个建立EIGRP邻居关系的路由器的直链接口, 其IP地址必需在同一个IP子网, 否则无法建立邻居.

5. 即使EIGRP的Hello/hold计时器不完全匹配, 只要自己Hello的间隔不超过对方的Hold间隔, 邻居是可以建立的. 但默认情况下, 建议不要修改这两个计时器.

### EIGRP计时器:

Hello Timer:

在大于T1(1.544Mbps)链路, 点对点链路上, 默认5秒发送一次Hello包.

在小于/等于T1的多点链路上, 默认60秒发送一次Hello包.

Hold Timer:

Hold timer默认是Hello Timer的3倍

如果Hold Timer所定义的时间内, 收不到对方的Hello包, 邻居关系就会Reset.

在本路由器上设置hello time和hold time是告诉邻居的!!也就是说在本路由器上设置的hello time和hold time是影响邻居的!!

只要在本路由器上设置的hello time 小于hold time, 那么邻居还是可以建立起来的, 而不管邻居的hello time是多少!!

### EIGRP的可靠传输报文:

Update/Query/Reply, 收到此包后, 需要发送ACK进行确认.

EIGRP的非可靠传输报文:

Hello/Ack, 收到此包后, 不需要进行确认.

### EIGRP Retransmission Policy/重传机制:

稳定的网络:

如果Hold Timer所定义的时间内,收不到对方的Hello包,邻居关系就会Reset.

不稳定的网络:

EIGRP对于可靠传输报文,有必需的确认机制,

如果没有收到对方的确认,那么可靠传输包就会发生重传,极限是16次,

如果达到极限,都没有收到对方的确认,那么就RTO.

RTO:Retransmission TimeOut重传超时

因为EIGRP的是windows size of one (stop-and-wait mechanism)

所以如果在对一组邻居,进行组播的路由更新时,

有个别路由器响应特别慢,

可能导致整个EIGRP网络的收敛效率低下.

解决方案是:

对正常的大部分路由器做组播更新,

对特别慢的路由器,单独进行单播更新.

### EIGRP的DUAL算法:(EIGRP Diffusing Update Algorithm离散更新算法)

AD是通告距离

FD可行性距离

### Successor:(后继路由器)

当前的,去往特定目标网络的,转发路由器  
(正在工作运行的轮胎)

成为Successor的条件:

通过某个路由器,去往特定目标网络的开销/cost/Metric/FD最小的  
路由器,那么这个路由器就是Successor.

通过Successor到达目标网络的这条路径,就能进入路由表,成为去往  
这个目标网络的路由.



Step1:确定路由的入口:  
(也就是访问该目标网络的数据包的出口)

Step2:收集各个路由的入口的BW/DL信息:

```
R2#show interfaces serial 1
      BW 1544 Kbit, DLY 20000
```

(这两个参数是用来控制三层协议的选路,而clock rate是反应链路上真实的速度,这两个参数可以在接口上更改,但仅仅是影响路由协议的选路)

```
R2#show interfaces ethernet 1
      BW 10000 Kbit, DLY 1000
R2#show interfaces loopback 1
      BW 8000000 Kbit, DLY 5000
```

Step3:EIGRP的路由Cost/Metric的计算公式

$$\text{Metric} = \{ (10,000,000/\text{BW}) + (\text{DL}/10) \} * 256$$

BW:单位是:Kbps,多个路由入口中的带宽最小值.

DL:单位是:us(微秒),多个路由入口的延迟之和.

## 05晚-动态路由协议-EIGRP

### LAB3:Wildcard Mask in EIGRP

(通过反掩码,控制运行EIGRP的接口的范围  
有哪些接口在运行EIGRP)

~~~~~  
~~~~~

Wildcard Mask/反掩码的匹配原则:

0:表示准确匹配

1:表示忽略不计

当EIGRP passive接口时,不会向外路由,也不会接收邻居发过来的路由.

(在所有路由器上, 关闭自动汇总)

Step0:

network 0.0.0.0 255.255.255.255 (路由器的所有接口都运行EIGRP)

Step1: 要求以155.\*.\*.\*的所有接口都运行EIGRP:

network 155.0.0.0 0.255.255.255

Step2: 要求以155.2.\*.\*的所有接口都运行EIGRP:

network 155.2.0.0 0.0.255.255

(反掩码不表示网络长度!!)

Step3: 要求以155.2.3.\*的所有接口都在运行EIGRP:

155.2.3.0 0.0.0.255

Step4:

对于A类接口, 不写反掩码时, 其默认的反掩码是0.255.255.255

对于B类接口, 不写反掩码时, 其默认的反掩码是0.0.255.255

对于C类接口, 不写反掩码时, 其默认的反掩码是0.0.0.255

Step5: 要求以215.5.5.5的这个特定接口运行EIGRP:

R5(config-router)#network 215.5.5.5 0.0.0.0

215.5.5.0 0.0.0.0 (这个写法是错误的, 因为路由器上根本找不到这样的接口)

结论:

EIGRP的反掩码

不控制EIGRP的接口在路由条目中, 表现出来的路由长度.

EIGRP/OSPF都是通过这种反掩码的方式,只控制运行协议的接口范围.

LAB4:Automatic Summarization:

~~~~~  
~~~~~

EIGRP在主类网络的边界/(On major network boundaries),  
会发生自动汇总,是默认产生的.

自动汇总会将该子网的路由,汇总到该主类的网络边界:

LAB5:Summarization: Manual(手工汇总,是基于接口的)

~~~~~  
~~~~~

step0:

R3(config)#router eigrp 90

R3(config-router)#no auto-summary

Step1:在汇总路由的出接口中,手工汇总:

R3(config-if-S0)#ip summary-address eigrp 100 175.10.0.0

255.255.192.0 (5)

show ip route 172.16.0.0 255.255.248.0

察看某条路由的详细信息.

Step2:在生成汇总的EIGRP路由器上,既有明细路由,也有汇总:

D 175.10.0.0/18 (AD:5) is a summary, Null0

D 175.10.12.0/24 (AD:90), Ethernet0

但汇总路由一定比明细路由“短”,  
按照“最长配置原则”,一定不会发到空接口,  
而是按照特定明细路由所指示的接口,发送出去.

路由条目的比较步骤:

- 1: 首先按照“最长配置原则”,优先选择路由长度最长的路由.
- 2: 假如,有多条长度相同的路由,才按照AD最小进行比较.

3:如果,连AD也相同,才比较每条路由的Metric值.

Step3:

在生成汇总路由的R3上,

```
R3#show ip route 175.10.0.0 255.255.192.0
```

LAB6:Default-network for EIGRP(在用户连接ISP的边缘路由器上/R3)

~~~~~  
~~~~~

Step0:网络背景:

通常ISP与用户之间是不运行IGP的:

0-1:

在ISP R3上做去往用户的静态路由:

```
R3(config)#ip route 0.0.0.0 0.0.0.0 35.0.0.5
```

0-2:

```
R5(config)#ip route 175.10.0.0 255.255.0.0 serial 0
```

R3#

```
router eigrp 90
```

```
redistribut static metric 1544 2000 255 1 1500
```

Step3:内部的EIGRP路由器上察看由EIGRP生成的默认路由:

R1/2/4#

```
D*EX 0.0.0.0/0 [170/.....]via R3
```

Step4:

内网的EIGRP路由器上察看同EIGRP生成的默认路由:

R1/2/4#

```
D*EX 0.0.0.0/0 [170/.....]Via R1
```

EIGRP产生默认路由的方法B:

~~~~~

Step1:同方法1.

Step2:创建一个私网地址, 提供默认网络:

```
R3#interface loopback192
ip ad 192.168.1.1 255.255.255.0
```

Step3:将私网宣告到EIGRP中:

```
R3#
router eigrp 90
network 192.168.1.0
```

Step4:指定这个网络是默认网络, 它可以在EIGRP域中, 以EIGRP路由的形式存在和传播:

```
R3(config)#ip default-network 192.168.1.0
```

Step5:R1/2/4上察看路由:

```
D* 192.168.1.0/24 [90/
```

LAB7:EIGRP Equal-cost Load Balancing /等价负载均衡:

~~~~~

```
R2#show ip protocols
```

```
EIGRP maximum metric variance 1
```

```
show ip eigrp topoloty detail-linkks
```

```
show ip eigrp topology
```

```
show ip route iegrp
```

LAB8:EIGRP Unequal-cost Load Balancing/不等价

R4观察175.10.12.0/24

P 175.10.12.0/24, 1 successors, FD is 2195456

R4(config)#router eigrp 90

R4(config-router)#variance 2

LAB9:EIGRP Route Authentication(路由协议的, 链路级别的, 链路认证)

~~~~~  
~~~~~

Step1:定义Key-chain

R1/R3#

key chain CCNP

key 1

key-string CISCO

Step2:

R1/R3#

interface s0/s1

ip authentication mode eigrp 100 md5

ip authentication key-chain eigrp 100 CCNP

如果EIGRP的认证失效, 连EIGRP邻居都无法建立.

LAB10:Adjusting the EIGRP Metric Weights(调整EIGRP的K值)

~~~~~

R3#show ip protocols

EIGRP metric weight k1=1, k2=0, k3=1, k4=0, k5=0

Router(config-router)#metric weights tos k1 k2 k3 k4 k5

在全AS的EIGRP路由器, 都必须有相同的K值, 建议不要轻易更改:

R5/R3(config-router)#metric weights 0 1 2 3 4 5

在全AS的EIGRP路由器, 都必须有相同的K值, 建议不要轻易更改:

R5/R3(config-router)#metric weights 0 1 2 3 4 5

实际上, K值是EIGRP协议衡量:

BW/DL/Reliability/Loading/MTU, 这5个衡量路径优劣的不同参数的权重

默认只考虑BW/DL,

∴K1=K3=1 K2=K4=K5=0

06晚-动态路由协议-OSPF

OSPF

理论:

OSPF三张表 (OSPF AD:110)

1. Neighbor table (列出了所有和本路由器直接相连的OSPF邻居)

2. Topology table (LSDB链路状态数据库)

列举了所有从自己的邻居那得到的LSA, 在同一个OSPF区域中的路由器, 都有完全一致的OSPF Database。一个OSPF区域, 就对应着一个OSPF Database。

3. Routing table (从OSPF这个路由协议, 学到的路由。)

在OSPF的数据库中, 通过SPF算法, 计算得到了路由。也称为:

Forwarding Database

区域划分, 及划分的目的:

划分的目的:

1. 提高路由效率:

缩减部分路由器的OSPF的路由条目。

对某些**特定的LSA**, 可以在区域边界 (ABR/ASBR) 上, 实现汇总/控制/过滤。

(通过OSPF的汇总路由/默认路由实现OSPF区域之间的全网互通)

2. 提高网络稳定性:

当某个区域内的一条OSPF路由出现抖动时, 可以有效控制受影响的波及面。

(对于大型的路由协议来说, **稳定**是很重要的一个因素。)

3. OSPF VS. IS-IS的路由可扩展性的对比:

OSPF: 以Area0为BackBone

ISIS: 以Level2的链路为BackBone

OSPF对不同物理链路的处理分类:

OSPF Routing updates and topology information are only passed between FULL adjacent routers.

1. P2P (HDLC/PPP Serial/ Point2Point Sub-if) 一定要求是Full状态.
(没有DR / BDR的选举的, 代表是E1, 两台路由器直连)

2. BMA (Ethernet/TR/FDDI) (代表是局域网)
Broadcast Multi-Access (有DR/BDR的选举的)

3. NBMA (FR/ATM/X.25) (代表是广域网, 如帧中继)
Non-Broadcast Multi-Access (有DR/BDR的选举的)

关于MA (Multi-Access) 网络的DR/BDR的选举:

Step1. 根据OSPF路由器的OSPF接口的优先级选举DR/BDR:

每个接口默认的优先级都是: 1。

其中优先级最大的成为DR, 次大的成为BDR, 其它的都是DR-

Other。

如果有路由器的Pri:0, 放弃DR/BDR的选举, 成为DR-Other。

(OSPF Priority:0~255)

Step2. 如果接口的优先级相同, 将使用Router-ID来进行DR/BDR的选举:

其中Router-ID最大的成为DR, 次大的成为BDR, 其它的都是DR-

Other。

在选出DR/BDR后, 如果有新的优先级更高的路由器加入, 那么新加入的路由器并不会成为DR/BDR, 需要在下次选举中才能生效。

OSPF的SPF算法:

1. 每一个OSPF区域, 就对应着一个独立的OSPF Database (LSDB)

2. 每一个OSPF路由器, 都生成了以自己为根的, 一棵SPF树。

3. 从本路由器出发, 到特定目标网络的整体开销最小的那个路径, 成为最佳路径。

4. 那么这条最佳路径, 就成为OSPF这个协议, 提交给路由表的, 到达这个目标网络的路由。

最长匹配, AD比较, Metric比较

先比较最长匹配如 (10.2.2.0/24, 10.0.0.0/8, 将选10.2.2.0/24, 而不管路由协议的AD), 然后比较AD, 最后比较Metric。

10.2.2.0/24，而不管路由协议的AD），然后比较AD，最后比较Metric。

LSA的传播更新规律（OSPF是LS协议，无需遵循水平分割，DV协议才遵循水平分割。）

Step1. 如果本路由器从来没有收到过此LSA，那么路由器就将其加入LSDB，并且转发/泛洪此LSA，同时继续SPF计算，得出达到此目标的最佳路由。

Step2. 如果本路由器，曾经受到过描述同一个网络的LSA：

2-1. 如果新收到的LSA序号与自己已有的相同，则丢弃此LSA。

2-2. 如果新收到的LSA序号比自己已有的更新，则同Step1，去计算最佳路由。

2-3. 如果新收到的LSA的序号，比自己的更旧，就将自己较新的LSA，发送给源。

OSPF的5种数据包

1. Hello（建立和维护邻居（Neighbor）关系，路由器发送Hello包的缺省时间间隔是10秒）

2. DBD(Database Description)

3. LSR(LinkStatus Request)

4. LSU(LinkStatus Update) (LSA是包含在LSU中的)

5. LSAck

OSPF的Protocol ID:89（EIGRP:88）

在OSPF的Hello包中，影响建立邻居关系的4个关键因素：

Hello/dead interval

Area-ID（链路所在的Area ID）

Authentication password（OSPF认证的密码）

Stub area flag（NSSA标示位）

这四个信息必须匹配才能建立邻居

在BMA网络和点对点网络上，默认的Hello Interval值是10秒，Dead Interval值是40秒。在NBMA网络上，默认的Hello Interval值是30秒，Dead Interval值是120秒。

修改Hello Interval和Dead Interval的值：（在接口上修改）

R1(config-if)#ip ospf hello-interval time(time的取值为1-65535秒)

R1(config-if)#ip ospf dead-interval time(time的取值为1-65535秒)

OSPF邻接关系的建立过程

OSPF邻接关系的建立过程

1. Down（路由器A从运行OSPF的接口以组播地址224.0.0.4发送Hello数据包）
2. Init（所有收到从路由器A发送来的Hello数据包的路由器，都把路由器A添加到自己的邻居Neighbor列表中）
3. Two Way（所有收到路由器A的Hello包的路由器都向其发送一个单点传送的回复Hello包，其中包含有它们的信息。路由器A收到这些信息后，检查这些数据包，把哪些Hello包的邻居域中有自己ID的路由器也加入自己的邻居列表中。在这个过程中同时选举出DR和BDR）
4. Exstart（DR和BDR与其他的路由器建立相邻关系（Adjacency）。）
5. Exchange（由DR向其他路由器发送数据库描述数据包（DBD, Database Description）。DBD有序号，由DR决定DBD的序号）
6. Loading（发送链路状态请求包的过程）
7. Full（路由器及哪个新的链路状态条目添加到它们的链路状态数据库中。当所有的LSR都得到答复时，相邻的路由器就被认为达到了同步并处于“Full”状态了。路由器必须在达到Full状态后才能正常转发数据。此时区域内的每个链路应该都有相同的数据链路状态数据库。）

OSPF数据包的发送地址

DR/BDR notifies LSU on 224.0.0.5（映射到二层MAC地址：010005e00000005）

DR-Other notifies LSU to OSPF DR on 224.0.0.6（映射到二层MAC地址：010005e00000006）

DR负责宣告整个网络的路由更新，BDR或DR-Other只能先把路由更新先发给DR，然后再由DR发给BDR和DR-Other

每次收到LSU，路由器在重新计算路由表之前等待一段时间，默认是5秒。每个LSA都有一个老化（Aging）计时器，到期时由产生该LSA的路由器再发送一个有关该网络的LSU以证实该链路仍然是活跃的，这个Aging时间默认是1800秒。

DR和BDR是在交换Hello数据包的过程中选举出来的，然后其他路由器都与DR和BDR建立相邻关系。每台DR-other路由器都只与DR和BDR建立相邻关系（Adjacency），交换链路状态信息。

LAB1. OSPF的Router-ID（要求全网唯一）

一旦启动OSPF，立刻确定Router-ID

通过此命令察看Router-ID:

```
R1#show ip ospf
```

在OSPF路由器上，确定Router_ID的3个优先级别:

Step1. (建议使用router-id命令来确定Router-ID)

通过router-id命令, 修改Router-ID, 其优先级别最高, 也是建议的。

```
R1(config)#router ospf 110
```

```
R1(config-router)#router-id 100.0.0.1
```

Step2. 假如没有通过router-id命令指定router-id, 那么路由器会自动的将自己的环回口的IP, 作为router-id. 如果存在多个环回口, 那么路由器会自动的悬着一个IP地址最大的那个IP作为自己的Router-ID。

Step3.

如果路由器上, 连一个环回口都没有, 那么路由器会自动从当前是Active (激活状态下: UP/UP) 的物理接口中, 选择IP地址最大的那个接口的IP作为自己的Router-ID。这是很不稳定的, 不建议的方法。

LAB2. 通过反掩码控制有哪些接口, 在运行OSPF (在OSPF/EIGRP中, network命令中携带的反掩码不表示接口网络长度, 而表示运行路由协议的接口的范围, 即有哪些接口在运行OSPF)

反掩码/通配符:wild card bits

反掩码原则:

0:表示准确匹配

1:表示忽略不计

LAB3.

show ip ospf neighbor (detail) (查看路由器的OSPF邻居表, 当前有哪些OSPF的邻居, DR/BDR/DR-other状态)

show ip ospf interface (查看有哪些接口在运行OSPF, 本路由器是DR, 或者BDR, 还是DR-other, 还有优先级)

```
show ip ospf database
```

```
show ip route ospf
```

LAB4. DR/BDR的选举: (只发生在多路访问网络/Multi-Access Network, BMA和NBMA)

1. 在点对点链路, 是没有DR/BDR的选举

2. 在BMA网络中:

2-1. OSPF首先通过优先级, 控制DR/BDR的选举:

优先级越大, 越可能成为DR。OSPF路由器的优先级, 默认是1。

如果需要进行DR的人为控制, 应该建议, 通过OSPF的接口优先级进行控制。

修改特定接口的优先级

```
R1(config)#int s0
```

```
R1(config-if)#ip ospf priority 10
```

```
R1(config-if)#ip ospf priority 10
```

```
R1#clear ip ospf process (清OSPF进程)
```

特别注明：OSPF的优先级是针对某个特定的MA接口而言的，不是针对整个路由器的。

2-2. 如果OSPF路由器的优先级，全部都是默认值1，路由器默认通过Router-ID, 选举DR/BDR，Router-ID最大的成为DR，次大的成为BDR。其余的统统都是DR-other。

3. 在Hub&Spoke的NBMA网络中，中心点（HUB）应该成为DR。

结论：

1. 同一个路由器的不同MA接口，可能在不同的MA网络中，充当不同的DR/BDR/DR-other.

2. 在一个MA网络中：

DR/BDR与所有的邻居都是Full状态，DR-Other与DR/BDR是Full的，但与别的DR-Other是2way状态。

特别注意：

只有Full状态才能交换路由信息。

07晚-动态路由协议-OSPF

Advance OSPF Controlling

OSPF的基本实验配置

建议宣告每一个Router-ID的网络，便于网管Telnet。如Router-ID是195.100.0.1，则宣告网络network 195.100.0.0 0.0.0.255 area 0。

影响OSPF Metric计算的3种操纵方法

在路由协议计算Metric/Cost值时，只计算路由的入口

OSPF计算Cost的公式：

$Cost = 100,000,000 / BW$ ($10^8 / BW$ ，BW的单位是bps)

R1访问35.0.0.0网络的Cost是 $10^8 / BW + 10^8 / BW$ (BW是bps, 即1544Kbps=

R1访问35.0.0.0网络的Cost是 $10^8/BW+10^8/BW$ (BW是bps, 即1544Kbps=1,544,000)

Show interfaces serial 1 (BW 1544 Kbit)

show interfaces ethernet 0 (BW 10,000 Kbit)

show interfaces fast-ethernet 0 (BW 100,000 Kbit)

show interfaces loopback 5 (BW 8,000,000 Kbit, 环回口的OSPF Cost固定为1, 而不管参考带宽的值是多少)

方法1: 直接修改接口Cost值

R2(config)#int s0

R2(config-if)#ip ospf cost 100 (用这个命令修改后, 就不再用 $10^8/BW$ 这个公式计算Cost了)

方法2: 修改接口的带宽BW

R3(config)#int s0

R3(config-if)#bandwidth 2048 (单位是Kbps, 2.048Mbps/E1)

方法3: 修改参考带宽 (分子)

工程/题目需求:

考虑到为了的网络发展, 要求将来再10Gbps链路上, 其OSPF Cost为:

10, 所以我们要把分子更改为100G, 即 10^{11} 。

R1(config)#router ospf 110

R1(config-router)#auto-cost reference-bandwidth 100000 (单位是Mbps)

要求: 一旦准备更改OSPF的参考带宽, 就必须在所有的OSPF路由器 (包含不同区域) 上一起更改。

"ip ospf cost"设置的值要优先于 "auto-cost reference-bandwidth"命令计算出来的值。

特别注意:

1. 路由入口的定义。

2. 同步串行的同步时钟速率: clock rate 2400, 只会影响链路的真实物理速率 (带宽), 不会影响OSPF的Cost的计算。

但接口种配置的 "Bandwidth 2048", 只影响OSPF的Cost的计算, 影响OSPF选路, 不影响真实物理速率

3. 对于以太网, 其速率就等于其接口的带宽。

改接口速率:

int fastethernet 0/1

speed 10(100)

OSPF的路由汇总

RIP和EIGRP的路由汇总是设置在接口上的，它们是DV协议。
链路状态路由协议的路由汇总需要在路由进程中设置，链路状态协议没有自动汇总的特性。

因为在何种路由协议的路由汇总中：
生产的汇总路由包含范围过大，则很可能形成路由黑洞。
生成的汇总路由包含范围过小，则很可能丢失部分明细路由。

所以，默认情况下，在工程/题目没有指定汇总的长度的时候，应该进行最精准的汇总。

1. OSPF的域间汇总，发生在连接不同OSPF区域的ABR上。

0 IA (IA:inter-area)

ABR: (互联了2个，或者2个以上的OSPF区域的路由器。)

2. OSPF的域外汇总，发生在OSPF与别的路由协议相连的ASBR上。

0 E1 (OSPF external type 1)

0 E2 (OSPF external type 2)

ASBR: (在OSPF区域的边界上，互连了OSPF和别的其它路由协议的路由器。)

LAB3. OSPF的域间路由的汇总

Step1. 在Area24中的路由器R4上，模拟4条/26的路由：

175.14.14.1/26

175.14.14.65/26

175.14.14.129/26

175.14.14.193/26

Step2. 在R4上，将明细路由宣告到OSPF中

```
R4(config)#router ospf 110
```

```
R4(config-router)#network 175.14.14.0 0.0.0.255 area 24
```

Step3

Step4. 在明细路由所在的区域Area 24的ABR上，进行OSPF域间路由汇总：

```
R2(config)#router ospf 110
```

```
R2(config-router)#area 24 range 175.14.14.0 255.255.255.0
```

原明细路由所在区域

Step5.

R1/R3, 只有汇总路由

0 IA 175.14.14.0/24

R2, 即有明细路由, 也有汇总路由

R4, 没有汇总, 只有明细。R4代表整个Area24中的所有路由器。

路由汇总黑洞 (null0), 只出现在做路由汇总的路由器上。

在OSPF协议下向区域内产生一条默认路由的语法:

R1 (config-router)#default-information originate [always]

使用Default-information originate命令产生缺省路由的前提是, 使用该命令的路由器必须存在一条默认路由。

如果不使用参数(always), 那么路由器上必须存在一条0/0的默认路由, 它把默认路由通过到整个区域, 否则该命令不起作用。但使用参数(always)时, 无论路由器上是否存在0/0的默认路由, 使用该命令的路由器总会向区域内注入一条默认路由。

LAB4. OSPF的域外路由的汇总。

R3/R5上运行EIGRP,

R5上,

178.15.15.177/28

178.15.15.161/28

178.15.15.129/28

177:10110001

161:10100001

129:10000001

在R3 (ASBR) 上将域外的EIGRP路由, 重分布 (Redistribute) 到OSPF中。

R3(config)#router ospf 110

R3(config-router) redistribute eigrp 90 subnets

在R3 (ASBR) 上将域外的路由, 做OSPF的域外汇总:

```
R3(config)#router ospf 110
R3(config-router)#summanry-address 178.15.15.128
255.255.255.192 (路由长度, 即/26)
```

E1 - OSPF external type 1,
(在路由传播的路径上, OSPF的Cost会随着路径的远进叠加链路的开销,
其Cost会发生改变)

LAB5. OSPF Virtual Links(虚链路)

Step1. 在Area0和非连续区域 (Area35) 的ABR上 (R2/R3)

在virtual link穿越的区域内 (area 123) 的ABR上, 互相指向对方的
Router-ID

R2的Router-ID是192.100.0.2

R3的Router-ID是192.100.0.3

```
R2(config-router)#area 123 virtual-link 192.100.0.3
```

```
R3(config-router)#area 123 virtual-link 192.100.0.2
```

Step2. 检查OSPF的virtual link的连接状态:

```
process 110,
```

```
nbr 192.100.0.2 on ospf_vl1 from loading to full, loading done
```

```
R2#show ip ospf virtual-links
```

```
virtual link ospf_vl0 to router 192.100.0.2 is up Adjacency
```

```
State Full(检查Full, 不能看UP)
```

Step3. 在骨干区域的路由器, 就可以接受到非连续区域的路由。全网路由正常。

Step4. 每个路由器的数据库的个数:

R4:1个 (Area 0)

R5:1个 (Area 35)

R1:1个 (Area 123)

R2:2个 (Area 0, Area 123)

R3:3个 (Area0, Area 35, Area 123)

Step5. OSPF virtual link的应用要点:

5-0. By default, all areas must connect to area 0

5-1. 在大型网络工程中, 由于历史原因, 导致网络扩展不佳, 迫于网络

5-1. 在大型网络工程中，由于历史原因，导致网络扩展不佳，迫于网络扩容的原因，被迫新建OSPF区域，使用OSPF虚链路。

5-2. 在大型网络中，处于网络冗余考虑，选择合适的ABR做OSPF-VL，避免因个别物理链路的中断，导致整个OSPF区域的全网中断。

5-3. 导致OSPF的Area0区域出现双BackBone

09晚-动态路由协议-OSPF

LAB1.3口的FR-SW

Step1. 在R2上关闭路由功能，启用FrameRelay交换功能。

Step2.

配置接口时钟

```
interface serial 1/0
clock rate 2000000
```

```
encapsulation frame-relay
frame-relay lmi-type cisco
frame-relay lmi-type dce
```

Step3. 配置FR的路由：（PVC401/PVC104）

在FR-SW的角度观察：从S1，以PVC401进入交换机的数据，将以PVC104，从S0离开交换机

```
FR-SW2(config-if-S1)#frame-relay route 401 interface serial 0
104
```

从S0，以PVC104进入交换机的数据，将以PVC401，从S1离开交换机

```
FR-SW2(config-if-s0)#frame-relay route 104 interface serial 1
401
```

两端口的FR-SW配置完成。

在FR交换机上检查FR路由：

show frame-relay route（status必须是active才是正确的。）

Step4. 开始配置FR-SW的第3个口，构建R2-R3之间的Tunnel.

```
FR-SW3(config-if-e0)#ip ad 23.0.0.3 255.255.255.0
```

```
FR-SW2(config-if-e0)#ip ad 23.0.0.2 255.255.255.0
```

```
FR-SW3(config)#  
interface tunnel 3  
tunnel source 23.0.0.3  
tunnel destination 23.0.0.2
```

```
FR-SW2(config)#  
interface tunnel 2  
tunnel source 23.0.0.2  
tunnel destination 23.0.0.3
```

Step5. 在FR-SW3的S1口，也要有接口的配置（同Step2）

```
Step6. PVC405/PVC504  
FR-SW2(config-if-s1)#Frame-relay route 405 interface tunnel 2  
1000  
FR-SW3(config-if-s1)#Frame-relay route 504 interface tunnel 3  
1000
```

```
PVC105/PVC501  
FR-SW3(config-if-s1)#Frame-relay route 501 interface tunnel 3  
1001  
FR-SW2(config-if-s0)#Frame-relay route 105 interface tunnel 2  
1001
```

3口FR SW配置完成

FR-SW的基本检查:

1. show frame-relay route
2. 在FR的用户端R1/R4/R5上，show frame-relay pvc, PVC Status必须是Active
3. 在FR的用户端配置IP地址，进行ping的L3测试。show frame-relay map（观察FR的自动反向ARP是否能够成功建立，L3的IP地址，与L2的DLCI的映射）

OSPF Over FR(NBMA网络)

LAB4. Broadcast: (Over FULL Mesh 的NBMA网络中: FR)

Full Mesh: 在N节点的网络中，每个节点都有到别的所有节点的，直接相连的连接/PVC.

在FR网络里面，动态映射dynamic默认允许广播数据通过，手动/静态映

映射 (static) 需要添加broadcast来允许广播数据通过。

Step1. Show frame-relay pvc

Step2. 确认FR支持广播数据流通过

在接口中的映射语句中:

```
frame-relay map ip 100.0.0.5 105 broadcast
```

Step3. 启动OSPF, 指定运行模式是Broadcast:

```
int s0
```

```
ip ospf network broadcast
```

注意: Serial接口如果封装了FR, 其OSPF的运行模式, 默认是NBMA

(NON_BROADCAST), 默认情况下, 不发送OSPF的组播包的, 所以无法自动建立邻居。

如果Serial接口封装的是HDLC/PPP, 其OSPF的运行模式是: P2P。

Step4. 邻居问题:

```
show ip ospf neighbor 察看OSPF邻居
```

Step5. DR问题

确认DR/BDR的正确选择。

因为PVC是Full mesh的, 所以每个路由器都能收到所有别的路由器的Hello包, 所有能够正确的选举出DR和BDR

Step6. 下一跳问题:

手工next-hop:

(通过手工做映射或者自动FR-inverARP, 得到OSPF的路由的下一跳, 的可达性)

LAB5. RunMode:NBMA (NON-BROADCAST)

(Full Mesh或Hub&Spoke的NBMA网络中, 不支持广播流量通过)

Hub&Spoke

Step1. 邻居问题:

当前无法建立邻居, 解决办法: OSPF的单播更新 (通过单播建立OSPF邻居)

在中心点R4上:

```
R4#
```

```
router ospf 110
```

```
neighbor 100.0.0.1
```

```
neighbor 100.0.0.5
```

无需在对端进行同样的nei命令

虽然邻居已经成功建立, 但出现了DR/BDR选举错误, 很可能导致OSPF数

数据库混乱。

```
{  
单播方法:  
RIP: 1.passive interface 2.neighbor xxxxx (两端都要做)  
EIGRP: 直接neighbor xxxxx(两端都要做)  
OSPF:直接neighbor xxxxx(在中心, 单边做)  
}
```

Step2. DR问题

在Hub&Spoke网络拓扑中: 的DR选举原则:

一定要让Hub成为DR, 其余的Spoke成为DR-Other, 无BDR。

在中心点R4#

```
interface Serial 0  
ip ospf priority 10 (通过OSPF的接口优先级, 控制DR的选举)
```

在其余的Spoke, R1/R5

```
Interface Serial 0  
ip ospf priority 0
```

DR/BDR选举正常了, OSPF数据库出现了能够学到OSPF路由, 在中心点访问所有的分支点都是可通的。但是在分支点访问别的分支点时, 路由的下一跳不可达。所有在Spoke之间, 不能互通。

Step3. Next-Hop下一跳问题

对于分支点R1/R5, 其下一跳不可达问题, 不是没有路由, 而是在进行二层封装时, 没有合适的FR的2层封装。

解决办法:

应该通过FR map这个命令, 完成L3:IP L2:DLCI进行映射, 来解决

到达别的分支点的IP数据包, 统统都发往去中心点的PVC

```
R1#frame-relay map ip 100.0.0.5 104
```

```
R5#frame-relay map ip 100.0.0.1 504
```

LAB6. OSPF Run Mode:Point-to-Multipoint Non-broadcast (适合Hub&Spoke的NBMA网络, 不支持广播流量通过的)

Step1: 在每个分支点, 都只做到达中心点的映射, 无需做分支点之间的映射。

Point-to-Multipoint Non-Broadcast的优点之一

Step2: 指定OSPF的运行模式

```
int s0
ip ospf network point-to-multipoint non-broadcast(不发组播)
```

邻居问题：
解决方法同LAB5.

DR问题：
在P2MP No-BR0，是没有DR/BDR的概念的，根本不存在DR问题。

下一跳问题：
由于P2MP的自动建立下一跳特征，导致分支点间无需做映射，

LAB7. OSPF RunMode: Point-to-Multipoint
(Hub&Spoke的NBMA网络中，要求支持广播流量通过)

指定OSPF的运行模式：P2MP

```
int s0
ip ospf network point-to-multipoint
```

邻居问题：
当前模式是P2MP，OSPF会自动发送组播，去尝试建立邻居，同时L2 FR也允许广播，所有邻居自动建立。

DR: 问题
同LAB6

下一跳问题：
同LAB6

10晚-动态路由协议-OSPF

帧中继的子接口选用原则：

0. 在一个封装FR的物理接口上，可以同时承载多条PVC。

为了网络的可扩展性，建议不论在考试环境还是在工程环境中，都应该优先考虑使用子接口

1. 应该创建几个子接口：在一个物理接口中，对应着几个网络，就应该建几个子接口。一个IP子网对应着一个子接口。

2. 选用哪种子接口？（点对点/多点）

2. 选用哪种子接口？（点对点/多点）

在一个子接口中：如果对应着一个点，那么子接口类型应该是P2P。默认情况下，其OSPF Running Mode是Point-to-Point。OSPF对待这种子接口就像对待点对点串行链路一样。

如果对应着多个点，那么子接口类型应该是Multi Point。默认情况下，其OSPF的Running Mode是NBMA。OSPF对待这种子接口就像对待FR的主接口/物理接口一样。

LAB8. 此实验是将主接口配置，全部迁移到多点子接口中：

在同一个网络的NBMA网络中，其主接口或多个子接口的配置，是完全一样的。

LAB9. 通过P2P子接口，实现NBMA网络的优化/简化配置：

Point-to-Point子接口：

默认就有传输广播流量的能力。

默认就可以Ping通自己的接口。

不能使用Frame-relay map，应该使用frame-relay interface-dlci的命令。

Step1. 为每一条PVC，规划一个不同的IP子网：

有多少条PVC，就有多少个点对点子接口，多少个子网。

Step2. 在每一条PVC的两端，都构建一个P2P子接口。

封装主接口（物理接口），然后关闭RARP，不需要配IP地址。

```
encapsulation frame-relay
```

```
no frame-relay revert-arp
```

配置子接口

```
R4(config-if)#interface s0.401 point-to-point
```

```
R4(config-subif)#ip address 100.0.0.2 255.255.255.252
```

```
R4(config-subif)#frame-relay interface-dlci 401 （不能用frame-relay map命令）
```

Step3. 运行OSPF

Step4. 邻居问题：

因为，只要是P2P子接口，其就具备让广播或组播通过的能力（L2）。

只要是P2P子接口，其L3 OSPF的运行模式：默认就是P2P。

所有，OSPF邻居一定可以建立。

Step5. DR问题

在P2P链路上，运行P2P模式，根本没有DR, 所有无DR问题。

Step6. 下一跳问题

因为，每个分支点收到的OSPF路由的下一跳，都是本分公司的哪条PVC的对端，（也就是中心点/总公司）
所有，下一条不在成为问题。

LAB10. OSPF认证（保证网络安全）

按照参与认证的成员，进行分类：

1. 链路认证

（只能确保两路由设备之间的链路是安全的）

链路：两个路由器之间的物理链路。

2. 区域认证

（能够确保整个OSPF区域的所有路由器的安全性）

3. 虚链路认证（Virtual Link）

（能保证跨越多个路由器的，Virtual Link的连接安全性）

按照认证的加密方式，进行分类：

1. 明文认证

2. 密文认证

步骤：

Step1. 在接口配置密码

Step2. 在进程（区域认证）启用认证。或：在接口（链路认证）启用认证。

1-1:链路的明文认证

（int S 1）ip ospf authentication-key cisco1（配置认证密码）

（int S 1）ip ospf authentication（启动明文认证）

1-2. 链路的密文认证

（int S 1）ip ospf message-digest-key 1 md5 cisco1（配置认证密码）

（int S 1）ip ospf authentication message-digest（启动链路认证）

2-1:区域的明文认证

对于区域认证：凡是Area0的路由器，都必需参与认证，如果不参与认证/认证不成功，将无法交换OSPF路由信息。

```
(int s 0)ip ospf authentication-key cisco2
(router ospf 110)area 0 authentication
```

如果两台路由都没有在接口上打上密码, 而且启用了区域或链路的认证, 则也可以正常地建立起邻居.

存在问题:

Area0中的路由器, 没有Area35的路由

原因是: R1-R3之间的VL, 出现中断,

R3相当于是Area0的路由器, 但是没有参加Area0的区域认证。

解决方法: 在建立VL的R3上, 只要宣告:

R3#

```
router ospf 110
```

```
area 0 authentication
```

即使R3没有配置密码, 也可以成功与BackBone进行认证。

又带来了VL的安全问题。

2-2: 区域的密文认证

```
(int s 0)ip ospf message-digest-key 3 md5 cisco2
(router ospf 110)area 0 authentication message-digest
```

(明文认证对应明文的密码!!!密文认证对应密文的密码!!!)

3:key ID 两端的key ID不一样, 即使密码一样也会认证失败!!所以, key ID和密码要两端都一样!!

存在问题:

同: 区域的明文认证

解决VL安全性的办法: 单独进行基于VL的认证。

3-1. VL的明文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3
authentication
(router ospf 110)area 123 virtual-link 100.0.0.3
authentication-key cisco3
```

3-2. V1的密文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3
authentication message-digest
(router ospf 110)area 123 virtual-link 100.0.0.3 message-
digest-key 1 md5 cisco3
```

查看验证类型:

R1#

```
00:18:30: OSPF: Rcv pkt from 192.168.10.65, FastEthernet0/1
: Mismatch Authentication type. Input packet specified type 1,
we use type 0
```

type0:不验证, 默认状态

type1:明文验证

type2:密文验证

IS-IS

能同时支持IP和CLNP网络

Link-state routing protocol

SPF algorithm

Level1:local area route(system ID)

Level2:the route between areas (Area ID)

1. L1 Router

(OSPF internal nonbackbone router): Intra-area routing, L1 LSDB

2. L1/L2 Router (OSPF BackBone ABR)

Inter-area & Inter-area routing

separate L1 and L2 LSDBs

advertises default route to L1 routers in the Area.

3. L2 Router(OSPF backbone routers):

Inter-area routing.

L2 area LSDB.

LAB1. 使用IS-IS协议构建集成的IP网络:

Step1. 规划IS-IS区域, 规划每个路由器的NET (network entity title) /网络实体标识。

NET: -----|-----|-----
 Area-ID, System-ID, NSEL
 1-13byte, 6byte, 1byte(路由器固定是00)

```
router isis
net 49.0035.0000.0000.0002.00
```

Step2. 配置IP地址:

Step3. 在接口中激活IS-IS的运行 (让IS-IS为这个接口所在的网段进行路由)

```
int s0
ip router isis
```

Step4. 察看IS-IS的邻居建立:

```
show clns neighbors
show ip route
```

Step5. 控制接口的IS-IS的Level类型 (默认是L1L2)

```
int s0
isis circuit-type level-2/Level-1
```

11晚-路由控制

(已完成实验)

Route-Controlling

Redistributing/重分布、重分发

将A的路由, 重分布到B。

使A路由协议中的路由, 以外部路由的形式, 进入B路由协议。

default-metric/默认的度量值 (种子度):

A协议中的路由, 进入B路由协议后, 在B协议中表现出来的, 默认的Metric值 (度量值)

default-metric在不同的路由协议中的默认取值:

如果有外部路由（不管此路由源自哪种协议），进入以下的路由协议，默认在这些协议中表再找 Metric是：

1:DV协议（RIP/IGRP/EIGRP），其默认Metric是无穷大(不可达不可用)

Default Seed Metrics

<u>Protocol</u>	<u>Default Seed Metric</u>
RIP	infinity
IGRP/EIGRP	infinity
OSPF	20 for all execept BGP, which is 1
IS-IS	0
BGP	BGP metric is set to IGP metric value.

在两种路由协议进行重分布时：
要特别注意路由的控制/过滤，避免路由环路的出现。
如果控制不慎，可能出现路由环路。

一般不建议使用双向出口的重分布

三种建议解决办法：

- 1) 在一个方向上做重分布，另外一个方向做静态、默认路由。
- 2) 在一个方向上做重分布，反方向做带过滤的重分布。
- 3) 在一个方向上做重分布，反方向做带修改AD的重分布。

LAB2: 将外部路由，重分布EIGRP中：

=====

Step1: 在路由协议的边界路由器R1:

```
R1 (config) #router eigrp 90
R1 (config-router) #redistribute rip
```

没有指定Metric, 默认就是无穷大
EIGRP 区域中的R3/R5都没有RIP的路由。

Step2: 添加Metric参数(把RIP重分布到EIGRP里面)

Step2: 添加Metric参数(把RIP重分布到EIGRP里面)

```
R1 (config-router) #redistribute rip metric 1544 2000 255
1 1500
```

BW DL

Reliability Loading MTU

1544(Bandwidth metric in Kbits per second)

2000(IGRP delay metric, in 10 microsecond units)

255(IGRP reliability metric where 255 is 100% reliable)

1 (IGRP Effective bandwidth metric (Loading) where 255 is 100% loaded)

1500 (IGRP MTU of the path)

R4#ping 35.0.0.5 !!!!!

R4#ping 5.5.5.5 !!!!!

把EIGRP重分布到RIP中:

```
redistribute eigrp 90 metric 3
```

90: Autonomous system number

3: Default metric,,, (本路由器会把Metric为3的外部路由发出去, 也就是说, 下一跳收到重分布过来的路由是3跳, 而下一跳的下一跳收到的就是4跳!)

LAB3: 将外部路由, 重分布OSPF中:

=====

Step1:

```
R1 (config) #router ospf 110
```

```
R1 (config-router) #redistribute rip
```

"Only classful networks will be redistributed"

只能让主类的路由重分布到OSPF, 子网的路由将不被重分布到OSPF中

R5#

```
0 E2 14.0.0.0/8
```

Step2:

Step2:

R1 (config-router) #redistribute rip subnets

所有路由，包括子网路由都被重分布到OSPF中

R5#

0 E2 25.0.0.0/16

0 E2 24.0.0.0/24

0 E2 12.0.0.0/24

0 E2 14.0.0.0/8

Step3:进入OSPF后的外部路由的类型：E1/E2:

默认是E2型，其OSPF的cost/Metric值，不会随着路径远近的变化而变化。（无法反映路径的远近）

R3# 0 E2 14.0.0.0/8 [110/20]

R3# 0 E2 14.0.0.0/8 [110/20]

R1 (config-router) #redistribute EIGRP 90 Subnets metric-type
1

OSPF E1, 会随着路径的远近，其Cost会累加：

R3# 0 E1 14.0.0.0/8 [110/84]

R3# 0 E1 14.0.0.0/8 [110/148]

Step4:进入OSPF后的路由的Metric值：（默认是20）

R1 (config-router) #redistribute EIGRP 90 Subnets metric-type
1 metric 100

LAB 4：将外部路由，重分布IS-IS中：

=====

Step1: 将rip路由重分布到ISIS中

R2 (config) #router isis

R2 (config-router) #redistribute rip

Step2:将ISIS路由，重分布到RIP:

R2 (config) #router rip

R2 (config-router) #redistribute isis metric 2

Step3:IS-IS路由重分布到别的协议时，特殊情况：

在边缘路由器R3上，没有将直链的34.0.0.0路由重分布到外部路由协议RIP中

解决方案：进行重分布直链路由

```
R4 (config) #router rip
```

```
R4 (config-router) #redistribute connected metric 1
```

LAB5：重分布直链路由：

=====

```
R4(config)router A
```

```
R4 (config-router) #redistribute connected metric 1
```

在RIP中, 如果重分布直链路由的话, 在进程中

```
Redistribute connected = redistribute connected 1
```

这时,RIP会把直链路由为1跳发给对方, 也就是说, 在对方的路由表里面显示的是1跳!!

```
redistribute connected 2
```

这时,RIP会把直链路由为1跳发给对方, 也就是说, 在对方的路由表里面显示的是2跳!!

LAB6：重分布静态/默认路由：

=====

```
R4#
```

```
router rip
```

```
redistribute static metric 1
```

```
ip route 0.0.0.0 0.0.0.0 34.0.0.3
```

```
or
```

```
ip route 4.0.0.0 255.0.0.0 34.0.0.3
```

```
ip route 35.0.0.0 255.0.0.0 34.0.0.3
```

```
ip route 34.0.0.0 255.0.0.0 34.0.0.3
```

12晚-路由控制

(已完成实验)

LAB7：RIP中的被动接口 (passive)

=====

被动接口：
只接收RIP路由，而不发送RIP路由

R5
router rip
passive-interface s0

R5#
router rip
passive-interface default
no passive-interface s0
除了s0外，所有接口都不往外发送路由

单播更新：（R1-R2）
router rip
passive-interface s0
neighbor 12.0.0.2（链路对方的RIP）

LAB8：在DV协议（RIP/EIGRP）中，实现基于接口的路由过滤：

=====

EIGRP 90 ---- RIP V2
(4) - (2) - (1) - (3) - (5)

8-1: Distribute-list调用ACL，比较落后：

Step1:通过ACL，定义所需要过滤的路由：
R3(config)#Access-list 3 deny 5.5.5.0(这里的deny跟数据包过滤是一样的意思，都是拒绝的意思)

R3(config)#Access-list 3 permit any(这里的permit跟数据包过滤是一样的意思，都是允许的意思)

Step2:

R3(config)#router rip
R3(config-router)#distribute-list 3
out serial 0
IP access list number
in Filter incoming routing updates

out Filter outgoing routing updates

8-2:比较先进, 通过Distribute-list调用前缀列表
(用RIP举例, 也可用于EIGRP) ACL的缺陷, 无法定义路由的长度

Step1: 使用prefix-list定义所需要过滤的路由: (SEQ5为首, 预留空间方便编辑)

```
ip prefix-list R-10 seq 5 permit 10.1.0.0/24 (跟过滤数据包一样的
permit表示允许, deny表示拒绝)
ip prefix-list R-10 seq 10 deny 10.1.0.0/16
ip prefix-list R-10 seq 15 permit 0.0.0.0/0 le 32 (any)
```

以下是错误的说法:

```
access-list 10 deny 10.1.0.0
                    wildcard
```

```
ip prefix-list R-10 seq 10 deny 10.1.0.0/16
```

```
ip prefix-list R-10
```

Step2:

```
R3(config)#router rip
```

```
R3(config-router)#distribute-list prefix R-10 in serial 1 (调用
了整个R-10的访问列表, 往下匹配)
```

distribute-list(可以调用访问列表, 也可以调用前缀列表!!)

out方向的控制:

只在路由出口方向进行控制, 影响路由下游方向的路由器, 而不影响本路由器.

In方向的控制:

在路由器的入口方向进行控制, 直接影响到本路由器.

```
distribute-list 3 out serial 0
```

Step 3:

R3:

```
R3(config)#router rip
```

```
R3(config-router)#distribute-list 3 in serial 1
```

```
R1# clear ip route * (清除/复位路由表)
```

LAB9:对用户数据包的过滤:

在R3上,对来自本机S0口的,访问目标网络是15.0.0.0/8的数据包,进行过滤:

Step1:通过ACL定义需要进行过滤的数据包.

```
R3 access-list 100 deny ip any 15.0.0.0 0.255.255.255
```

LAB10:在路由协议之间的重分布中,实现路由重分布时的基于协议的路由过滤:

~~~~~

在EIGRP和RIP之间做双向重分布:

```
R1(config)#router eigrp 90
```

```
R1(config-router)#redistribute rip metric 1544 2000 255 1 1500
```

```
R1(config)#router rip
```

```
R1(config-router)#redistribute eigrp 90 metric 1
```

10-1:通过ACL,实现路由控制:

Step1:通过ACL,定义所需要控制的路由:

```
R1(config)#access-list 5 deny 5.5.5.0 0.0.0.255
```

```
R1(config)#access-list 5 permit any
```

```
R1(config)#access-list 5 permit any
```

Step2:在EIGRP进程中:

```
R1(config)#router eigrp 90
R1(config)#distribute-list 5 out rip (理解成rip out)
```

意思是: 禁止把rip中的5.5.5.5路由重分布到EIGRP里面, 也就是说除了R1之外, 其它的EIGRP的路由器都没有5.5.5.5的路由!!!!

distribute-list: Filter networks in routing updates

15: IP access list number

```
out:
  in   Filter incoming routing updates
  out  Filter outgoing routing updates
```

10-2:通过prefix-list ,实现路由控制:

Step1:通过Prefix-list, 定义所需要控制的路由:

```
R1(config)#ip prefix-list DN-15 deny 15.0.0.0/8
R1(config)#ip prifix-list DN-15 permit 0.0.0.0/0 le 32
```

```
Step2:
R1(config)#router eigrp 90
R1(config-router)#distribute-list prefix DN-15 out rip
```

调用一个空的, 不存在的prefix-list, 相当是permit any:

```
R1(config-router)#distribute-list prefix D-15 out rip
```

prefix-list 命名, 都是大小写敏感的:

```
R1(config-router)#distribute-list prefix dn-15 out rip
```

```
~~~~~
~~~~~
```

route-map

如果没有set,则表示set nothing

如果没有match,则表示match any

route-map的特征:

1.route-map 由一组statements/声明,所组成,  
每句声明中,可以包含了Match set 语句.

每句statements有个编号,10/20/30.....  
路由器就按照statements的编号,按从小到大(自上而下)顺序执行.

2.  
match commands specify criteria to be matched

set commands modify matching routes.

一旦匹配/符合特定某些条件

就立刻执行由set所定义参数/操作,并且离开route-map.(而不再往下执行)

3:

the first match found for a route is applied.

once there is a match, leave the route map.

路由器在向下检查匹配条件时,不断地与每个statements中的匹配条件进行比较,  
如果不匹配,则向下继续检查下一个statements;  
如果匹配,则按照set所定义的参数,进行操作,然后立刻离开route-map  
(即使下面还有statements没有进行匹配检查,也不往下检查了)

4:route-,map 的逻辑关系:

in the same line uses a logical OR(水平方向:或关系)  
vertical match uses a logical AND (垂直方向:与关系)

5:在route-map中:

5:在route-map中:

如果没有match,表示:match any (上面statements剩余的any)

如果没有set,表示: set nothing !! (保留其原始的默认值)

#### LAB11:通过route-map,实现路由协议之间的路由过滤/控制:

Step1:通过前缀列表定义所需控制的路由:(通过ACL也可以,但比较落后)

```
R1(config)#ip prefix-list R-15 permit 15.0.0.0/8
R1(config)#ip prefix-list R-25 permit 25.5.0.0/16
R1(config)#ip prefix-list R-35 permit 35.0.0.0/24
```

注意:这里的permit是表示:"匹配",而不是"允许".

Step2:创建用于重分布的路由控制的route-map:

```
R1(config)#route-map RP-SPF permit 10 (permit:允许)
R1(config)#match ip address prefix-list R-15 R-25
R1(config-route-map)#set metric 100
R1(config-route-map)#set metric-type type-1

R1(config)#route-map RP-SPF deny 20 (deny 不允许重分布)
R1(config-router-map)#match ip address prefix-list R-35 没有
Set:Set nothing!)
R1(config)#router-map RP-SPF permit 30
```

step3:在重分布的协议进程中,调用route-map RP-SPF:

```
R1(config)#router ospf 110
R1(config-routiter)#redistribute rip route-map RP-SPF subnets
```

Step 4 :如果没有最后的这句空的route-map,将有什么效果?

```
R1(config)#route-map RP-SPF permit 30
```

剩余的部分路由,都将被Deny.

Step5:在调用时出错:

```
router ospf 110
```

Step5:在调用时出错:

```
router ospf 110
redistribute rip route-map RP-SP subnets
```

空的route-map,意味着deny any.

## 13晚-路由控制与BGP开头

(已完成实验)

administrative distance is a way of ranking the trustworthiness of routing information.administrative distance is expressed as an integer,from 0 to 255.Lower administrative distance is more trustworthy.

administrative distance 会影响路由的选路。

路由选择:

1. longest match(最长匹配)
2. lowest administrative distance(不同协议之间的)
3. lowest metric(rip:hop) (同一种协议)

LAB11:双向/双出口的重分布(在“我的文档”里面有详细的Word文档)

通过调整AD,防止次优路径的产生:

~~~~~

step0:

R1/R2/R4运行OSPF

R1/R4/R3/R5运行RIP

Step1:在RIP/OSPF间,做双向/双出口的重分布:

R1/4#

```
router ospf 110
redistribute rip subnets
```

```
router rip
redistribute ospf 110 metric 1
```

Step2:在观察路由:(次优路径)

```
R1#  
0 E2 34.0.0.0 [110/20] via 12.0.0.2  
0 E2 35.0.0.0/8 [110/20] via 12.0.0.2
```

R4#

```
0 E2 13.0.0.0/8 [110/20] via 24.0.0.2
```

step3:通过ACL, 定义出需要更改AD的路由(RIP路由)(不支持前缀列表)
定义RIP路由:

```
R1/R4#  
access-list 10 permit 13.0.0.0  
access-list 10 permit 34.0.0.0  
access-list 10 permit 35.0.0.0
```

Step4:针对源自RIP, 进入OSPF的路由, 修改AD, 比RIP的AD:120大. (125)

R1/R4#

(AD的修改仅对本路由器生效)

```
router ospf 110  
  
distance 125 0.0.0.0 255.255.255.255 10  
          AD:125      路由的源:any          ACL:10
```

~~~~~

PBR

Policy-Based Routing/策略路由

PBR allows you to implement policies that selectively cause packets to take different

PBR的核心目的:实现网管的人为的网络操纵意图/策略.

PBR的核心目的:实现网管的人为的网络操纵意图/策略.

PBR has the following benefits:

策略路由是基于源的.

普通正常IP路由是基于目标的.

PBR的应用:

Source-based      SP 提供区分服务

Qos

Load sharing

Defining Policies Using a Route map

PBR只应用于路由器入口的数据包.

所有的PBR都应该在入口实现.

PBR大量使用Route-map这个工具.

```
set ip next-hop ip-address.....  
(have an explicit route to the destination)
```

优于路由表

```
set ip default next-hop ip-addredss.....  
(have no explicit route to the destination)
```

次于路由表

LAB12:PBR-(R3#)

~~~~~

Step1:通过ACL, 定义用户群:

```
access-list 10 permit 10.0.0.0 0.255.255.255(permit:匹配)
```

```
access-list 20 permit 20.0.0.0 0.255.255.255
```

Step2:创建route-map, 对不同的用户, 进行不同的策略.

```
route-map PBR permit 10  
match ip address 10  
set ip next-hop 13.0.0.1  
route-map PBR permit 20  
match ip address 20  
set ip next-hop 34.0.0.4  
route-map PBR permit 30
```

Step3:在数据包的入口, 调用route-map PBR

```
R3(config)#inter s1  
R3(config-if)#ip policy route-map PBR
```

Step4:

```
R3#debug ip policy
```

如果匹配的statements中是permit, 意味着进行策略路由

如果匹配的statements中是Deny, 意味着不进行策略(要进行正常路由)

如果连一个statements都匹配不上, 意味着不进行策略路由(正常路由)

Step5:如果不配置

```
“route-map PBR permit 30”
```

BGP:

1:BGP基本理论

2:BGP邻居建立(物理接口)

3:BGP路由宣告

4:BGP路由的优化基本条件(下一跳/同步)

5.....

AS:自治系统

An **AS** is a collection of networks under a single technical administration. (独立的一个技术/管理域)

An **AS** is a collection of networks under a single technical administration. (独立的一个技术/管理域)

IGP:

包括RIP/IGRP/EIGRP/ODR/OSPF/ISIS
运行在**同一个AS中**.

IGP是通过Cost/**Metric**判断路由的优劣, 越小越好.

IGP的主要任务:

更快/更好的正确描述路由信息, 尽快地将数据包送到目的地.
IGP强调收敛速率.

BGP: (v4版本)

运行在**不同的AS之间**, 用于对BGP路由进行控制/策略,

BGP的主要任务: 网管的**人为意图**的集中体现, 强调策略控制, 不强调收敛.

BGP是通过BGP的**属性/Attributes**, 判断多条路径的优劣, 从中进行择优选取.

BGP可以通过网管定义的**策略/Policies**, 实现数据或路由的控制/操纵.

AS: (autonomous Systems/自治系统) (**大**)

独立的技术/管理域, 通常是一个大型公司, 或者组织, 或者国家.

这是区别于IGRP/EIGRP的AS的概念. (**小**)

BGP本身就是一种策略路由/PBR(Policy-Based Routing)
实现网管的**人为意图**的集中体现.

BGP是一种**AS BYAS**的**高级距离向量/DV协议**.

BGP认为, 每经过一个AS是一跳

RIP认为, 每经过一个router是一跳

应该使用BGP的情况:

应该使用BGP的情况:

1:ISP:

当允许AS 1 的数据包穿越AS 2,
但是不允许AS 1的用户访问AS2内部的时候:

2:Multihome/多宿主:

对于一个用户的AS, 如果他同时连接到多个AS/ISP

3:PBR:

当需要对BGP路由/数据, 进行人为控制/操纵的时候.

不该使用BGP的情况:

1. 与ISP只有单连接, 没有同时连接到多个ISP
2. 如果网络硬件设备的档次不够(内存/CPU).
- 3:对BGP路由操纵理解有限, 无法预计BGP的后果.
4. 链路带宽不足.

BGP特征:

1:BGP是高级DV协议.

2:BGP工作在TCP/IP协议栈的4层:TCP179端口.

3:BGP是触发/增量更新的协议

4:BGP通过周期性的发送Keepalive信息, 保证TCP连接的可靠性.

5:BGP使用多种"BGP属性/Attribute", 来衡量路径的优劣.

6:BGP是为巨型网络设计的, 意味着这种路由协议, 可能带来海量的路由和数据.

BGP协议的三张表:

1: BGP的邻居表(BGP neighbor table)

BGP的邻居可以**直接相连**, 也可以**凌空建立**.

2: BGP表(BGP forwarding table/database)

从BGP邻居那里收到的BGP路由, 如果能够满足“优化”的条件, 就可以提交给路由表

3: IP路由表:

上面BGP提交的“优化/最佳”路由, 在经过不同寻路协议之间的“AD的竞争/竞选”之后, 如果获胜, 才能成功的送进路由表.

BGP的4种信息包类型:

1: Open

2: Keepalive

3: Update

4: Notification

Peers=neighbors

Any two routers that have formed a TCP connection, to exchange BGP routing information, are called **BGP peers or neighbors**.

1:

EBGP Neighbor

两个BGP Neighbor分别属于**两个不同的AS**.

EBGP Neighbor通常是**直接相连**的.

2:

IBGP Neighbor

两个BGP Neighbor同时属于**一个相同的AS**.

IBGP Neighbor可以是**直连的**, 也可以是**非直连**的.

14晚-BGP

LAB1: BGP的基本配置:

~~~~~  
~~~~~

Step1: 确认L1/L2的连通性.

Step1:确认L1/L2的连通性.
可以通过Ping, 进行链路测试.

Step2:确保L3的IP路由的通达(是通过IGP实现):

Step3:启动BGP, 检录BGP邻居 (L4 TCP: 179)

Step4: 宣告BGP路由

Step5: 判断BGP优化基础条件

Step6:
(一般都是在同一个AS中完成)

(两个AS之间是不运行IGP的)

=====

在AS123中的所有路由器(R2/R3), 运行IGP:RIP V2

```
router rip
version 2
network 23.0.0.0
no auto-summary
```

测试L3的IGP网络(ping)

```
show ip route
```

step3:启动BGP, 并且立刻指定全局唯一的bgp router-id.

```
R2(config)#router bgp 123
R2(config-router)#bgp router-id 123.0.0.2
```

Step4:构建BGP Neighbor

~~~~~

R5-R3 EBGp

```
R5(config-router)#neighbor 35.0.0.3 remote-as 123
```

```
R3(config-router)#neighbor 35.0.0.5 remote-as 150
```

R3-R2 IBGP:

```
R3(config-router)#neighbor 23.0.0.2 remote-as 123
```

```
R2(config-router)#neighbor 23.0.0.3 remote-as 123
```

R2-R4 EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
```

```
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

Step5:

R4#debug ip bgp 观察BGP邻居建立的过程, 及其经历的5个状态:

1-1:Idle:router is searching

1-2:connect:Completed three-way TCP 179 handshake.

1-3:Open sent: Open message sent

1-4:Open confirm:router received agreement

1-5:Established: Peering is established;BGP Routing begins!!

R5#show tcp brief(察看TCP连接的摘要信息)

| Local Address<br>(state) | Foreign Address<br>state/PfxRcd |       |
|--------------------------|---------------------------------|-------|
| 35.0.0.5.11001           | 35.0.0.3.179                    | ESTAB |

收到的路由条目

R5#show ip bgp summary(察看BGP邻居的简要信息)

BGP router identifier 150.0.0.5, local AS number 150

| Neighbor       | V | AS  | ***** |
|----------------|---|-----|-------|
| State / PfxRcd |   |     |       |
| 35.0.0.3       | 4 | 123 | * * * |

(如果邻居已经建立好了, 这里必需空白) / 0(收到的路由条目)

```

*                               (如果邻居已经建立好了, 这里必需空白) / 0(收到
的路由条目)
18.0.0.8                        4                        100      *   *   *
/ 3

```

0: 没有从这个邻居收到BGP路由, 但是和这个邻居的BGP邻居关系已经建好了.

3: 从这个邻居收到三条BGP路由, 同时和这个邻居的BGP邻居关系已经建好了.

Step6: 通过 network 命令, 宣告BGP路由

BGP Network配置:

```
R5(config)#router bgp 150
```

```
R5(config-router)#network 105.5.0.0 mask 255.255.0.0
                        (接口所在的网络号)      (这个网络的准确路由长度)
```

network命令的含义:

在IGP中:

- 1: 激活接口, 在这个接口中运行此IGP路由协议.  
(在IGP中, 路由协议是向运行的接口发送路由更新)
- 2: 这个路由器的IGP路由协议, 正在为此接口所在的网段进行路由.

在BGP中:

1. 这个路由器的BGP路由协议, 正在为此BGP网段进行路由.  
(在BGP中, BGP路由协议是向BGP Neighbor发送路由更新.  
所以, 没有了“激活接口”的含义)

在BGP路由器上, 检查自己的BGP路由, 是否已经成功地宣告到BGP进程中:

```
R5#show ip bgp (察看BGP表, BGP数据库)
```

```

Network                Next Hop (是指更新源)
*>105.1.0.0/16         0.0.0.0 (源自本路由器)

```

```
R3#sh ip bgp
```

```

Network                Next Hop                Metric

```

| Network            | Next Hop           | Metric |
|--------------------|--------------------|--------|
| LocPrf Weight Path |                    |        |
| * i104.4.4.0/24    | 24.0.0.4 (源自AS140) | 0      |
| 100 0 140 i        |                    |        |
| *> 105.5.0.0/16    | 35.0.0.5 (源自AS150) | 0      |
| 0 150 i            |                    |        |

R2#sh ip bgp

| Network         | Next Hop | Metric | LocPrf | Weight |
|-----------------|----------|--------|--------|--------|
| Path            |          |        |        |        |
| *> 104.4.4.0/24 | 24.0.0.4 | 0      |        | 0      |
| 140 i           |          |        |        |        |
| * i105.5.0.0/16 | 35.0.0.5 | 0      | 100    | 0      |
| 150 i           |          |        |        |        |

Step7:IBGP路由的优化`

6-1:

BGP路由器, 把自己学到的BGP路由, 转发给别的BGP邻居的必要条件: (R3)

每个BGP路由器, 对于特定的某条BGP路由,  
必须是自己已经优化的路由, 才具备转发给别的BGP邻居的能力.

注意:

这是必要条件, 不是充分条件!!!

这意味着:即使自己已经优化, 但此路由器, 可能转发, 也可能不转发.

察看R3向R2发送了哪些BGP路由条目:

R3#show ip bgp neighbors 23.0.0.2 advertised-routes

7-2: (在R2上, 观察从R3这个IBGP邻居获得的BGP路由)

IBGP路由的**必要优化条件**: (以下是必要条件, 而不是充分条件)

- 1: 下一跳可达
- 2: 路由的同步

## 2:路由的同步

提醒：只有在BGP表已经“优化”的才有可能被送进路由表

6-3:察看R2上, 在R3没有作任何更改前, 当前的BGP路由:

注意观察:下一跳问题:

1:本路由器R2, 是否能够到达这条BGP路由的下一跳?

问题的本质: R2有没有去105.0.0.5/16的下一跳路由: 35.0.0.0/24

因为在BGP路由通告中, 当BGP路由器收到相邻的AS发来的路由  
其下一跳是: 相邻的AS的边缘节点 (EBGP的更新源)

所以整个AS123中的所有路由器 (R2/R3) 现在收到的AS150中的所有BGP  
路由, 的下一跳都是35.0.0.5

但是

因为2个AS之间的链路是不运行IBGP的

所以R2没有到达这个下一跳的路由 (35.0.0.0/24)

R2#show ip bgp

| Network        | Next Hop  |
|----------------|-----------|
| * 105.5.0.0/16 | 35.0.0.5  |
|                | 当前下一跳它不可达 |

解决方案:更改下一跳为: 本AS内可以通过IGP学到的路由

R3(config)#router bgp 123

R3(config-router)#neighbor 23.0.0.2 next-hop-self

R3#clear ip bgp \*soft out

R2#show ip bgp

| Network         | Next Hop | Metric | LocPrf | Weight |
|-----------------|----------|--------|--------|--------|
| Path            |          |        |        |        |
| * i105.5.0.0/16 | 23.0.0.3 | 0      | 100    | 0      |
| 150             |          |        |        |        |

7-4: 同步问题 (在新版的IOS里面, 默认是关闭同步的)  
(本质: R2是否能够通过IGP, 得到这条105. 1. 0. 0/16路由)

所谓同步, 是指:

如果R2能够通过IGP (RIP), 学到这条路由 (105. 5. 0. 0/16), 那么就同步.  
(也就是: 满足同步要求).

如果R2不能够通过IGP, 学到这条路由 (105. 5. 0. 0/16), 那就不满足同步,  
(也就是: 不满足同步要求).

BGP路由器, 对于从IBGP邻居学到的路由, 默认要求同步.

但是,

∴实际上R2是不可能通过IGP, 学到另外的一个AS中的路由.

∴只能关闭同步规则, 也就是不遵循同步规则.

解决方案: 关闭同步规则:

R2#

```
router bgp 123
```

```
no synch
```

```
clear ip bgp * (硬清除)
```

(切断原来的TCP179 Session的邻接, 然后重新建立)

R2#show ip bgp

```
*>i 105. 5. 0. 0/16 23. 0. 0. 3
```

step8: EBGp邻居的路由优化问题:

1: 下一跳问题:

∴BGP路由的下一跳是其直连路由,

∴不存在下一跳不可达的问题.

(整个AS140中的100个路由器, 察看到这条BGP路由的下一跳都是:24. 0. 0. 2)

(整个AS123中的100个路由器, 察看到这条BGP路由的下一跳都是:35. 0. 0. 5)

2: 同步问题:

对于EBGP邻居,

∴ 无需遵循同步规则,

∴ 没有同步问题!!

结论: 对于EBGP, 只要是从EBGP邻居那里传来的路由, 在不考虑BGP属性的情况下, 是肯定可以优化的.

```
R4#show ip bgp
*>105.5.0.0/16    24.0.0.2
```

Step9:

EBGP

```
R3#show ip bgp
```

|    | Network      | Next Hop |
|----|--------------|----------|
| *> | 105.5.0.0/16 | 35.0.0.5 |

```
R3#show ip route
B      105.5.0.0 [20/0] via 35.0.0.5,
```

IBGP:

```
R2#show ip bgp
```

|     | Network      | Next Hop |
|-----|--------------|----------|
| *>i | 105.5.0.0/16 | 23.0.0.3 |

```
R2#show ip route
B      105.5.0.0 [200/0] via 23.0.0.3
```

BGP的更新源: (BGP Neighbor Update Source Address)

原则1:

在默认情况下,  
BGP路由器以自己路由表中, 到达对方BGP邻居的地址的那条路由所指示的  
出接口的地址, 作为自己的BGP更新源(源地址)

原则2:

当BGP路由器, 收到邻居发来的BGP信息时, 会检查其源地址,  
然后和自己宣告的Neighbor的目标地址进行比较,  
如果一致, 这个BGP Session才可建立起来.

~~~~~  
~~~~~

LAB2: 验证通过物理接口, 构建IBGP邻居的不稳定性:

Step1: 确认L1/L2通达

Step2: 确认L3的IGP通达 (RIP), 在AS123内

Step3: 通过物理接口, 在R2/R3之间构建IBGP邻居

```
show run | begin router bgp
```

```
R2#neighbor 23.0.0.3 remote-as 123
```

```
R3#neighbor 23.0.0.2 remote-as 123
```

结论:

在IBGP中, 如果使用物理接口构建邻居, 是很不稳定的.  
很可能因为某条物理链路的抖动, 导致IBGP邻居的Flapping/抖动:

建议:

使用环回口/Loopback接口, 构建IBGP邻居.

~~~~~  
~~~~~

LAB3:以Loopback接口作为BGP更新源, 构建稳定的IBGP Session

Step1:为每个IBGP路由器, 构建一个环回口:

Step2:把此Loopback接口, 宣告到IGP (RIP) 中.

R2/R3#

```
router rip
```

```
network 2.0.0.0
```

or

```
network 3.0.0.0
```

step3:

在R2/R3上, 删除原来的, 通过物理接口构建的邻居.

Step4:通过环回口构建IBGP邻居:

4-1:

以对方的环回口, 作为IBGP的目标地址:

```
R2#neighbor 3.3.3.3 remote-as 123
```

```
R3#neighbor 2.2.2.2 remote-as 123
```

4-2:

更改了IBGP邻居后需要把下一跳也做相应更改

```
R2(config-router)#nei 3.3.3.3 next-hop-self
```

```
R3(config-router)#nei 2.2.2.2 next-hop-self
```

注意:删除原物理接口所做的IBGP邻居时, 相应的下一跳将自动删除.

4-3:

以自己的环回口, 作为IBGP连接的源地址:

```
R2#neighbor 3.3.3.3 update-source loopback 2
```

```
R3#neighbor 2.2.2.2 update-source loopback 3
```

step5:

任意切断本AS123 中的物理链路,

只要两个IBGP路由器R2/R3之间, 还有最后一条能够到达, 对方的环回口的路由, IBGP邻居都不会中断.

建议:

凡是构建IBGP, 默认都使用环回口做更新新源, 以构建稳定的IBGP.

## 15晚-BGP

(已完成实验)

LAB4:在两AS间, 存在多条冗余链路的网络环境中:

以LoopBack接口作为EBGP更新源, 构建稳定的EBGP Session.

~~~~~  
~~~~~

Step1:在两AS之间回口.  
, 构建多条冗余链路.

如果两AS间, 没有多条冗余链路, 就使用物理接口构建EBGPP即可.

Step2:为两AS间的EBGP路由器, 构建环

Step3:在各自路由器上, 指定到达对方环回口静态路由:

∴有两条冗余链路,  
∴要有两条到达对方环口的静态路由

```
R5:(config)#  
ip route 3.3.3.3 255.255.255.255 35.0.0.3  
ip route 3.3.3.3 255.255.255.255 100.0.0.3
```

测试:

```
R3#ping 5.5.5.5 source 3.3.3.3 !!!!!!!!!!!!!!!
```

Step4:建立邻居EBGP邻居:

```
R3#router bgp 123  
neighbor 5.5.5.5 remote-as 150  
R5#router bgp 150  
neighbor 3.3.3.3 remote-as 123
```

step5:告知对方,自己的更新源:

```
R3#neighbor 5.5.5.5 update-source loopback 3
```

```
R3#neighbor 3.3.3.3 update-source loopback 5
```

Step6:更改EBGP的TTL值(Time to Live)

∴EBGP TTL值默认是1,

∴EBGP的TTL值最少要设为2

而实际上EBGP多跳这个命令,在不指定其取值时,会自动默认指定为255

```
R5#neighbor 3.3.3.3 EBGP-multihop 2
```

```
R3#neighbor 5.5.5.5 EBGP-multihop(255)
```

TTL:time to live是L3的IP包头中的一个特定字段, IP包每经过一个路由设备,其TTL会自动减1.

如果TTL减到为0,即使路由器有去往目标的路由,也不会继续转发这个IP包.

Step7:测试

```
R3#ping 105.5.5.5 source 3.3.3.3 repeat 1000000 size 15000
```

结论:

在一般情况下,EBGP的邻居关系,是不需要使用环回口构建邻居的.

默认都直接使用物理接口,

在只有单链路的时候,都是使用物理接口构建邻居.

只有在两AS之间,存在多条冗余链路的时候,才需要考虑使用环回口构建EBGP邻居,以确保其EBGP的稳定性.

BGP路由黑洞的解决方案:

解决BGP路由黑洞,可供选择的解决方案/Solution:

1:Redistribute Selected BGP Route into IGP

2:Full-mesh IBGP

3:Part-mesh IBGP+Reflector

4:Confederation

4:Confederation

5:MPLS

LAB5:

part-mesh IBGP, Redistribute Selected BGP Route into IGP, with  
sync (同步/synchronization)

~~~~~  
~~~~~

Step1: 定义需要重分布到IGP中的, BGP的路由:

R3(config)#ip prefix-list B-105 permit 105.1.0.0/16

R2(config)#ip prefix-list B-104 permit 104.1.0.0/24

Step2: 通过Route-map, 控制重分布到IGP的范围,

route-map R3-BGP-RP permit 10

match ip address prefix-list B-105

set metric 1

R2#

route-map R2-BGP-PR permit 10

match ip address prefix-list B-104

set metric 1

小提醒:

不要配置: "route-map R3-BGP-RP permit 20"

一旦配置, 意味着所有一切BGP路由都进入IGP!

Step3: 按照route-map所定义的条件, 将BGP路由注入RIP:

R3#

router rip

redistribute bgp 123 route-map R3-BGP-RP

Step4: 在R2上观察, 105.5.0.0/16,

R2: 同时从RIP和BGP, 都能学到路由, 但因为AD竞争原因,

从RIP所获得的路由, 成功进入路由表,

而从BGP所获得的路由, 不能进入路由表.

R2#show ip route

```
R      105.5.0.0[120/2]
```

```
R2#show ip bgp
r>i105.5.0.0/16      3.3.3.3
```

Step5:在R2观察,如果R2,此时启动了BGP的“同步”,是否还能优化?

结果:可以优化~~~!!!

因为:R2此时通过RIP,学到105.5.0.0/16,  
结果:可以优化!!!!

Step6:在R1观察两条路由:

```
R      104.4.4.0[120/1] via 12.0.0.2
R      105.5.0.0[120/1] via 13.0.0.3
```

R1不再是黑洞!!

Step7:在R2上观察路由的递归查询:

```
R      105.5.0.0[120/2] via 12.0.0.1
C      12.0.0.0 is directly connected, serial0
```

Step8:测试:

```
R4#ping 105.5.5.5 source 104.4.4.4 !!!!!!!!!!!!!
```

LAB6:

Full-mesh IBGP with No-Sync/(Peer Groups)  
~~~~~

Step0:启动BGP进程

```
R1(config)#router bgp 123
```

Step1: 在R1上, 使用Peer-Group(一个模版), 对R2/R3建IBGP邻居:

1-1: 定义peer-group:

```
neighbor R1-PG peer-group
neighbor R1-PG remote-as 123
neighbor R1-PG update-source loopback 1
```

1-2: 对不同的IBGP邻居, 调用peer-group:

```
neighbor 2.2.2.2 peer-group R1-PG
neighbor 3.3.3.3 peer-group R1-PG
```

peer-group 只是一种模版, 只影响本路由器的, 邻居建立的方法.

在R2/R3上, 与R1的邻居建立, 仍然可以使用普通方法建立.

Step2: 确保整个AS123中的所有BGP路由器的, 下一跳, 同步问题能够解决:

2-1:

R1/R2/R3#关闭同步

2-2:

```
R2上, 对R1/R3 Say next-hop-self
R3上, 对R1/R3 Say next-hop-self
```

Step3: 在R1上, 观察所有BGP路由:

```
*>i103.3.3.0/24      3.3.3.3
*>i104.4.4.0/24      2.2.2.2
*>i105.5.5.0/16      3.3.3.3
```

Step4: 观察在R2上的BGP路由的递归查询:

```
R2#
B      105.5.0.0[/0]via 3.3.3.3
R      3.3.3.3[120200/2]via 12.0.0.1
```

```
R    3.3.3.3[120200/2]via 12.0.0.1
C    12.0.0.0 is directly connected, serial0
```

Step5:测试:

```
R4#ping 105.5.5.5 source 104.4.4.4
```

LAB7:

part-mesh IBGP+"RP"(BGP Route-Reflector)

~~~~~  
~~~~~

Step1:

删除R2-R3IBGP Session

删除之前:R2有105.5.0.0/16路由

```
R3(config-router)#no neighbor 2.2.2.2
```

```
R2(config-router)#no neighbor 3.3.3.3
```

删除之后

IBGP Split Horizon Rule:/IBGP的水平分割原则:

(已完成实验)

IBGP的水平分割原则,

by default, routes learned via IBGP are never propagated to
othe IBGP peers.

默认情况下:

对于一个BGP路由器R1来说, 从一个IBGP邻居R3那里学到的BGP路由,

是不是传递给另外的一个IBGP邻居R2的

提醒:EBGP是没有这种规则的!!

Step2:解决方法:BGP Route Reflector /RR
在R1上, 定义R2/R3为自己的“路由反射器的客户端”

2-1:
如果使用Peer-group:

```
R1#neighbor R1-PG route-reflector-client
```

2-2:
如果没有使用peer-group:

```
R1(router bgp 123)#  
  
neighbor 2.2.2.2 route-reflector-client  
  
neighbor 3.3.3.3 route-reflector-client
```

Step3:
问题:在R1上, 定义R2为自己的“路由反射器的客户端”, 但R3不是.
R4能收到105的路由吗?

21晚-BGP

LAB8:

Confederation(联邦) /(community)

~~~~~  
假定:子AS65003/AS65012之间, 是不运行IGP的.

AS65012内部运行的IGP是RIP.

Step1:子AS内的IGP是RIP

```
router rip  
version 2  
network 1.0.0.0
```

```
net 12.0.0.0
no auto-summary
```

step2:

构建R3—R5之间的EBGP:

```
R3(config-router)#neighbor 35.0.0.5 remote-as 150
```

```
R5(config-router)#neighbor 35.0.0.3 remote-as 123
```

构建R2—R4之间的EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
```

```
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

提醒:

在联邦以外的EBGP邻居，他们能查看到的是联邦的大AS号，而不是子AS号。

Step3:在联邦内部的路由上，都要标识他们同属于AS123

```
R1/2/3#bgp confederation identifier 123
```

Step4:在子AS的边界路由器R1/R3上，互相指定对方的子AS号:

```
R1(config-router)#bgp confederation peers 65003
```

```
R3(config-router)#bgp confederation peers 65012
```

Step5:构建联邦子AS中的联邦IBGP:

R1/R2的联邦IBGP:

```
R2#neighbor 1.1.1.1 remote-as 65012
```

```
R2#neighbor 1.1.1.1 update-source loopback 2
```

```
R1#neighbor 2.2.2.2 remote-as 65012
```

```
R1#neighbor 2.2.2.2 update-source loopback 1
```

Step6:构建联邦EBGP:

```
R1#(config-router)#neighbor 13.0.0.3 remote-as 65003
```

```
R3#(config-router)#neighbor 13.0.0.1 remote 65012
```

以上建立BGP邻居的步骤

~~~~~

以下是观察BGP的路由的传递:

Step7:

```
R3# *> 105.5.0.0/16 35.0.0.5
```

但是由于R1无法通过IGP, 获得到达35.0.0.0这个网络的路由,

```
∴R1# * 105.5.0.0/16 35.0.0.5
```

解决办法:

```
R3(config-router)#neighbor 13.0.0.1 next-hop-self
```

```
R1# *>105.5.0.0/16 13.0.0.3
```

Step8:子AS内部到IBGP路由器:

```
R2# *i105.5.0.0/16 13.0.0.3
```

```
R1#(config-router)#neighbor 2.2.2.2 next-hop-self
```

R2

```
*>i105.5.5.0/24 12.0.0.1 0 100 0  
(65003) 150 i
```

```
R4#*> 105.5.0.0/16 24.0.0.2
```

结论:

联邦子AS之间的EBGP的下一跳, 不像普通EBGP那样每经过一个AS, 都发生改变.

而保留原始的BGP下一跳.

Step9:联邦内的同步问题:

```
R2/R3# no sy
```

```
R1(config-router-65012)#sy
```

```
R1#show ip bgp (当R1启动"同步")
```

```
* i104.4.4.0/24 2.2.2.2 (来自联邦IBGP)
```

```
* >105.5.0.0/16 13.0.0.3(来自联邦EBGP)
```

```
R2#sh ip bgp
```

```
R2#(config-router)#sync
```

结论:

联邦子AS之间的同步问题:

如果路由来自联邦IBGP,则需要审查同步条件.

如果路由来自联邦EBGP,则不需要审查同步条件.

Step10:

假定:子AS65003/AS65012之间,是运行可以互通的IGP的.

在R1-R3上,运行IGP:RIP V2

```
R3(config)#router rip
```

```
R3(config-router)#version 2
```

```
R3(config-router)#no auto-summary
```

```
R3(config-router)#network 13.0.0.0
```

查看RIP路由:

```
R2# R 13.0.0.0/24
```

```
R3# R 2.2.2.2/32
```

删除原R1/R3的更改下一跳命令:

```
R1(config-router)#no neighbor 13.0.0.3 next-hop-self
```

```
R3(config-router)#no neighbor 13.0.0.1 next-hop-self
```

```
R3# *>104.4.4.0/24 2.2.2.2
```

LAB9:

community/团体

(相当于一种BGP路由的标识位,常用于标识这条BGP路由应该传播的范围)

Step1:通过前缀列表,定义BGP路由:

```
ip prefix-list B-1 seq 5 permit 40.1.0.0/16(permit:表示匹配)
```

```
ip prefix-list B-2 seq 5 permit 40.2.0.0/16
```

```
ip prefix-list B-3 seq 5 permit 40.3.0.0/16
```

Step2:通过调用前缀列表.

Step2:通过调用前缀列表.

为每条路由,指定其Community:

```
router-map CMM permit 10
```

```
match ip address prefix-list B-1
```

```
set community no-advertise(do not advertise to any peer)  
R4通知R2, 不要发给任何BGP邻居
```

```
router-map CMM permit 20
```

```
match ip address prefix-list B-2  
set community local-AS (Do not send outside local AS)  
联邦的子AS/小AS
```

```
route-map CMM permit 30  
match ip address prefix-list B-3  
set community no-export (do not export to next AS)  
联邦的AS
```

```
route-map CMM permit 40  
(match any, set Nothing!)
```

(如果没有这句话,除了这三条路由以外,其它的路由都不发给R2!!)

Step3:在BGP进程中,对R2调用route-map CMM:

```
R4:neighbor 24.0.0.2 route-map CMM out
```

in:入方向的控制,影响本路由器.

out:路由的出方向的控制,影响对方路由器

Step4:

将community这些标签,发送给对方,

每向前走一个BGP Router,就要"Send-community"推一下.

```
R4#neighbor 24.0.0.2 send-community
```

```
R2#neighbor 1.1.1.1 send-community
```

```
R1#neighbor 13.0.0.3 send-community
```

```
clear ip bgp *
```

```
R2#show ip bgp community
```

```
R5#show ip bgp community
```

```
show ip bgp community no-advertise
```

```
show ip bgp community local-AS
```

```
show ip bgp community no-export
```

```
show ip bgp 40.3.0.0/16
```

Step5:

如果Route-map中, 没有最后的那句空的route-map:

```
"no route-map CMM permit 40"
```

R4向R2通告的bgp路由只有3条:

```
R4#show ip bgp neighbor 24.0.0.2 advertisedp-routes
```

```
*> 40.1.0.0/16
```

```
*> 40.2.0.0/16
```

```
*> 40.3.0.0/16
```

BGP路由过滤的方法之一.

BGP-Summarization-15"

LAB1:非专业汇总(network命令, 是不需要宣告明细路由的.)

Step1:在BGP进程中, 使用network命令, 宣告所需要汇总的路由

```
R1(config-router)#router bgp 140
```

```
R4(config-router)#network 40.0.0.0 mask 255.252.0.0
```

Step2:手工配置静态的汇总路由, 指向空接口,

∴BGP进程, 在路由宣告前,
会检查路由表, 如果能够查到汇总路由,
才会将此汇总路由宣告到BGP进程中.

∴

```
R4(config)#ip route 40.0.0.0 255.252.0.0 null 0(空接口)
```

在R5上, 可以查看到明细路由/汇总路由,
实际上, 明细路由是不需要的:

Step3:删除原明细BGP路由的宣告:

```
R4(config-router)#router bgp 140
R4(config-router)#no network 40.0.0.0 mask 255.255.0.0
R4(config-router)#no network 40.1.0.0 mask 255.255.0.0
R4(config-router)#no network 40.2.0.0 mask 255.255.0.0
R4(config-router)#no network 40.3.0.0 mask 255.255.0.0
```

R5#show ip bgp

*>40.0.0.0/14

LAB2:专业汇总(推荐方法)

~~~~~

Step1:宣告准确的明细路由:

```
R4(config)#router bgp 140
R4(config-router)#network 40.0.0.0 mask 255.255.0.0
R4(config-router)#network 40.1.0.0 mask 255.255.0.0
R4(config-router)#network 40.2.0.0 mask 255.255.0.0
R4(config-router)#network 40.3.0.0 mask 255.255.0.0
```

不要使用network命令,宣告汇总路由:

```
R4(config-router)#no network 40.0.0.0 mask 255.255.0.0
```

Step2:aggregate-address命令是不需要事先配置汇总路由的.

```
R4(config-router)#aggregate-address 40.0.0.0 255.252.0.0
```

明细路由/汇总路由,都传播出去了

Step3:为了不让明细路由传播出去,调用"summary-only":

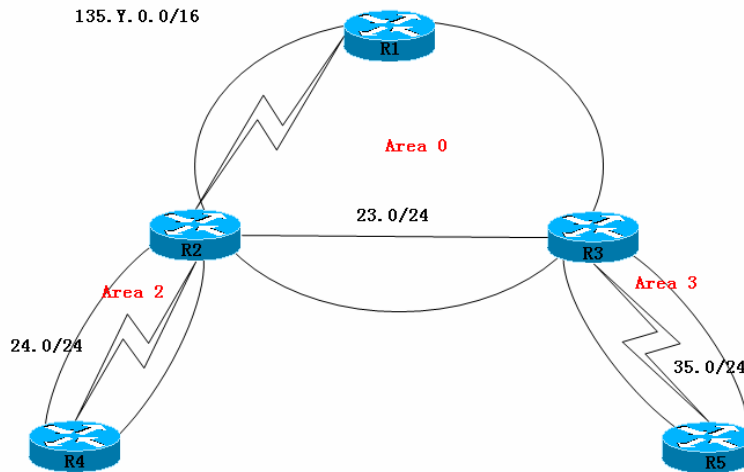
```
R4(config-router)#
aggregate-address 40.0.0.0 255.252.0.0 summary-only
```

只有汇总路由传播出去,而明细路由就被抑制了,不往外发送了.

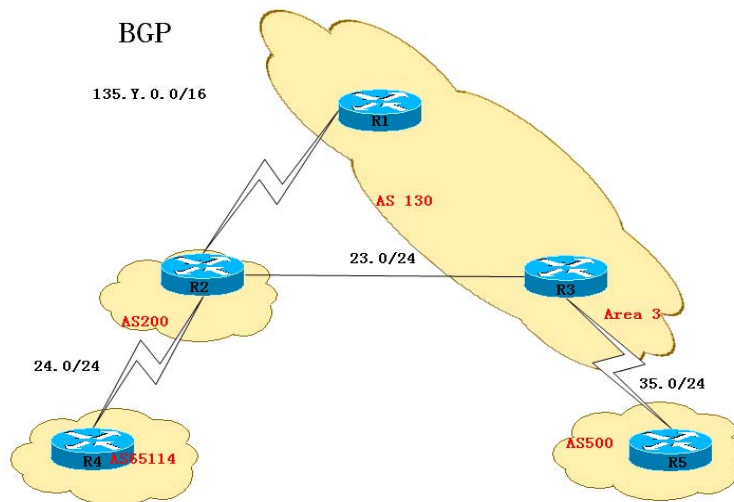
**CCNP+**  
**BGP**

**BGP小结实验一**

## IGP



## BGP



总需求:

1不允许使用静态路由。否则扣20分

2测试实验: 2小时候

IGP需求: 30分钟

1为各router建立loopback0接口 (ip: 135.y.x.x/24,y为RACK号,x为router编号。例如,对r1,x=1),发布在适当的IGP路由进程中,在IGP配置完成后,在每台router上都应该看到所有reouter的loopback0路由。(2)

2。R4上loop8----loop11, ip 4.4.8.4-4.4.11.4/24,重分布到ospf中并汇总(5)

3. 汇总路由在OSPF中传递时COST要发生改变.

4 如果以后运行OSPF的路由器上存在GIBYTE以太口时,其COST值要为5.其他借口COST值也要相应的改变(5)

5area2 的路由条目要尽量优化,并且LSDB中要求存在7类默认路由. (5)

6area 3 中不能存在外部路由,但是应该存在域间路由. (5)

7 R3的E0口永远不能成为DR或者BDR

BGP需求 (70)

1 所有BGP邻居关系均使用LOOPBACK接口做为update sources (5)

2 EBGp: R1--R2 R2--R4 R3-R5) (5)

3 IBGP: R1--R3 (5)

4 R4公布10.0.0.0/24, R2传递给R1时要去掉所有AS信息

第一种方法, 联盟, 第二种方法, remove-private-AS

5 R1上公布10.0.0.0/24. IP 10.0.0.0/24; R2公布10.0.0.0/24 ip: 10.0.0.0/24

6 R3上只能将10.0.0.0/24和10.0.0.0/24传递给R5

route-map 过滤(不好用) 指neighbor加prefix-list直接匹配这两条

7 R5公布10.0.0.0/24, ip: 10.0.0.0/24

8 R3对其进行聚合, 只要聚合路由发给邻居, 不能使用summary-only参数, 不能使用路由过滤.

用抑制列表. AS-PATH(自己试). 社团公告属性.

9 R1, R2, R3, R4都可以PING通R5公布的明细路由.

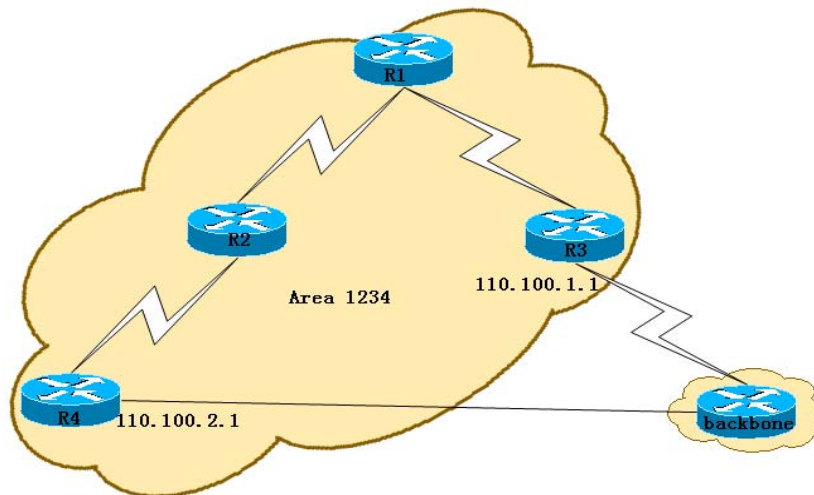
有环路, 可以用PBR(无D技术含量) 修改nexthop(方法最好.) 抑制R1的路由(次好)

10 R2监控R1的10.0.0.0/24 振荡一次给1000惩罚值, 单惩罚值超过2500该路由被抑制, 每30分钟惩罚降低到原来的一半, 当惩罚值降到1200, 该路由被重新启用

半衰期, 重用门限, 抑制门限. 最大抑制时间.

## BGP小结实验二

BGP实验二



IGP全网互通，请用OSPF作为IGP协议，主网段为170.16.0.0/16。不能配置BACKBONE。

IBGP

R1、R2、R3、R4在AS1234里，R1/R2、R1/R3、R2/R4、R2/R3、R3/R4互为IBGP邻居关系，R2是RR，R3、R4是它的客户。IBGP邻居关系要尽可能的稳定。（表示用10.0作更新源）

EBGP

R3、R5分别和BACKBONE建立EBGP邻居关系，BACKBONE属于AS 2，BACKBONE只与AS 1建邻居。

R4与R5直连口的地址为110.100.2.1，R3与R5直连口的地址为110.100.1.1

策略

R3上有一个环回口，10.20.200.200.3/24，宣告进BGP，只能在本AS内传递。（network时设置团体属性）

R1、R2上分别有一个环回口：R1:10.20:

200.200.1.1/27，R2:10.20:200.200.2.2/27。分别将其宣告进BGP。R3/R4分别只发送200.200.1.0/27和200.200.2.0/27的路由给BACKBONE。让BACKBONE访问R1的环回口走R3，访问R2的环回口走R4。

聚合

在R4上面聚合，R4的BGP转发表如下

| Network | Next Hop | Metric | LocPrf | Weight |
|---------|----------|--------|--------|--------|
|---------|----------|--------|--------|--------|

| Weight          | Path          |       |
|-----------------|---------------|-------|
| *>198.16.0.0/19 | 0.0.0.0       | 32768 |
| 2 i             |               |       |
| s>198.16.1.0    | 110.100.2.254 | 1000  |
| 2 i             |               |       |
| *>198.16.2.0    | 110.100.2.254 | 1000  |
| 2 i             |               |       |
| *>198.16.3.0    | 110.100.2.254 | 1000  |
| 2 i             |               |       |
| s>198.16.5.0    | 110.100.2.254 | 1000  |
| 2 i             |               |       |
| *>198.16.22.0   | 110.100.2.254 | 1000  |
| 2 i             |               |       |
| *>198.16.23.0   | 110.100.2.254 | 1000  |
| 2 i             |               |       |

在R3上做策略，当R3与BACKBONE之间链路正常的时候，R1访问BACKBONE走R3；当R3与BACKBONE失去连接时，R1访问BACKBONE走R4。

方法一:1. R3选择EBGP, 设weight  
 bgp bestpath as-path ignore  
 方法二: bgp default local-preference

R3只接受源自于AS 2或源自于AS 2直连AS的路由。

R4上监控198.16.3.0这条路由，如果其FLAPPING，就对其进行惩罚，当198.16.3.0，FLAPing一次后，经过20分钟，惩罚值降到500，当惩罚值超过2500时对其进行抑制，当惩罚值降到700时，将重新起用这条路由。

```

R5
conf t
ho Backbone1-2
no ip do lo
li 0
no exec-t
logg s
end

conf t
int s0
no sh
ip add 110.100.1.254 255.255.255.0

```

```
no sh
ip add 110.100.1.254 255.255.255.0
```

```
int e0
no sh
ip add 110.100.2.254 255.255.255.0
```

```
int lo1
ip add 198.16.1.1 255.255.255.0
int lo2
ip add 198.16.2.1 255.255.255.0
int lo3
ip add 198.16.3.1 255.255.255.0
int lo4
ip add 198.16.5.1 255.255.255.0
int lo5
ip add 198.16.22.1 255.255.255.0
int lo6
ip add 198.16.23.1 255.255.255.0
```

```
router bgp 2
no au
no sy
bgp router-id 170.16.5.5
net 198.16.1.0
net 198.16.2.0
net 198.16.3.0
net 198.16.5.0
net 198.16.22.0
net 198.16.23.0
nei 110.100.1.1 remote 1
nei 110.100.1.1 route-map AS out
nei 110.100.2.1 remote 1
exit
```

```
route-map AS per 10
set as-path prepend 3
```

```
exit
```

```
~~~~~
~~~~~
```

## BGP笔记1

www.net130.com

BGP知识点:

1、BGP的一些特性以及BGP的三张表

IGP: 一个AS内部;

单一的选路, 简单的策略

要求收敛快

相对于BGP能保存的路由数量少

BGP: AS之间

丰富的策略

稳定, 收敛慢

BGP的特性:

1、基于TCP连接, 所以可靠, 端口179

2、定时发送keepalive来检验链路的连通性 (默认60s)

3、增量的触发更新

4、丰富的度量值

5、可以组建大型网络

IGP:

它执行拓扑发现

它尽力完成快速收敛

它需要周期性的更新来确保路由选择信息的准确性

它受同一个管理机构的管理

它采取了共同的路由选择策略

它提供了有限的策略控制能力

IGP不适合当域间路由选择协议

一种域间路由选择协议应该能够提供广泛的策略控制

当前缀的数量处于internet

三张表:

路由表: sh ip route

转发表: sh ip bgp

邻居表: sh ip bgp summary

二、EBGP和IBGP的比较, 各有哪些需求, 解决IBGP路由黑洞问题的几种方法

1、一个路由器只能运行1个bgp的进程

router bgp AS号

1--65535, 其中, 64512--65535是私有, 不能在公网上传输

IBGP: 在一个AS里使用IBGP可以减少分区域IGP路由条目  
IBGP不需要直连  
下一跳需要可达

EBGP: 丰富的选路策略  
需要直连N

IBGP表和IGP表要求同步, 只要从IGP学到的路由, AS内的路由器都认为目的可达

关闭同步作用: 关闭同步, 使其同步 (含义: 关闭BGP同步, 使IP路由表和BGP转发表同步)

当IBGP全互联或包的走向链路全部运行BGP时, 可以关闭同步。

关闭同步后所执行的动作: 把BGP转发表中的路由条目放进IP路由表中

EBGP和IBGP的比较:

EBGP运行在AS与AS之间的边界路由器上, 默认情况下, 需要直连, 如果不是直连, 必须指EBGP多跳

### 三、neighbor和network的含义

neighbor x.x.x.x remote-as as号

对方访问我: 允许这个地址来访问我的179端口

我访问对方: 访问这个地址的179端口

所有有两条TCP会话

基本配置: router bgp 1  
no au  
no s  
bgp router-id

查看TCP的会话: sh tcp bri

BGP neighbor 语句的含义:

1、允许邻居以neighbor这个源地址访问我的179端口, 但并没有指明访问本路由器的那个地址, 只检查源地址

2、本路由器以自己的更新源地址访问neighbor这个地址的179端口, 是否可以建立tcp连接要看邻居是否允许我的源地址

network的含义:  
IGP:

BGP:

- 1、把条目引入BGP，使可在bgp中传递
- 2、可以告诉对方，到这个网络可以走我

在auto情况下，当用net 10.1.1.0 mask 255.255.255.0  
net 10.0.0.0

此时会将路由表里有的主类条目引入BGP

在no-auto情况下，需要严格匹配

BGP的防环:

- 1、AS-Path 用于EBGP
- 2、IBGP的水平分割
- 3、class-id
- 4、RR会产生origin-id
- 5、从IGP邻居学到的路由，再将其network。

邻居的FLAPPING:

BGP的包:

open  
keepalive  
update  
notification

BGP的状态:

idle 初始，终止状态，当收到出错时就回到idle状态  
connect 尝试三次握手  
active 正在建立三次TCP链接  
open send 发open，AS号，版本  
open confirm 版本向下兼容，确认  
establish 建立

holdtime 180s  
keepalive 60s  
hold 0时不发 keepalive  
hold 3倍于keepalive  
两边不匹配，以小的为准

dampening路由惩罚

当一次up后又一次down，叫一次flapping，会进行一次路由惩罚。（一次惩罚默认1000）

抑制门限：默认2000（抑制了会在转发表内打上D）

最大抑制时间：  $2 * (\text{最大抑制时间} / \text{半衰期}) * \text{重用门限} = \text{最大抑制时间}$   
（最大抑制时间是半衰期的4倍）

每隔一段时间，其值会降到该值的一半，叫半衰期（15分钟）

重用门限：750

dampening每10s降一次

BGP的dampening只能对EGBP的路由产生效果

```
router bgp 100
```

```
bgp dampening
```

 后可接router-map对某条路由进行控制

如果没有对某条路由设置dampening，它会直接把路由删掉的。

联邦：为了减少IBGP全互联的条数。要求联邦内的所有路由器都要运行联邦，

即每个路由器都要打：bgp confederation identifier AS号

联邦的配置顺序：

```
router bgp 65001
```

```
no au
```

```
no s
```

```
bgp router-id
```

```
bgp confederation in peers 65002
```

```
nei 12.2 remote 65002
```

选路时：EBGP>联邦EBGP>IBGP

IBGP发路由信息5s，EBGP是30s

路由反射器：也是为了解决IBGP的全互联

成为RR的命令：nei 2.2.2.2 route-reflector-client 指明了2.2.2.2  
这条路由器就成为了我的客户，它自己并不知道

同一个簇（指class-id相同）的客户

非客户不能通过客户与RR来建立邻居关系；

客户不能通过非客户与RR建立邻居关系

大型的AS里面，经过不同的簇，簇ID会集中在簇产生的path中，传递给我的EBGP邻居时，就把簇id的path删掉了

在老版本的IOS，必须客户全互连才能起用对等体

命令：`no bgp client-to-client` 表示当我的客户全互连时，不要将客户来的路由信息反射回给该客户

对等体组：减少配置命令行，拥有共同的出站策略，可以设置不同的进站策略。对等体组可以不在一个AS内

BGP实验列表：

实验1、向BGP中注入IGP路由（宣告，重分发）

实验2、向IGP中注入BGP路由（注入EBGP路由和注入）

实验3、没有

实验5、EBGP多跳

实验7、使用LOCAL\_PREF

实验8、使用MED

nei 2.2.2.2 maximum-prefix(指希望从邻居收到多少条路由)

限制允许自己接收的最大前缀条目来保护它的边界路由器

实验9、BGP选路绕圈（几种方法）

修改holdtime（默认是多少，最小是多少）和advertisement-interval(EBGP, EBGP)

neighbor命令的体会试验（一边指环回口，一边指接口地址，没有更新源）

隐藏命令：`bgp bestpath as-path ignore` 忽略as-path在选路中使用

`bgp always-compare-med`（不同AS传给我的metric值，默认是不比较的，当打了这条命令时就比较）

`bgp bestpath med confed` 可以比较联盟内的metric值

`bgp bestpath compare-routerid`

邻居的FLAPPING

auto-summary与network命令相互作用

BGP选路时一些特定的BGP命令

产生默认路由与router-map相结合

BGP的负载均衡

过滤列表与正则表达式

收多少前缀

BGP的链路认证

BGP的链路认证

删除私有AS号的两种方法（联盟和remove-as）

RR与关闭RR的路由

## BGP笔记2

### BGP知识点

1. BGP的一些特性以及BGP的三张表
2. EBGp和IBGP的比较, 各有那些需求, 解决IBGP路由黑洞问题的几种方法.
3. nei和network语句的含义. 以及邻居的FLAPPING, 下一跳在多路访问网络总的实现, 解决IBG同步的几种方法,
4. BGP的各种包, 以及建邻居的各种状态.
5. BGP聚合路由以及各参数
6. BGP的各种属性, BGP选路原则, 以及路由最优的必要条件.
7. dampening. 联邦, 反射器, 对等体组

### 什么时候需要用到BGP

1. 穿越ISP的时候
2. 对路由进行策略控制
3. 多出口

不能用BGP

硬件  
带宽  
单出口

不能熟练操作BGP的时候.

~~~~~  
~~~~~

### BGP IGP 的比较

BGP特性

AS之间的.

有丰富的策略选路.

稳定, 收敛慢

INTERNET

IGP

一各AS内部

单一的选路, 做简单的策略

收敛快

相对BGP的路由条目少

BGP特性

1. 可靠传输, 基于TCP协议179端口.

2. 定时发送keepalive (60秒发送一次)

3. 增量的触发更新

4. 丰富的度量值

5. 可以构架扩展的巨大的网络.

IGP的特性

他执行拓扑发现

它尽力完成快速收敛

它需要周期性的更新来确保路由选择信息的准确性

它受同一个策略机构的控制

采取了共同的路由选择策略

提供了有限的策略控制能力.

IGP不适合单域间路由选择协议

一种域间路由协议应该能够提供广泛的策略控制, 当前缀的熟练处于internet水平时, IGP路由的周期性刷新特性是不具有扩展能力的

route-server. ip. att. net

~~~~~  
~~~~~

BGP的三张表

show ip route bgp 路由表

show ip bgp 转发表

show ip bgp summary 邻居表

clear ip bgp \* soft 软清影响对方, 硬清影响自己.

~~~~~  
~~~~~

~~~~~

router bgp 1~65535

64512~65535是私有的,不能在公网上传递

EBGP是指两个路由器分属于两个AS的路由器建立的BGP邻居叫EBGP

IBGP是指相同的AS内的两台路由器建立的BGP邻居

EBGP,多跳.用于冗余

建议写精确的多少跳

关闭同步的要求

IBGP 全互连的时候

开启BGP重分布的时候

EBGP运行在AS与AS之间的边界路由器上,默认情况下,需要直连,如果不是直连,必须指EBGP多跳,定义了AS的边界,既定义了管理的边界,体现了管理控制

IBGP运行在AS内部,不需要直连,IBGP建议full mesh,只要减少了区域IGP中的路由.

~~~~~

~~~~~

neighbor命令的含义.

neighbor X. x. x. x remote-ad X

允许对方的X. X. X. X来访问我的179端口

我也是访问x. x. x. x的179端口.

network命令的含义

BGP

IGP

把路由条目引入BGP

匹配一个网段

告诉对方,到该网络可以走我这

宣告进这个协议

auto summar命令.

在no au的情况下network的时候必须严格匹配 不能network主类

在au的情况下,可以将主类的也宣告进来.

~~~~~

~~~~~

BGP防环

1 AP-PATH ebgp

2 IBGP 水平分割 从EBGP学到的路由不会发给另外的IBGP邻居

3 class-ID. 在IBGP中不会最佳as-path. 就引入class-id如果路由反射器从非客户端收到的路由,的class-id是自己的,它不收.

4 origin Id 标识了路由是从某某路由器产生的. RR在origin的地方

- 3 class-ID. 在IBGP中不会最佳as-path. 就引入class-id如果路由反射器从非客户端收到的路由, 的class-id是自己的, 它不收.
- 4 origin Id 标识了路由是从某某路由器产生的. RR在origin的地方产生一个ID是产生该路由的RID. 当源收到该路由的origin的ID是自己时. 自己不收.
- 5 从IGP学到的路由下一跳时自己的接口. 就抑制. 不收

思考MA网络下一跳,,,根指neighbor有关.

如果使用了next-hop unchanged 冲IGP学到的路由就无发发送出去。也会写入路由表失败。

~~~~~  
~~~~~

Open

Keepalive

Update

Notification

Idle State	初始状态
Connect State	尝试3次握手
Active State	上次握手
OpenSent State	发open包
OpenConfirm State	确认OPEN
Established State	建立成功

keepalive 默认60秒 可以时0秒既不发

holddown 默认180秒. 默认keepalive的3陪.

如果两边的open包的keepalive合holddown时间不一致, 就以小的为准.

收到路由条目, 发给IBGP邻居. 时5秒. 发给EBGP邻居是30秒

~~~~~  
~~~~~

同步

从BGP学到的路由, IGP也要有该路由。

当IBGP全互联（指的是在包的走向连路上全部运行BGP的时候, 可以关闭同步。

关闭同步, 使其同步.

以关闭同步。

关闭同步, 使其同步.

就是说关闭BGP的同步, 把BGP的转发表的条目放入IP ROUTE, 使其BGP转发表与IP ROUTE同步.

~~~~~  
~~~~~

BGP的聚合... 相当于汇总.

IGP的汇总.

RIPv1 { 自动 可以
 { 手工 可以

RIPv2 { 自动 基于database的汇总
 { 手工 可以

EIGRP { 自动 只能对本路由器的直连接口产生路由
 { 手工 可以本路由器也可以是学来的路由. 当它还是会发收到的
 明细

OSPF { 手工

~~~~~  
~~~~~

sm

对network有关:指要IP ROUTE里有以条明细, 就可以将这条明细的主类也network进来....

还有就是对重分布的时候有关... 只重分布, 主类.

BGP的汇总

 {自动
 {net 写以条到null0的汇总路由, 然后在network进BGP
 aggregate 只要在BGP的转发表中有一条明细. 必须要在
BGP转发表中), 就可以聚合.

R1(config-router)#aggregate-address ?

advertise-map Set

condition to advertise attribute

被advertise-map匹

配的就继承它的属性. 没背匹配的就不继承.

as-set

Generate AS set path information

继承每条明细的AS-

PATH {里面的算一个}参与选路.

attributes of aggregate

Nlri aggregate applies to

parameters of aggregate

能清除AS-PATH属性.

Filter more specific routes from updates

总不发明细

Conditionally filter more specific routes from updates

发汇总抑制指定的
抑制 在指邻居的时候可以在后面加unsuppress-map对指定邻居解除抑制

AS-SET

将汇总内的明细. 的AS-PAST以以个乱序排序的起源的 {} 内. {} 内的AS算以个, 并参与选路.

可以继承as-path

local-prf

commu的属性

如果继承了commu属性, 要去掉的话, 只有在聚合的同时用route-map把commumiyt去掉.. 用none

继承每条明细的AS-

attribute-map Set

同route-map

nlri

route-map Set

可清除所有属性, 不

summary-only

抑制明细 只发汇

suppress-map

发汇总抑制指定的

~~~~~  
~~~~~

weight

local-preference

本地起源

as-path

起源代码

med

EBGP优于IBGP

最近的IGP邻居

最老的

最低的router-id

最老的
最低的router-id

```
~~~~~  
~~~~~  
  
route-map  
set metric-type intenal  
把IGP的metric作为MED的metric的值
```

```
~~~~~  
~~~~~  
  
dampening 路由惩罚 只能对EBGP的邻居产生效果。  
默认0 down up一次加1000  
抑制门限. 到一定数值后就惩罚(默认2000. 最大抑制门限有公式下面.)  
就抑制. 在转发表内打D  
过一段时候后(手工配置, 默认15分钟), 会降到原来的一半(每10秒较少  
一次数值), 叫半衰期. 降到一定数值后就重新启用该路由. 叫重用门限  
(默认750).
```

重用门限*(最大抑制时间*2/半衰期)=最大抑制门限.
router)#bgp dampening route-map 可以控制单条路由的dampening的
数值,....
如果每有对路由进行DEMPENING的话就不抑制, 直接删除.

半衰期, 重用门限, 抑制门限. 最大抑制时间. s

```
~~~~~  
~~~~~  
  
联邦
```

用于解决IBGP的全互连问题.
要求区域内所有路由器支持联邦. 要敲入BGP confederation
identifier (ASid(大的区域ID.))
有联邦内Ebgp关系的路由器要打.
BGP confederation peer (对方联邦内AS号)

```
~~~~~  
~~~~~  
  
路由 反射器  
用起源ID 产生路由的route-ID 用于防环.  
用cluster ID防环
```

EBGP	EBGP	过来的路由	都可以传
给其他客户上上			
IBGP客户	RR	IBGP客户	过来的路由
EBGP邻居, 传给IBGP客户, IBGP非客户			可以传给
IBGP非客户	IBGP非客户	过来的路由	可以传给
EBGP邻居, IBGP客户, 不可传给IBGP非客户			

nei x.x.x.x route-reflector-client 打上这句话就指明了x.x.x.x是我的客户, 我就是RR了. 客户不知道自己成了客户的.

同一个簇标识cluster-id相同.

非客户不能通过客户与RR建立邻居关系.
客户也不能通过非客户跟RR建立邻居关系
里两点有争议.

通过cluster ID防环.

在大型的AS内, 经过不同的簇, 所经过的簇都会记录在簇ID内. 传递给EBGP邻居的时候就把簇ID删除. 传出联邦呢?

思考起源ID会不会传递出区域.

客户发送给RR的路由也会反射给客户, 会再发撤回消息. 所在对等体组的时候, 会撤回所有的. 老版本会出现.

单客户全互连的时候. 打 no bgp client-to-client 这样就不会把客户的路由反射给客户.

~~~~~  
~~~~~

对等体组. 最大的用处. 减少命令行. 拥有共同的出站策略. (减少系统资源, 对于对外策略只缓存一个buffer)可以设置不同的, 进站策略.

重IGB学到的路由下一条不改边, 那么会不会学会自己重IGP公布出去的路由.

~~~~~  
~~~~~

进站过滤, 出站过滤

nei x.x.x.x capability orf prefix-list ? 协商我需要或我只发什么路由.

跟我邻居协商. 我只能发送什么路由过去, 接收, 我只希望从邻居学到什么路由, 每匹配的不要发过来.

到什么路由, 每匹配的不要发过来.
有一条件, 必须在network之前打...

~~~~~  
~~~~~

试验一 向BGP中注入IGP路由(宣告, 重分布)
试验二 向IGP中注入BGP路由(注入EBGP路由合注入IBGP路由)
试验三 没有IGP的BGP(静态)
试验四 IGP三的BGP(同步, 下一跳)(什么情况下可以关闭同步)
试验五 EBGp多跳
试验六 聚合路由(各种map)
试验七 使用LOCAL_PERF
试验八 使用MED
试验九 限制允许自己接收的最大前缀条目来保护它的边界路由器 nei
x.x.x.x maximum-prefix指定从我这个邻居最多接收多少跳路由. i反向
的. 默认75%会警告. |
试验十 BGP选路绕圈(几种办法)
试验十一 修改holdtime (默认十多少, 最小十多少)和advertisement-
interval (IBGP, EBGp)
试验十二 neighbor命令的体会试验(一边指环回口, 一边指物理接口, 没
有更新源)
试验十三 邻居的flaping
试验十四 auto-summary 和network的相互作用
试验十五 BGP选路时一些特定的BGP命令
试验十六 产生默认路由与route-map相结合
试验十七 BGP的负载均衡
maximun-paths 1-6
IBGP要打多个IBGP
默认是EBGP的.
负载均衡默认一条。最多6条

跟IGP相互作用。
IBGP的时候IGP必须负载. BGP才负载
EBGP的时候只要多多少下一跳就就可以负载.
试验十八 过滤列表与正则表达式
nei x.x.x.x filter-list 1 [in|out}
试验十九 收多少前缀
nei x.x.x.x maximum-prefix ?
试验二十 BGP的链路认证
nei x.x.x.x password
试验二十一 删除私有AS号的两种方法(联盟和remove-as)
nei x.x.x.x remove-private-AS 对对等体的成员能不能单独设置
试验二十二 RR与关闭反射RR的路由
实验十三 soft-reconfiguration inbound

实验十三 `soft-reconfiguration inbound`

可以对这个邻居软清。对内存产生一定的影响。会多建一个buffer

试验二十四`keepalive and holdtime`

试验二十五`network`和`backdoor`

`backdoor`当我从EBGP和IGP同时学来一条路由，优选IGP学来的路由。BGP的管理距离改成200

试验二十六`rr`与联盟

实验二十七拆分路由

把聚合的变成明细的。

`bgp inject-map 111 exist-map 222 copy-attributes`

当我有222的时候就拆成111....

`copy-attributes`可以打可以不打。

实验二十八`AS-PATH PREPEND`

只能在EBGP下做. `in`和`out`的反向做是不同效果的。

试验二十九BGP的下一跳

试验三十允许接收经过多少AS的路由

`bgp maxas-limit ?`

试验三十一`LOCAL_AS`

`nei x.x.x.x local-as 1`

我本地的AS是1. `show ip bgp` as是起源与自己的AS号, 然后加三LOCAL-AS的AS号再到PEER那的

进的时候AS也一样会加上, 要去掉的话在进程打`nei x.x.x.x local-as xx no-prepend`

试验三十二 条件路由

`nei 10.1.25.2 advertise-map 111 exist-map 222`

如果222存在就公告111

`nei 10.1.25.2 advertise-map 111 non-exist-map 222`

如果222不存在就公告111

~~~~~  
~~~~~

隐藏命令 `BGP bestpath as ignore` 第四跳选路

`BGP laway-compare-med` 默认不同AS传递给我的med是不比较的. 使用这命令了就比较了,

`BGP bestpath med confed` 可以比较联盟内的不懂MED值.

`BGP bestpath med missing-as-worst` 如果EBGP邻居向我发路由, 如果

BGP bestpath med missing-as-worst 如果EBGP邻居向我发路由, 如果没设Metric. 默认就是最大.

bgp bestpath compare-routerid 忽略最老路由这一选路条件.

~~~~~  
~~~~~

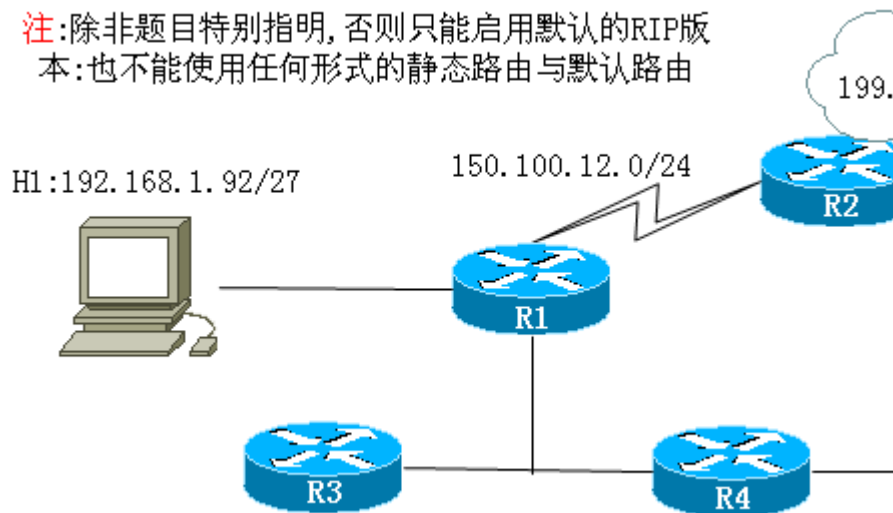
set as-path pre pend last-as 3 只能影响穿过的路由, 可在入方向和出方向做, 表示将最后一个AS号重复出现3次

local AS只能在小的as内传递, no-export能在联邦的as的之间传递.

IGP

IGP实验1

注:除非题目特别指明, 否则只能启用默认的RIP版本:
本:也不能使用任何形式的静态路由与默认路由



IBG实验一

1, r3不能向R4通告路由更新 (20分)

2, R1是链接到internet的路由器, 在R1上面通过RIP向内网注入一条默认路由; (有关上internet的配置不在rip考试范围, 可以不进行配置)

3, 修改R4 rip计时器时间update interval 为5s; invalid 为10S; holddown为20s; 但要求路由在没有收到update后的15s就删除 (20分)

4, 在删除R1上发送出来的缺省路由后, R1与R4之间分别链接主机H1和H2

4. 在删除R1上发送出来的缺省路由后, R1与R4之间分别链接主机H1和H2 (环回口代替);发现不能互相访问对方, 解决

5. R1的S0口上级联一个RIP邻居, 要求仅在R1S0接口上面发送和接收RIPV2的更新, 并且要求使用MD5加密验证;加密为"wolf";配置成功的话, 你应该从R2上面收到一些路由(199. 172. X. 0 X为任意数) (10分);R1向R2发送10网段的明细路由;要求R1/R3/R4能PING通这些路由的IP地址 (如:199. 172. 1. 254) (5分)

6. 你不能用任何手段登陆到R2上查询或修改配置, 但要求在R1上配置-能且只能允许R2成功链接TELNET到R1的"enable"模式(既无需输入密码和键入enable命令), R2将使用2. 2. 2. 2这个地址发出telnet (10分);其他路由器的访问将被拒绝 (10分)

7. R3收到R1发送过来的199. 172. X. 0/24网段路由, 要求奇数路由在路由表里面. metric为10 (5分)

R2初始配置

```
key chain wolf
  key 1
    key-string wolf
interface Loopback0
  ip address 2.2.2.2 255.255.255.255
interface Loopback1
  ip address 199.172.2.254 255.255.255.0
interface Loopback2
  ip address 199.172.3.254 255.255.255.0
interface Loopback3
  ip address 199.172.4.254 255.255.255.0
interface Loopback4
  ip address 199.172.1.254 255.255.255.0
interface Serial0
  ip address 150.100.12.2 255.255.255.0
  ip rip authentication mode md5
  ip rip authentication key-chain wolf
  clockrate 64000
  no shutdown
router rip
  version 2
  network 2.0.0.0
  network 150.100.0.0
  network 199.172.1.0
```

```
network 199.172.2.0
network 199.172.3.0
network 199.172.4.0
```

>q

```
r1#sho run
Building configuration...

*Mar 1 01:49:04.343: %SYS-5-CONFIG_I: Configured from console by
console
Current configuration : 1110 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
key chain huahua
key 1
  key-string wolf
!
!
!
interface Loopback0
ip address 192.168.1.193 255.255.255.224
!
interface Ethernet0
ip address 192.168.1.2 255.255.255.224 secondary
ip address 10.1.134.1 255.255.255.0
ip summary-address rip 192.168.1.192 255.255.255.224
!
interface Serial0
ip address 150.100.12.1 255.255.255.0
ip rip send version 2
ip rip receive version 2
ip rip authentication mode md5
ip rip authentication key-chain huahua
!
interface Serial1
no ip address
```

```
interface Serial1
  no ip address
  shutdown
!
interface BRI0
  no ip address
  shutdown
!
router rip
  network 10.0.0.0
  network 150.100.0.0
  network 192.168.1.0
  no auto-summary
!
no ip http server
ip classless
ip default-network 150.100.0.0
ip route 150.100.0.0 255.255.0.0 150.100.12.0
!
!
!
access-list 1 permit 2.2.2.2
!
!
line con 0
  exec-timeout 0 0
  logging synchronous
line aux 0
line vty 0
  access-class 1 in
  no login
line vty 1 4
  login
!
end
```

```
-----
r2#sho run
Building configuration...
```

```
Current configuration : 1103 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r2
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
```

```
ip subnet-zero
no ip domain lookup
!
!
key chain wolf
key 1
  key-string wolf
!
!
!
interface Loopback0
ip address 2.2.2.2 255.255.255.255
!
interface Loopback1
ip address 199.172.2.254 255.255.255.0
!
interface Loopback2
ip address 199.172.3.254 255.255.255.0
!
interface Loopback3
ip address 199.172.4.254 255.255.255.0
!
interface Loopback4
ip address 199.172.1.254 255.255.255.0
!
interface Ethernet0
no ip address
shutdown
!
interface Serial0
ip address 150.100.12.2 255.255.255.0
ip rip authentication mode md5
ip rip authentication key-chain wolf
clockrate 64000
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router rip
version 2
network 2.0.0.0
network 150.100.0.0
network 199.172.1.0
network 199.172.2.0
network 199.172.3.0
network 199.172.4.0
!
```

```
network 199.172.4.0
!
no ip http server
ip classless
!
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

```
-----
r3#sho run
Building configuration...
```

```
*Mar  1 01:49:09.583: %SYS-5-CONFIG_I: Configured from console by
console
Current configuration : 662 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r3
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Ethernet0
ip address 10.1.134.3 255.255.255.0
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
```

```
interface BRI0
  no ip address
  shutdown
!
router rip
  passive-interface default
  offset-list 1 in 8
  network 10.0.0.0
  neighbor 10.1.134.1
!
no ip http server
ip classless
!
!
!
access-list 1 permit 199.172.1.0 0.0.254.0
!
!
line con 0
  exec-timeout 0 0
  logging synchronous
line aux 0
line vty 0 4
!
end
```

```
-----
r4#sho run
Building configuration...
```

```
Current configuration : 711 bytes
```

```
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r4
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
  ip address 192.168.1.33 255.255.255.224
!
interface Ethernet0
  ip address 192.168.1.1 255.255.255.224 secondary
  ip address 10.1.134.4 255.255.255.0
```

```
ip address 192.168.1.1 255.255.255.224 secondary
ip address 10.1.134.4 255.255.255.0
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router rip
timers basic 5 180 10 15
network 10.0.0.0
network 192.168.1.0
!
no ip http server
ip classless
!
!
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

IGP实验1总结

1.rip更新是分别从自己直链的所有接口发广播出去,发出去的路由条目是除本接口以外的network条目.

R4#

```
00:11:51: RIP: sending v1 update to 255.255.255.255 via
Ethernet0 (10.1.134.4)
00:11:51: RIP: build update entries
00:11:51:      network 192.168.1.0 metric 1
00:11:51: RIP: sending v1 update to 255.255.255.255 via
Loopback1 (192.168.1.33)
00:11:51: RIP: build update entries
00:11:51:      network 10.0.0.0 metric 1
```

R4#

```
00:12:23: RIP: received v1 update from 10.1.134.1 on Ethernet0
00:12:23:      192.168.1.0 in 1 hops
```

当路由器上没接直链路由条目时,会抑制发空的更新包.

R3#

```
*Mar  1 00:14:37.639: RIP: sending v1 update to
255.255.255.255 via Ethernet0 (10.1.134.3)
*Mar  1 00:14:37.643: RIP: build update entries - suppressing
null update
```

2. R4(config)#router rip

```
R4(config-router)#timers basic      5      10      20
15
                                update  invalid  holddown
flush time
```

3. By default, there are three privilege levels on the router.

- privilege level 1 = non-privileged (prompt is router>), the default level for logging in
- privilege level 15 = privileged (prompt is router#), the level after going into enable mode
- privilege level 0 = seldom used, but includes 5 commands: disable, enable, exit, help, and logout

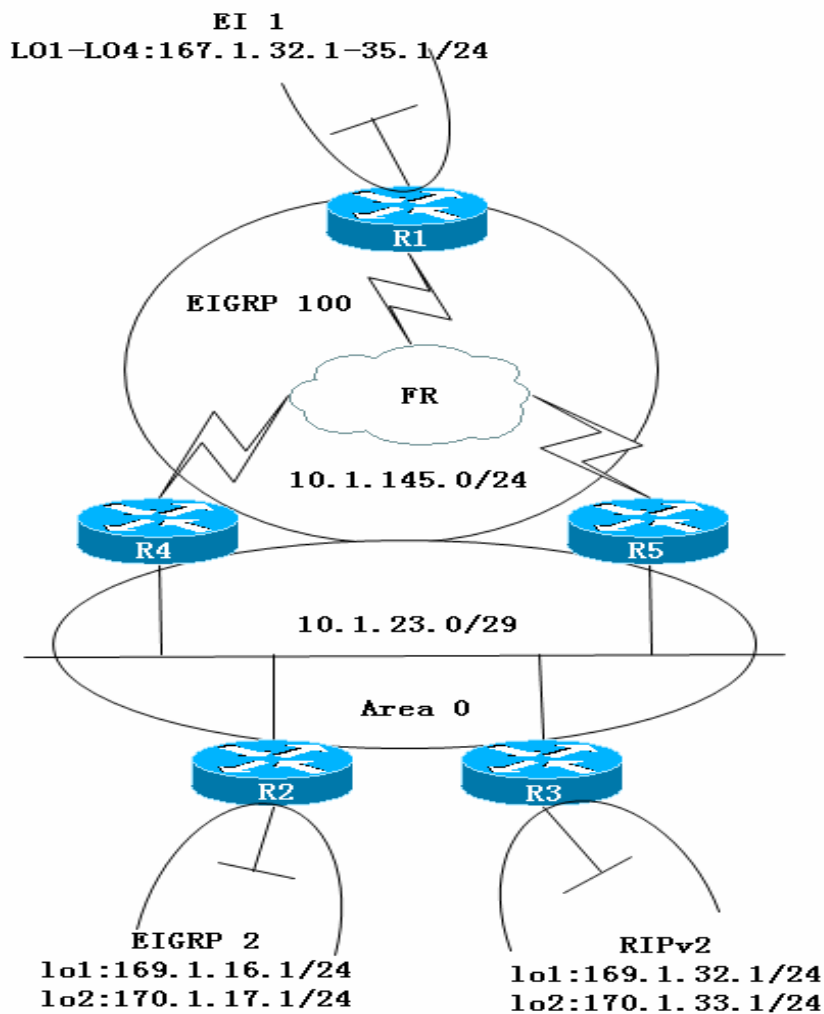
Levels 2-14 are not used in a default configuration, but commands that are normally at level 15 can be moved down to one of those levels and commands that are normally at level 1 can be moved up to

commands that are normally at level 15 can be moved down to one of those levels and commands that are normally at level 1 can be moved up to one of those levels. Obviously, this security model involves some administration on the router.

To determine the privilege level as a logged-in user, type the show privilege command. To determine what

commands are available at a particular privilege level for the version of Cisco IOS® software that you are using, type a ? at the command line when logged in at that privilege level.

IGP实验2



1. 桥接, 帧中继中R1的S0接口只用于子接口 (仅用图中所提供的DLCI)
2. RIP: R3的L01, L02运行RIPv2, RIP和OSPF做双向重分布.
3. EIGRP:

EIGRP 1

R1上4个环回口

L01:167.1.32.1/24

L02:167.1.33.1/24

L03:167.1.34.1/24

L04:167.1.35.1/24

宣告在EIGRP 1中

EIGRP 100

R1/R4/R5在EIGRP 100中, R1/R4/R5的环回口以及VLAN_A为EIGRP 100域内路由 有隐藏需求, 要求passive E0口

R4上看R1/R5的环回口

D 10.1.1.1/32[90/256000] VIA 10.1.145.1

D 10.1.5.5/32[90/256000] VIA 10.1.145.5

R5上看R1/R4的环回口

R5上看R1/R4的环回口

D 10.1.4.4/32[90/256000] VIA 10.1.145.4

D 10.1.1.1/32[90/256000] VIA 10.1.145.1

R1上看R4/R5的环回口

D 10.1.5.5/32[90/1657856] VIA 10.1.145.5

D 10.1.4.4/32[90/1657856] VIA 10.1.145.4

由于hub端metric要大于sopke. 所以只能各自指neighbor

R4/R5收到167.1.32.0/22的一条汇总路由, EIGRP 1 向L01发送一条10.1.0.0/16的路由, 向其他接口公告明细路由信息. 有隐藏需求. 注意要deny10.10.0/16

EIGRP 2

R2的L01, L02口在EIGRP2中, EIGRP 2与OSPF做双向重分布

4. OSPF: R2/R3/R4运行OSPF, R3/R4/R5之间都只能形成TWO-WAY状态, 当VLAN_A中再加入一台新设备时, R2还是DR.

R2/R3只看到167网段和10.1.145.0网段负载均衡, R2访问R1的环回口正常情况下走R4, 当R2/R4间的链路断了后走R5. R3访问R1的环回口走R5, 当R3/R5之间断了后, 丢包

5. 过滤: EIGRP 1 和EIGRP 2 不接受RIP的路由, RIP也不接受他们的路由.

注: 要求全网互相通, 所有L0接口的地址为10.1.X.X/32 (X为路由器号), 本实验主网段为10.1.0.0/16. 不允许出现静态路由.

点到多点, 主接口和多点子接口都要求把自动反向解析关闭

题目只要求R1使用子接口

帧中继交换机 持续

1 启用帧中继交换

2 接口封装, 两类型CISCO ???

3 frame lmi-type 自动的

4 frame intf-type DCE(二层的DCE) DTE NNI (帧中继交换机以帧中继交换机之间的接口模式)

5 clock rack 同步与异步速率(一层的概念, 以帧中继无关)

6 frame-relay route

7 dlci号是本接口有效的. 我入和出都可以相同, 当一个接口必须要有不同的DLCI

同步与异步概念

就是我们两同时发信息, 当不固定包的大小, 必须时间要匹配

异步: 就是两不同时发信息, 但包的大小要匹配

~~~~~  
~~~~~

要求指neighbor, 必须双向指. rip可以当向

```
~~~~~  
~~~~~  
r1# sho run  
Building configuration...  
  
Current configuration : 1773 bytes  
!  
version 12.2  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname r1  
!  
logging queue-limit 100  
!  
ip subnet-zero  
no ip domain lookup  
!  
!  
!  
!  
interface Loopback0  
ip address 10.1.1.1 255.255.255.255  
!  
interface Loopback1  
ip address 167.1.32.1 255.255.255.0  
ip summary-address eigrp 1 167.1.32.0 255.255.252.0 5  
ip summary-address eigrp 1 10.1.0.0 255.255.0.0 5  
!  
interface Loopback2  
ip address 167.1.33.1 255.255.255.0  
!  
interface Loopback3  
ip address 167.1.34.1 255.255.255.0  
!  
interface Loopback4  
ip address 167.1.35.1 255.255.255.0  
!  
interface Ethernet0  
no ip address  
shutdown  
!  
interface Serial0  
no ip address  
encapsulation frame-relay  
no frame-relay inverse-arp  
!  
interface Serial0.1 multipoint  
ip address 10.1.145.1 255.255.255.0  
frame-relay map ip 10.1.145.4 104
```

```
frame-relay map ip 10.1.145.4 104
frame-relay map ip 10.1.145.5 105
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router eigrp 1
redistribute eigrp 100 metric 100 1000 255 1 1500
network 167.1.32.0 0.0.3.255
no auto-summary
!
router eigrp 100
redistribute eigrp 1 metric 100 1000 255 1 1500 route-map EIGRP1-100
passive-interface Loopback0
network 10.1.1.1 0.0.0.0
network 10.1.145.0 0.0.0.255
neighbor 10.1.145.5 Serial0.1
neighbor 10.1.145.4 Serial0.1
metric weights 0 1 0 0 0
auto-summary
!
no ip http server
ip classless
!
!
!
ip prefix-list 3 seq 5 permit 167.1.32.0/22
!
!
route-map EIGRP1-100 permit 10
match ip address prefix-list 3
set tag 10
!
route-map EIGRP100-1 deny 10
match tag 30
!
route-map EIGRP100-1 permit 20
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
```

```
line vty 0 4
!
end

r1#config t
Enter configuration commands, one per line. End with CNTL/Z.
r1(config)#end
r1#sho run
Building configuration...

*Mar 1 02:43:49.739: %SYS-5-CONFIG_I: Configured from console by
console
Current configuration : 1773 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip address 167.1.32.1 255.255.255.0
 ip summary-address eigrp 1 167.1.32.0 255.255.252.0 5
 ip summary-address eigrp 1 10.1.0.0 255.255.0.0 5
!
interface Loopback2
 ip address 167.1.33.1 255.255.255.0
!
interface Loopback3
 ip address 167.1.34.1 255.255.255.0
!
interface Loopback4
 ip address 167.1.35.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
```

```
interface Serial0
  no ip address
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial0.1 multipoint
  ip address 10.1.145.1 255.255.255.0
  ip summary-address eigrp 100 167.1.32.0 255.255.252.0 5
  frame-relay map ip 10.1.145.4 104
  frame-relay map ip 10.1.145.5 105
  no frame-relay inverse-arp
!
interface Serial1
  no ip address
  shutdown
!
interface BRI0
  no ip address
  shutdown
!
router eigrp 1
  redistribute eigrp 100 metric 100 1000 255 1 1500
  network 167.1.32.0 0.0.3.255
  no auto-summary
!
router eigrp 100
  redistribute eigrp 1 metric 100 1000 255 1 1500 route-map EIGRP1-100
  passive-interface Loopback0
  network 10.1.1.1 0.0.0.0
  network 10.1.145.0 0.0.0.255
  neighbor 10.1.145.5 Serial0.1
  neighbor 10.1.145.4 Serial0.1
  metric weights 0 1 0 0 0
  auto-summary
!
no ip http server
ip classless
!
!
!
ip prefix-list 3 seq 5 permit 167.1.32.0/22
!
!
route-map EIGRP1-100 permit 10
  match ip address prefix-list 3
  set tag 10
!
route-map EIGRP100-1 deny 10
  match tag 30
!
route-map EIGRP100-1 permit 20
```

```
!  
route-map EIGRP100-1 permit 20  
!  
!  
line con 0  
  exec-timeout 0 0  
  logging synchronous  
line aux 0  
line vty 0 4  
!  
end
```

```
~~~~~  
~~~~~  
~~~~~
```

```
r2# sho run  
Building configuration...
```

```
Current configuration : 1562 bytes  
!  
version 12.2  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname r2  
!  
logging queue-limit 100  
!  
ip subnet-zero  
no ip domain lookup  
!  
frame-relay switching  
!  
!  
!  
interface Loopback0  
  ip address 10.1.2.2 255.255.255.255  
!  
interface Loopback1  
  ip address 169.1.16.1 255.255.255.0  
!  
interface Loopback2  
  ip address 170.1.17.1 255.255.255.0  
!  
interface Tunnel1  
  no ip address  
  tunnel source 10.1.23.2  
  tunnel destination 10.1.23.3  
!  
interface Ethernet0  
  ip address 10.1.23.2 255.255.255.248
```

```
ip address 10.1.23.2 255.255.255.248
ip ospf priority 2
!
interface Serial0
no ip address
encapsulation frame-relay
clockrate 56000
frame-relay intf-type dce
frame-relay route 104 interface Serial1 401
frame-relay route 105 interface Tunnel1 1000
!
interface Serial1
no ip address
encapsulation frame-relay
clockrate 56000
frame-relay intf-type dce
frame-relay route 401 interface Serial0 104
!
interface BRI0
no ip address
shutdown
!
router eigrp 2
redistribute ospf 10 metric 100 1000 255 1 1500 route-map DRIP
network 169.1.16.0 0.0.0.255
network 170.1.17.0 0.0.0.255
no auto-summary
!
router ospf 10
router-id 10.1.2.2
log-adjacency-changes
redistribute eigrp 2 subnets route-map EIGRPTAG
passive-interface Loopback0
network 10.1.2.2 0.0.0.0 area 0
network 10.1.16.0 0.0.7.255 area 0
!
no ip http server
ip classless
!
!
!
!
route-map DRIP deny 10
match tag 30
!
route-map DRIP permit 20
!
route-map EIGRPTAG permit 10
set tag 10
!
!
```

```
!  
line con 0  
  exec-timeout 0 0  
  logging synchronous  
line aux 0  
line vty 0 4  
!  
end  
~~~~~  
~~~~~  
~~~~~  
r3#sho run  
Building configuration..  
  
Current configuration : 1533 bytes  
!  
version 12.2  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname r3  
!  
logging queue-limit 100  
!  
ip subnet-zero  
no ip domain lookup  
!  
frame-relay switching  
!  
!  
!  
interface Loopback0  
  ip address 10.1.3.3 255.255.255.255  
!  
interface Loopback1  
  ip address 169.1.32.1 255.255.255.0  
!  
interface Loopback2  
  ip address 170.2.33.1 255.255.255.0  
!  
interface Tunnel1  
  no ip address  
  tunnel source 10.1.23.3  
  tunnel destination 10.1.23.2  
!  
interface Ethernet0  
  ip address 10.1.23.3 255.255.255.248  
  ip ospf priority 0  
!  
interface Serial0
```

```
ip ospf priority 0
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
encapsulation frame-relay
clockrate 56000
frame-relay intf-type dce
frame-relay route 501 interface Tunnel1 1000
!
interface BRI0
no ip address
shutdown
!
router ospf 10
router-id 10.1.3.3
log-adjacency-changes
redistribute rip subnets route-map RIPTAG
passive-interface Loopback0
network 10.1.3.3 0.0.0.0 area 0
network 10.1.23.0 0.0.0.7 area 0
!
router rip
version 2
redistribute ospf 10 metric 1 route-map DEIGRP
network 169.1.0.0
network 170.2.0.0
no auto-summary
!
ip local policy route-map POLICYR3
no ip http server
ip classless
!
!
!
access-list 101 permit ip any host 10.1.1.1
!
route-map DEIGRP deny 10
match tag 10
!
route-map DEIGRP permit 20
!
route-map RIPTAG permit 10
set tag 30
!
route-map POLICYR3 permit 10
match ip address 101
set ip next-hop 10.1.23.5
```

```
!  
route-map POLICYR3 permit 10  
  match ip address 101  
  set ip next-hop 10.1.23.5  
!  
!  
line con 0  
  exec-timeout 0 0  
  logging synchronous  
line aux 0  
line vty 0 4  
!  
end
```

```
~~~~~  
~~~~~
```

```
r4#sho run  
Building configuration...
```

```
Current configuration : 1368 bytes  
!  
version 12.2  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname r4  
!  
logging queue-limit 100  
!  
ip subnet-zero  
no ip domain lookup  
!  
!  
!  
interface Loopback0  
  ip address 10.1.4.4 255.255.255.255  
!  
interface Ethernet0  
  ip address 10.1.23.4 255.255.255.248  
  ip ospf priority 0  
!  
interface Serial0  
  bandwidth 10000  
  ip address 10.1.145.4 255.255.255.0  
  encapsulation frame-relay  
  frame-relay map ip 10.1.145.1 401  
  frame-relay map ip 10.1.145.5 401  
  no frame-relay inverse-arp  
!
```

```
frame-relay map ip 10.1.145.5 401
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router eigrp 100
passive-interface Ethernet0
passive-interface Loopback0
network 10.1.4.4 0.0.0.0
network 10.1.23.0 0.0.0.7
network 10.1.145.0 0.0.0.255
neighbor 10.1.145.5 Serial0
neighbor 10.1.145.1 Serial0
metric weights 0 1 0 0 0
no auto-summary
!
router ospf 10
router-id 10.1.4.4
log-adjacency-changes
redistribute eigrp 100 subnets route-map EIGRP100-OSPF
network 10.1.23.0 0.0.0.7 area 0
distance 171 0.0.0.0 255.255.255.255
!
no ip http server
ip classless
!
!
!
ip prefix-list 1 seq 5 permit 10.1.1.1/32
!
!
route-map EIGRP100-OSPF permit 10
match ip address prefix-list 1
set metric 19
!
route-map EIGRP100-OSPF permit 20
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

end

~~~~~  
~~~~~  
~~~~~

r5#sho run

Building configuration...

Current configuration : 1181 bytes

!

version 12.2

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname r5

!

logging queue-limit 100

!

ip subnet-zero

no ip domain lookup

!

!

!

!

interface Loopback0

ip address 10.1.5.5 255.255.255.255

!

interface Ethernet0

ip address 10.1.23.5 255.255.255.248

ip ospf priority 0

!

interface Serial0

bandwidth 10000

ip address 10.1.145.5 255.255.255.0

encapsulation frame-relay

frame-relay map ip 10.1.145.1 501

frame-relay map ip 10.1.145.4 501

no frame-relay inverse-arp

!

interface Serial1

no ip address

shutdown

!

interface BRI0

no ip address

shutdown

!

router eigrp 100

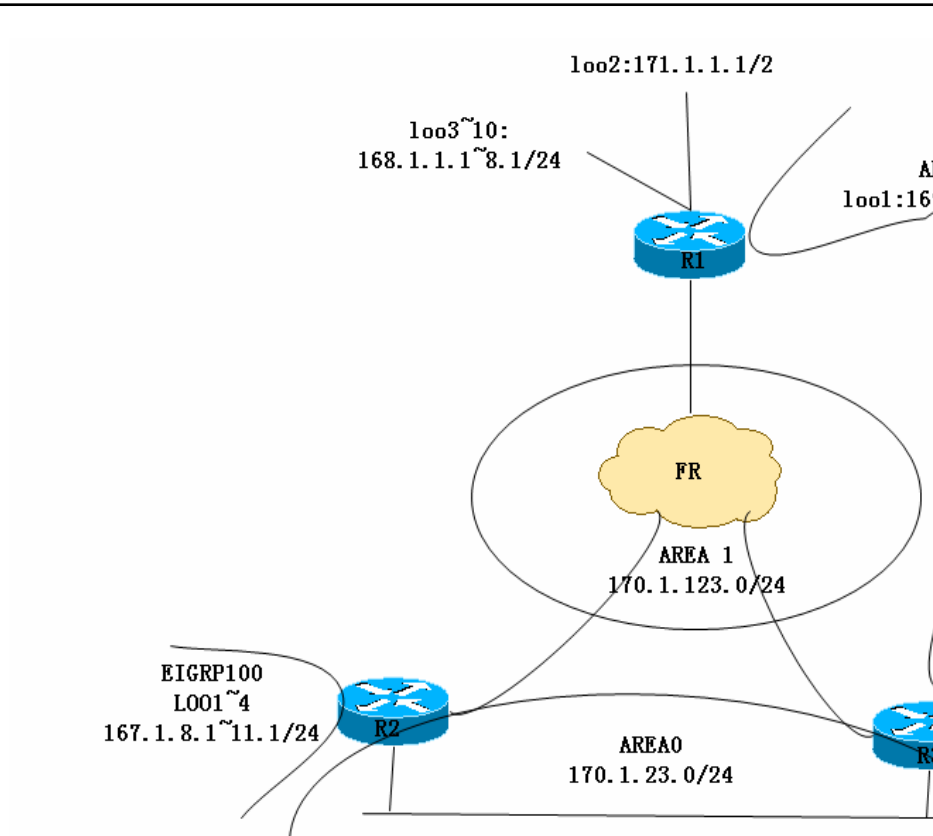
passive-interface Ethernet0

passive-interface Loopback0

network 10.1.5.5 0.0.0.0

```
network 10.1.5.5 0.0.0.0
network 10.1.23.0 0.0.0.7
network 10.1.145.0 0.0.0.255
neighbor 10.1.145.4 Serial0
neighbor 10.1.145.1 Serial0
metric weights 0 1 0 0 0
no auto-summary
!
router ospf 10
router-id 10.1.5.5
log-adjacency-changes
redistribute eigrp 100 subnets
network 10.1.23.0 0.0.0.7 area 0
distance 171 0.0.0.0 255.255.255.255
!
no ip http server
ip classless
!
!
!
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

### IGP实验3



### 一 桥接

帧中继中R1的S0接口只用于子接口，要求ping通所有接口（仅使用图中提供的DLC）

### 二`OSPF

u

2, ospf的 帧中继中不允许使用nbma和广播模式.

3, area2知接收ospf的inter和intra路由

4, r3在日后会重area3中接收到一些lsa7类型的路由以及area3中会有一条默认路由.

5 raea0使用明文认证, area1使用更安全的认证方式, 验证密码为cisco.

6 所有loopback0接口均在ospf域内.

7r1的loopback3到loopback10接口不允许直接宣告进ospf域内.

### 二EIGRP

R2的loopback1到loopback4在EIGRP100中.

2eigrp和ospf在r2上做双向重分布, eigrp只向ospf发送一条路由(不允许是167. 1. 0. 0/16) eigrp和ospf在r2上做双向重分布, eigrp只向ospf只发送一条路由

(不允许是167. 1. 0. 0/16)

### 三rip

1, r3的loopback2到loopback5在rip域中

2, rip和ospf在r3做双向重分布.

3在r1和r2上只能看到rip域过来的一条汇总路由(不能是

2, rip和ospf在r3做双向重分布.

3在r1和r2上只能看到rip域过来的一条汇总路由(不能是166.1.0.0/16).要和eigrp的汇总用不同的方法.

四

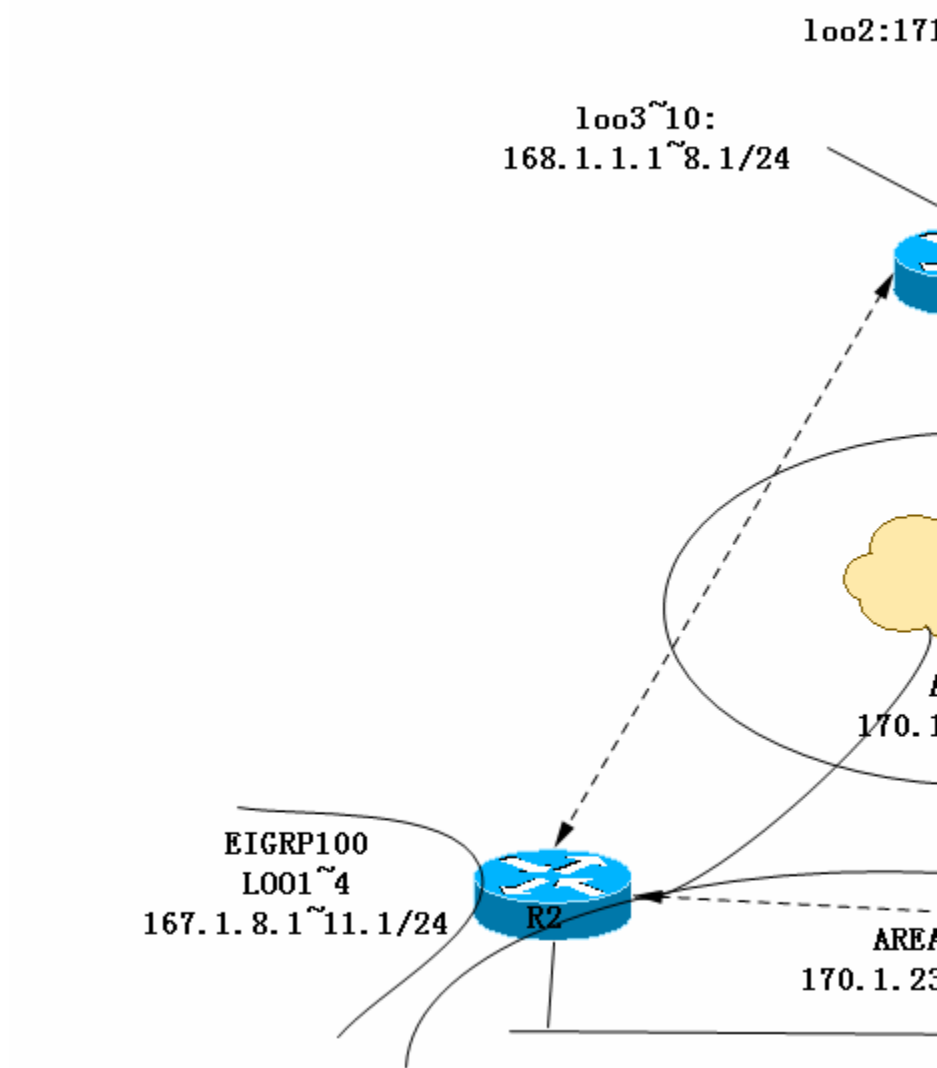
1, R1的loopback2接口不允许宣告在任何路由协议中.

五 过滤

1 在R2上可看见这样的一些路由168.1.X.0(x为奇数)

2 在R3上可看到这样的一些路由168.1.y.0(y为偶数)

注:要求全网全通,不允许出现任何主机路由,除P2M外,所有loopback0接口的地址为170.1.x.x/24(x为路由器号),本试验主网段为170.1.0.0/26.



要求一:要求ping通所有接口,既包括自己的接口.做试验应该由2层开始做,所以即使是P2MP也要做到各节点的映射.

虚链路要求都做上,冗余用... 问

重分布进ospf的loopbacl口不用打ip ospf net ptp也是显示为网段路由..

如果要求重分布直连接口就match接口

area3 要求不重分布,因为由默认了,就没必要有外部路由.

odr重分布到ospf的时候要挂route-map, 重分布直连.

rip如果题目没有明确要求,rip基本配置就是版本2 no auto

~~~~~  
~~~~~  
~~~~~

r1# sho run

Building configuration...

Current configuration : 2043 bytes

!

version 12.2

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname r1

!

logging queue-limit 100

!

ip subnet-zero

no ip domain lookup

!

!

!

!

interface Loopback0

ip address 170.1.1.1 255.255.255.0

ip ospf network point-to-point

!

interface Loopback1

ip address 169.1.1.1 255.255.255.0

ip ospf network point-to-point

!

interface Loopback2

ip address 171.1.1.1 255.255.255.0

!

interface Loopback3

ip address 168.1.1.1 255.255.255.0

!

```
interface Loopback4
  ip address 168.1.2.1 255.255.255.0
!
interface Loopback5
  ip address 168.1.3.1 255.255.255.0
!
interface Loopback6
  ip address 168.1.4.1 255.255.255.0
!
interface Loopback7
  ip address 168.1.5.1 255.255.255.0
!
interface Loopback8
  ip address 168.1.6.1 255.255.255.0
!
interface Loopback9
  ip address 168.1.7.1 255.255.255.0
!
interface Loopback10
  ip address 168.1.8.1 255.255.255.0
!
interface Ethernet0
  no ip address
  shutdown
!
interface Serial0
  no ip address
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial0.100 multipoint
  ip address 170.1.123.1 255.255.255.0
  ip ospf message-digest-key 1 md5 cisco
  ip ospf network point-to-multipoint
  frame-relay map ip 170.1.123.1 105 broadcast
  frame-relay map ip 170.1.123.2 104 broadcast
  frame-relay map ip 170.1.123.3 105 broadcast
  no frame-relay inverse-arp
!
interface Serial1
  no ip address
  shutdown
!
interface BRI0
  no ip address
```

```
shutdown
!
router ospf 10
router-id 170.1.1.1
log-adjacency-changes
area 0 authentication
area 1 authentication message-digest
area 1 virtual-link 170.1.3.3 authentication
area 1 virtual-link 170.1.2.2 authentication
area 2 stub
redistribute connected subnets route-map REC
network 169.1.1.1 0.0.0.0 area 2
network 170.1.1.1 0.0.0.0 area 1
network 170.1.123.1 0.0.0.0 area 1
!
no ip http server
ip classless
!
!
!
!
route-map REC permit 10
match interface Loopback3 Loopback4 Loopback5 Loopback6
Loopback7 Loopback8 Loopback9 Loopback10
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
~~~~~
~~~~~
rl#sho ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
       * - candidate default, U - per-user static route, o -
```

i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
* - candidate default, U - per-user static route, o -
ODR
P - periodic downloaded static route

Gateway of last resort is not set

170.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
O 170.1.3.0/24 [110/65] via 170.1.123.3, 00:11:25,
Serial0.100
O 170.1.2.0/24 [110/65] via 170.1.123.2, 00:11:25,
Serial0.100
C 170.1.1.0/24 is directly connected, Loopback0
O 170.1.23.0/24 [110/74] via 170.1.123.2, 00:11:25,
Serial0.100
[110/74] via 170.1.123.3, 00:11:25,
Serial0.100
C 170.1.123.0/24 is directly connected, Serial0.100
O 170.1.123.2/32 [110/64] via 170.1.123.2, 00:11:25,
Serial0.100
O 170.1.123.3/32 [110/64] via 170.1.123.3, 00:11:26,
Serial0.100
171.1.0.0/24 is subnetted, 1 subnets
C 171.1.1.0 is directly connected, Loopback2
168.1.0.0/24 is subnetted, 8 subnets
C 168.1.8.0 is directly connected, Loopback10
C 168.1.1.0 is directly connected, Loopback3
C 168.1.3.0 is directly connected, Loopback5
C 168.1.2.0 is directly connected, Loopback4
C 168.1.5.0 is directly connected, Loopback7
C 168.1.4.0 is directly connected, Loopback6
C 168.1.7.0 is directly connected, Loopback9
C 168.1.6.0 is directly connected, Loopback8
169.1.0.0/24 is subnetted, 1 subnets
C 169.1.1.0 is directly connected, Loopback1
166.1.0.0/22 is subnetted, 1 subnets
O E2 166.1.16.0 [110/20] via 170.1.123.3, 00:11:16,
Serial0.100
167.1.0.0/22 is subnetted, 1 subnets
O E2 167.1.8.0 [110/20] via 170.1.123.2, 00:11:16,
Serial0.100
165.1.0.0/24 is subnetted, 1 subnets
O IA 165.1.1.0 [110/65] via 170.1.123.3, 00:11:26,
Serial0.100

```
165.1.0.0/24 is subnetted, 1 subnets
0 IA    165.1.1.0 [110/65] via 170.1.123.3, 00:11:26,
Serial0.100
```

```
~~~~~
~~~~~
r2#sho run
Building configuration...
```

```
Current configuration : 1879 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r2
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 170.1.2.2 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback1
 ip address 167.1.8.1 255.255.255.0
 ip summary-address eigrp 100 167.1.8.0 255.255.252.0 5
!
interface Loopback2
 ip address 167.1.9.1 255.255.255.0
!
interface Loopback3
 ip address 167.1.10.1 255.255.255.0
!
interface Loopback4
 ip address 167.1.11.1 255.255.255.0
!
interface Ethernet0
 ip address 170.1.23.2 255.255.255.0
 ip ospf authentication-key cisco
```

```
ip address 170.1.23.2 255.255.255.0
ip ospf authentication-key cisco
!
interface Serial0
ip address 170.1.123.2 255.255.255.0
encapsulation frame-relay
ip ospf message-digest-key 1 md5 cisco
ip ospf network point-to-multipoint
frame-relay map ip 170.1.123.1 401 broadcast
frame-relay map ip 170.1.123.2 401 broadcast
frame-relay map ip 170.1.123.3 401 broadcast
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router eigrp 100
 redistribute ospf 10 metric 1000 100 255 1 1500
 network 167.1.8.0 0.0.3.255
 no auto-summary
!
router ospf 10
 router-id 170.1.2.2
 log-adjacency-changes
 area 0 authentication
 area 1 authentication message-digest
 area 1 virtual-link 170.1.3.3 authentication
 area 1 virtual-link 170.1.1.1 authentication
 redistribute eigrp 100 subnets route-map EIGRP-OSPF
 network 170.1.2.2 0.0.0.0 area 0
 network 170.1.23.2 0.0.0.0 area 0
 network 170.1.123.2 0.0.0.0 area 1
 distribute-list 1 in Serial0
!
no ip http server
ip classless
!
!
!
ip prefix-list 1 seq 5 permit 167.1.8.0/22
```

```
ip prefix-list 1 seq 5 permit 167.1.8.0/22
!
access-list 1 deny 168.1.0.0 0.0.254.0
access-list 1 permit any
!
route-map EIGRP-OSPF permit 10
 match ip address prefix-list 1
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

```
~~~~~
~~~~~
r2#sho ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
       * - candidate default, U - per-user static route, o -
ODR
       P - periodic downloaded static route
```

Gateway of last resort is not set

```
170.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
O      170.1.3.0/24 [110/11] via 170.1.23.3, 00:08:51,
Ethernet0
C      170.1.2.0/24 is directly connected, Loopback0
O      170.1.1.0/24 [110/65] via 170.1.123.1, 00:08:51,
Serial0
C      170.1.23.0/24 is directly connected, Ethernet0
C      170.1.123.0/24 is directly connected, Serial0
O      170.1.123.1/32 [110/64] via 170.1.123.1, 00:08:51,
Serial0
O      170.1.123.3/32 [110/128] via 170.1.123.1, 00:08:51,
Serial0
```

```
Ethernet0
C      170.1.123.0/24 is directly connected, Serial0
O      170.1.123.1/32 [110/64] via 170.1.123.1, 00:08:51,
Serial0
O      170.1.123.3/32 [110/128] via 170.1.123.1, 00:08:51,
Serial0
      168.1.0.0/24 is subnetted, 4 subnets
O E2   168.1.1.0 [110/20] via 170.1.123.1, 00:08:52, Serial0
O E2   168.1.3.0 [110/20] via 170.1.123.1, 00:08:52, Serial0
O E2   168.1.5.0 [110/20] via 170.1.123.1, 00:08:52, Serial0
O E2   168.1.7.0 [110/20] via 170.1.123.1, 00:08:52, Serial0
      169.1.0.0/24 is subnetted, 1 subnets
O IA   169.1.1.0 [110/65] via 170.1.123.1, 00:08:52, Serial0
      166.1.0.0/22 is subnetted, 1 subnets
O E2   166.1.16.0 [110/20] via 170.1.23.3, 00:08:52,
Ethernet0
      167.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
C      167.1.10.0/24 is directly connected, Loopback3
C      167.1.11.0/24 is directly connected, Loopback4
C      167.1.8.0/24 is directly connected, Loopback1
D      167.1.8.0/22 is a summary, 00:29:35, Null0
C      167.1.9.0/24 is directly connected, Loopback2
      165.1.0.0/24 is subnetted, 1 subnets
O IA   165.1.1.0 [110/11] via 170.1.23.3, 00:08:52, Ethernet0
~~~~~
~~~~~
~~~~~
r3# sho run
Building configuration...

Current configuration : 2062 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r3
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
```

```
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r3
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 170.1.3.3 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback1
 ip address 165.1.1.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback2
 ip address 166.1.16.1 255.255.255.0
!
interface Loopback3
 ip address 166.1.17.1 255.255.255.0
!
interface Loopback4
 ip address 166.1.18.1 255.255.255.0
!
interface Loopback5
 ip address 166.1.19.1 255.255.255.0
!
interface Ethernet0
 ip address 170.1.23.3 255.255.255.0
 ip ospf authentication-key 1 cisco
!
interface Serial0
 ip address 170.1.123.3 255.255.255.0
 encapsulation frame-relay
 ip ospf message-digest-key 1 md5 cisco
 ip ospf network point-to-multipoint
 frame-relay map ip 170.1.123.1 501 broadcast
 frame-relay map ip 170.1.123.2 501 broadcast
```

```
frame-relay map ip 170.1.123.3 501 broadcast
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router ospf 10
router-id 170.1.3.3
log-adjacency-changes
area 0 authentication
area 1 authentication message-digest
area 1 virtual-link 170.1.2.2 authentication
area 1 virtual-link 170.1.1.1 authentication
area 3 nssa no-redistribution no-summary
summary-address 166.1.16.0 255.255.252.0
redistribute rip subnets
network 165.1.1.1 0.0.0.0 area 3
network 170.1.3.3 0.0.0.0 area 0
network 170.1.23.3 0.0.0.0 area 0
network 170.1.123.3 0.0.0.0 area 1
distribute-list 1 in Serial0
!
router rip
version 2
redistribute ospf 10 metric 1 route-map OSPF-RIP
network 166.1.0.0
no auto-summary
!
no ip http server
ip classless
!
!
!
ip prefix-list 1 seq 5 permit 166.1.16.0/22
!
access-list 1 deny 168.1.1.0 0.0.254.0
access-list 1 permit any
!
route-map OSPF-RIP deny 10
match ip address prefix-list 1
```

```

!
route-map OSPF-RIP permit 20
!
!
line con 0
  exec-timeout 0 0
  logging synchronous
line aux 0
line vty 0 4
!
end
~~~~~
~~~~~
r3#sho ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
       * - candidate default, U - per-user static route, o -
ODR
       P - periodic downloaded static route

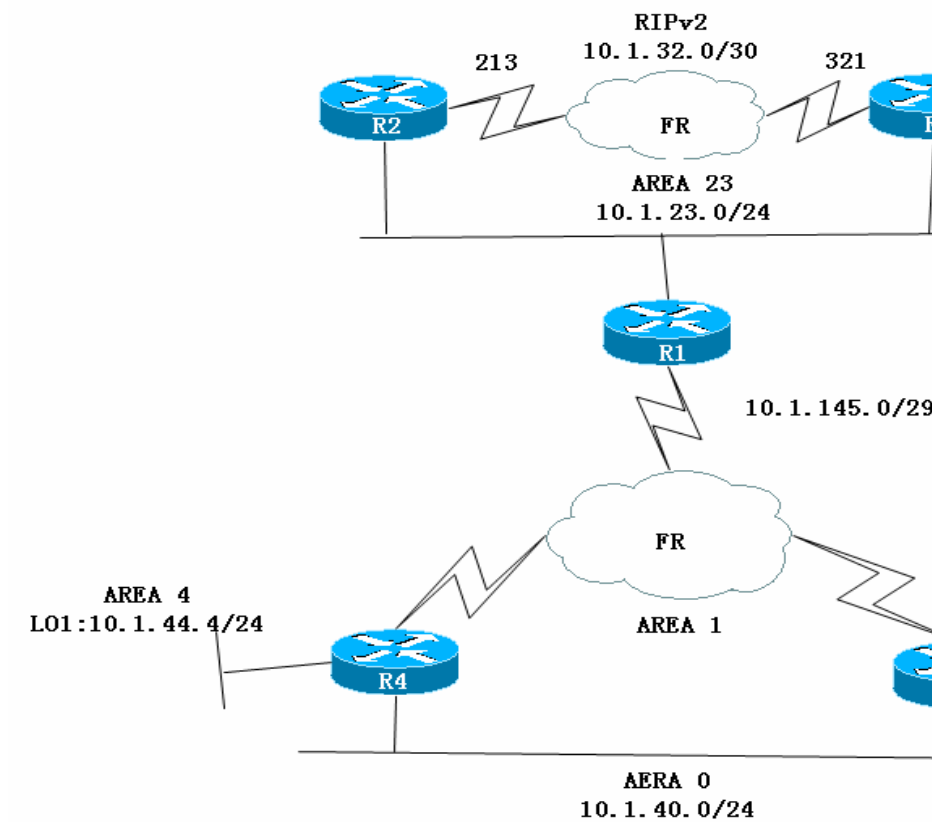
Gateway of last resort is not set

      170.1.0.0/16 is variably subnetted, 7 subnets, 2 masks
C       170.1.3.0/24 is directly connected, Loopback0
O       170.1.2.0/24 [110/11] via 170.1.23.2, 00:10:38,
Ethernet0
O       170.1.1.0/24 [110/65] via 170.1.123.1, 00:10:38,
Serial0
C       170.1.23.0/24 is directly connected, Ethernet0
C       170.1.123.0/24 is directly connected, Serial0
O       170.1.123.1/32 [110/64] via 170.1.123.1, 00:10:38,
Serial0
O       170.1.123.2/32 [110/128] via 170.1.123.1, 00:10:38,
Serial0
      168.1.0.0/24 is subnetted, 4 subnets
O E2    168.1.8.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
O E2    168.1.2.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
O E2    168.1.4.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
O E2    168.1.6.0 [110/20] via 170.1.123.1, 00:10:39, Serial0

```

```
0      170.1.123.2/32 [110/128] via 170.1.123.1, 00:10:38,
Serial0
      168.1.0.0/24 is subnetted, 4 subnets
0 E2   168.1.8.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
0 E2   168.1.2.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
0 E2   168.1.4.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
0 E2   168.1.6.0 [110/20] via 170.1.123.1, 00:10:39, Serial0
      169.1.0.0/24 is subnetted, 1 subnets
0 IA   169.1.1.0 [110/65] via 170.1.123.1, 00:10:39, Serial0
      166.1.0.0/16 is variably subnetted, 5 subnets, 2 masks
C      166.1.19.0/24 is directly connected, Loopback5
C      166.1.18.0/24 is directly connected, Loopback4
C      166.1.17.0/24 is directly connected, Loopback3
C      166.1.16.0/24 is directly connected, Loopback2
0      166.1.16.0/22 is a summary, 00:10:38, Null0
      167.1.0.0/22 is subnetted, 1 subnets
0 E2   167.1.8.0 [110/20] via 170.1.23.2, 00:10:39, Ethernet0
      165.1.0.0/24 is subnetted, 1 subnets
C      165.1.1.0 is directly connected, Loopback1
```

IGP实验4



IGP实验四

一，桥接

桥接, 帧中继中R1的S0 R2的S0接口只用于子接口 (仅用图中所提供的 DLCI)

二，EIGRP

R5的lo02-lo07运行在EIGRP 100 中

eigrp->ospf要求路由表如下:

E02 169.1.16.0 [110/0]

E02 169.1.17.0 [110/150]

E02 169.1.18.0 [110/0]

E02 171.1.32.0 [110/150]

E02 171.1.33.1 [110/150]

E02 171.1.12.0 [110/150]

三 RIP

R2 R3运行RIP v2, R2 R3的lo0口以及R3的lo1口是RIP域内路由, R3的lo1口地址为192.168.1.3/24

R2, R3间用很安全的认证, 密码为CISCO. R2 R3间在路由稳定时不发送路由更新.

正常情况下R2访问RIP域外的网络时走R3, 当R3的E0口DOWN掉后, 走R1.

R3访问RIP域外的网络时走R1, 当自己的E0口当掉后, 走R2.

要求R2 R3间链路全网可见. 且每一跳. cost值会改变.

四, OSPF

ospf的帧中继不允许使用neighbor命令, R1, R4 R5的环回口运行在OSPF域内; R4 R5的E0口运行在area0内; R1 R2 R3的E0口运行在area 23内; R1 R4 R5的串口运行在area1中; R4的lo1运行在area4中, R5的lo1运行在area5中, 由于我们不能配置交换机且在以太网三跑ospf, 所以请正确选择ospf的接口网络类型. area0明文认证, 密码cisco. R4和R1互相看对方的lo0口路由为24位. R4不向area 4发送任何LSA. R5会向area5中的其他路由器发送一条LSA-7类的默认路由.

R4的E0口一秒发送4个hello包. 当R2在150秒内没收到hello包, 邻居关系也不会当.

五 过滤

RIP不接收171. 1. 12. 0这条路由. 不能用列表匹配, 但RIP的路由器能通过R2来访问171. 1. 12. 0网段内的主机.

6 feature

在R3三配置一个lo1口将器宣告进RIP域内. 掩码为24位
假定这个环回口连一台主机, 地址位192. 168. 1. 133. 当这台主机上trace R4的lo1时. trace信息为
192. 168. 1. 134
10. 1. 4. 4

注: 要求全网全通, 除lo0外, 不允许出现任何主机路由, 摸索有looback0接口地址为10. 1. x. x/32(x为路由器号), 本试验主网段为10. 0. 0. 0/8. 不允许出现任何静态路由

R2子接口用P2P 点到点子借口不用关闭反向解析

R1子接口用P2MP

R1两条map

R2没有

R3一条

R4两条

R5两条

RIP

记得passvis default

no pass loo 1

密码要大写

注意

rip要用触发更新, 因为要不周期更新. 打了触发更新后, 他会自动生成

timer base 30 180 0 240

在帧中继环境下, 主接口的水平分割默认时关闭的... 要开启.

注意 指有帧中继的主接口默认时关闭的, 其他都时开的.,

在多路访问下. 指neiboor的话要关闭水平分割.

OSPF

R1 R4 R5帧中继用接口用广播模式.

area0的密码是小写的

R1 R4互相看到对方的接口是24为. 就用汇总的办法解决.

area4的接口都要过滤database loopback0 loopback1

R2 R1 R3 的dead-interval 改成150

改dead hello不会边

改hello dead会变

redistribut metri用150可以少一个route-map

抓16 18 用0.0.2.0的通配符

RIP重分布到ospf用重分布直连, 不过会有问题.... 因为如果R2 R3之间

有其他路由就无法公布到OSPF中, 如果重分布的是RIP就会出现环路

新办法就是用tag, metric-type 1.

feature用tunno完成.

tunnel地址要用借用地址完成.

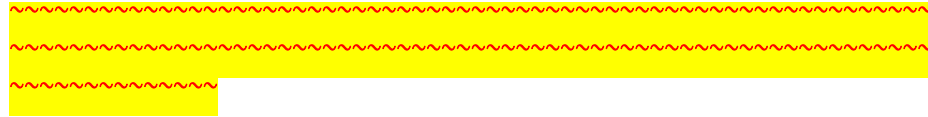
~~~~~  
~~~~~  
~~~~~

```
r1#          sho run
Building configuration...
```

```
Current configuration : 1458 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
frame-relay switching
!
!
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
```

```
ip address 10.1.1.1 255.255.255.255
!
interface Ethernet0
ip address 10.1.23.1 255.255.255.0
ip ospf network point-to-multipoint non-broadcast
ip ospf dead-interval 151
!
interface Serial0
no ip address
encapsulation frame-relay
no frame-relay inverse-arp
frame-relay lmi-type cisco
frame-relay route 213 interface Serial1 312
!
interface Serial0.1 multipoint
ip address 10.1.145.1 255.255.255.0
ip ospf network broadcast
frame-relay map ip 10.1.145.4 104 broadcast
frame-relay map ip 10.1.145.5 105 broadcast
no frame-relay inverse-arp
!
interface Serial1
no ip address
encapsulation frame-relay
no frame-relay inverse-arp
frame-relay lmi-type cisco
frame-relay intf-type dce
frame-relay route 312 interface Serial0 213
!
interface BRI0
no ip address
shutdown
!
router ospf 10
router-id 10.1.1.1
log-adjacency-changes
area 1 virtual-link 10.1.5.5
area 1 virtual-link 10.1.4.4
network 10.1.1.0 0.0.0.255 area 1
network 10.1.23.0 0.0.0.255 area 23
network 10.1.145.0 0.0.0.255 area 1
neighbor 10.1.23.2
neighbor 10.1.23.3
distance 111 0.0.0.2 255.255.255.0
!
```

```
no ip http server
ip classless
!
!
!
!
!
line con 0
  exec-timeout 0 0
  logging synchronous
line aux 0
line vty 0 4
!
end
```



```
r2#sho run
Building configuration...
```

```
*Mar  1 03:58:58.863: %SYS-5-CONFIG_I: Configured from console
by console
Current configuration : 1879 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r2
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
frame-relay switching
!
key chain jiaobaba
  key 1
    key-string cisco
!
!
!
interface Loopback0
```

```
    ip address 10.1.2.2 255.255.255.255
!
interface Tunnel1
    no ip address
    tunnel source 10.1.23.2
    tunnel destination 10.1.23.3
!
interface Ethernet0
    ip address 10.1.23.2 255.255.255.0
    ip ospf network point-to-multipoint non-broadcast
    ip ospf dead-interval 151
!
interface Serial0
    no ip address
    encapsulation frame-relay
    clockrate 56000
    no frame-relay inverse-arp
    frame-relay lmi-type cisco
    frame-relay intf-type dce
    frame-relay route 104 interface Serial1 401
    frame-relay route 105 interface Tunnel1 1000
!
interface Serial0.1 point-to-point
    ip address 10.1.32.2 255.255.255.0
    ip rip triggered
    ip rip authentication mode md5
    ip rip authentication key-chain jiaobaba
    frame-relay interface-dlci 213
!
interface Serial1
    no ip address
    encapsulation frame-relay
    clockrate 56000
    no frame-relay inverse-arp
    frame-relay lmi-type cisco
    frame-relay intf-type dce
    frame-relay route 401 interface Serial0 104
!
interface BRI0
    no ip address
    shutdown
!
router ospf 10
    router-id 10.1.2.2
    log-adjacency-changes
```

```
network 10.1.23.0 0.0.0.255 area 23
neighbor 10.1.23.1
neighbor 10.1.23.3
!
router rip
version 2
timers basic 30 180 0 240
redistribute ospf 10 metric 1 route-map OSPF_RIP
passive-interface Ethernet0
passive-interface Loopback0
network 10.0.0.0
default-information originate
distance 109 0.0.0.3 255.255.255.0
no auto-summary
!
no ip http server
ip classless
!
!
!
!
route-map RR permit 10
!
route-map OSPF_RIP deny 10
match tag 1
!
route-map OSPF_RIP permit 20
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
~~~~~
~~~~~
~~~~~
r3# sho run
Building configuration...

Current configuration : 1969 bytes
!
version 12.2
```

```
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r3
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
frame-relay switching
!
key chain jiaobaba
 key 1
 key-string cisoc
!
!
!
interface Loopback0
 ip address 10.1.3.3 255.255.255.255
!
interface Loopback1
 ip address 192.168.1.3 255.255.255.0
 ip policy route-map POLICY
!
interface Tunnel1
 no ip address
 tunnel source 10.1.23.3
 tunnel destination 10.1.23.2
!
interface Ethernet0
 ip address 10.1.23.3 255.255.255.0
 ip ospf network point-to-multipoint non-broadcast
 ip ospf dead-interval 151
!
interface Serial0
 no ip address
 encapsulation frame-relay
 clockrate 56000
 no frame-relay inverse-arp
!
interface Serial0.1 point-to-point
 ip address 10.1.32.3 255.255.255.0
 ip rip triggered
```

```
ip rip authentication mode md5
ip rip authentication key-chain jiaobaba
frame-relay interface-dlci 312
!
interface Serial1
no ip address
encapsulation frame-relay
clockrate 56000
no frame-relay inverse-arp
frame-relay lmi-type cisco
frame-relay intf-type dce
frame-relay route 501 interface Tunnell 1000
!
interface BRI0
no ip address
shutdown
!
router ospf 10
router-id 10.1.3.3
log-adjacency-changes
redistribute rip metric-type 1 subnets
network 10.1.23.0 0.0.0.255 area 23
neighbor 10.1.23.1
neighbor 10.1.23.2
!
router rip
version 2
timers basic 30 180 0 240
redistribute ospf 10 metric 1 match internal external 1
external 2 route-map OSPF_RIP
passive-interface Ethernet0
passive-interface Loopback0
network 10.0.0.0
network 192.168.1.0
no auto-summary
!
no ip http server
ip classless
!
!
!
access-list 101 permit ip 192.168.1.0 0.0.0.255 host 10.1.44.4
!
route-map POLICY permit 10
match ip address 101
```

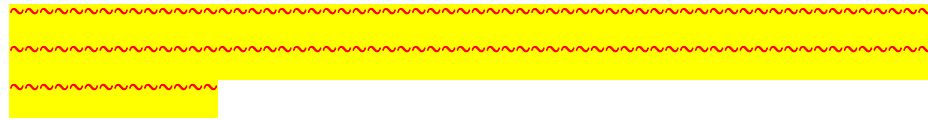
```
match ip address 101
set ip next-hop 10.1.4.4
!
route-map OSPF_RIP deny 10
match tag 1
!
route-map OSPF_RIP permit 20
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

```
~~~~~
~~~~~
~~~~~
r4#sho run
Building configuration...
```

```
*Mar  1 04:01:51.267: %SYS-5-CONFIG_I: Configured from console
by console
Current configuration : 1210 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r4
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
ip address 10.1.4.4 255.255.255.255
!
interface Loopback1
```

```
ip address 10.1.44.4 255.255.255.0
ip ospf database-filter all out
!
interface Ethernet0
ip address 10.1.45.4 255.255.255.0
ip ospf network non-broadcast
ip ospf dead-interval minimal hello-multiplier 4
!
interface Serial0
ip address 10.1.145.4 255.255.255.0
encapsulation frame-relay
ip ospf network broadcast
frame-relay map ip 10.1.145.1 401 broadcast
no frame-relay inverse-arp
frame-relay lmi-type cisco
!
interface Serial1
no ip address
shutdown
!
interface BRI0
no ip address
shutdown
!
router ospf 10
router-id 10.1.4.4
log-adjacency-changes
area 1 virtual-link 10.1.5.5
area 1 virtual-link 10.1.1.1
network 10.1.4.0 0.0.0.255 area 0
network 10.1.44.0 0.0.0.255 area 4
network 10.1.45.0 0.0.0.255 area 0
network 10.1.145.0 0.0.0.255 area 1
neighbor 10.1.45.5 priority 1
!
no ip http server
ip classless
!
!
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
```

```
line aux 0
line vty 0 4
!
end
```



```
r5#      sho run
Building configuration...
```

```
Current configuration : 2036 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname r5
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 10.1.5.5 255.255.255.255
!
interface Loopback1
 ip address 10.1.55.5 255.255.255.0
!
interface Loopback2
 ip address 169.1.16.1 255.255.255.0
!
interface Loopback3
 ip address 169.1.17.1 255.255.255.0
!
interface Loopback4
 ip address 169.1.18.1 255.255.255.0
!
interface Loopback5
 ip address 171.1.32.1 255.255.255.0
!
```

```
interface Loopback6
  ip address 171.1.33.1 255.255.255.0
!
interface Loopback7
  ip address 171.1.12.1 255.255.255.0
!
interface Ethernet0
  ip address 10.1.45.5 255.255.255.0
  ip ospf network non-broadcast
!
interface Serial0
  ip address 10.1.145.5 255.255.255.0
  encapsulation frame-relay
  ip ospf network broadcast
  frame-relay map ip 10.1.145.1 501 broadcast
  no frame-relay inverse-arp
  frame-relay lmi-type cisco
!
interface Serial1
  no ip address
  shutdown
!
interface BRI0
  no ip address
  shutdown
!
router eigrp 100
  network 169.1.16.0 0.0.1.255
  network 169.1.18.0 0.0.0.255
  network 171.1.12.0 0.0.0.255
  network 171.1.32.0 0.0.1.255
  no auto-summary
!
router ospf 10
  router-id 10.1.5.5
  log-adjacency-changes
  area 1 virtual-link 10.1.4.4
  area 1 virtual-link 10.1.1.1
  area 5 nssa no-redistribution default-information-originate
  redistribute eigrp 100 subnets route-map EIGRP_OSPF
  network 10.1.5.0 0.0.0.255 area 0
  network 10.1.45.0 0.0.0.255 area 0
  network 10.1.55.0 0.0.0.255 area 5
  network 10.1.145.0 0.0.0.255 area 1
  neighbor 10.1.45.4 priority 1
```

```
network 10.1.145.0 0.0.0.255 area 1
neighbor 10.1.45.4 priority 1
!
no ip http server
ip classless
!
!
!
access-list 1 permit 169.1.16.0
access-list 1 permit 169.1.18.0
access-list 2 permit 171.1.12.0
!
route-map EIGRP_OSPF permit 10
match ip address 1
set metric 0
!
route-map EIGRP_OSPF permit 20
match interface Loopback7
set metric 150
set tag 1
!
route-map EIGRP_OSPF permit 30
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
endrouter eigrp 100
network 169.1.16.0 0.0.1.255
network 169.1.18.0 0.0.0.255
network 171.1.12.0 0.0.0.255
network 171.1.32.0 0.0.1.255
no auto-summary
!
exit

router ospf 10
router-id 10.1.5.5
log-adjacency-changes
area 1 virtual-link 10.1.4.4
area 1 virtual-link 10.1.1.1
area 5 nssa no-redistribution default-information-originate
```

```
originate
 redistribute eigrp 100 subnets route-map EIGRP_OSPF
 network 10.1.5.0 0.0.0.255 area 0
 network 10.1.45.0 0.0.0.255 area 0
 network 10.1.55.0 0.0.0.255 area 5
 network 10.1.145.0 0.0.0.255 area 1
 neighbor 10.1.45.4 priority 1
!
no ip http server
ip classless
!
!
!
access-list 1 permit 169.1.16.0
access-list 1 permit 169.1.18.0
access-list 2 permit 171.1.12.0
!
route-map EIGRP_OSPF permit 10
 match ip address 1
 set metric 0
!
route-map EIGRP_OSPF permit 20
 match ip address 2
 set metric 150
 set tag 1
!
route-map EIGRP_OSPF permit 30
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

### IGRP-EIGRP

IGRP 和rip的比较:

更具有扩展性

使用复合度量值

周期性广播更新，每90s一次（rip的三倍），与rip相比占用cpu少，收敛速度也较慢

收敛速度也较慢

更有效的更新包格式

使用进程域：是指在一个域里只能运行一个路由协议（比较：路由域是指可以用多个路由协议的区域）

cisco私有，支持多协议（支持ip, ISL CLNP）

不需要建立邻居关系

利用ip, 协议号9进行传输

HOP Count

更改hop的命令：metric maximum-hops

Bandwidth & Delay

可配置，但是不影响实际的物理性能，是一个接口参数，而不是链路参数。

注意单位和在路由更新包中的保存形式，不要轻易更改，但应该更改串行接口的带宽值，默认为1544kbit,

应根据你自己设置的clock rate.

metric的计算是沿路由的出接口方向计算，即是路由更新的入接口方向  
loopback: BW:8Gbit 但它在更新包中保存形式是 $10^7/8\text{Gbit}$

Delay:5000微妙，但它在更新包中的保存形式是“5000除以10”

非等价负载均衡：

最大6条，默认4条；

通过variance更改。（variance的计算是通过用大的metric值除以小的metric值，取整）

做非等价负载均衡时下一跳要更接近源。（可以理解为AD<FD）

命令：

```
router igrp as
```

```
traffic-share balanced | min    用路由metric值小的做负载均衡
```

```
no validate-update-source
```

```
timers basic 90 270 280 630
```

EIGRP特性

高级的DV协议，因为它的路由更新包发的是路由条目

无类的

使用DUAL算法

广播/组播更新，用224.0.0.10(rip v2用224.0.0.9)

100%无环

非周期的，增量的更新

可靠的，有重传机制

非周期的，增量的更新  
可靠的，有重传机制  
支持等价和不等价负载均衡。

RIB: routing information database

EIGRP 的包:

hello 此包每五秒发一个，hello包包含k值，进程号  
update是增量更新  
query 此包包含：所要查询的路由  
reply  
ack 可靠重传机制

update, query, reply需要ack确认

eigrp重传时使用单播，重传最大次数为16次

debug eigrp packet  
debug ip eigrp

EIGRP的汇总:

手工汇总: 是基于接口进行的

当做了汇总时，路由器会马上生成一条指向null 0 的汇总路由，用于防环。

当最后一条明细路由down了，汇总路由会自动删除。

汇总路由使用明细路由中最小的metric

手工路由与自动汇总谁更优先?

对哪些明细路由进行汇总?

是否抑制被汇总的明细路由?

是否产生用于防环的指向null0接口的汇总路由

有明细，汇总才会生效。且该明细需要宣告在该路由协议中

EIGRP的认证:

只支持MD5;

配置和RIP一样;

是一个关于邻居的认证，即是双方都必须认证成功，邻居才能建成；而

rip是一个包的认证

注意：关闭eigrp水平分割需用：no ip split horizon eigrp 100

EIGRP独有的: debug eigrp

# 重分布:

- 1、基于路由表
- 2、寻找被直连取代的路由
- 3、redistribute connected  
还有个router map

路由器是按照路由协议的优先级依次重分布的（且路由器认为一条路由只能重分布一次）

对于链路状态协议的来讲更新源就是router-id  
对于距离矢量来讲更新源就是出包的接口地址

ospf选路由的顺序: 0-->0 IA-->0 E1-->0 E2

## OSPF

OSPF的知识点:

一、三张表: neighbor table  
topology table  
route table

二、网络分层, 各种区域及各种LSA类型  
ospf的hello包全是发往224.0.0.5  
对于ospf来讲, 域内是链路状态, 域间是距离矢量。因为OSPF 只知道去往目标网络的ABR的路径

问题1: 是不是建立了邻接关系就能学到各自的路由?  
不一定

问题2: DR和BDR之间如何看LSA更新地址?

IGP的选路原则:

- 1、下一跳可达
- 2、最长匹配
- 3、AD最小
- 4、cost最小

LSA4是有ASBR所在的ABR产生，它是向其他区域产生的路由条目，但本区域不会有LSA4。

OSPF的区域0和普通区域拥有全路由（即有1类，2类，3类，4类，5类）。唯一不同点，就是区域0可以传递其他区域的LSA。

默认情况下，stub，total stub，和total nssa的默认路由是3类的

### 三、ospf的包类型

hello是以组播形式发的。

LSACK是单播发送的。

（看看其他几种包的传输发式？）

### 四、OSPF接口的各种网络类型

|                | 32位主机路由 | hello    | nei |
|----------------|---------|----------|-----|
| hello LSA      |         |          |     |
| 1、环回口          | 有       | 没hello间隔 | 可以打 |
| neighbor命令，但不到 | ？       |          |     |
| 2、p2p          | 没有      | 10s      | 不可以 |
| 打              | 组 组     |          |     |
| 3、p2mp         | 有       | 30s      | 可以  |
| 组 组            |         |          |     |
| 4、NBMA         | 没有      | 30s      | 可以打 |
| 单 单            |         |          |     |
| 5、b            | 没有      | 10s      | 不可以 |
| 打              | 组 组     |          |     |
| 6、p2mp-non     | ---     | 30s      | 可以  |
| 单 单            |         |          |     |
| 7、虚            | 没有      | 10s      |     |
| 单 单            |         |          |     |

对于虚链路来讲，hello只在虚链路建起来之前发，但，当虚链路FULL之后，hello被抑制，LSA也不发了，当链路有变化时，才会发LSA。

当虚链路建起后，更改两端的dead-interval，虚链路不会down，再一边做认证，一边不做认证，虚链路也不会down，但当另外给条路由给它时，虚链路就down了，因为当虚链路当建立成功后，就不会再发hello包，但是

它时, 虚链路就down了, 因为当虚链路当建立成功后, 就不会再发hello包, 但是当发一条路由给它, 即给它一个路由更新消息, 因为更新消息中包含认证的字段, 当这个包去匹配时, 虚链路就认证不成功, 所有就down了.

P2P子接口, 串口(直连)默认是P2P  
帧中继, 以及帧中继多点子接口等默认是NBMA  
以太网默认是B

neighbor这条命令只能在NBMA, P2PMP网络中的使用

问题3: LSA与HELLO发送的模式是不是必须要一样? 还是可以有区别?

问题4: LOOPBACK会不会发LSA??? (它是不发hello包的)

五、OSPF建邻居的必要条件以及各种过程

OSPF 建邻居的必要条件:  
hello间隔  
区域号  
验证  
stub

结论: 在多进程时, 哪个进程先起动, 就和哪个建邻居。 (在NBMA模型中试试这个结论?)

先起动的发hello包, 后启动的不发hello包。  
一条链路只能宣告在一个进程里。  
后启动的进程是失效的。

ROUTER(config-if)#ip ospf mtu-ignore

用mtu X 最小是64;  
用ip mtu X 最小是68;  
(这两条命令有什么不同?)

当MTU不匹配的时候, 会停在EXSTART状态!

当网络类型选DR或BDR时, 对子网掩码有严格要求。

改变了hello 时间, dead会自动改变;

改变了hello 时间，dead会自动改变；  
改变dead时间，hello不会自动改变

rip的触发更新只能用在串口，而不可以使用在以太网口。因为是为了在低速链路上减少带宽的消耗。

**问题5：P2MP为什么会产生一条32位的主机路由？有什么用？**

LSA1包括我自己直链的网络，邻居的ROUTER-ID，当LSA1对同一条链路描述的网络的类型不一样时，它不能计算路由，所以不能写进路由表，但可以建邻居，因为建邻居是通过hello包建立的，建邻居时和hello包的发包形式是无关的。

在NBMA中，如果spoke端（默认优先级为0）手工指定neighbor，不会在sh run 中看到neighbor这条命令！

但是在sh ip ospf neighbor时可以看到该命令，会显示尝试连接，应该要删除掉此命令的，只要将接口的优先级改为比0大，就可以在sh run 中看到。

当NBMA的网络模型指定neighbor之后，再将该接口改为另一个网段的ip地址，将不能取消原来指定neighbor的命令，除非将接口改回原来的ip地址。

**用debug ip ospf adj看看各个网络模型的建立邻居的过程！！！！**

为什么要建立这么多网络类型？？

NBMA：

二层不支持B条件：当关闭反向解析，其次，在fram-relay map中不加Broadcast；

点到多点相当于多个点到点；NBMA相当于以太网模型（多访问网络），只是不能够发包；

P2MP的下一跳在HUB端

帧中继环境下，主接口和多点子接口默认网络类型是NBMA，应该在子接口下关闭反向解析。

SWITCH#sys mtu 1200 (用于改变交换机的MTU，需改变后保存重启)

当本路由和ASBR在同一个区域，不需要四号LSA，也可以学到外部路由；  
当本路由和ASBR不在同一区域，要四号LSA，才可以学到外部路由

类型四是由ASBR所在区域的ABR产生的，当经过边界路由器时（即穿越区域时），它的公告路由器将会改变

几种LSA所描述的东西：

- 1 自己直连的
- 2 描述的是那条链路
- 3 路由
- 4 ASBR的router-id
- 5 路由

| 邻居?有路由?             | P2P     | P2MP     | Broadcast | NBMA   | P2MP-non |
|---------------------|---------|----------|-----------|--------|----------|
| P2P                 | -----   | Yes/ Yes | Yes/No    | Yes/No | Yes/Yes  |
| P2MP                | Yes/Yes | -----    | Yes/NO    | Yes/No | Yes/Yes  |
| Broadcast<br>Yes/No | Yes/No  | -----    | Yes/No    | Yes/No |          |
| NBMA                | Yes/No  | Yes/No   | Yes/Yes   | -----  | Yes/No   |
| P2MP-non            | Yes/Yes | Yes/Yes  | Yes/No    | Yes/No | -----    |

P2P和Broadcast只能建立邻居,但是没有路由,因为P2P描述网络是一个stub区域,而Broadcast描述网络是一个transit区域,由于对该网络类型的描述不一致,所以不会通LSA计算出路由.

p2p, p2mp, p2mp-non都描述一个区域为stub区域

Broadcast和NBMA都描述一个区域为a transit区域

重分布直连时一定要挂router-map, 把需要重分布的接口匹配进协议。

当database能看到五号类型，却看不到四号分 类型，路由表也能看到这条路由，说明这个ASBR在我的本区域里。

如果在database能看五号类型，在路由表中不显示这条路由时，说明我本区域内没有一个ABR和公告这条路由的

ASBR在同一个区域。

虚链路的作用：1、将没有和 area 0 相直连的区域拉近area 0

2、用于area 0 的备份

- 2、用于area 0 的备份
- 3、将两个不连续的 area 0 连起来。





## RIP

## RIP

思路

协议类型

有类还是无类

管理距离/metric

使用bellman-ford算法

发送“全部”路由条目

使用udp传输, 端口520

距离矢量:

distance(metric)

vector(接口或者下一跳)

DV-的本质就是指路由条目;

DV协议的本质是指更新包发的是路由条目;

“有类”路由协议

路由更新中不携带子网掩码.

在相同的网络, 必须配置相同的子网掩码(收发机制)

自动在主类网络边界(汇总了会不会抑制明细)

不支持手工汇总.

ip classless

影响路由器查找路由表方式, 不影响路由协议工作

no ip classless 有类的查找 : 先查找主类网络, 有主类了再找子网, 没子网就丢, 不会出来找默认(即使有默认路由也不会查找默认路由). 如果没有主类就走默认.

ip classless 无类查找. 即最长匹配, 在如果找不到明细路由的情况下, 最终也肯定会匹配默认路由.

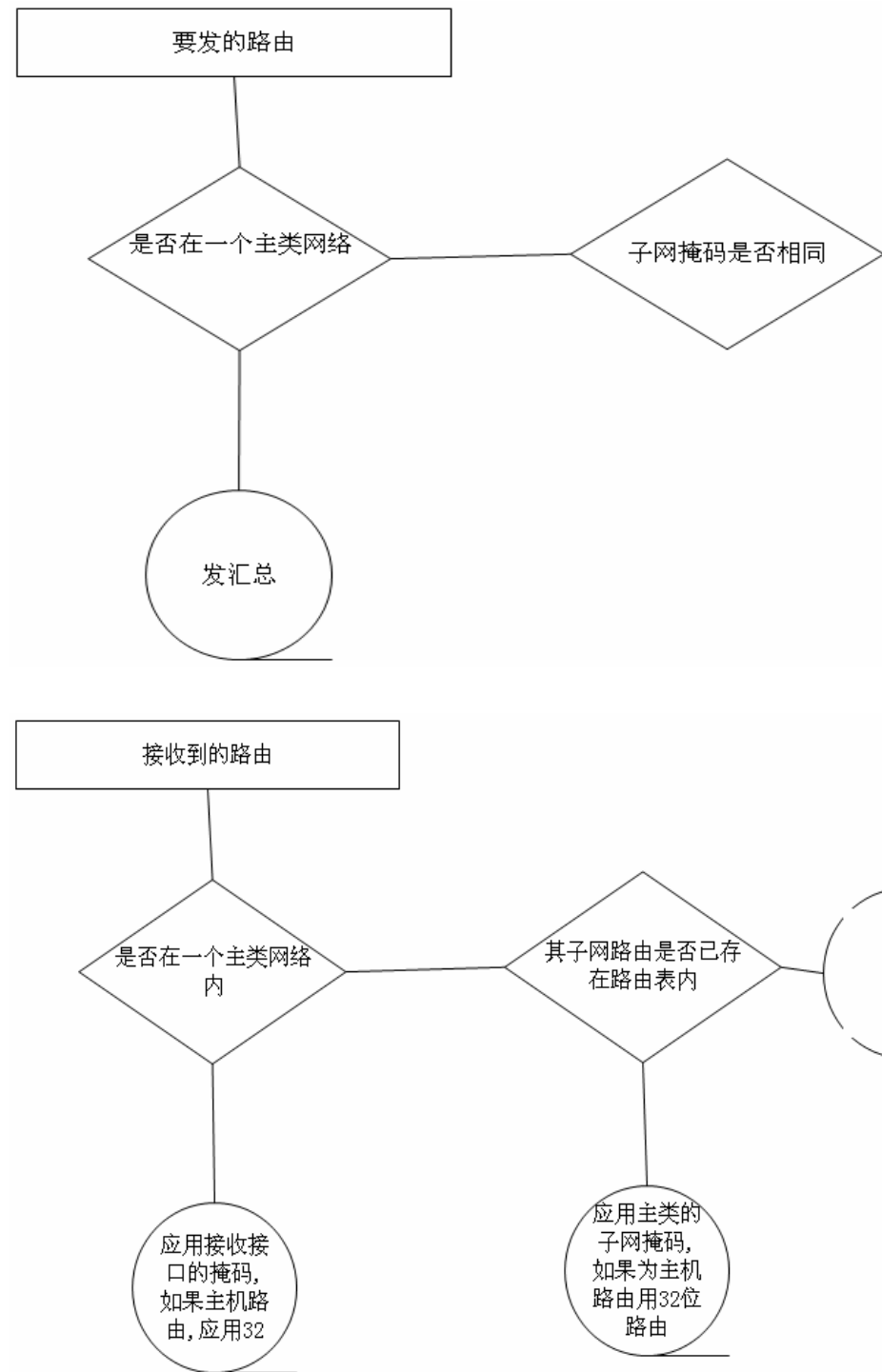
路由更新不带子网掩码:

掩码以接收口的为主.

不连续子网:(一个主类网络的子网被另一个不同的主类网络分开了)

了)

有类边界:指两个不同的的主类网络...(不是以一个A类、B类来定义的.  
主类网络里的网络的不同网络号.)



LAB1:

L01:172.16.1.1/24 ->发(子)

L02:172.16.2.1/24 ->发(子)

L03:172.16.3.129/25 ->不发

L04:172.16.4.1/23 ->不发

L05:172.31.1.1/24 ->发(主)

L06:172.31.2.1/25 ->发(主)

L07:172.32.3.1/23 ->发(主)

L08:172.33.1.1/32

L09:172.16.5.1/32

发接口:172.16.16.1/24 s0

扩展:接收方加已loopback:172.32.4.1/23, 观察发送方lo7路由条目

同一主网发明细, 但要求子网掩码相同, 或者为主机路由; 不同主网发  
汇总

接收同一主网路由应用接口掩码;

接收不同主网路由应用主类掩码;

但要求没有该主网的子网路由在路由表中如果主机位不全为零, 则应用  
32位主机掩码。

timers of RIP

update:30s (控制自己发的)

invalid timer:180s (即是路由表在180s内没有收到关于这个路由的  
任何更新。等待对方发来的)

invalid timer之后rip所执行的动作:

1、将该路由的跳数改变为16, 并向所有运行RIP的接口发送更新。

2、该路由在路由表中被显示为possible-down。

3、该路由在路由器中仍可用, 即仍然可以被路由。

hold-down timer:180s (跳数16跳的路由发送时间)

当180s的Invalid time过后, 就进入了hold-down时间, 在这个时间内  
不管来的跳数是多少, 路由器都

不与理睬, 但是再过60s, 该路由条目已经在路由表中被删除, 因为  
flush计时器与invalid计时器是同时

开启的。

flush timer:240s (此计时器与invalid计时器是同时开启的)

即过完invalid计时器的180s后, 再过60s此路由条目在路由表中被删  
除。

即过完invalid计时器的180s后, 再过60s此路由条目在路由表中被删除。

sleep:触发更新的间隔(即收到触发更新后隔多久再更新)触发更新用, 默认马上发. 挫开更新时间, 防止同时更新. (单位毫秒)  
全网的计时器必须一致....

要更改定时器时间的话用:

```
router rip
timer basic 30 180 180 240
```

~~~~~  
~~~~~

version2

classless  
multicast update  
route tag  
next-hop  
更新包的认证

classless routing protocol  
include the subnet mask with the route advertisement .  
support vlsn  
support manual router summarization  
does it support CIDR? how does it support CIDR(rip V2 支持, 可以接收)不支持汇总成超网...

RIP Basic Configuration

```
router rip
version 2(一个宏~~~所有接口上都收发version2的更新)
(config-if)#ip rip send | receive
即启用自动汇总. 又使用手工汇总, 手工汇总不起作用....
neighbor X.X.X.X(如果不passive, 广播、单播一起发)
passive(建议先passive接口再network, 建议在network前先考虑是否要passive某些接口)
(passive是只接收路由而不以广播的方式发送出去)
passive-interface default (passive所有接口)
network:宣告接口(告诉路由器那些接口启用rip协议), 并将该接口的网络通告出去.
```

口的网络通告出去.

(eigrp与ospf的network的反掩码必须连续的零与一(控制列表的是通配符))

(config-if)#ip summary address rip x.x.x.x x.x.x.x 只在该接口上发汇总 ----试

水平分割

只有在封装了帧中继的**主接口**上, 水平分割才是默认关闭的. 只有在NBMA (帧中继环境)的hub-spoke模型的

hub端才需要关闭水平分割,

在spoke端要手工关闭

sho ip interface (查看IP层面的东西)

show int 是物理层面的...

多点子接口水平分割默认开启的.

maximum-paths 支持多少条等价负载的条目

validate-update-source

在接地址的接口情况下, 默认是关闭源监测机制的

timer basic 30 180 180 240 20  
sleep

对于rip来讲, 启动协议的是network命令。

查看rip的三个命令: sh ip protocols

sh ip route

sh ip rip database

## 交换

### 交换

知识点:

1. 网络设计的三层结构

2. VLAN/TRUNK/VTP
3. Spanning-Tree protocol
4. 多层交换
5. 高可用性, 冗余
6. 组播
7. QoS

一. 访问层: 提供端口密度, 能做简单的安全, vlan的划分  
分布层: 做路由, 策略  
核心层: 高速的包交换

~~~~~  
~~~~~

二. VLAN: 就是不受地理限制的一个广播域

把接口划进vlan的方法:

1. 静态(基于接口的)

```
if#switchport mode access
if#switchport access vlan vlannumber
```

2. 动态(基于MAC表) (VMPS VLAN管理策略服务器)

a. 配置VMPS服务器在哪: 进程下: `vmps server 10.1.1.1` 此命令使该交换机成为VMPS的一个客户端

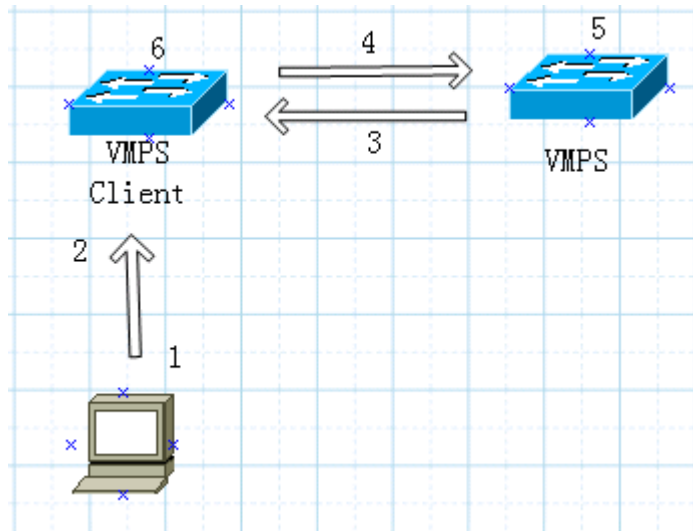
```
#vmps server x.x.x.x
```

b. 将接口划进动态vlan

```
if#sw mo ac
if#sw ac vlan dynamic
```

动态vlan的建立步骤:

1. 接口属于动态vlan
2. 接入一台设备
3. 向mps查询



步骤:

- 1 the pc send a frame to the switch
- 2 the VMPS client learns the PC MAC address ont the dynamic port
- 3 the VMPS client sends a VQP request to the VMPS.the request:contains the VMPS client: ip address, the PC MAC address, the PC port number, and the VTP Domain.
- 4 the VMPS parses its database file for PC VLAN assignment
- 5 the VMPS sends a VQP response to the VMPS client
- 6 if the VQP response contains a VLAN assignment, the VMPS client assigns it to the VLAN. Otherwise, it denies the PC access.

测试MAC地址表:

sh mac-address-table

静态指定MAC地址到某个vlan:

```
sw1(config)#mac-address-table staic 00e0.b05a.5ca8 vlan 1 int f0/1
```

查看 sh mac-address-table int f0/1

mac地址老化时间是5分钟

sh arp

在路由器上将某个地址放进arp表中

```
#arp x.x.x.x h.h.h.h arpa
```

~~~~~  
~~~~~TRUNK

思科建议trunk的模式为点到点链路, 不建议为共享链路  
在一条链路上可以承载多个vlan的流量为trunk

二层接口的几种模式:

access  
trunk  
nonegotiate  
dynamic desirable  
dynamic auto

~~~~~  
~~~~~

802.1q两边必须要一样, 否则cdp会报错, 因为会产生桥接环路, cdp会看到vlan的信息.

当trunk作冗余时, 会block端口, 当不同vlan选的根桥不一样的时候, 两个vlan block的端口也不一样.

假如上边的交换机native vlan为vlan 1, 下边的交换机native vlan 是vlan2, 上面的交换机从左边的端口发出一个vlan1 的数据帧, 这边会误以为是vlan2的, 假如是一个广播帧, 那么它会往vlan2的端口去泛洪. vlan2上面那条链路会往左头, 阿么就原始以太网帧就发回去了, 这边收到原始以太网帧会以为是vlan1的数据帧, 然后vlan1的数据帧再进行泛洪.

#sh vlan id vlannumber

1025--4094与vtp模式有关

Q-in-Q:

if#switchport mode dot1q tunnel  
if#switchport access vlan 30

~~~~~  
~~~~~

vtp:

vtp协议的作用

运行vtp的条件:

运行vtp的条件:

1. domain一致
2. trunk端口
3. c/s

sh vtp status

修订版本号是大的覆盖小的版本号

vtp有三种报文类型:

summary 由server发过来的  
subset 子设置报文, 主要是cline去同步  
request 负责请求vtp信息的

vtp域名不一样, trunk协商会否失败.  
DTP中也包含vtp的信息

~~~~~  
~~~~~  
vtp 修剪为了增大可用带宽, 可以减少不必要的泛洪流量

~~~~~  
~~~~~  
交换机并不知道根桥在哪里, 也不知道网络拓扑, 知道根桥是谁, 根桥应该放在哪里应该由人来涉及, 从而使网络更优化.

~~~~~  
~~~~~  
CST:所有的VLAN的流量是京过同一条链路转发的

PVST:为了让链路负载, 通过改变根桥负载. 提供了一定的灵活性, 但并不知道所有vlan的流量是多少, 并不能做到完全的负载均衡.

pvst的四种方法:

改变根桥  
路径 cost:改指定根端口  
send bridge id:选指定根桥  
改pri-id 选指定端口

MST(多生成树):把一组vlan定义在一个生成树里.

```
spanning-tree mst configuration
instance 1 vlan 20-30          (1代表一个组, 里面的vlan
包括20--30)
exit
```

```
spanning-tree mst 1 priority
```

MST存在域的概念, 即域内的所有交换机都要配置, 维护同一个组.

```
~~~~~
~~~~~
```

```
spanning-tree vlan 10 root promary 默认情况下降了两级, 减了两个
4096
```

```
~~~~~
~~~~~
```

spt

SPT是为了打破交换机冗余的交换环路而出现的一种技术

不用生成树会发生

-广播风暴

由于二层无TTL概念, 所以一发生环路, 就会无休止的打环

-多帧复制. -拷贝

主机分别收到两个由两个链路收到的数据帧

-mac地址表的翻动

由于会环这发送帧, 所以会在不同接口接收到相同的数据帧, 使得mac地址表的不稳定

一个mac地址对应一个端口

show version 由一个basemac地址, 这是接口mac地址中最小的.

cisco 有一个特性. 可以对风暴进行控制

```
int f0/1
storm-control broadcast level 30(百分比)
```

通知管理员 广播对链路的消耗  
超过百分比后, 交换机会发trap信息, 同构snmp发送给snmp服务器

storm-control action shutdown  
超过百分比自动shutdown接口

spanning-tree protocol

发送BPDU

形成一个无环的拓扑  
树的特点:无环, 最短 (节点到树根)

生成树只有一个树根:root bridge

one root bridge per network  
one root port per nonroot bridge  
one designated port per segment  
nondesignates ports are blocked

root bridge all port are designated port  
非根网桥只有一个根端口  
每个端只有一个指定端口

bridge ID -8字节  
2字节6字节  
bridge priority MAC Address  
BPDU有8个字节, 前两个字节为优先级, 后6个字节为MAC地址, 优先级默认为32768最大为65530 (0X8000)  
优先级是以4096递增的. 因后面的12尾为是必须为0的.

port-id  
两字节,  
第一字节是优先级, 后一字节是编号  
优先级后4比特必须为0, 既以16递增

整个BPDU的格式  
bytesfield  
2 protocol-ID  
1 version  
1 message Type

```
1   flags
8   root ID
4   cost of poth
8   bridge
2   prot-id
2   message Age
2   maximum time
2   hello time
2   porwoed delay
```

message age:bpdu的老化. 交换机每转发一次BPDU,message age就加1 根桥发出的BPDU message age为0

message type 有两种,a.configuration bpdu b.tcn bpdu

flags:在802.1D里,有8位 只有前后两位可以使用,前面以为叫TCA,一个TA 用于拓扑变化

发送的时候不加本端口的链路cost,既进的时候加cost

maximun time 和端口状态有关

bridge id:谁发送的BPDU,就是这个bridge id

port id 发送放的port id (0X80=128 0xXX)

hello time 和端口状态有关

所有的交换机都要跟跟桥的计时器保持一致. 而跟桥的计时器是用来防止环路的.

生成树是如何进行选举的. 如何去进行收敛.

端口状态,生成树如何打破桥接环路呢 ,是将一个端口给阻塞,这个端口数据是不敏感的,很此段的.(发也不发,收也不收,学习也不学习) 既block

转发端口:既收发也学习, forward

一个刚接入的交换机

listening 发出bpdu进行选举,这是收到用户数据是不发送的,mac地址也

址也不学习, 不会字节变为forwarding状态, 因为每有mac地址, 所有的帧都会是广播帧.

forwarding 可以对用户数据根据mac地址表进行转发.

block状态下, 是做什么的  
不处理用户数据, 但可以处理bpdu, 接收bpdu, 不发送bpdu.

第一个比  
root id  
path cost  
sender bridge id  
sender port id  
receiver port id

当链路断了后, 它认为自己是跟桥. block的端口收到一个劣质的bpdu

生成树的选举原则.  
lowest root BID  
lowest path to root bridge  
lowest sender BID  
lowest sender port id

show spanning-tree int f0/2 detail  
看bpdu的值  
我这个端口属于那个vlan, 它就在那个vlan的stp里选举根端口或者是指定端口lan多少它就加多少, 因为vlan一共有4096个, 所以前面说到的, BID的优先级以4096递增的原因.

如果做了port-channel port-id会变

port-priority默认128

每有可能以一个交换机收到的port-id 一样  
self-loop 不建议

比较DP

RP和block的端口是不会发送BPDU的, 只会从DP发BPDU

show spanning-tree  
查看各个vlan的spanning tree的状况

show spanning-tree vlan1

查看特定vlan的spanning-tree的状况

```
show int status
```

-----  
发送tcn的情况

- 1 当一个交换机,的端口被选为block端口时
- 2当以一个block被选位根端口或指定端口时
- 3进入转发状态的时候会发TCN

发TCN往根端口发.发到上一个交换机时,肯定时指定网桥确认TCA 发送下一个BPDU的时候TCA会置位.不断的传递到根桥

为什么会有TCN的存在

当拓扑发生变化.就从RP发TCN,在下以个2秒会收到以个回复TCA (BPDU的置位),如果每有回复会一直发.当收到TCN,会把TCNRP发送出去  
当RB收到TCN的时候它就发送TC.持续发送.持续35秒.不需要确认

跟桥知道后,会在后面的configurtaion bpdu的tc置位.收到tc置位的BPDU的交换机会将他们的MAC aging-time.改位15秒.,正好是一个forward-delay. TC置位的BPDU会持续发送35秒,既为max-aging 和 forwoed-delay 让block的端口改变

tcn是从叶往根发的  
tca是对tcn的恢复  
tc是从根往叶发的.

-----  
交换机不知道整网络的拓扑,也不知道跟桥在那里  
只知道跟桥是谁,所以跟桥应该放在那里,应该由我们人来设计

CST所有的vlan都是根据一个生成树的.  
pvst(per vlan spanning-tree) cisco私有  
基于每个vlan就有一个自己的生成树对象.以一个交换机最多只能由 64棵树  
资源占用比较多  
pvst+完全一样,就可以基于.q1的支持  
从流量的地方考虑,来以个负载均衡  
-最常用的方法改变跟桥的位置  
PVST提供了一定的灵活性.当并不能知道所有vlan的流量是多少,并不能

PVST提供了一定的灵活性. 当并不能知道所有vlan的流量是多少, 并不能做到完全的负载均衡

-改变指定端口

方法一. cost 在接口下打spanning-tree cost

方法二 sener port-id

spanning-tree vlan 10 root primary

比本域内最小的优先级. 降两级

如果由不是默认情况的话, 就比那个值降一级

secendry 就比默认的降一级

MST

m-单一的独立的

m-多生成树

多个vlan共同使用以个生成树.

前提是不会有干扰

802.1s

spanning tree mst

定义实例名只能定义0-15个组

spanning-tree configurtion

instance 1 vlan 20-30, 31, 40

spanning-tree mst 1 priority

~~~~~  
~~~~~

if#spanning-tree cost xx 改所有vlan的cost

if#spanning-tree vlan cost 更改某个vlan的cost

spanning-tree vlan 10 root primary diameter 定义生生树的直径  
(即生成数从根到末节经过多少台交换机)

Port-Fast

debug spanning-tree events

bpdu guard

if#spanning-tree bpduguard enable/disable

此接口不接收BPDU. 如果收到bpdu, 就将接口变为err-disable状态.

errdisable recovery interval 更改errdisable的时间间隔

```
#sh int f0/1
```

```
~~~~~  
~~~~~使用BPDU-filter功能，将  
能够防止交换机在启用portfast特性的接口上发送BPDU。
```

Catalyst交换机支持以每个端口或全局为基础而配置BPDU-filter。

a. 接口上明确配置了BPDU-filter的功能，那交换机将不发送任何BPDU，并且将把接收到的所有BPDU都丢弃。

b. 如果全局配置了BPDU-filter的功能，并且如果端口在各自接口接收到任何BPDU，那么交换机将会把接口更改回正常的STP操作。

警告：如果在连接到其他交换机的端口上配置BPDU-filter，那么可能导致桥接环路；在部署BPDU过滤的时候，用户应当格外小心。BPDU过滤不是一种推荐配置。

| 每端口配置<br>态 | 全局配置 | portfast状态 | portfast BPDU过滤状<br>态 |
|------------|------|------------|-----------------------|
| 默认         | 启用   | 启用         | 启用 (*)                |
| 默认         | 启用   | 禁用         | 禁用                    |
| 默认         | 禁用   | 不适用        | 禁用                    |
| 禁用         | 不适用  | 不适用        | 禁用                    |
| 启用         | 不适用  | 不适用        | 启用                    |

\*端口至少发送10个BPDU。如果端口接收到任何BPDU，那么交换机将禁用portfast和BPDU过滤特性。

如果在与启用BPDU-filter的相同接口上启用BPDU-guard，那么因为BPDU-filter的优先级高于BPDU-guard，所以BPDU-guard将不起作用。

全局性启用BPDU-filter：

```
spanning-tree portfast bpdupfilter default
```

```
show spanning-tree summary totals
```

```
~~~~~  
~~~~~
```

```
uplink fast: debug spanning-tree uplinkfast
```

访问层交换机的rp丢失后，另外一

访问层交换机的rp丢失后, 另外一

#spanning-tree uplinkfast max-update-rate  
max-update-rate每秒更新多少个包

backbone fast  
做在核心层

L1断电后, 收敛时间50秒





## 其它知识

### IPv6

IPv4现在存在的缺陷:

地址短缺(2的32次方)

复杂的包头

复杂的路由器和主机配置

重编址比较困难

很大的路由表(CIDR没有做好)

部署安全, 组播, mobile-ip比较麻烦

解决方案:

NAT(一个NAT表项会耗费CPU的64K, 由于隐藏内部ip地址, 却破坏了端到端的模型, 不能快速的重路由)

DHCP

CIDR

### IPv4 IPv6的比较

| 服务   | Ipv4          | Ipv6                            |
|------|---------------|---------------------------------|
| 地址空间 | 32bit         | 128bit                          |
| 自动配置 | DHCP          | 无状态自动配置/DHCP                    |
| 安全   | Ipssec        | 可以做端到端的ipsec加密                  |
| 移动性  | Mobile-ip     | Mobile-ip with direct routing   |
| QoS  | 区分服务/集成服务     | 区分服务/集成服务                       |
| 组播   | IGMP/PIM/MBGP | MID/PIMMBGP,scope<br>identilser |

### IPv6的特性

大的地址空间

提供了autoconfigure的技术, 即插即用

简单高效的包头

提高了安全和移动性

有丰富的IPv4到IPv6的转换途径

大的地址空间

128bit

8段, 每个段16bit

书写IPv6地址时, 前导0可省略, 连续0省略(用::表示, 在ipv6地址里只能出现一次::)

ipv6自动配置的技巧:

路由器会向网络内发送一些网络信息(包括前缀, 缺省路由...)以组播的方法去发送信息

新加进网络的设备, 会收到这样一些网络前缀, 并以自己的MAC地址来构造IPv6地址

IPV6自动配置编址技术  
 IPV6地址=前缀+接口标识  
 接口标识:相当于v4地址中的主机id

在MAC地址中, 第七位为0, 代表全球唯一;  
 为1, 代表本地唯一;

在EUI接口中的第七为: 1 代表全球唯一  
 0 代表本地唯一

在路由器里用ipv6需打:ipv6 unicast-routing

ipv6地址分类

单播地址(Unicast Address)

组播地址(Multicast Address)

泛播地址(Anycast Address) 表示为:前缀::

特殊地址

| 地址类型   | 二进制前缀           | Ipv6标识    |
|--------|-----------------|-----------|
| 未指定    | 00...0(128bits) | ::/128    |
| 环回地址   | 00....1(128bit) | ::1/128   |
| 组播     | 1111,1111       | FF00::/8  |
| 本地链路地址 | 1111,1110,10    | FE80::/10 |
| 本地站点地址 | 1111,1110,11    | FEC0::/10 |
| 全局单播   | (其他)            |           |

单播地址:

ipv6单播地址分类:

全局单播地址 2001:A304:....

本地链路地址 FE80::E0:F726... (本地链路地址要唯一)

本地站点地址 FEC0:.....

全局单播地址层次结构

001 全局路由前缀 子网ID 接口ID  
           45位        16位        64位

001可表示为:2000--3FFF, 这是全局单播地址第一段的范围

配置ipv6地址

int e0

ipv add fe80::2 link-local 这是配置本地链路地址需在后面加link-local, 配置其他类型的地址则不用加

(ipv6一个接口上可以打N个地址, 且都生效)

本地链路地址:设备自动生成,在本地网络中使用  
本地站点地址:相当于ipv4网络中的私网地址(需要配置)

组播地址:

1111, 1111 | flags | scop | reserved must be zero | group ID  
8 4 4 80 32

flags:用来表示永久地址(0000)(代表全认识)或临时地址(0001)

scope:表示组播的范围:0001—本地接口, 0010—本地链路, 0011—本地子网, 0100—本地管理,

0101—本地站点, 1000—组织机构, 1110—全球

group ID:组播组ID

组播指定地址

| 组播地址    | 范围   | 含义    | 描述             |
|---------|------|-------|----------------|
| FF01::1 | 节点   | 所有节点  | 在本地接口范围的所有节点   |
| FF01::2 | 节点   | 所有路由器 | 在本地接口范围的所有路由器  |
| FF02::1 | 本地链路 | 所有节点  | 在本地链路范围的所有节点   |
| FF02::2 | 本地链路 | 所有路由器 | 在本地链路范围的所有路由器  |
| FF05::2 | 站点   | 所有路由器 | 在一个站点范围内的所有路由器 |

被请求节点组播地址

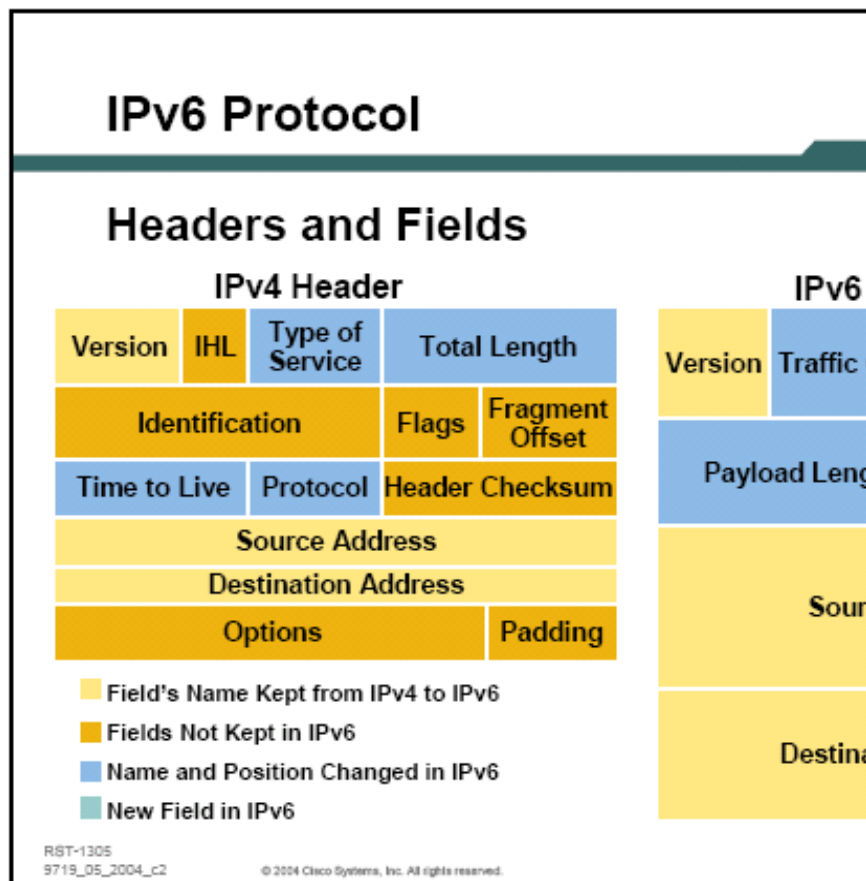
此类地址由前缀FF02::1:FFxx:xxxx/104和单播或泛播地址的低24位(即是MAC地址的低24位)组成,主要为两类特定类型的地址:替代ipv4中的ARP和重复地址检测(DAD)

泛播地址:是一种机制,它使到最近点的发现机制成为可能.泛播地址可使用可聚合的全球单播地址,也能够使用本地站点或本地链路地址.

IPV4兼容IPV6地址

一个设备会对将要设置的全局单播地址,本地链路地址,本地站点地址都产生一个被请求节点组播地址

ipv4和ipv6包头比较:



IPv4定义一个流是用源, 目的的ip地址, 源目的的MAC地址, 端口号

邻居发现协议 (NDP): ARP协议的替代协议, 无状态自动配置, 路由器重定向

其中无状态自动配置又包括前缀公告, 重复地址检测 (DAD), 前缀重新编址

(ARP协议的替代协议用于知道对方的地址)

自动配置:

无状态自动配置

有状态自动配置

无状态自动配置是ipv6中最吸引力和最有用的特性之一, 它涉及以下几种机制:

前缀公告: 在本地链路上公告前缀和参数. ipv6节点利用前缀公告信息来配置ipv6地址

重复地址检测 (DAD): 确保在接口上无状态自动配置的每个ipv6地址在本地链路的范围内唯一的

前缀重新编址: 在本地链路上公告修改了的前缀或新的

~~~~~  
~~~~~  
在没配ipv6地址前可以出现link-local, 是用MAC地址自动产生的Link-local

```
int s0
ipv6 enable
```

或

```
int s0
ipv6 add autoconfigure
```

或者直接配置Link-local地址:

```
int s0
ipv6 add fe80::1 link-local
```

接口配置地址时跟eui-64参数, 是使用link-local地址来填充后面的位数, 具体填充多少位, 要看前缀配置了多少位.

~~~~~  
~~~~~  
icmp v6(因特网控制消息协议)

icmp向源节点报告关于向目的地址传输IP数据包的错误和信息

| 消息    | 类型号 | 消息类型 | 定义                                |
|-------|-----|------|-----------------------------------|
| 目标不可达 | 1   | 错误   | 目的主机中的ip地址或者端口未处于活动状态             |
| 数据包超长 | 2   | 错误   | 数据包长度超过发送链路的最大MTU                 |
| 超时    | 3   | 错误   | 当存活时间ttl 字段为0的时候,数据包丢弃,中间路由器通知源主机 |
| 回应请求  | 128 | 错误   | 发送到目的地的消息,请求一个回应消息                |
| 回应该答  | 129 | 错误   | 用来回答回应请求消息的错误                     |

icmp v6的协议号是58, ipv6认为icmp v6数据包是一个上层协议, 像TCP和UDP一样, 意味着它必须被放在ipv6数据包中所有可能的扩展包头之后

在ipv6中, 协议的几种机制和功能使用icmp v6 消息

替代地址解析协议(ARP)用新的icmp v6消息

无状态自动配置, 用新的icmp v6消息

重复地址检测(dad) 用新的icmp v6消息

前缀重新编址 用新的icmp v6消息

路径发现协议

ndp用到的几种icmp v6的消息

RS(router solicitation) (icmpv6 type 133)路由请求(主机发)

RS(router solicitation) (icmpv6 type 133)路由请求(主机发)  
RA(router advertisement) (icmpv6 type 134)路由应答/公告(把前缀和一条缺省路由通告给PC)  
NS(neighbor solicitation) (icmpv6 type 135)邻居请求  
NA(neighbor advertisement) (icmpv6 type 136)邻居公告/应答  
Redirect(icmp type 137)重定向

```
debug ipv6 packet
debug ipv6 ndp
sh ipv6 neighbor
clear ipv6 nei
sh ipv6 mtu
```

刚配置了地址进行冲突检测时,是用未指定地址来发的,会进行多次重复检测,先进行link-local的检测,再进行自己所配置的地址地址进行检测

无状态自动配置通过发RA来执行,

思科认为串口不连主机,所以它不往串口发RA路由公告

ipv6 nd ?

参数:ra-interval:在以太网链路上,确认唯一的,间隔多少发RA,如果是路由器就不会收到RA里的默认路由

ra-lifetime

suppress-ra 抑制RA的发送

prefix 前缀 使用特定的前缀来配置主机ip

默认发送RA有效时间是30天

DAD和替代ARP都是用NS/NA, DAD的NS/NA是没有源地址的,是::,而替代ARP的NS是有源地址的,源地址是自己接口的地址,目的地址都是被请求节点地址

DAD检测:先检测link-local有没有冲突,即发link-local的NS,以未指定地址为源,以link-local组合而成的被请求地址为目的地址.(如果link-local冲突了,还会不会发单播地址的检测??)

其中,在检测link-local和单播地址时,发的NS的源地址都未指定(::)

在接口进行no shutdown,自动配置时就会进行DAD.

替代ARP协议: 当2001::1 ping 2001::2 时

第一步:2001::1是无法ping通2001::2,由于没有MAC的映射

第二步:2001::1这个地址发一个NS包,以源:2001::1,目的:ff02::1:ff00:2(是通过2001::2构成的一个被请求节点地址)

替代ARP协议: 当2001::1 ping 2001::2 时

第一步:2001::1是无法ping通2001::2, 由于没有MAC的映射

第二步:2001::1这个地址发一个NS包, 以源:2001::1, 目的:ff02::1:ff00:2(是通过2001::2构成的一个被请求节点地址)

第三步:由于2001::2检测ff02::1:ff00:2, 所以它会回一个NA包, 以源:2001::2, 目的:2001::1(包中包含了2001::2的MAC地址)

第四步:此时2001::1收到包后, 就建立了2001::2与其MAC地址的映射

~~~~~  
~~~~~

ipv6协议

RIPng:以本接口的link-local地址为源地址, 默认以FF02::9为目的地址, 源端口和目的端口都是521

RIPng以link-local来当下一跳的

step1:启动RIPng

ipv6 router rip CISCO(进程里)

step2:在接口里启用rip进程

int s0

ipv6 rip CISCO enable

debug ipv6 rip

clear ipv6 route \*

~~~~~  
~~~~~

int s0

ipv rip CISCO default-information

only 只发默认不发明细

or 都发

~~~~~  
~~~~~

ipv6 router rip CISCO

port 更改发更新的接口

只有在Frame-relay 环境主接口默认是关闭水平分割的.

~~~~~  
~~~~~

~~~~~  
~~~~~

ospf v3

ipv6 router ospf 1(进程号)

if#ipv6 ospf process area area-id

ipv6的ospf v3:

增加了2个LSA:Intra-area-prefix-lsa link-lsa

原来的2个LSA被改名:summary-lsa被改为inter-area-prefix-lsa

asbr-summary-lsa被改名为inter-router-

prefix-lsa

组播地址变化:

ALLSPFROUNTER FF02::5

DR/BDR: FF02::6

OSPF V3 一定要配置router-id,用点分十进制来配置

~~~~~  
~~~~~

ISIS for ipv6 基于draft-ietf-isis-ipv6-05.txt

为了携带IPV6路由信息,引入2个新的TLV,

IPV6 Reachability TLV(0xEC)

IPV6 interface TLV(0xE8)

一个新的网络层协议标识符(NLPID)

~~~~~  
~~~~~

MBGP:同样有ibgp,ebgp的概念;  
用地址簇来做的

BGP-4基于RFC1771

BGP-4 plus基于RFC2858,BGP-4的多协议扩展

BGP update 中只有以下字段与IPV4具有相关性

next-hop aggregator NLRI

BGP-4 plus增加2个新的属性:

MP\_REACH\_NLRI

MP\_UNREACH\_NLRI

no bgp default ipv4-unicast 不默认启用IPV4的BGP功能

进程:address-family

参数: ipv4

ipv6

需将其激活:

router bgp 1

add ipv6

nei 2012::2 active

~~~~~  
~~~~~

ipv4和ipv6网络的共存与整合

双栈协议(Dual stack)网络中的主机,服务器和路由器可以同时使用

ipv4和ipv6协议

隧道协议(Tunneling)隧道使孤立ipv6主机,服务器,路由器和域利用现有的ipv4基础设施与其他ipv6网络通信

建tunnel的主机或路由器必须是一个双栈.

协议转换:使ipv6网络上的ipv6单协议网络节点与ipv4网络上的ipv4单协议网络节点进行通信

ipv4和ipv6的转换;

可以用GRE来做,GRE可以封装一些ipv4不能封装的ip层信息,ipv6/ip tunnel,外层的封装,有固有的ipv4包头.

tunnel配置方法:

6to4(基于包的,以前的GRE,MPLS,IP基于链路的)

自动配置

ISATAP(主机对主机)

compatible ipv6 address

GRE tunnel

int tunnel 0

ipv6 enable

ipv6 address 3ffe:b00::3/128

tunnel source 192.168.99.1

tunnel destination 192.168.30.1

```
tunnel mode gre ipv6

ipv6 to ip的tunnel

int tu 0
ipv6 enable
ipv6 address 3ffe:b00::3/128
tunnel source 192.168.99.1
tunnel destination 192.168.30.1
tunnel mode ipv6ip
```

6to4的原理:

先定义前16bit, 然后用将公网ipv4 32bit地址转换成ipv6的后32bit, 接着的子网位任意

tunnel source 写1.1.1.1, destination不写, 写了就使静态的了. 哪怎么建tunnel?用包来出发

6to4隧道:

1. 根据tunnel source来改造我的内部的ipv6的网络, 前16bit要一样(所有需要建立动态tunnel的ipv6网络前16bit都要一样)

紧跟着16bit的32bit是以tunnel source的ipv4地址转换成十六进制后来填充.

```
int tunnel 0
tunnel source lo (1.1.1.1)
ipv6 unnumbed e0 (2002:101:101::1/64)
```

2. 建立一条静态路由

```
ipv6 route 相同的16bit/16走tunnel 0
```

3. 发包

访问2002:202:202::202, 首先, 看路由表, 看到静态路由, 走tunnel口, tunnel还没有建立起来, 把目的地址17到48位(即32位)提取出来, 转化为ipv4地址, 然后以这个地址为tunnel destination, 建立tunnel.











## 列表

### 访问控制列表

**if#ip access-group {NO./WORD} {in/out}**

不管是permit还是deny什么,最后都要 **access 101 per ip any any**

更改路由器时间:clock set

### 时间访问控制列表:

两个步骤:

#### 1. 定义时间范围

**time-range WORD**

后面的参数:1. absolute 定义绝对开始时间和结束时间,必须跟 start/end (这两个参数必须跟一个)

**absolute start hh:mm date month year**

**end hh:mm:date month year**

没有start就立即生效,然后盗结束

如期为止

没有end就没有停止的时间,从开始

永远生效

#### 2. periodic 定义一星期内,每天什么时间生效

一个时间范围里只有有一个absolute,可以有多个periodic

#### 2. 把时间范围和访问控制列表绑定起来

**access 101 permit ip a a time-range WORD**

自反访问控制列表:允许内部主动发起包,外部响应.

动态访问列表:从哪个接口telnet进来,只能在哪个接口生效,而且先写的列表生效

**clear access-tem a host 2.2.2.2**

```
clear access-tem a host 2.2.2.2
```

## 组播

### 组播

## Unicast（单播）/Multicast（组播）

push model(推的模式)()

1、一开始的流量被泛洪到网络上的所有用户上

路由器一收到组播包就会建立（S，G），如果有（\*，G）的话，就在它下面建

要建（S，G），一定先要有（\*，G）。

（\*，G）是一个共享树的概念。（\*，G）就是在密集模式下收到PIM HELLO 包的接口

（S，G）应该由组播包维护。而（\*，G）应该由它的子项维护，即（S，G）

在密集模式下，（\*，G）只是一个辅助的作用。

注意一点：谁创建谁维护！！！！

每5s会重查先组播表，决定消息来的接口

~~~~~  
~~~~~

## 组播2

1. 组播最大的目的是为了了解决链路的流量, 使其最优化传送.
2. 组播是基于UDP的. (TCP只能用于单播, 要求两个终端需连接起来)
3. 组播路由协议的分类:
  - 密集模式协议:DVMRP, PIN-DM
  - 稀疏模式协议:PIM-SM, CBT
  - 链路状态协议:MOSPF
4. 组播的设备:
  - 完全不支持组播
  - 只支持发组播
  - 可发, 可收
5. 组播的组成员是基于接口的, 是一个接口特性(一个接口可以同时属于多个组)
6. 被保留的本地链路地址:224. 0. 0. 0--224. 0. 0. 255, 其TTL=1, 不能穿越一个网段

另一些被保留的地址:224. 0. 1. 0--224. 0. 1. 255, 可以跑在公网  
上, TTL>1

|               |                           |
|---------------|---------------------------|
| 224. 0. 1. 1  | ntp network time protocol |
| 224. 0. 1. 32 | mtrace routers            |
| 214. 0. 1. 78 | tibco multicast1          |

管理地址(即私有地址)  
239. 0. 0. 0--239. 255. 255. 255

~~~~~  
~~~~~

协议号 1 是ICMP

IGMP:

一台主机加入组的命令:

```
int e0/0
```

```
ip igmp join-group 239.255.1.1 (模拟一个接收者)
```

另一条命令:ip igmp static-group 使路由器上某个可以转发组播流量的接口一直转发流量,不管有没有组员

启用路由器转发组播:ip multicast-routing

ICMPV1 查询包的地址是224.0.0.1

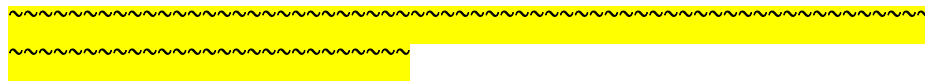
回应包的地址是要加入的组的地址。

查询路由器:

对于IGMP V1来说不根据IP地址的大小进行选举,可以通过dr-priority来指定;查询路由器需要根据第三层的协议来指定,

对于IGMP V2来说自己会选出查询路由是地址小的为查询路由器,V2的查询路由器的选举与第三层组播路由协议无关

sh ip pim interface e0 可以看到选出的DR



IGMP个人小结:

1. IGMP是多播主机与多播路由器之间的一种协议,协议号为2(ICMP协议为1);IGMP的信息是本地有效,路由器不会

将其前转,所以将TTL设为1;现有三个版本:IGMP V1, IGMP V2和IGMP V3

2. IGMP V1

V1主机的消息类型:membership report

V1路由器的消息类型:general query

**V1的抑制机制:**a) V1路由器会子网中的所有设备为发一个查询包(general query),查询包的源地址是V1路由器

一个具有转发组播流量功能的接口,而目的地址是224.0.0.1;

b) 子网上的设备当收到V1路由器的查询包后,会产生一个1s至10s随机计时器,随机计时器先超时

的设备会向路由器发送一个回应包(发回应包的目的是表明它要加入某个组). 回应包的源地址是

要加入组的接口地址,目标地址是所要加入组的组地址;

组地址;同时这个回应包会发向子网中的其他设

备,当其他设备收到这个回应包后,它会马上删除随机计时器,并且不会再V1路由器发回应包.

附注:224. 0. 0. 1是指子网上的所有系统,可以是主机,也可以是路由器的一个接口;另外组播是一个接口特性.

**V1的查询器选举**:RFC没有具体的定义V1查询器的选举过程,其次V1这个版本它不会自动的选举出查询器,而会根

据第三层协议来指定. 第三层协议会选出一个指定路由器,可以用dr-priority来进行指定. 可以

说,对V1来讲,指定路由器就是查询器.

**V1主机的退出机制**:当v1主机要退出组时,它并不想V2那样会发送一个leave group消息,而直接退出,而查询者需

要在发了三次查询包后都没有组员回应时,才认为子网没有成员了,而终止向该子网发组播流

量,即需要3分钟的时间才能完成主机的退出确定过程,这个时间是比较漫长的.

### 3. IGMP V2

V2主机的消息类型:membership report消息

vl membership report消息

leave group消息

V2路由器的消息类型:general querey

group-specific querey (特定组查询消息)

**V2的抑制机制**和v1的抑制机制是相同的.

V2查询器的选举:V2会自己选择出查询器,会选择出接口地址**最小**的作为查询器.

对v2来说,主机多了一个**leave group消息**,路由器多了一个**特定组查询消息**;

a) 当有主机(附注:这个主机是刚才路由器发查询包时,发回应包给路由器同时抑制其他主机发包的那个主机)要

退出组时,它会发一个leave group消息给路由器,leave group消息的源地址是所要退出组的接口的地址,目

标地址是224. 0. 0. 2;

此时,V2查询路由器会马上回一个group-specific querey消息(这个消息平均在2s内会连续发两个,有是会是

1. 3s,有时会是1. 03s,有时会是1. 7s),这个group-specific querey消息的源地址是查询路由器的接口地

址,目的地址是组地址;而子网上的其他成员在收到group-specific querey消息后,同样会产生一个1s至10s

的随机计时器,先超时的会跟查询路由器一个回应包.

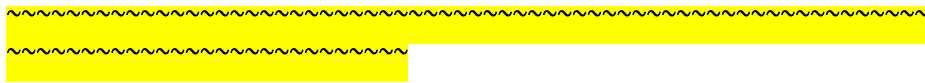
b) 当有其他主机退出组时(附注:这个主机并不是刚才发抑制消息的那个主机),这个主机不会发送一个leave

group消息,而查询路由器此时也不会发group-specific querey消息

group消息, 而查询路由器此时也不会发group-specific query消息 (可以理解为, 因为查询路由器知道有刚才响应它的主机存在, 所以当其他组员离开组时, 它仍然认为有其他组员存在, 就是刚才响应它的那个主机, 因此, 它仍然是要向这个组转发组播流量的, 因此, 就不会发group-specific query消息)

当非查询路由器在一段时间内都没有收到查询者的查询时, 那么非查询路由器会认为查询者不存在了, 而自己充当查询者, 这段时间默认是查询时间的两倍, 即120s

当查询路由器没有发查询包时, 当主机首次想加入组, 所以它会向查询路由器发一个membership report消息, 表示它要加入该组, 但只会发一次.



命令解释:

1. ip igmp query-timeout (60s--300s) 非查询器的等待查询器的时间  
ip igmp query-interval 更改查询间隔



PIN

dense-mode:use push model(一个“推”的模式, 就像所有对象都发)  
traffic flooded throughout network  
pruned back where it is unwanted  
flood & prune behavior(typically every 3 minutes)

sparse-mode:use pull model



## my plan

### 疑问回答

有无合适的模拟器？目前没有可做全面实验的模拟器。

问马老师的问题：

1. ppp multilink 最多支持几路？理论上没有限制。
2. 不同设备上可以做吗？（不可以。）

G703是什么？马老师：是E 1

一端是G 7 0 3，另一端V 3 5，会不兼容吗？马老师：通常不这样做，试一下。联合证券确认一下。确认过了，可以兼容。联合证券从7507router一总部G 7 0 3到证通公司1 5 5 M线路上分时隙下来，然后转接到v 3 5的协议转换器件—2811router。

isdn理论上比A D S L 安全，因为它是二层的直接到对端。没有上internet.

一个设备从一个接口收到frame去除他的frame头，并加上新的frame头，从另一个接口除去的过程叫做forwarding.

trunk端口的 5 种模式：on off auto desirable nonegotiate

nonegotiate 怎么理解？

EC FEC GEC端口的4种模式：on off auto desirable  
auto-desirable, desirable-desirable, on-on这 3 中组合可以做成F E C.

## 我的单词库

1)

optimal

[5CptimE1]

adj.

最佳的，最理想的

例子:the optimal route

2)

stick

[stik]

n.

棍，棒，手杖

v.

粘住，粘贴

vt.

刺，戳

例子:router in a stick

3)

desirable

[di5zaiErEbl]

adj.

值得要的，合意的，令人想要的，悦人心意的

例子:port mode : auto on off desirable

4)negotiate

4)negotiate

[ni5^EuFieit]

v.

(与某人)商议, 谈判, 磋商, ,买卖, 让渡(支票、债券等), 通过, 越过

5)

revision

[ri5viVEn]

n.

修订, 修改, 修正, 修订本

例子:VTP的revision

6)

segment

[5se^mEnt]

n.

段, 节, 片断

v.

分割

例子:网段的有三种解释.

7)

draft

[drB:ft]

n.

草稿, 草案, 草图

vt.

起草, 为...打样, 设计

v.

草拟

8 )

dedicated

[5dedIkeItId]

adj.

专注的, 献身的

举例: 专线, 比如 D D N .

也叫Leased line.

9 )

broadband

[^brR:dbAnd]

宽带

10)

narrowband

[讯] 窄带

10)

payload

净荷: 一个packet的data部分.

11)

backward compatibility

n.

向后兼容

**WOLF-2**

**NP**

**NP-ROUTE**

**NP-OSPF**

**OSPF理论及应用**

**OSPF**

OSPF三张表 (OSPF AD:110)

1 neighbor table (列出了所有和本路由器直接相连的OSPF邻居)

2 topology table (LSDB链路状态数据库)

2 topology table (LSDB链路状态数据库)

列举了所有从自己的邻居那得到的LSA, 在同一个OSPF区域中的路由器, 都有完全一致的OSPF database, 一个OSPF区域, 就对应着一个ospf database

3 Routing table (从OSPF这个路由协议, 学到的路由)

在OSPF的数据库中, 通过SPF算法, 计算得到了路由, 也称为: forwarding database

区域划分的目的:

1 提高路由效率:

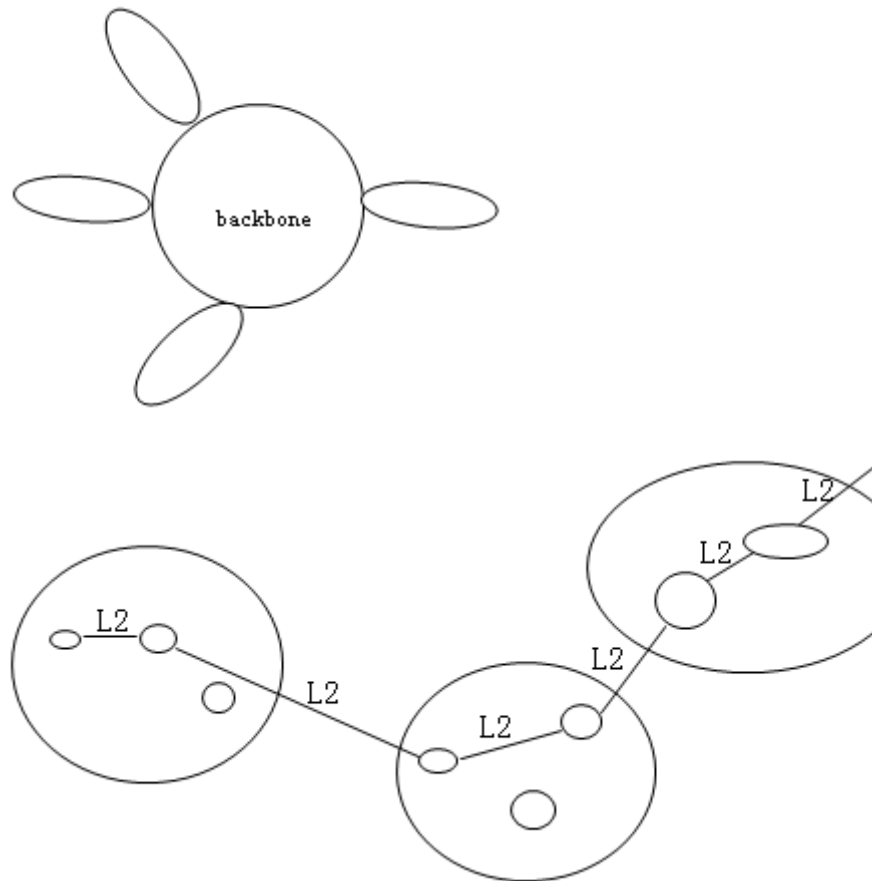
缩减部分路由器的OSPF的路由条目, 对某些特定的LSA, 可以在区域边界 (ABR) 上, 实现汇总/控制/过滤 (通过OSPF的汇总路由/默认路由/实现OSPF区域之间的全网互通)

2 提高网络稳定性:

当某个区域内的, 一条OSPF路由出现抖动时, 可以有效控制受影响的波及面。(对于大型的路由协议来说, 稳定时很重要的因素)

3 OSPF VS ISIS 的区域可扩展性的对比:

图 3 march 17



ISIS扩展性优于OSPF

注意: OSPF是以Area0为backbone的, 而ISIS以Level2的链路为backbone

因为一个成熟的OSPF区域如果要新增路由器，那么可能引起路由混乱，而ISIS可以无限制扩展，ISIS是以链路形态出现的，所以ISIS的区域可扩展性优于OSPF

### OSPF对不同的物理链路的处理分类

OSPF routing updates and topology information are only passed between **full** adjacent routers

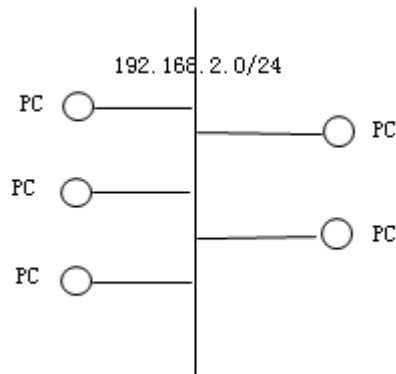
1 P2P (HDLC/PPP/Serial/Point2point sub-if) 一定要求是FULL状态, 注意的是此没有DR/BDR的选举的

图3 match 19

2 **BMA (Ethernet/TR/FDDI)** (**局域网为代表**)

broadcast multi-accwss (有DR/BDR的选举的)

**什么是MA (多路访问)? 见下图**



那么BMA呢? 允许广播通过的MA

3 **NBMA (FR/ATM/X.25)** (**广域网为代表**)

non-broadcast multi-accwss (有DR/BDR的选举的)

不允许广播通过的MA (多路访问)

### 关于MA (multi-access) 网络的DR/BDR的选举

1 根据OSPF路由器的OSPF **接口的优先级**选举, 每个接口默认的优先级都是1, 其中优先级最大的称为DR, 次大的成为BDR, 其他的都是DR-other, 如果有路由器的PRIority为0, 放弃DR/BDR的选举 (OSPFpriority:0~255)

基于接口的意思: 图3 match 21

同一个路由器有可能在不同网段做不同的角色

2 如果接口的优先级相同, 将使用router-id来决定DR/BDR的选举, 其中router-id最大的成为DR, 次大的成为BDR, 其他的都是DR-other.

### OSPF算法

- 1 每个OSPF区域, 都对应着一个独立的OSPF database (LSDB)
- 2 每个OSPF路由器, 都生成了以自己为根的, 一棵SPF树
- 3 从本路由器出发, 到特定目标网络的整体开销最小的那个路径,

成为最佳路径

4 那么这条最佳路径，就成为OSPF这个协议，提交给路由表的，到达这个目标网路的路由

最先比较最长匹配/然后AD比较/最后metric比较（选路规则）

注意

1 最长匹配是指：掩码长度越长，越有可能是最佳路径，在EIGRP和RIPV2同时运行在某网络中，且目标网络是5.5.5.0/24，那么如果EIGRP是AUTO-summary, 而RIP是no auto-summary, 那么最佳路径是经过RIP访问目标网络的，因为EIGRP中自动汇总后，5.5.5.0/24会汇总为5.5.5.0/8，又因为RIP是关闭自动汇总的，那么在RIP中会有明细路由5.5.5.0/24出现，那么根据最长匹配原则，路径优选为RIP

2 如果最长匹配相同，那么比较AD，RIP的管理距离为120，EIGRP的管理距离为90，所以优选EIGRP

3 如果最长匹配和AD都相同，那么就根据METRIC判断，假设全网运行RIP，那么根据HOP的数目选择路径，如果EIGRP，那么根据bandwidth来选择路径

LSA的传播更新规律（OSPF是LS协议，无需遵循水平分割）

1 如果本路由器从来没有收到过此LSA, 那么路由器就将其加入LSDB, 并且转发/泛洪此LSA, 同时进行SPF计算，得出到达此目标的最佳路由

2 如果本路由器，曾经收到过描述同一个网络的LSA:

2-1 如果新收到的LSA序号，与自己已有的相同，则丢弃此LSA

2-2 如果新收到的LSA序号，比自己的更新，则同1，去计算最佳路由

2-3 如果新收到的LSA序号，比自己的更旧，就将自己较新的LSA, 发送给源

OSPF数据包的5种类型

1 HELLO

2 DBD(database description)数据库描述

3 LSR (linkstatus request) 链路状态请求

4 LSU (linkstatus updata) 链路状态更新 (LSA是包含在LSU中的)

5 LSACK (linkstatus ack) 链路状态确认

OSPF的protocol id:89 (EIGRP:88, BGP: TCP的179端口, RIP: UDP的520端口)

在OSPF的HELLO包中，影响建立邻居的4个关键因素

1 HELLO/Dead interval

2 链路所在的area id

3 OSPF认证密码

4 stub(nssa)区的标示位

以上4个因素，必须一致，否则无法建立OSPF邻居

以上4个因素，必须一致，否则无法建立OSPF邻居

OSPF邻接建立过程

```
1 down
2 init      (初始化)
3 two way
4 exstart
5 exchange  (交换)
6 loading
7 full
```

OSPF数据包的发送地址

DR notifies LSU on 224.0.0.5

DR-other notifies LSU TO OSPF DR on 224.0.0.6

LAB1:OSPF的router-id

STEP1:通过router-id命令，修改ID号

R1:router ospf 110

router-id 100.0.0.1(或建立LP地址，ID和LP地址相同)

STEP2:假设没有通过router-id指定路由器的ID，那么自动会将自己的环回口地址做为路由器ID，如果有多个LP口，那么会自动会指定IP地址最大的那个LP口地址做为路由器ID

STEP3:如果没有LP口，那么会自动从当前为Active(激活)状态下的物理接口，选择IP地址最大的IP，做为自己的路由器ID，注意：这种方法是不稳定的，不建议

LAB2:通过反掩码，控制OSPF（你会很奇怪为啥这两个实验都没图聂？答案很简单，朕太懒了！）

R1:network 10.0.0.5 0.0.0.0 area 0 （此命令只允许这一个接口运行OSPF

R2:network 10.0.0.0 0.255.255.255 （此命令宣告掩码长度为24位的网络10.0.0.0运行OSPF）

反掩码匹配原则结论：

0：准确匹配

1：忽略不计

在OSPF和EIGRP中，network宣告路由命令中带的反掩码，只表示路由协议的接口范围，而不表示网络长度

提示：默认情况下，OSPF的LP口都以LP做为“网络类型（OSPF的运行模式）”，所以路由长度32的主机路由：3.0.0.0/32（本身的是3.0.0.0/16），如果要准确反映其长度，需要以下操作：

首先查看：

R1:sh ip ospf int lo 3

其查看结果为：network type: loopback

那么要准确反映其长度，需以下操作：

R1:in lo 3

```
ip ospf network point-to-point
```

再查看结果为: 3.0.0.0/16

OSPF四张表:

```
sh ip ospf database
```

```
sh ip ospf interface
```

```
sh ip ospf neighbor
```

```
sh ip route ospf
```

注意在指定DR后, 要清进程, 才能正确显示

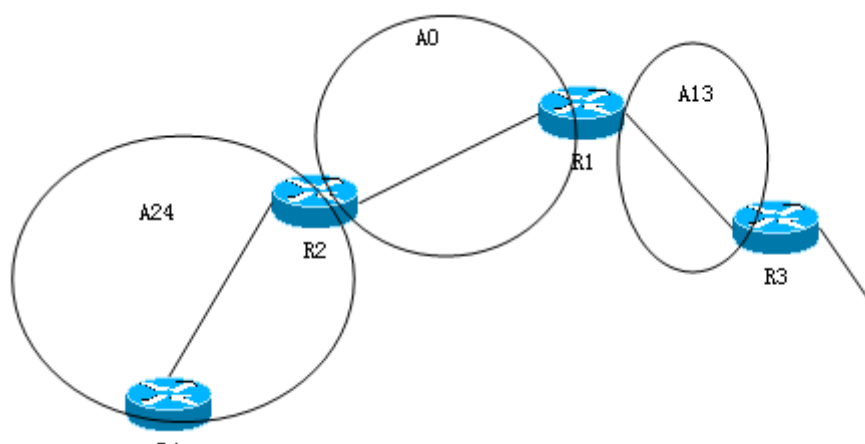
```
R1:clear ip ospf process
```

```
advanced
```

## 高级OSPF控制

### advanced ospf controlling(高级OSPF控制)

OSPF的metric计算:



公式:  $cost = 100,000,000 / BW$  (带宽为路由入口, 其单位为bps)

以上计算是一个接口的cost值, 如果是N个接口的cost值, 那么就是N个cost之和

注意OSPF的cost计算主要是查看BW的值, 接口BW通常有以下几种:

```
R1:show interface s1 (1544K) 1.544M
```

```
show interface e0 (10000K) 10M
```

```
show interface fast-ethernet 0 (100,000K) 100M
```

```
show interface lo 1 (8000,000K) 8G
```

修改OSPF metric的3种方法:

方法1: 直接修改接口的cost值

```
R2:in s1
```

```
ip ospf cost 100
```

sh ip ospf interface s0 (用此命令查看此运行了OSPF的接口的cost值)

方法2: 修改接口的BW

方法2: 修改接口的BW

```
R1:in s0
```

```
    bandwidth 2048
```

方法3: 修改参考带宽 (修改cost公式中的分子大小)

工程需求, 考虑到网络发展, 要求将来在10Gbps链路上, 其OSPF的COST值为10

那么用逆运算推出COST公式中的分子大小:  $X/10G(10,000,000,000) = 10$ , 那么 $X=100,000,000,000=100G$

R1/2/3/4/:router ospf 110 注意这里必须所有路由器上都修改参考带宽(基于进程修改)

```
    auto-cost reference-bandwidth 100000 (注意这里的单位是M, 所以数值是100000M=100G)
```

特别注意:

1 路由入口的定义

2 同步串行链路的同步时钟频率, 不影响OSPF的COST计算, 它只影响链路的真实物理速率(带宽), 这不同与接口中的BW参数, BW参数只影响OSPF的选路(影响COST值的计算), 不影响真实的物理速率(带宽)

3 对于以太网, 其速率等于其接口带宽(我日, 折腾!)

如果在2600上支持修改带宽的设备上, 那么用以下修改

```
in fast-ethernet 0/1
```

```
speed 10(100)
```

## OSPF的汇总

inter-area(IA), 域间汇总

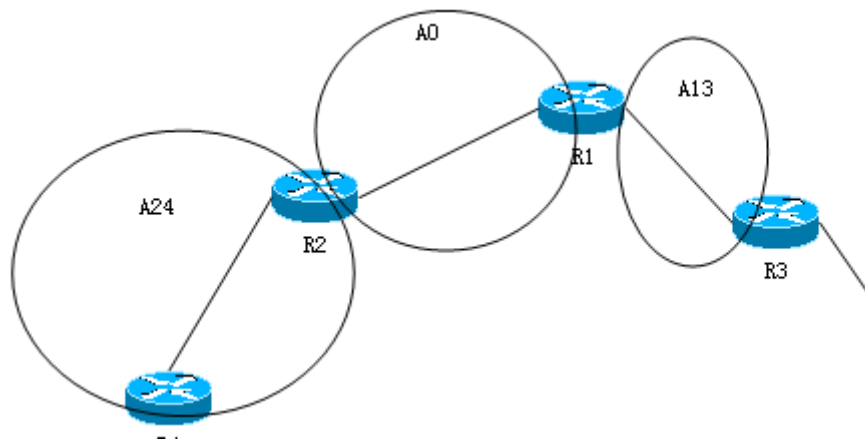
1 OSPF的域间汇总, 发生在连接不同OSPF区域的ABR上 (ABR: 互联了两个, 或两个以上的OSPF区域的路由器)

OSPF external type1(2):域外汇总 0 E1(E2)

2 域外汇总, 发生在OSPF与别的路由协议相连的ASBR上 (ASBR: 在OSPF区域的边界上, 互联了OSPF和其他路由协议的路由器)

## LAB1:OSPF的域间汇总

因为, 无论在何种路由协议的汇总中, 生成的汇总路由包含范围过大, 则很可能形成路由黑洞, 反之, 生成的汇总路由包含的范围过小, 则很可能丢失部分明细, 所以, 默认情况下, 没有指定汇总长度的时候, 应该进行最精确的汇总



STEP1:在area24中R4路由器上，模拟4条/26的路由

R4:LP1:175.14.14.65 255.255.255.192  
 LP2:175.14.14.129 255.255.255.192  
 LP3:174.14.14.193 255.255.255.192  
 LP4:174.14.14.118 255.255.255.192

注意要得到精确路由必须以下操作：

```
in lo 1/2/3/4
ip ospf network point-to-point
```

STEP2:那么将明细路由宣告到OSPF进程中

R4: network 175.14.14.0 0.0.0.255 area 24 (注意这里虽然宣告的是/24位的，但查看路由表时是明细路由/26的条目)

R1/2/3:sh ip route ospf

其结果都是/26的明细路由

结论：链路状态协议(LS协议)，没有类似于DV协议那样的“自动汇总”的特征

STEP3:全网的OSPF上，都可以看到明细路由

STEP4:在R2上做汇总（在明细路由区域中的ABR上做汇总）

R2:router ospf 110

area 24 range 175.14.14.0 255.255.255.0 (新增的汇总路由及掩码/24)

sh ip route 会出现汇总路由信息

0 IA 175.14.14.0 (0-OSPF, IA-OSPF INTER AREA)

STEP5:R2 (ABR)上观察有明细路由和汇总路由，但R1和R3上只有汇总路由，R4上是没有汇总路由的，R4在这里代表area24中所有的路由器，也就是说在area24上，除了ABR路由器(R2)，其他路由器上只有明细路由，没有汇总路由

## LAB2:OSPF的域外汇总

图与LAB1相同（灭哈哈。。。又不用画图列）

STEP1:R3/5运行EIGRP，且no auto-summary

STEP2:在R5上建立LP口

LP1:178.15.15.161/28

LP2:178.15.15.177/28

LP3:178.15.15.129/28

STEP3:R3上观察路由 (EIGRP)

D 178.15.15.176/28

D 178.15.15.160/28

D 178.15.15.128/28

STEP4:在R3 (ASBR)上,将域外路由,重分布到OSPF中

R3:router ospf 110

redistribute eigrp 90 subnets

STEP5:在R1/2/4上观察路由

OSPF域内路由

O E2 178.15.15.176/28

O E2 178.15.15.160/28

O E2 178.15.15.128/28

在R3上观察有以上路由,但没O E2标示,因为AD小 (EIGRP: 90, OSPF:110), 优选EIGRP路由, 故在R3上看到这些路由不认为是域外路由

STEP6:在R3 (ASBR)上,也只有在ASBR上,将域外路由,做OSPF的域外汇总

R3:router ospf 110

summary-address 178.15.15.128 (/26) 255.255.255.192 (回

去多琢磨这些汇总的问题,为什么汇总成了/26)

R1/2/4上查看路由

汇总成功

提示:只查看某种路由的命令:

sh ip route | include O E2 (关键字: O E2)

用此命令是搜索关键字包含 O E2 的路由信息

E1-OSPF external type 1

在传播路径上, E1的COST会叠加,随着路径的远近叠加开销,其COST值会改变

E2-OSPF external type 2

在路由传播路径上, OSPF的COST不变,默认是20

把E2改为E1的方法:

R3:router ospf 110

redistribute eigrp 90 subnets metric-type 1

## OSPF虚链路

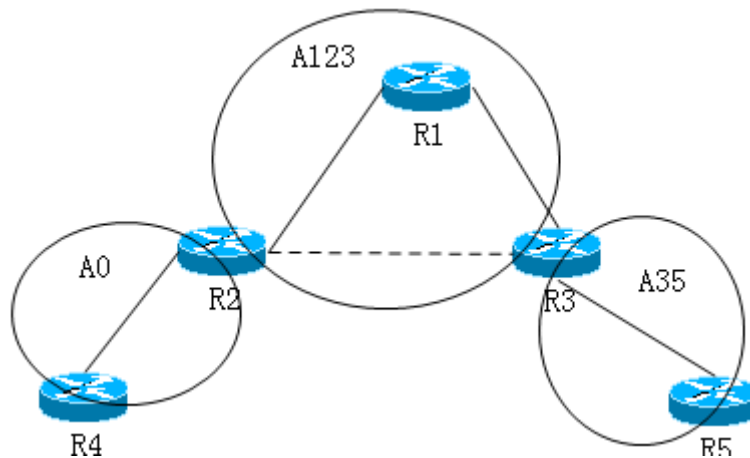
1 在大型网络工程中,由于历史原因,导致网络不佳,迫于网络扩容的原因,被迫新建OSPF区域,使用OSPF虚链路

默认情况下,所有区域必须连接在area0的

2 出于网络冗余考虑,选择合适的ABR做OSPF-VL,避免因个别物理链路的中断,而导致整个OSPF区域的全网中断

- 2 出于网络冗余考虑，选择合适的ABR做OSPF-VL，避免因个别物理链路的中断，而导致整个OSPF区域的全网中断
- 3 出于网络冗余考虑，选择合适的ABR做OSPF-VL，避免因个别物理链路的中断，而导致OSPF的area0区域出现双backbone

OSPF Virtual links (虚链路)



由于A35没有与A0直接相连，所以要在A35和A0之间做虚链路，即在A0和A35的ABR上做虚链路，也就是说在A123上的两个ABR上做虚链路

STEP1:全网运行OSPF，且router-id:195.100.0.\*

R2/3在A0和A非连续区域A35的ABR上做（即在VL穿越的区域内A123的ABR上，互相指向对方）

```
R2: router ospf 110
    area 123 virtual-link 195.100.0.3
```

```
R3: router ospf 110
    area 123 virtual-link 195.100.0.2
```

用此命令观察虚链路状态：

```
R2:ip ospf virtual-link
```

显示以下信息是正常

```
virtual link ospf-vl0 to router 195.100.0.3 to up
adjacency state full
```

## OSPF的多区域及LSA类型

OSPF的多区域

1 internal router（内部路由器）

所有接口/网段都在同一OSPF区域中

2 backbone router（骨干路由器）

至少有一个接口，运行在area0的路由器

3 ABR（区域边界路由器）

承担两个或多个OSPF区域的互联的路由器

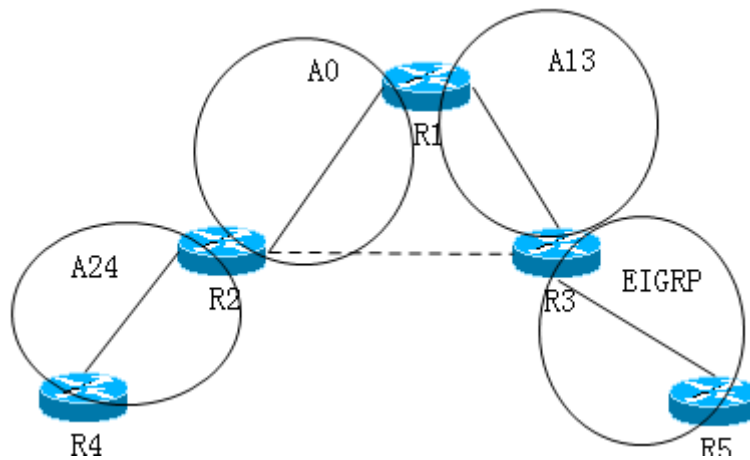
4 ASBR（自治系统边界路由器）

至少有一个接口不是运行在OSPF，它可将非OSPF路由传入OSPF的区域

至少有一个接口不是运行在OSPF，它可将非OSPF路由传入OSPF的区域中

## LSA类型

首先看图：



R1/2/3/4:router-id 100.0.0.\*

注意LP口也要宣告进OSPF，测试需要，且R3/5运行EIGRP

注意LP口配置后，在接口模式下：

```
R1:in lo 1
```

```
ip ospf network point-to-point (此举是在配置后可看到LP的  
明细长度)
```

查看域内0路由，和域间0 IA路由

```
R1:sh ip ospf database
```

那么进入正题：LSA类型！

### LSA1 (LSA1型)

```
sh ip ospf database router
```

查看和本路由器相邻的直连链路的路由

originating router

只在本区域内传递，而不穿越ABR

### LSA2:

```
sh ip ospf database network
```

本区域内的BMA/NBMA的网络路由，由DR所生成的，只在本区域内传递，而不穿越ABR

### LSA3:

```
sh ip ospf database summary
```

指的是OSPF的域间路由 (0 IA)

原LSA1所描述的路由，会由原区域的ABR将其转换为LSA3，LSA3可以传播到整个OSPF的所有区域（特殊区域例外）

```
R3:router ospf 110
```

```
redistribute eigrp 90 subnets
```

将EIGRP重分布到OSPF

#### LSA4

```
sh ip ospf database ASBR-summary
```

ASBR的路由器ID，是由ASBR所在的那个区域的ABR产生并通告的，LSA4可以传播到整个OSPF的所有区域（特殊区域例外），LSA4协助LSA5共同工作，让访问AS外的网络路由器，寻找到ASBR所在的位置，LSA4每穿越一个区域，其advertising router(通告路由器)都会改变（是下一区域的ABR）

#### LSA5:

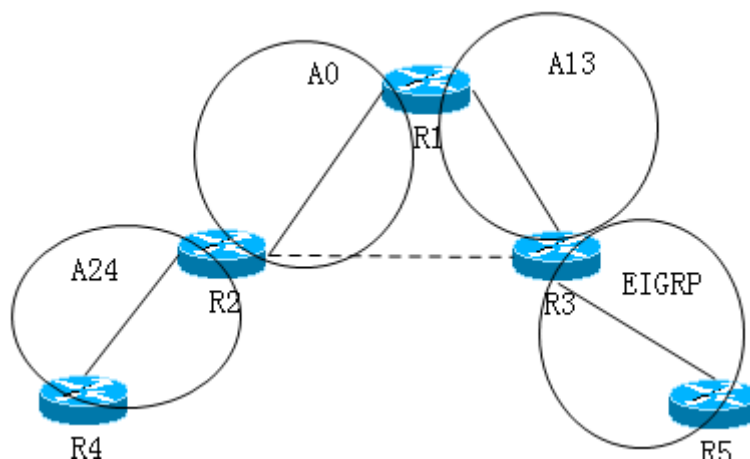
```
sh ip ospf database external
```

显示域外路由，就是OSPF区域以外的路由

由ASBR产生并通告，LSA5可以传播到整个OSPF的所有区域（特殊区域例外），LSA5需要LSA4协同工作，让访问的网络路由器，寻找到ASBR的所在位置

The advertising ASBR router id is unchanged throughout the as

OSPF的特殊区域



第一种级别：stub区

假设A24为stub区域，stub区域的路由器，只有域内/域间的路由表，没有域外路由表（上图中的EIGRP路由），阻止了LSA5/LSA4进入区域，减少了A24中的路由条目，去掉域外的，保留域内和域间的操作方法：

LAB1:在area24中的所有路由器上

```
R4:router ospf 110
```

```
area 24 stub
```

```
R2:router ospf 110
```

```
area 24 stub
```

注意必须在stub区的所有路由器上都需要做，否则会DOWN掉

在R4上 sh ip route 观察，多了一条默认路由

注意在area24中所有路由器上，ABR（R2）除外，都有以下路由信息：

```
0* IA 0.0.0.0/0 VIA 24.0.0.2
```

0\* IA 0.0.0.0/0 VIA 24.0.0.2

第二种级别: total stub 区

在total stub区的路由器上, 只有区域内的路由表, 没有域间和域外的路由表, 阻止了LSA5/4/3进入total stub区

操作方法: 只在STUB区的ABR上修改, STUB区内的路由器, 完全与LAB1相同

LAB2: 在ABR上 (R2) 做 (前提是已经定义了STUB区)

```
R2:router ospf 110
```

```
area 24 stub no-summary
```

ABR所产生的默认路由, 与标准的STUB情况完全一致

提示: 无论是stub/total stub区, 都不会有域外的路由表

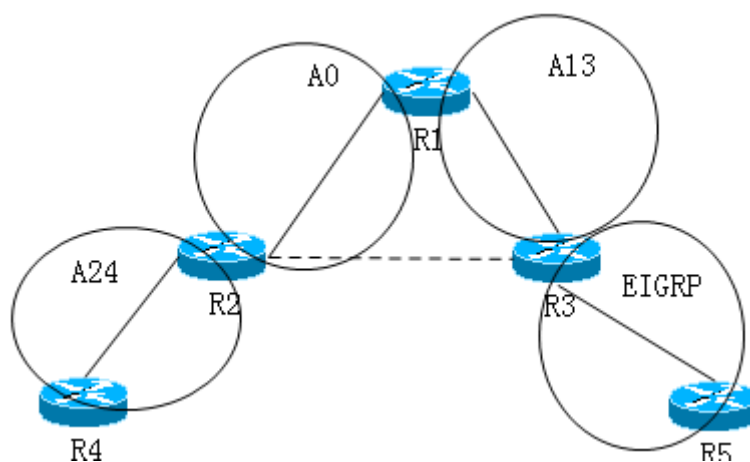
NSSA区域 (no-so-stubby area) 不完整的stub区域

NSSA=stub区域+能够引入外部路由的功能

第一种级别:

NSSA区域的路由器, 只有域内/域间的路由, 没有其他区域的ASBR所引入的域外路由, 但本NSSA区域却可以引入域外路由 (与本区域直接相连的其他协议) (标准的STUB和NSSA的区别)

例如下图:



假设A13为NSSA区域, R2上建立LP口, 模拟域外路由, LP运行的是EIGRP, 那么R2就成为ASBR

那么, 其结论:

R3是A13 (NSSA) 区域中的ASBR, 它只能引入与本区域 (R3/5) 相连的域外路由, 但不能引入R2上LP的路由协议

LAB3: R2上模拟外部路由的引入LP 20.0.0.0/24

STEP1:

```
R2:loopback 20
```

```
ip ad 20.0.0.2 255.255.0.0
```

```
redistribute connected subnets
```

```
R1:sh ip ospf datase
```

```
sh ip ospf
```

R3:sh ip route 也可以看到0 E2 20.0.0.0（注意此时A13区域还没有定义为NSSA区域）

如果A13不要其他的ASBR引入的外部路由进入路由表，但自己的区域可以引入外部路由表，那么只需在A13的所有路由器上定义NSSA区即可

STEP2:

```
R3:router ospf 110
```

```
area 13 nssa
```

```
R1:router ospf 110
```

```
area 13 nssa
```

注意NSSA区域中的所有路由器都必须定义

观察A13内的路由变化

STEP3:

```
sh ip route ospf database
```

0 E2 20.0.0.0 消失了（其他AS的所连接的域外路由不能进入）

那么，A13没有了其他的ASBR所引入的外部路由，但A13完成了对本NSSA区域的ASBR，所有连接的EIGRP路由的引入（150.0.0.0/24）

并且，A13仍然拥有域内/域间路由

```
R1:sh ip route | include N2
```

```
0 N2 35.0.0.0（LSA7类路由）
```

```
0 N2 150.0.0.0（LSA7类路由）
```

STEP4:

标准NSSA的ASBR不会自动为NSSA区域内的路由器生成路由，所以R3

ping 不同R2，解决此问题需手工加上

```
R1:area 13 nssa default-information-originate（生成默认路由的参数）
```

以上操作在NSSA的A13上做

在R1上观察

```
0 * N2 0.0.0.0/0（LSA7类）
```

第二种级别:

total NSSA区域，只有域内路由，没有域间和其他的区域的ASBR引入的域外的路由

操作方法:

在NSSA的ABR上（R1）

```
R1:router ospf 110
```

```
area 13 nssa no-summary
```

定义后会自动向NSSA区域，生成默认路由

```
D* IA 0.0.0.0/0（LSA3类）
```

LAB4: 假设R1与ISP相连，让其他路由器都通过R1访问ISP

STEP1: 在R1上建立LP(200.0.0.2)，以模拟ISP

首先建立虚拟环回口LP（200.0.0.1），目的是让骗取其他路由器都UP（什么意思？）

都UP（什么意思？）

```
R1:ip route 0.0.0.0 0.0.0.0 200.0.0.2
```

```
router ospf 110
```

```
default-information originate
```

在连接ISP的边缘路由器上，向整个OSPF区域，发生默认路由

## OSPF网络类型、认证、虚链路

### OSPF network types (OSPF的网络类型，也叫运行模式)

RFC分类：

NBMA

P2MA(点对多点)

CISCO分类：

P2P

Broadcast

P2MP

NBMA

分类：

1 P2P

PPP

HDLC

P2P Sub interface (点对点子网)

2 MA

BMA: E以太网

NBMA: 帧中继 (FR)

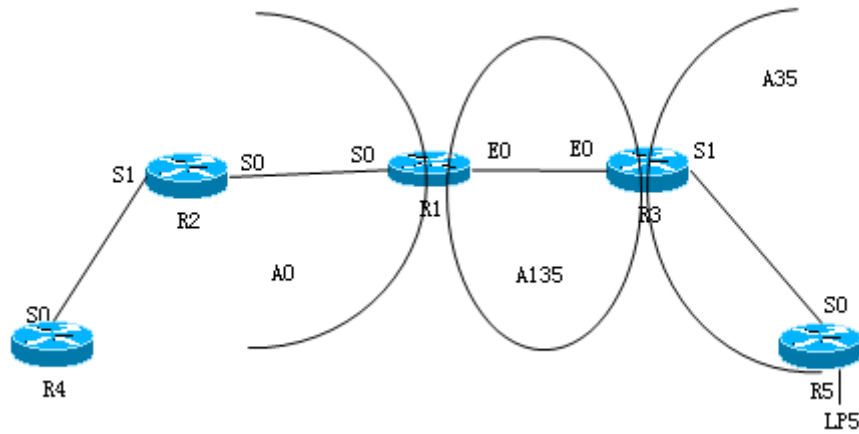
BMA模式为broadcast mode

3LP接口

Loopback mode, 进入OSPF时，只能是/32

笔记未完待续。。。

LAB0:在OSPF环境中建立虚链路，以适应工程需要



需求：按图配置后，R4不能正常访问R5，因为A35不是与A0相邻的区域

解决办法：在A0的ABR和A35的ABR上做虚链路（即在R1和R3之间做虚链路）

R3:area 135 virtual-link 100.0.0.1(100.0.0.1为R1的router-ID)

R1:area 135 virtual-link 100.0.0.3

此时观察路由，R4和R5都有对方的路由，可以正常访问到对方

## OSPF的认证

按照参与认证的成员进行分类：

- 1 链路认证：只能保证两个设备之间的链路是安全的，两个路由器之间的物理链路
- 2 区域认证：能够确保整个OSPF区域的所有路由器的安全性
- 3 虚链路认证：保证跨越多个路由器的虚链路的连接安全性

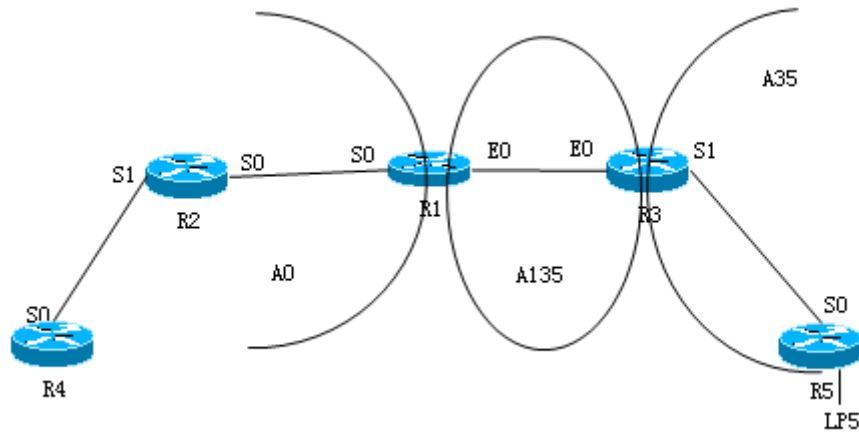
按照加密方式分类：

- 1 明文认证
- 2 密文认证

步骤：

- 1 在接口配置password
- 2 在进程（区域认证）启用认证
- 3 在接口（链路认证）启用认证

LAB1:1链路明文认证(无图也可以，好懒)



特别注意：在做认证前最好先SHUTDOWN接口，做好后再NO SHUTDOWN

STEP1: 定义认证密码

```

R4:in s0
    ip ospf authentication-key cisco1
R2:in s1
    ip ospf authentication-key cisco1

```

STEP2: 启用明文认证

```

R4:in s0
    ip ospf authentication
R2:in s1
    ip ospf authentication

```

LAB2: 链路密文认证

```

R1:in s0
    ip ospf message-digest-key 1 md5 cisco2
    ip ospf authentication message-digest (启用密文认证)
R2:in s0
    ip ospf message-digest-key 1 md5 cisco2
    ip ospf authentication message-digest

```

LAB3: 区域的明文认证

需求：凡是出在A135的路由器，都参与认证

```

R1:in e0
    ip ospf authentication-key cisco3
    ospf 110
    area 135 authentication
R3:in e0
    ip ospf authentication-key cisco3
    ospf 110
    area 135 authentication

```

LAB4: 区域的密文认证

需求：A35的所有路由器都参与密文认证

```

R3:in s1
    ip ospf message-digest-key 3 md5 cisco4
    ospf 110

```

```
ip ospf message-digest-key 3 md5 cisco4
ospf 110
area 35 authentication message-digest
R5:in s0
ip ospf message-digest-key 3 md5 cisco4
ospf 110
area 35 authentication message-digest
```

LAB5:虚链路明文认证（注意：此认证不是基于接口的）

R1和R3的虚链路（LAB0中的R1和R3之间的虚链路）

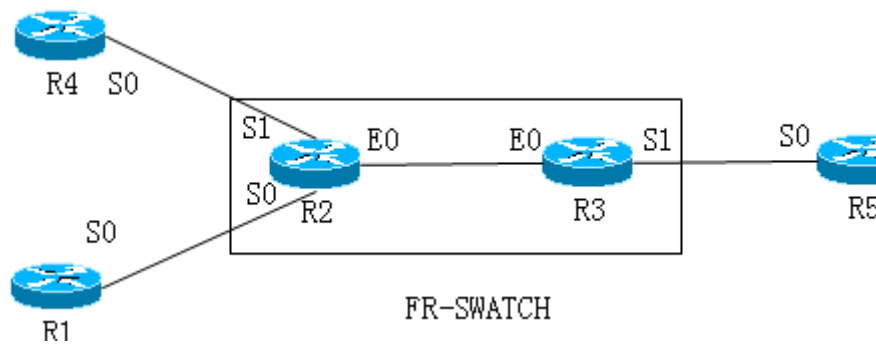
```
R3:ospf 110
area 135 virtual-link 100.0.0.1 authentication-key
cisco5
area 135 virtual-link 100.0.0.1 authentication
R1:ospf 110
area 135 virtual-link 100.0.0.3 authentication-key
cisco5
area 135 virtual-link 100.0.0.3 authentication
```

LAB6:虚链路的密文认证

```
R1:ospf 110
area 135 virtual-link 100.0.0.3 message-digest-key 4 md5
cisco6
area 135 virtual-link 100.0.0.3 authentication message-
digest
R3:ospf 110
area 135 virtual-link 100.0.0.1 message-digest-key 4 md5
cisco6
area 135 virtual-link 100.0.0.1 authentication message-
digest
```

## OSPF网络类型及帧中继

LAB1:帧中继原理及基本配置



STEP1:R2/3模拟帧中继交换机

R2:host FR-SW2

R2:host FR-SW3

STEP2:由于是路由器模拟的帧中继交换机，所以要关闭路由功能以模拟实验环境，并且启用帧中继交换功能）

FR-SW2/SW3:no ip routing

fram-relay switching

STEP3:注意R2/3要定义时钟速率，因为R2/3是DCE接口

STEP4:帧中继交换机封装帧中继，并且定义本地管理接口类型和端口类型

SW2/3:in s0/1

encapsulation fram-relay

fram-relay lmi-type cisco

fram-relay intfth-type dce

STPE5:配置帧中继的路由

此举保证了从FR-SW2的s0的104PVC进入的数据，将从FR-SW2的s1的401出去

在接口s0模式下:

FR-SW2:fram-relay route 104 in s1 401

双向都要做才可保证通达

在接口s1模式下:

FR-SW2:fram-relay route 401 in s0 104

STEP6:打开接口，注意，在做PVC前，最好先把接口DOWN掉，做完后再UP

FR-SW2:in s0/1

no sh

STEP7:在R1/4上的，与帧中继相连的接口，封装帧中继

R1:in s0

encapsulation fram-relay

R4:in s0

encapsulation fram-relay

STEP8:用户端的测试:

R4:sh in s0

sh fram-relay pvc (查看PVC状态)

sh fram-relay map (查看映射)

STEP9:开始构建FR-SW的第三个端口

由于实验环境限制，没有现成的设备，所以要在SW2/3虚拟一条tunnel管道，操作如下：

在FR-SW2/3上的E0端口操作：

```
FR-SW2:ip ad 23.0.0.2 255.0.0.0
```

```
FR-SW3:ip ad 23.0.0.3 255.0.0.0
```

```
SW2:in tunnel 2
```

```
    tunnel source 23.0.0.2
```

```
    tunnel destination 23.0.0.3
```

```
SW3:in tunnel 3
```

```
    tunnel source 23.0.0.3
```

```
    tunnel destination 23.0.0.2
```

STEP10:配置SW3的s1口，封装帧中继，并且设置本地管理接口类型和intf类型（同STEP4）

STEP11:配置R4/5之间的PVC（在tunnel中，以1000对接）

```
FR-SW2:in s1
```

```
    fram-relay route 405 in tunnel 2 1000
```

```
FR-SW3:in s1
```

```
    fram-relay route 504 in tunnel 3 1000
```

STEP12:配置R1/5之间的PVC（在tunnle中，以1001对接）

```
FR-SW2:in s0
```

```
    fram-relay route 105 in tunnel 2 1001
```

```
FR-SW3:in s1
```

```
    fram-relay route 501 in tunnel 3 1001
```

## NP-路由控制

### 路由重分布

Redistributing/重分布、重分发

定义：使A路由协议中的路由，以外部路由的形式，进入B路由协议

一：default-metric/默认度量值（种子度）

A协议中的路由，进入B路由协议后，在B协议中表现出来的，默认的metric值

二：default-metric在不同路由协议中的默认取值：

如果有外部路由（不管此路由源在哪里），进入以下的路由协议，

默认在这些协议中，表现的metric是：

1 DV协议（RIP/IGRP/EIGRP）无穷大（许多重分布不成功，都与此有

有关，分布后是不可达的，所以不成功)

2 OSPF 默认是20

3 IS-IS 默认是0

4 BGP 默认是IGP的原始的METRIC，此值通常会被人为的修改SET

三：两种些协议做重分布时，特别注意路由的控制/过滤，避免环路的出现，一般不建议使用双向重分布

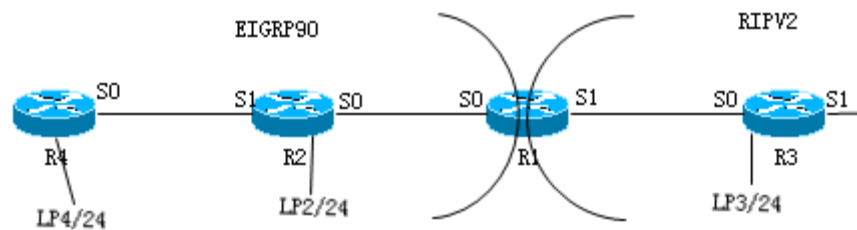
为了避免环路，通常有以下三种解决办法：

1 在一个方向上做重分布，另一个方向做静态默认路由

2 在一个方向上做重分布，反方向做带过滤的重分布

3 在一个方向上做重分布，反方向做带修改AD的重分布

LAB1:重分布/RIP和EIGRP (图8 August 6)



R1/3/5运行RIPV2 (no auto-summary), LP3和LP5也运行RIPV2

R1/2/4运行EIGRP (no auto-summary), LP2和LP4也运行EIGRP

STEP1:将EIGRP重分布到RIP中

在两个协议的边缘路由器上，做重分布

R1:router rip

redistribute eigrp 90

R3:sh ip route

看不到重分布后的路由，因为：对于DV协议来说，若不指定metric，默认就是无穷大，所以：路由没能成功进入RIP

STEP3:在重分布中，要携带metric参数，才能解决STEP1的问题

R1:redistribute eigrp 90 metric 2

R1:sh ip rip database

debug ip rip

显示都是2HOP，R3是2HOP，但R5是3HOP

STEP4:将RIP重分布到EIGRP中（如果没有SETP4，则R5ping通

4.4.4.4，因为没有回包路由）

R1:router eigrp 90

redistribute rip

此时查看路由，结果是重分布失败的，因为在DV协议中，metric默认是无穷大的，所以R2/4都没有RIP

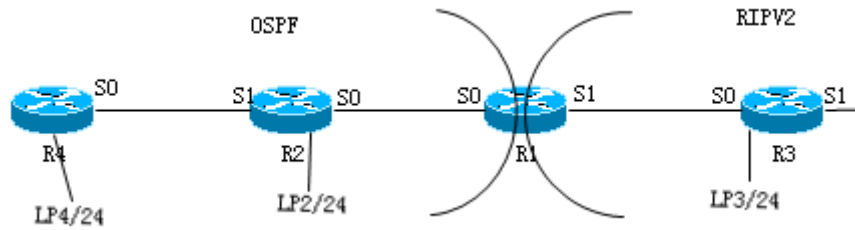
STEP5:增加metric才能解决问题

R1:router eigrp 90

redistribute rip metric 1544 2000 255 1 1500

DUAL算法：2681856

## LAB2:RIP重分布到OSPF中



STEP1:

```
R1:ospf 110
    redistribute rip
```

出现以下提示:

only classful network will be redistribute ! (仅使主类网络被重分布)

所以查看路由表时, 没发现有5.5.5.5/24被重分布到OSPF中

R3/5, 只能让主类的路由分布到OSPF中: OE2 10.0.0.0/8

STEP2: 为了让全部子网路由都能分布到OSPF中, 那么

```
R1:ospf 110
    redistribute rip subnets
```

此时, 5.0.0.0/24就显示在路由表中了

STEP3: 进入OSPF后外部路由的类型1/2控制

默认都是2型, 其OSPF的COST值, 不会发生变化: OE2 5.0.0.0 (120/20), COST值都是20, 且不会累加

STEP3: 将2型改为1型

```
R1:ospf 110
    redistribute rip subnets metric-type 1
```

```
R2:sh ip route
```

开销变化为: 20+64 (type1)

```
R2:OE1 4.4.4.0 [110/84]
```

```
R4:OE1 4.4.4.0 [110/148]
```

结论: OSPF的E1型, 会随着路径的远近, 其metric会累加

STEP4: 进入OSPF后改metric值的更改 (默认是2OE2型)

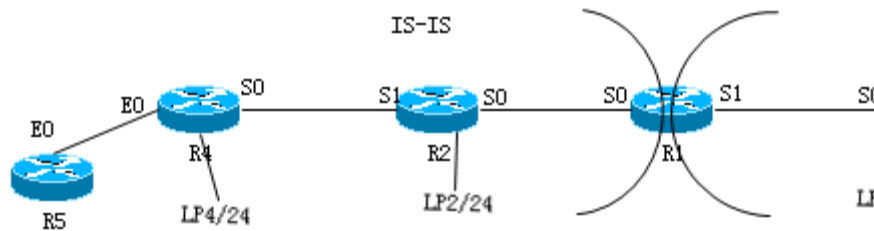
```
R1:ospf 110
    redistribute rip subnets metric-type 1 metric 50
```

改为E2后, metric不是20了, 而是50

```
OE2 [110/50]
```

都为50

LAB3: 重分布IS-IS (此图画错, 边缘路由器应该是R2, 太懒不想改)



STEP1:按图配置

R1/2/3:RIP V2

R5 (IS-IS):net 49.0245.0000.0000.0005.00

R4 net 49.0245.0000.0000.0004.00

R2 net 49.0245.0000.0000.0002.00

STEP2:激活ISIS

R2:in s1

ip router isis

R4:in e0

ip router isis

R5:in e0

ip router isis

sh clns nei 查看ISIS邻居

STEP3:重分布路由:

R2:router isis

redistribute rip

router rip

redistribute isis

结果在R3上看不到24.0.0.0的路由（注意，只要是ISIS重分布到其他路由协议里，那么其直连路由，默认是不参与重分布的）R3能PING通R5，但PING不通R4

解决办法：将此直链路由再重新分布一次

STEP4:

R2:router rip

redistribute connected

再R3上再查看，有了24.0.0.0这条路由了

有时需要用到重分布静态路由这个命令:

R2:router rip

redistribute static

## 路由过滤

Route Filters(路由过滤)

在路由控制过滤中，常见工具：

1 访问列表ACL，access-list

本身的设计意图不是为了进行路由控制，而是进行数据包的过滤，在路由控制中，如果使用这种方法，比较落后，实际使用中，并不推荐，除非是需要基于反掩码的路由控制，或者控制奇偶路由控制的时候

2 前缀列表，prefix-list

本身的设计意图是为了进行路由控制，路由控制的专业工具，推荐

3 route-map，高级路由操纵工具

可以实现多种功能，其中可用于路由控制

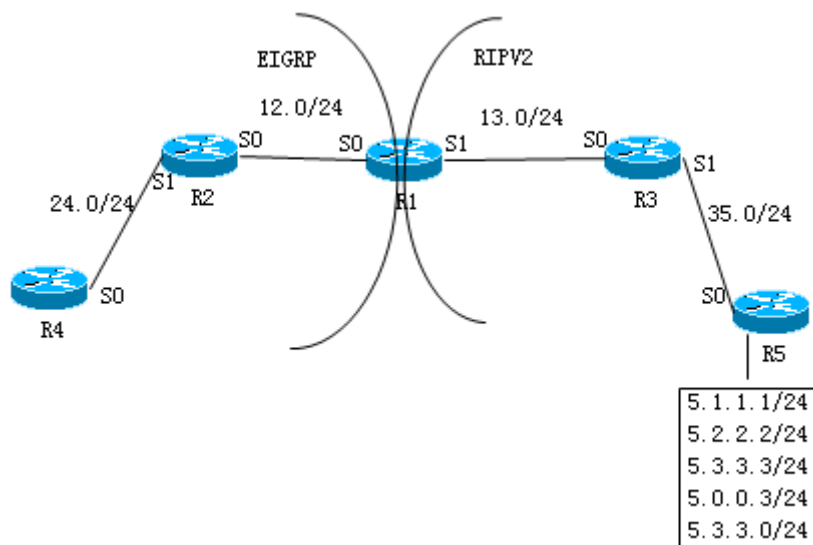
路由过滤的基本方法：

通过distribute-list/分布列表，实现路由过滤

1 在DV协议中，实现基于接口的路由过滤（接口级别控制）

2 在路由协议中的重分布时，实现路由重分布时的路由过滤

LAB1:在DV中，使用ACL，实现基于接口的路由过滤（分布列表+ACL）



需求：通过ACL，定义所需要过滤的路由，5.2.2.0，5.3.3.0可以通过R3，其他不要

如图配置：

STEP1: 配置IP，宣告路由

```
R3: access-list 1 deny 5.1.1.0
      access-list 1 deny 5.0.0.0
      access-list 1 permit any
```

STEP2: 基于接口的过滤（在R3上做）

```
R3: router rip
      distribute-list 1 out s0
```

R1: clear ip route \* 加速收敛

以上操作不影响R3本身，因为时out方向，R3路由条目不变，单如果时in方向，那么R3本身也会影响到

LAB2: 使用ACL的反掩码做基于反掩码的路由控制

需求：只允许5.\*.0.0的路由进入R3，其中\*是指奇数

```
R3: access-list 1 permit 5.1.0.0 0.254.255.0 (如果时偶数路由控制，那么 5.0.0.0 0.254.255.0)
```

在上述例子中，反掩码不能用于控制路由长度，只能用于控制每一个bit的0或1

LAB3: 先清掉以前的列表控制，用前缀列表控制路由（先定义prefix-list）

```
R3: ip prefix-list PRE-1 permit 5.1.0.0/24
      router rip
      distribute-list prefix PRE-1 out s0
```

LAB4: 协议之间的过滤（deny 5.2.2.2/24）

STEP1: 定义访问控制列表

```
R1:access-list 10 deny 5.2.2.0
      access-list 10 permit any
STEP2: 在EIGRP中, 调用distribute-list 10
R1:router eigrp 90
      redistribute rip metric 1544 2000 255 1 1500
      distribute-list 10 out rip
LAB5:需求同上, 但用前缀列表 (prefix-list)控制
STEP1:通过prefix-list定义需要控制的路由
需求: 让10.0.0.0/24可以进入EIGRP, 但10.0.0.0/16不能
R1:ip prefix-list PR-10 seq 5 permit 10.0.0.0/24
STEP2:
R1:router eigrp 90
      distribute-list prefix-list PR-10 out rip
STEP3:deny10.0.0.0/16, 其他都可以
R1:ip prefix-list PR-10 seq 5 deny 10.0.0.0/16
      ip prefix-list PR-10 seq 10 permit 0.0.0.0/0 le 32(允许
其他全部的意思)
```

ROUTE MAP (高级路由控制工具)

特征:

- 1 由一组statements/声明, 所组成每句声明中, 可能包含了match、set语句  
每条statements有个编号: 10/20/30  
路由就按照statements的编号, 按从小到大顺序执行
- 2 match commands specify criteria to be matched  
如果匹配特定某些条件, 就按照set所定义的参数, 进行操作
- 3 路由器在向下检查匹配条件时, 不断地与每个statements中的匹配条件进行比较, 如果不匹配, 则继续向下检查statements, 如果匹配, 则按照set所定义的参数, 进行操作, 然后离开
- 4 route-map逻辑关系  
同一水平线上: logical OR(或关系)  
垂直方面上: logical AND(与关系)
- 5 在route-map里, 如果没有match, 则表示match any (上面statements剩余的ANY)  
如果没有SET, 则表示SET noing

ROUTE-MAP的四步骤:

- 1 定义匹配 (prefix-list)
- 2 route-map
- 3 match ip address(匹配列表)
- 4 set需求

LAB6:需求: R4/2/1运行OSPF, R5/3/1运行RIPV2, 其中R5的LP中, 5.0.0.0/24进入OSPF后变为E1, 5.1.1.0/24不能通过R1进入OSPF, 5.2.2.0/24和5.3.3.0/24进入OSPF后metric从20变为100, 包括13.0.0.0和35.0.0.0

STEP1:用prefix-list控制

R1:ip prefix-list PR-5.0 seq 5 permit 5.0.0.0/24 (注意, 此时的permit不是允许与否, 而是匹配条件)

```
ip prefix-list PR-5.1 seq 5 permit 5.1.1.0/24
```

STEP2:创建用于重分布的路由控制的route-map

R1:route-map RP-SPF permit 10

```
match ip address prefix-list PR-5.0
```

```
set metric-type type-1
```

```
route-map RP-SPF deny 20
```

```
match ip address prefix-list PR-5.1
```

不设set, 意味set noing

```
route-map RP-SPF permit 20
```

不设match, 意味match any

```
set metric 100
```

STEP3:在重分布中调用route-map

R1:router ospf 110

```
redistribute rip route-map RP-SPF
```

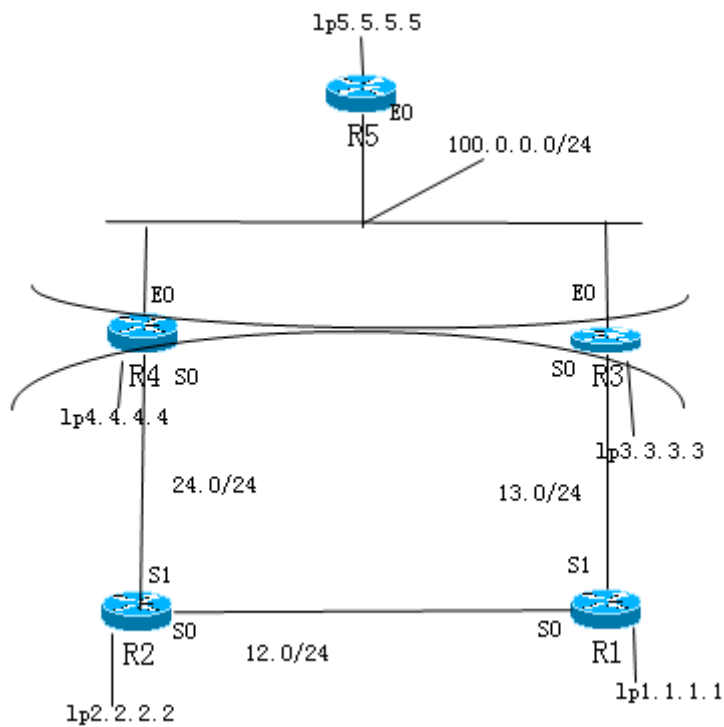
此时提示只允许主类路由进入, 那么

```
redistribute rip route-map RP-SPF subnets即可
```

## AD控制路由及策略路由

### 通过AD控制路由进程

按图014配置, 注意:R4/R3的LP都运行在RIP协议中, 注意no auto-summary



LAB1:通过修改AD，在双向/双出口的重分布种，优化路由（此实验假设R5在北京，R1/2/3/4在深圳）

STEP1:按图配置后做重分布

```
R4/3:ospf 110
```

```
    redis rip subnets
```

```
R4/3:rip
```

```
    redis ospf 110 metric 2
```

STEP2:观察路由（在路由协议的边界路由器上）次优路径

```
R3:OE2 4.4.4.0 via 100.0.0.4 [110/20]
```

此路由是不正常的，在本地路由器R3/4上传送数据，被首先传送给R5（北京），绕远了

原因：从OSPF学到的路由AD为110，比RIP的120小，故被首先传送到北京

解决办法：

STEP3:通过ACL，定义出需要更改的AD的路由，也就是说，在OSPF区域中，以OE2形式存在的原RIP路由

```
R3/4:access-list 1 permit 1.1.1.0
      access-list 1 permit 2.2.2.0
      access-list 1 permit 3.3.3.0
      access-list 1 permit 4.4.4.0
      access-list 1 permit 12.0.0.0
      access-list 1 permit 13.0.0.0
      access-list 1 permit 24.0.0.0
```

STEP4:针对源自RIP, 进入了OSPF的路由, 修改AD比RIP的AD (120) 大即可。(在OSPF里修改)

R3/4:ospf 110

distance 121 0.0.0.0 255.255.255.255 1

注意: 此命令是不管谁发给我的都修改为121, 1代表ACL1

以上修改只对本路由器生效

## 策略路由/policy-based routing (PBR)

PBR的目的, 是实现网管的人为的网络操纵意图

策略路由是基于源的, 而普通IP路由是基于目标的

PBR的应用

1 Source-based

2 Qos

3 Load sharing

PBR只用于路由器的入口的数据包, 所有PBR都应该在入口实现

PBR会大量使用route-map的3种操作逻辑:

1 如果match(匹配)的statement(声明)是permit, 就按照set语句进行策略路由

2 如果match的statement中是deny, 就不进行PBR, 而进行正常路由

3 如果连一个都match不上, 就不进行PBR, 而进行正常路由

优于路由表: (强制执行)

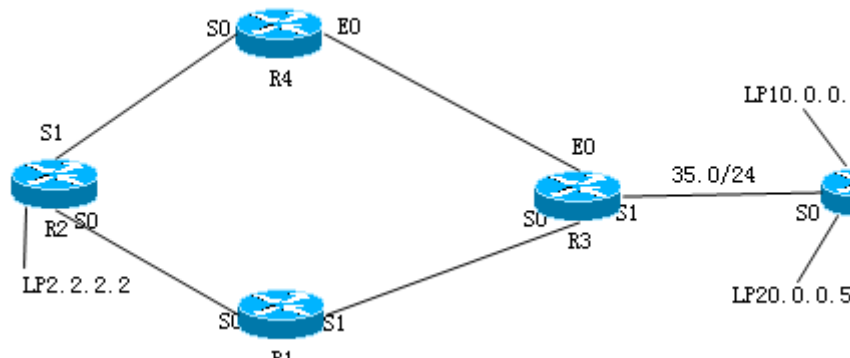
set ip next-hop 6.6.6.6

set interface s 1

次于路由表: (优先路由表, 再执行自定义) 只有当路由表里没有此路由时候, 才执行

LAB1: PBR (图015), 需求: LP10.0.0.0从R3-R4-R2访问,

LP20.0.0.0从R3-R1-R2访问



STEP1: 全局运行OSPF, 用ACL定义用户群 (用户数据包的源IP) IN 方向

R3:access-list 10 permit 10.0.0.0 0.255.255.255

access-list 20 permit 20.0.0.0 0.255.255.255

STEP2:创建route-map, 对不同用户, 进行不同策略

R3:route-map pbr permit 10

match ip address 10

```
match ip address 10
set ip next-hop 34.0.0.3
```

```
route-map pbr permit 20
match ip address 20
set interface s 0
route-map pbr permit 30 可加可不加
```

SETP3:在数据包的入口,调用route-map pbr

R3:in s1

```
ip policy route-map pbr
```

R5:ping 2.2.2.2 (扩展PING)

或者用R3:debug ip policy (跟踪策略信息)

## NP—BGP

### BGP概念及基本配置

BGP (border gateway protocol)

BGP是运行在不同的AS里面的,高级的DV协议

IGP: 包括 RIP/IGRP/EIGRP/ODR/OSPF/ISIS

IGP是通过cost/metric判断路由优劣,越小越好,主要任务:更快更好的描述路由信息,尽快地将数据包发送到目的地

BGP (V4)

运行在不同的AS之间,用于对路由的控制/策略,是对网管的人为意志的体现

BGP是通过BGP的属性/attributes,判断路径的优劣(不强调收敛),从中进行选择,BGP可以通过网管所定义的策略/policies实现数据或路由的控制和操纵

大AS: 独立的技术/管理域,通常是一个大公司或者组织或者国家,

家，这是区别与小as (EIGRP/OSPF) 的  
BGP是一种AS BY AS 的高级距离向量/DV协议  
BGP认为：每经过一个AS是一跳，而RIP认为：每经过一个路由器是一跳

#### 应该使用BGP的情况：

- 1 ISP, 当允许AS1的数据穿越AS2到达AS3, 但是不允许AS1的用户访问AS2的时候, 例如ADSL用户需要通过ISP来访问INTERNET的时候
- 2 Multihome/多宿主, 对于一个用户的AS, 如果他同时连接多个AS或者ISP的时候
- 3 PBR, 当需要对BGP路由/数据进行控制操纵的时候

#### 不应该使用BGP的情况

- 1 与ISP只有一条连接, 没有同时连到多个ISP
- 2 硬件档次不够, 内存/CPU
- 3 对BGP操纵理解有限
- 4 带宽不足

#### BGP特征：

- 1 BGP是高级DV协议
- 2 工作在TCP/IP协议栈的4层：TCP的179端口 （图BGP-004）
- 3 是触发/增量更新的协议
- 4 通过周期性的发送keepalive信息, 保证TCP连接的可靠性
- 5 BGP有丰富的metric值, 来衡量路径的优劣, 也称为BGP属性/attribute
- 6 BGP是为巨型网络设计的, 意味着可能有海量的路由和数据包

#### BGP的三张表

- 1 Neighbor table

BGP邻居表, 可以直接相连, 也可以凌空建立, 跨路由器建立邻居, 前提是可以PING通

- 2 BGP Forwarding table

BGP表, 一个BGP路由器, 可能从它的多个BGP Neighbor那里, 学到到达同一个目标的多条路由, 但是, BGP这种路由协议, 默认下是不进行负载均衡的, 也就是说, 到达一个目标, 只允许有一条可用路径, BGP路由器通过“BGP属性”, 从中优选一条它自己认为最好/最优/BEST的路径, 作为到达这个目标网络的“最佳路径”, 这条路由也会称之为“优化了的”(优化了的路由, 会打上“>”), 只有优化了的路由, 才能作为BGP送出来的最佳的种子选手, 去参加到达这个目标网络的AD竞选

- 3 BGP route table

BGP路由表, 上面BGP提交的优化了的路由, 在经过AD竞选后, 如果获胜, 才能送进路由表, 作为到达目标网络的路由

#### BGP Neighbors的两种分类

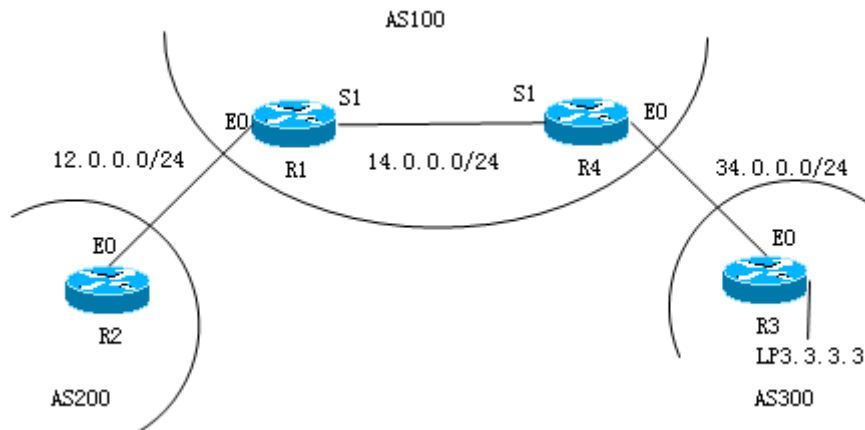
- 1 EBGp neighbors: 分别属于两个不同AS, EBGp通常是直链的

- 1 EBGp neighbors: 分别属于两个不同AS, EBGp通常是直链的
- 2 IBGP neighbors: 两个BGP neighbors同时属于一个相同的AS, IBGP neighbors可以是直连的, 也可以是非直连的

#### BGP的4种类型

- 1 open
- 2 keepalive
- 3 update
- 4 notification

#### LAB1: BGP的基本配置 (图BGP-005)



STEP1: 确认L1/L2的连通性 (物理层/数据链路层)

STEP2: 确保L3的路由通达 (ospf/eigrp/rip等), 在这里AS100中, 所有路由器运行IGP:RIPV2

注意: L3的通达是通过IGP实现的, 一般是在同一AS中完成, 注意: 两个AS之间是不运行IGP的

STEP3: 启动BGP, 并立即指定全局为唯一的BGP router-id

```
R1: router bgp 100
    bgp router-id 100.0.0.1
```

```
R4: router bgp 100
    bgp router-id 100.0.0.4
```

```
R2: router bgp 200
    bgp router-id 120.0.0.2
```

```
R3: router bgp 300
    bgp router-id 130.0.0.3
```

STEP4: 构建BGP邻居, R1-R4建立IBGP, R1-R2, R3-R4建立EBGP

```
R1: router bgp 100
    nei 14.0.0.4 remote-as 100
```

```
R4: router bgp 100
    nei 14.0.0.1 remote-as 100
```

```
R4: router bgp 100
    nei 34.0.0.3 remote-as 300
```

```
R3: router bgp 300
    nei 34.0.0.4 remote-as 100
```

同理R1-R2建立EBGP

建立后，通过以下命令查看信息：

```
sh tcp brief
```

```
sh ip bgp summary
```

其中，sh ip bgp summary后，提示信息里的state下必须是空的，如有信息，说明邻居尚未建立起来，prxrcd: 0，说明，没有从这个邻居收到BGP路由，但和这个邻居的关系已经确立

STEP5:通过NETWORK，宣告BGP路由

```
R3:in loop 113
```

```
    ip ad 113.0.0.3
```

```
R2:in loop 112
```

```
    ip ad 112.0.0.2
```

```
R3:router bgp 300
```

```
    network 113.0.0.0 mask 255.255.255.0
```

STEP6: 用以下命令查看路由的是否优化和传送

```
show ip bgp
```

```
show ip bgp summary
```

6-1: BGP路由器把自己学到的BGP路由，转发给别的BGP邻居的条件：

每个BGP路由器对于特定的某条BGP路由，必须是自己优化过的路由，才具备转发给其他BGP邻居的能力，特别注意的是：这是必要条件，但不是充分条件，也就是说，即时自己已经把收到的BGP路由优化，但此路由，可能转发，也可能不转发，这要看路由器的心情，我靠，还有这事！

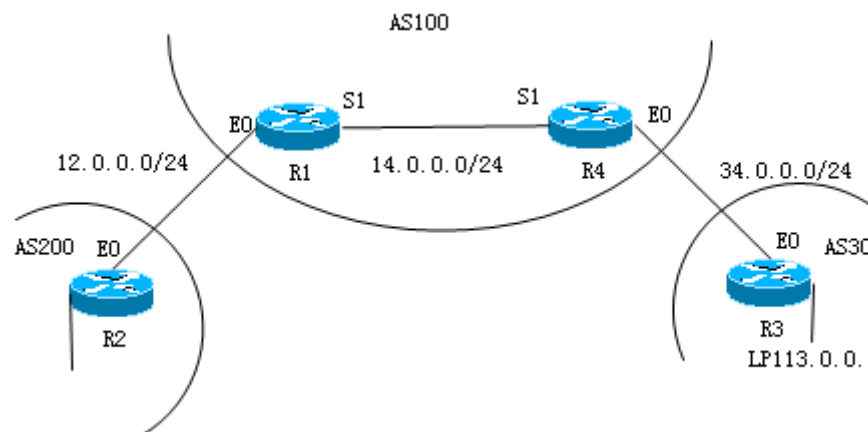
用此命令查看是否把BGP路由转发给邻居：

```
show ip bgp neighbor 14.0.0.1 adve route
```

6-2: IBGP路由中的必要优化条件：

- 1 下一跳问题
- 2 路由同步问题

图BGP-006



- 1 下一跳问题：

如图，R4上对R1BGP路由的下一跳，因为之前的R1下一跳34.0.0.3不可达，所以不能优化，那么

R4: nei 14.0.0.1 next-hop-self

收敛慢，所有用此命令加速收敛：软清除：clear ip bgp \* soft out，硬清除：clear ip bgp \*

下一跳变为了14.0.0.4，可达了！日，不容易

2 同步问题：

如图，同步是指：如果R1能够通过IGP（RIP）学到113.0.0.0/24路由，那么就是同步，R1不能通过IGP（这里是RIP）学到113.0.0.0/24，那么就不能满足同步的需求，所以，只能关闭同步规则，也就是不遵循同步规则

解决办法：关闭同步规则

R1: router bgp 100

no synchronization

结论：对于EBGP，不可考虑BGP属性下，肯定可以优化

## BGP黑洞及解决途径

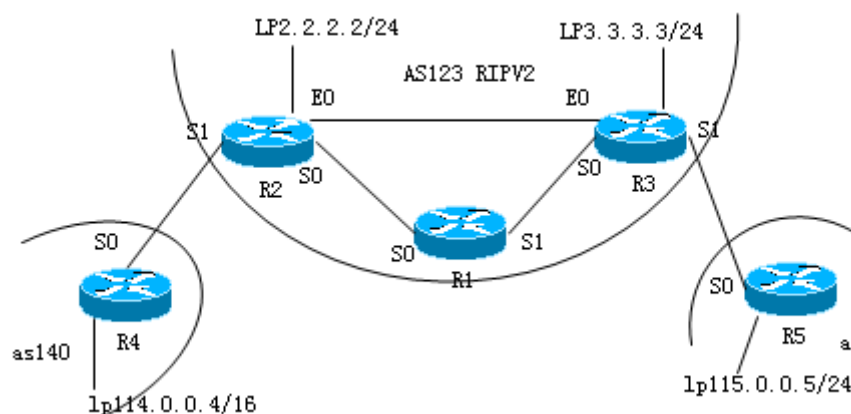
### BGP黑洞

#### 1、BGP的更新源

原则1、在默认情况下，BGP路由器以自己路由表中，到达对方BGP邻居的地址的路由所指示的出接口，作为自己的BGP更新源。

原则2、当BGP路由器，收到邻居发来的BGP信息时，会检查其源地址，然后和自己宣告的邻居的目标地址进行比较，如果一致，这个BGP session可以正常建立。

LAB1：验证通过物理接口构建IBGP/EBGP邻居的不稳定性



STEP1：按图配置

STEP2: AS123内运行RIPV2

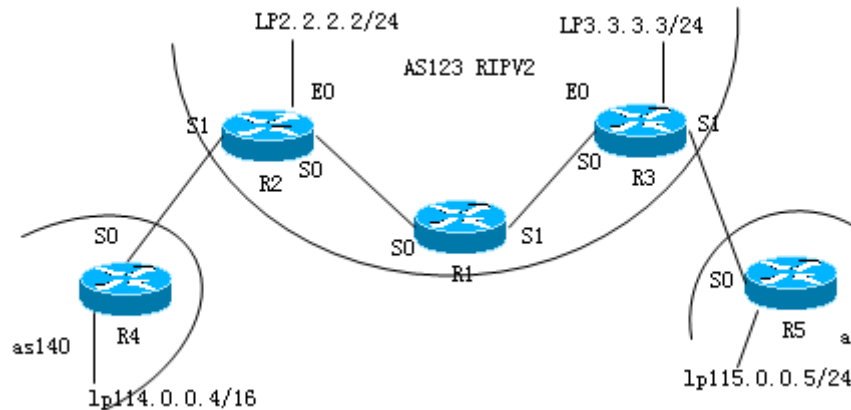
STEP3: 在R2/R3之间构建IBGP邻居

测试: 中断R2/R3的E太网连接, 那么IBGP会话就此中断

结论: 物理连接建立邻居, 不稳定, 建议使用环回口构建IBGP邻居

LAB2: 以LP口构建稳定的IBGP session

如图: 删除LAB1中R2和R3之间的E网连接



STEP1: R2/3构建LP, 并宣告进RIP (IBGP)

STEP2: R2/3上删除LAB1的物理接口邻居

STEP3: 通过LP口, 作为IBGP的目标地址 (以对方的LP口, IBGP的目标地址)

R2: nei 3.3.3.3 remot 123

STEP4: 以自己的LP口, 作为IBGP连接的源地址

R2: nei 3.3.3.3 update-source loopback 2

R3: nei 2.2.2.2 remot 123

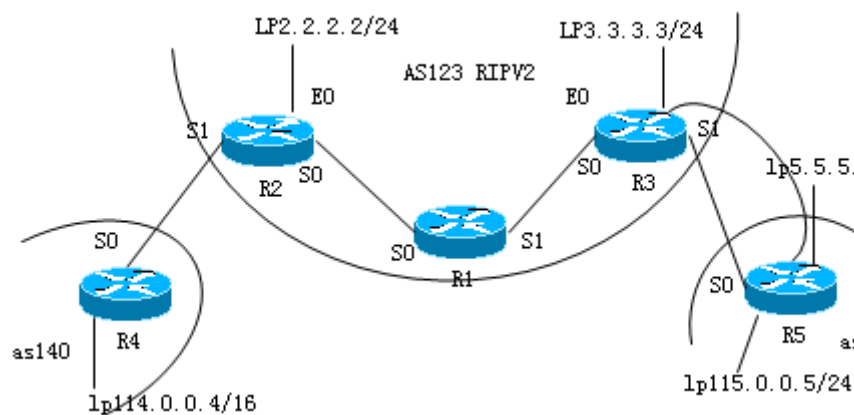
nei 2.2.2.2 update-source loopback 3

测试: 不管R2/3有没有直连链路, 邻居关系都可建立, IBGP都不会DOWN掉

建议: 凡是构建 IBGP默认都使用LP口做更新源, 更稳定、

LAB3: 以LP口做更新源, 构建EBGP session

如图: R3/R5之间有两链路, 两个E口, 两个串口S0和S1



STEP1: 在两个AS之间，构建冗余链路

STEP2: 为两个AS之间EBGP构建LP

STEP3: 在各自路由器上，指定到达对方LP口的静态路由

R5: ip route 3.3.3.3 255.255.255.0 35.0.0.3

ip route 3.3.3.3 255.255.255.0 100.0.0.1

R3: ip route 5.5.5.5 255.255.255.0 35.0.0.5

ip route 5.5.5.5 255.255.255.0 100.0.0.2

STEP4: 建立EBGP邻居, 注意告知对方，自己的更新源

R3: nei 5.5.5.5 remot 150

nei 5.5.5.5 update-source loop 3

R5: nei 3.3.3.3 remot 123

nei 3.3.3.3 update-source loop 5

STEP5: 更改EBGP的TTL值，因为EBGP用LP口建立邻居，其TTL值默认是1，只用一跳就为0了，但EBGP至少要两跳

R5: nei 3.3.3.3 ebgp-multihop (不设参数，意味其TTL设为255)

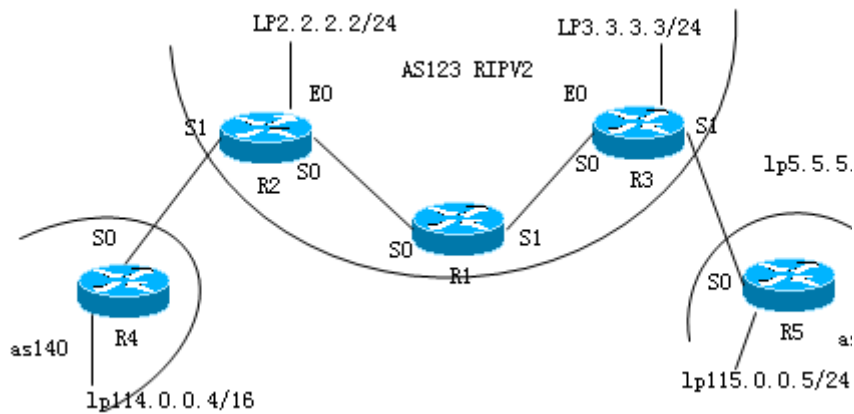
R3: nei 5.5.5.5 ebgp-multihop

**TTL值: 每经过一个路由器，其TTL值会自动减1，如果TTL值减少到0，就会停止转发（TTL在3层的IP包中）**

STEP6: 测试，中断R3/R5的串行链路，不会影响EBGP邻居的运行

结论：在一般情况下，EBGP是不用LP口构建的，默认都直接使用物理接口建立邻居（单链路情况下），只有在两个AS之间，存在冗余链路时，才考虑用LP口构建邻居，提高稳定性。

LAB4: BGP黑洞的形成



STEP1: 确认三个BGP邻居的建立, EBGp: R4和R2、R3和R5, IBGP: R2和R3, 并且AS123内都运行RIPV2

## STEP2: 宣告BGP路由

```
R5:  network 115.0.0.0 mask 255.255.255.0
```

```
R3:  network 114.0.0.0 mask 255.255.0.0
```

STEP3: 在R2和R3上指定对方到AS150和AS140的下以跳为自己，并且关闭同步，切记！

此时，R4和R5上都有对方的路由，但是用扩展PING不通，原因：R1只有AS123的RIP路由，没有AS140和AS150的路由

，所以R1收到114.0.0.0和115.0.0.0的路由后，不知道该送往何处，所以丢弃该包，注意R1没有运行BGP协议，也没和R2/R3建立IBGP邻居

### 解决黑洞方案:

- 1、redistribute
- 2、full-mesh ibgp
- 3、part-mesh ibgp+refilector
- 4、confederation
- 5、mpls

解决黑洞方法一: LAB5:part-mesh ibgp,redistribute selected  
bgp into igp,with sync

STEP1:定义需要进行重分布到IGP中的路由

```
R3: ip prefix-list bgp-115 permit 115.0.0.0/24
```

### STEP2: 通过route-map, 控制重分布到IGP的范围

```
R3: route-map bgp-rp permit 10
    match ip ad prefix-list bgp-115
    set metric 2
```

STEP3:按照route-map bgp-rp定义的条件,将BGP注入RIP

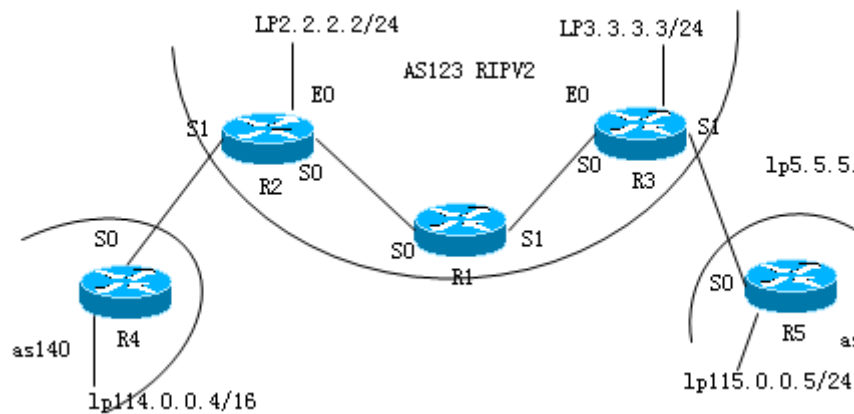
```
R3: router rip
    redistribute bgp 123 route-map bgp-rp
```

R2同理，将114.0.0.0重分布进RIP

#### STEP4: 测试, 扩展PING通过, 解决黑洞

解决黑洞方法二：LAB6:full-mesh ibgp with no sync

STEP1: 按图配置



STEP2:IGP (RIPV2)，且AS123中都用LP口构建邻居，更稳定，LP口也运行RIPV2

STEP3: 通过各自的环回口，构建IBGP邻居

R3: bgp 123

bgp router-id 123.0.0.3

R3: nei 1.1.1.1 remote-as 123

nei 1.1.1.1 update-source loop 3

R1/R3/R3相互构建邻居关系

sh ip bgp summary

构建EBGP邻居，R3和R5，R2和R4，注意：这里的EBGP关系用物理口建立邻居

STEP4: 在BGP进程中，宣告BGP路由

R5: network 115.0.0.0 mask 255.255.255.0

R4: network 114.0.0.0 mask 255.255.0.0

STEP5:解决下一跳和同步问题

同步问题：R2关闭同步（版本新的默认都是关闭的）

下一跳问题：

R3 nei 1.1.1.1 next-hop-self

nei 2.2.2.2 next-hop-self

clear ip bgp \* soft out BGP软清除

同理：R3关闭同步，然后解决下一跳问题

结论：FULL-MESH是可以解决黑洞问题的

STEP6: peer groups=template(模板)

由于full-mesh有太多的直连，所以启用peer-group定义模板解决大量手工输入问题

例如LAB6中：

R1: nei as123 peer-group(定义模板)

nei as123 remote-as 123

nei as123 update-source loop 1

对各个适合模板参数的nei进行调用

```
nei 2.2.2.2 peer-group as123
```

```
nei 3.3.3.3 peer-group as123
```

### LAB7:part-mesh ibgp+reflector

STEP1:删除R2/R3 IBGP session

删除之前有R2BGP路由

删除后在R2上观察115.0.0.0路由，没了，考！

R1有115.0.0.0优化过的路由，但是没有转发给R2，奇怪了我日

用命令sh ip bgp nei 2.2.2.2 adverti观察发给2.2.2.2的路由信息，发现是空的R1没有给R2发送，R1也没有给R3发送。为什么R1优化过的路由不转发呢？这是因为split horizon(水平分割)的原因

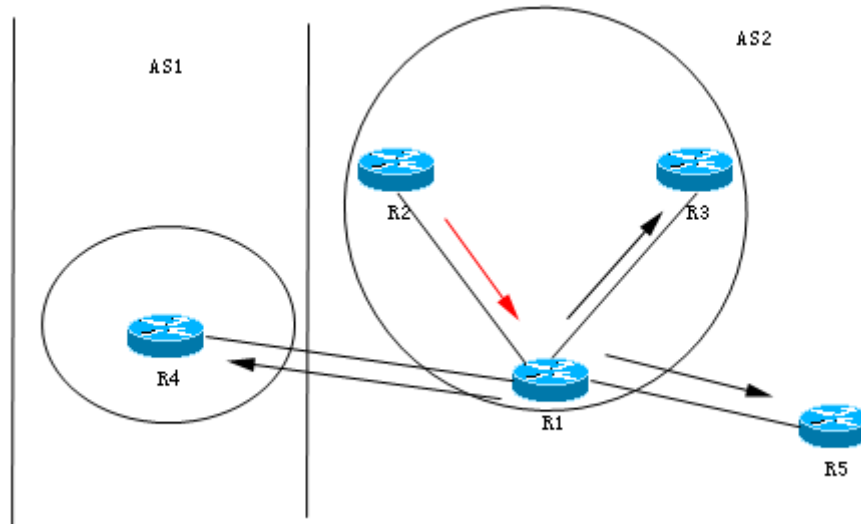
ibgp split horizon rule(水平分割原则)：默认情况下：(R1)从一个IBGP邻居邻居(R3)学到的BGP路由，是不会发送给另外(R2)一个IBGP邻居的

解决办法1:BGP route reflector /利用“路由反射器”规则 2: confederation/联邦规则 3: community/团体规则

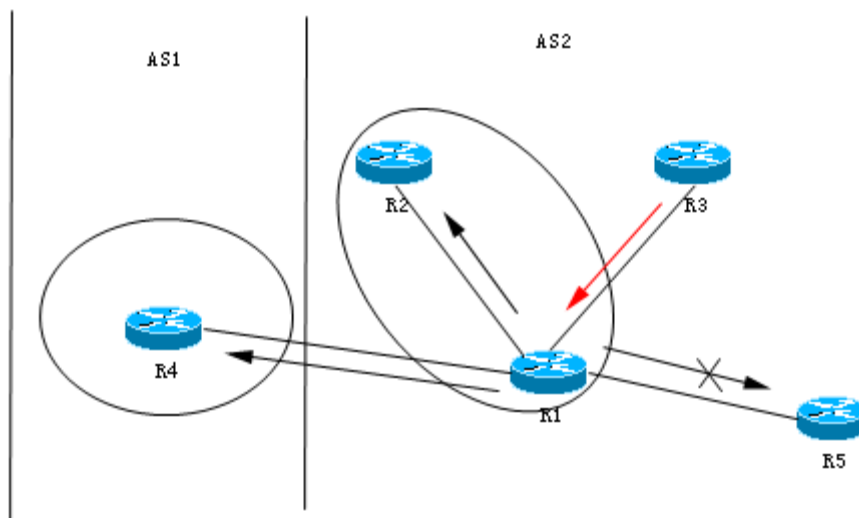
### 办法一：LAB7-1 BGP route reflector /利用“路由反射器”规则

路由反射器的三种规则

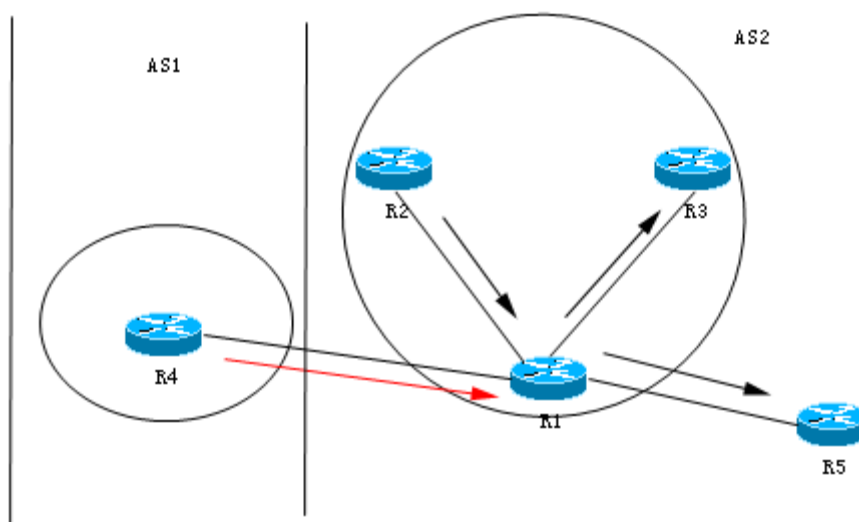
第一种：来自同一AS内的徒弟的路由，会反射给所有BGP session



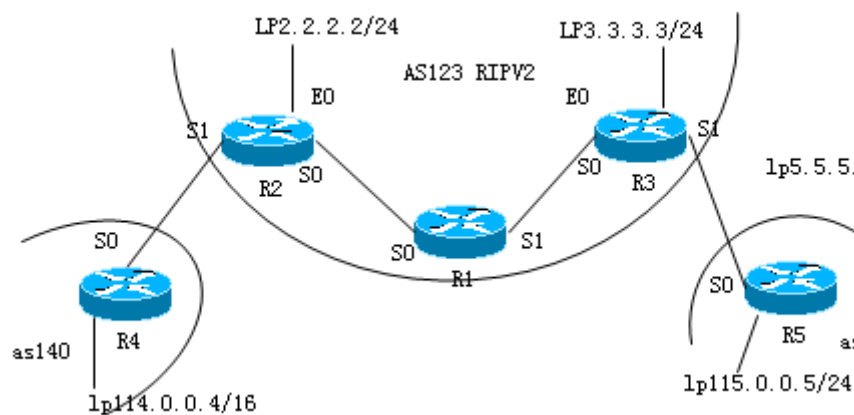
第二种：非徒弟的路由会发给自己的徒弟，而不会发送给非徒弟，但可以发送给其他AS



第三种：外部来的EBGP路由，可以发给所有本区内的徒弟和非徒弟



那么回到上个实验：



STEP2: 在R1上，定义R2/R3为自己的“路由反射客户端”（规则同图一，规则一）

如果没有使用peer-group，那么：

```
R1: nei 2.2.2.2 route-reflector-client
    nei 3.3.3.3 route-reflector-client
```

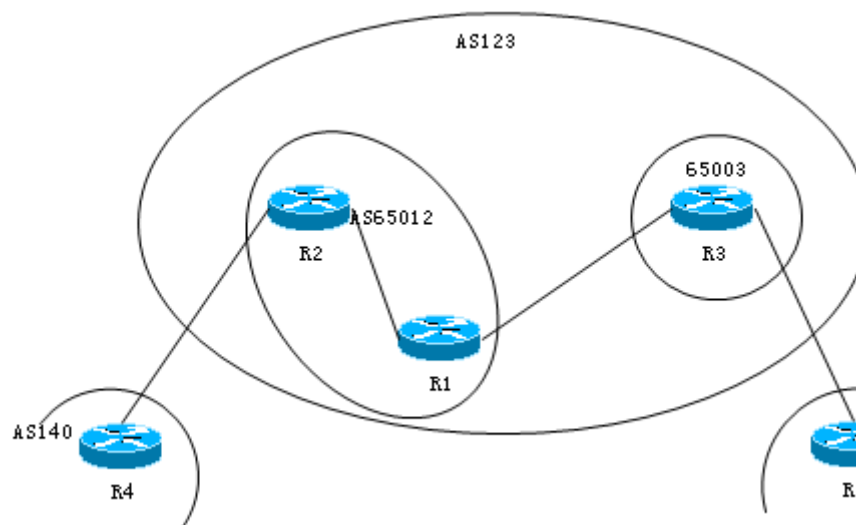
如果使用了peer-group（模板），那么：

```
R1: nei as123 route-reflector-client
```

R2/R3都可以得到优化过的路由

以上操作是让R2/R3成为R1的徒弟，那么R3优化过的路由，会发送给R2（规则一），同理R2优化过的路由，也会发送给R3

方法二: LAB7-2 confederation 联邦 (利用confederation解决问题)



STEP1: R1/R2/R3 删除以前的BGP设置

```
no router bgp 123
```

STEP2: 确认本小AS号

```
R1/R2: router bgp 65012
```

```
R3 : router bgp 65003
```

STEP3: R1/R2/R3都指定自己属于AS123这个联邦

STEP3:R1/R2/R3都指定自己属于AS123这个联邦

在BGP进程里 bgp router-id 123.0.0.\*

R1/R2/R3: `bgp confederation identifier 123`

STEP4:在两个小AS相邻的边缘路由器上，互相指定对方的子AS号

R1: `bgp confederation peers 65003`

R2: `bgp confederation peers 65012`

STEP5:在联邦里指定BGP邻居

R2/R4的EBGP邻居关系:

R2: `nei 24.0.0.4 remot-as 140`

R3: `nei 24.0.0.2 remot-as 123`

R3/R5的EBGP邻居关系:

R3: `nei 35.0.0.5 remot-as 150`

R5: `nei 35.0.0.3 remot-as 123`

R1/R3的联邦EBGP关系:

如果AS123内运行了IGP (RIP)，那么:

R3: `nei 1.1.1.1 remot-as 65012`

`nei 1.1.1.1 updata loop 3`

`nei 1.1.1.1 ebgp-multihop` (只有EBGP，且用LOOP口确认邻居关系的时候，才有必要设置ebgp-multihop, 默认为255)

R1: `nei 3.3.3.3 remot-as 65003`

`nei 3.3.3.3 updata loop 1`

`nei 3.3.3.3 ebgp-multihop`

R1/R2确认邻居关系:

R2: `nei 1.1.1.1 remot-as 65012`

`nei 1.1.1.1 updata loop 2`

R1: `nei 2.2.2.2 remot-as 65012`

`nei 2.2.2.2 updata loop 1`

此时，观察R2上有没有通过R1得到路由

R1上观察有路由，但没有优化，那么:

R2: `nei 1.1.1.1 next-hop-self`

观察R3得到优化过的路由

如果AS123没有运行RIP，那么R1也要指定下一跳为自己，R3才能收到路由

STEP6: 注意: 在联邦的子AS中，所有路由器看到的BGP下一跳都是相邻的AS的边缘节点，而不是本联邦内子AS的下一跳，所以在R1上看到: `* 115.0.0.0/24 35.0.0.5 (65003) 150i` (下一跳是AS150的边缘节点)

解决办法:

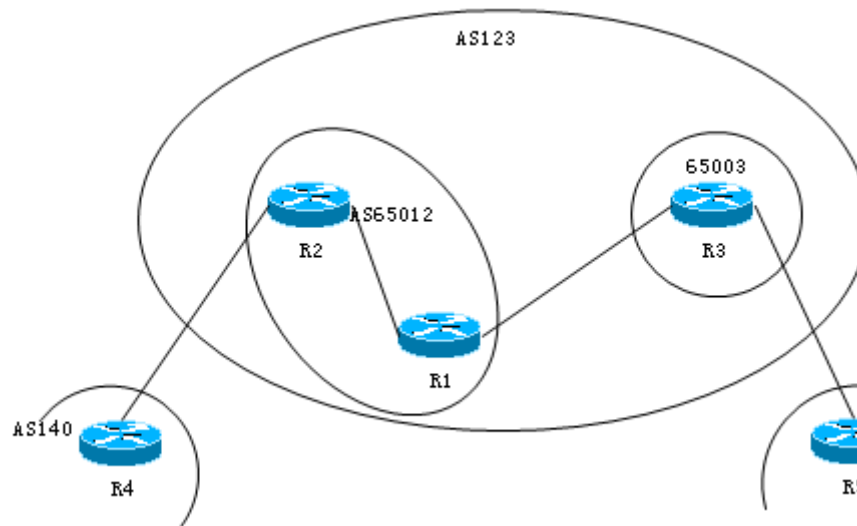
R3: `nei 1.1.1.1 nex-hop-self`

结论: 普通full-mesh和反射器太麻烦，所以考虑用联邦EBGP，划分成许多子AS，问题简单化，减少连接数量

方法三: LAN: 7-3 community/团体 (利用community解决问题)

community:作用: 控制路由的范围, 相当于一种BGP路由的标志位, 用于标示这条BGP路由应该传播的范围

位，用于标示这条BGP路由应该传播的范围



STEP1: 通过ACL/prefix-list, 抓出特定的BGP路由。为了LAB需要, 首先在R4上, 把添加的LP口(140.4.\*.4, \*代表1、2、3、4) 宣告进BGP

```
R4: router bgp 140
    network 140.4.1.0 mask 255.255.255.0
    network 140.4.2.0 mask 255.255.255.0
    network 140.4.3.0 mask 255.255.255.0
    network 140.4.4.0 mask 255.255.255.0
```

需求: 140.4.1.0 路由传送给R2后就停止, 不给其他路由器发送  
 140.4.2.0 路由只允许在AS65012内运行  
 140.4.3.0 路由只允许在AS123内运行  
 140.4.4.0 路由可以正常传给其他路由器

```
R4: ip prefix-list b-1 permit 140.4.1.0/24
    ip prefix-list b-2 permit 140.4.2.0/24
    ip prefix-list b-3 permit 140.4.3.0/24
```

STEP2: 通过route-map 设定路由的community种类

```
R4: route-map t-r2 permit 10
    match ip address prefix-list b-1
    set community no-advertise (do not advertise to any
peer)
```

```
route-map t-r2 permit 20
match ip address prefix-list b-2
set community local-as (do not send outside)
```

```
route-map t-r2 permit 30
match ip address prefix-list b-3
```

```
set community no-export (do not export to next AS)
```

```
route-map t-r2 permit 40
```

STEP3:在R4上, 对R2的BGP路由策略发生“出方向”的改变

```
R4: nei 24.0.0.2 route-map t-r2 out
```

STEP4:将这些communtiy送给下一个BGP路由器

```
AS140: R4: nei 24.0.0.2 send-community
```

```
as65012 R2: nei 1.1.1.1 send-community
```

```
as65012 R1: nei 3.3.3.3 send-coimmunity
```

## BGP汇总与过滤

### BGP-summarizntion(BGP汇总)

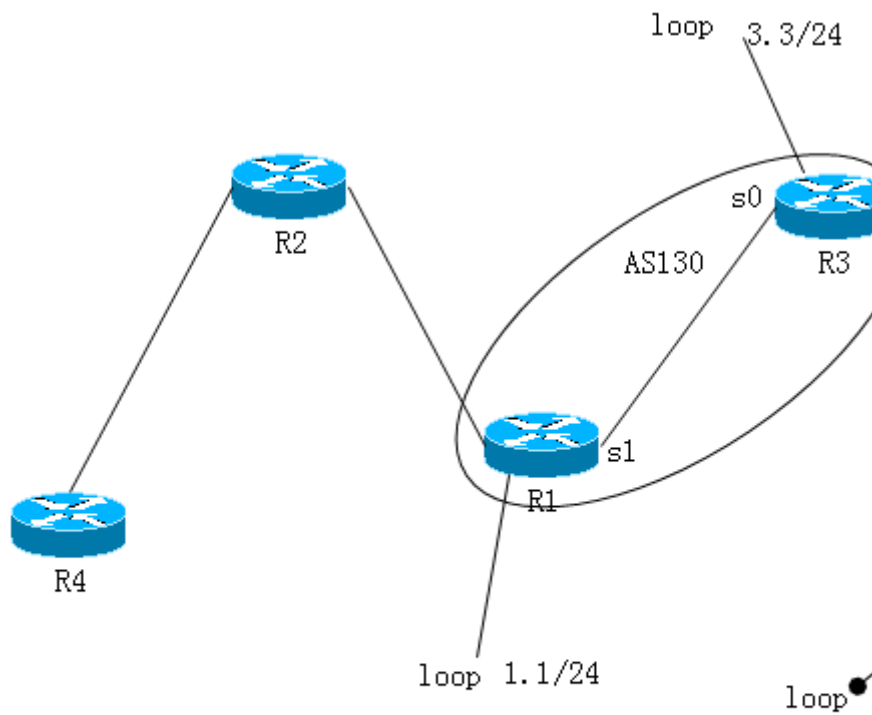
如图:

一: 1-3运行RIP, 注意关闭自动汇总: no auto-summary

二: 建立BGP关系, 1-3用环回口, 3-5用物理口

三: 宣告BGP路由, 注意R5的环回口也宣告进BGP

四: 在R3上宣告下一跳为自己, 否则R1上看不到优化过的路由



LAB1: 非专业汇总（将多的路由，汇总成数量较少的路由条目，减少资源浪费）

STEP1: 手工生成一个需要汇总的、静态的、空接口

R5: ip route 115.1.0.0 255.255.252.0 null 0

STEP2: 将上述路由，宣告到BGP中

R5: network 115.1.0.0 mask 255.255.252.0 注意如果只执行STEP2，而不执行STEP1，那么这个汇总路由不会生效

小结论: 如果用network命令做BGP汇总，是不需要宣告明细路由的

LAB2: 专业汇总（推荐）

STEP1: 首先删除LAB1中的汇总，和静态NULL0

STEP2: 宣告明细路由

R5: network 115.1.0.0 mask 255.255.255.0

network 115.1.1.0 mask 255.255.255.0

network 115.1.2.0 mask 255.255.255.0

network 115.1.3.0 mask 255.255.255.0

STEP3: 在BGP进程里，使用AGG-ADD命令汇总路由

R5: aggregate-address 115.1.0.0 255.255.252.0 summary-only

再观察R5路由，发现自动生成一条BGP路由，此时的R5，抑制了其他所有的明细路由，只发送了汇总路由给R3，实现了BGP路由的汇总目的

其中在R5上 sh ip bgp，会发现优化的BGP路由如下标示：

总目的

其中在R5上 sh ip bgp ,会发现优化的BGP路由如下标示:

s>115.1.0.0/24 s>:表示是被抑制的路由条目

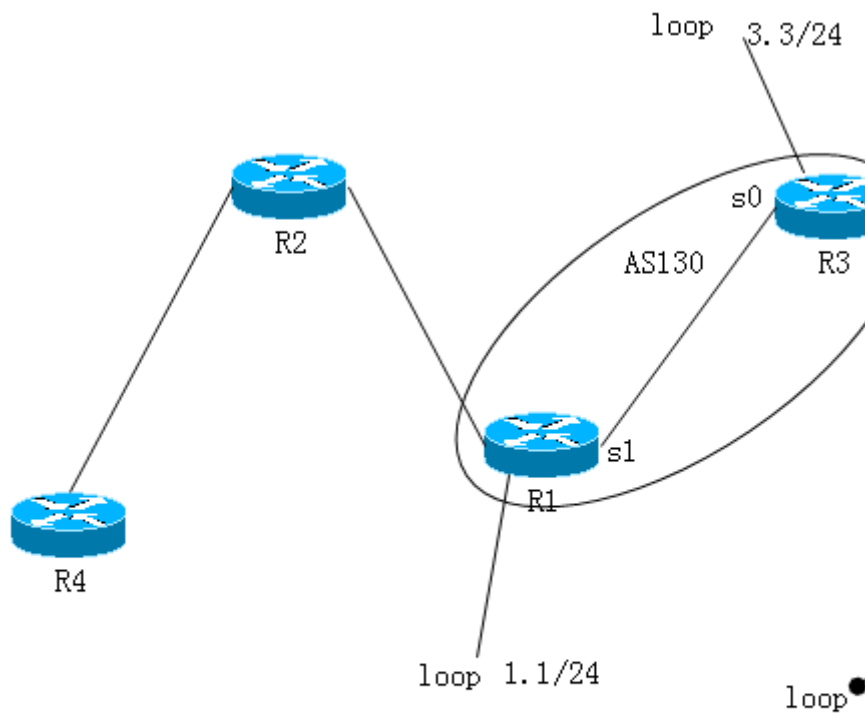
\*>115.1.0.0/22 \*:表示是正确的路由条目

~~~~~  
~~~~~

## BGP中的路由控制/过滤

LAB3:通过neighbor命令调用distribute-list+ACL(较落后)

需求: R5LP口的奇数路由不发送给R3, 偶数路由可以发送, 如图:



STEP1: 通过ACL定义BGP路由

R5: access-list 1 deny 115.1.1.0 0.0.254.0

access-list 1 permit any

STEP2:在BGP进程中,用neighbor命令,调用ACL

R5: neighbor 35.0.0.3 distribute-list 1 out

STEP3:用BGP软清除,刷新出口的改变

R5: clear ip bgp \* soft out

STEP4:在R5上查看发给R3有哪些路由

R5: sh ip bgp nei 35.0.0.3 adver

R5: sh ip bgp nei 35.0.0.3 adver

LAB4:通过prefix-list定义BGP路由,然后用NEI命令直接调用prefix-list,推荐(需求同LAB3)

STEP1: 通过prefix-list定义BGP路由

```
R5: ip prefix-list b-0 seq 5 deny 115.1.1.0/24
    ip prefix-list b-0 seq 10 deny 115.1.3.0/24
    ip prefix-list b-0 seq 15 permit 0.0.0.0/0 le 32
```

(0.0.0.0/0 le 32:表示any)

STEP2: 通过NEI命令, 直接调用prefix-list

R5: nei 35.0.0.3 prefix-list b-0 out

STEP3:clear ip bgp \* soft out

LAB5:nei+route-map+acl进行控制和过滤(需求同上)

STEP1: 通过ACL定义BGP路由(定义之前首先NO掉LAB4中的prefix-list列表和NEI调用)

R5: access-list 2 permit(匹配) 115.1.1.0 0 0.0.254.0

STEP2:通过route-map调用ACL

```
R5: route-map t-r3 deny(拒绝通过) 10
    match ip ad 2
    route-map t-r3 permei(允许通过) 20
    空的route-map表示: <match any, set nothing>
```

STEP3: 通过NEI调用route-map

R5: nei 35.0.0.3 route-map t-r3 out

LAB6:nei+route-map+prefix-list

STEP1:通过prefix-list定义路由

```
R5: ip prefix-list b-2 permit 115.1.1.0/24
    ip prefix-list b-2 permit 115.1.3.0/24
```

STEP2:通过ROUTE-MAP调用PREFIX

```
ROUTE-MAP SEND-R3 DENY 10
MATCH IP AD PREFIX B-2
route-map send-r3 permit 20
```

step3:通过nei调用prefix-list

R5: nei 35.0.0.3 route-map send-r3 out

~~~~~

BGP dampening (衰减)

在BGP路由器中, 宁可网络收敛慢, 也要确保BGP路由的稳定性

Dampening有4个默认值

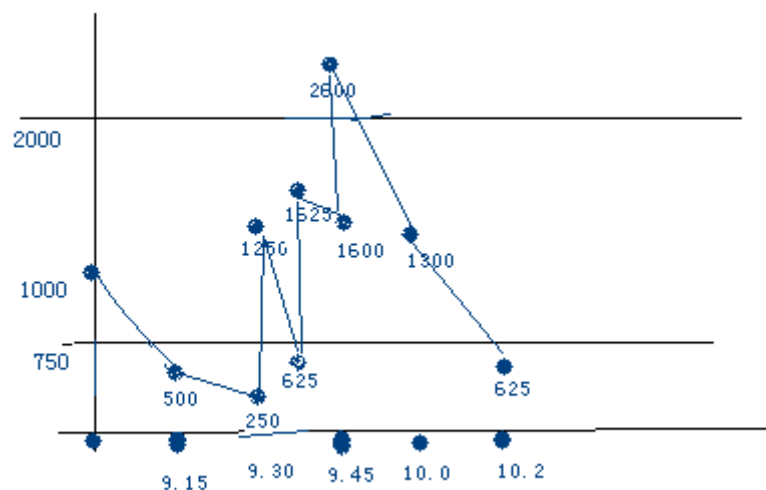
半衰期: 默认是15mins

开始抑制的阈值: 2000

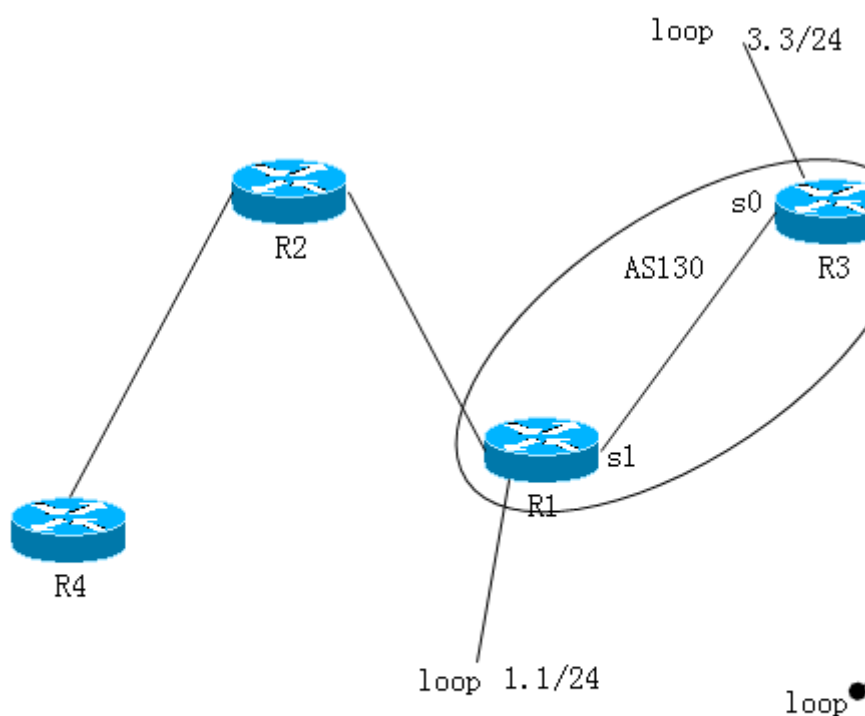
重新启用的阈值: 750

最大抑制时间: 60mins

如图:



LAB1: 观察dampening过程 TOP如图:



step1:通过prefix-list, 定义需要监控的/dampening的BGP路由

R3: ip prefix-list as-150-115 permit 115.1.0.0/24

ip prefix-list as-150-115 permit 115.1.2.0/24

step2:通过route-map, 对AS-150-115路由进行dampening

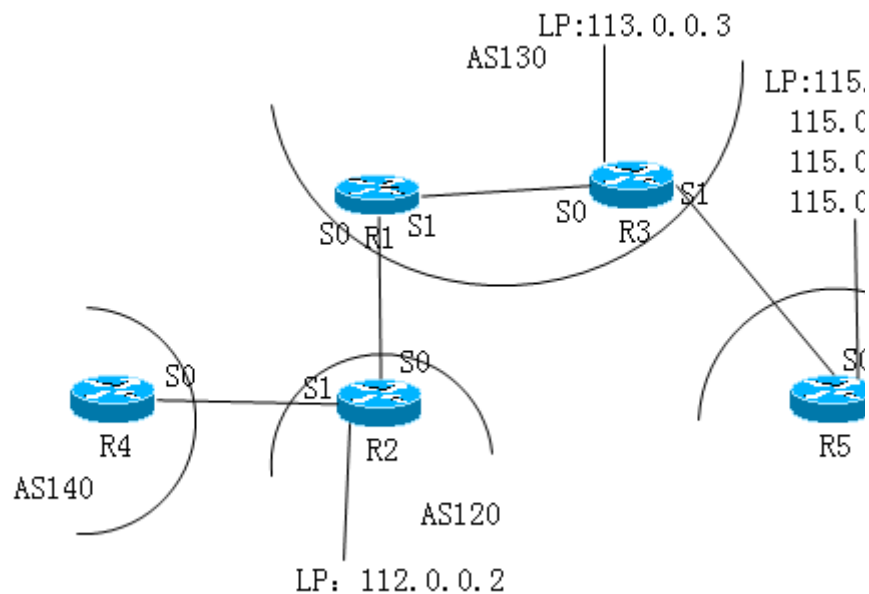
R3: route-map as150-damp(自定义)

```

match ip ad permit-list as-150-115
set dampening 15 750 2000 60
15:half-life
750:reusing/重新启用路由的阈值
2000: suppress/开始进行抑制的阈值
60:maximum duration to suppress/最大抑制时间
setp3: 调用BGP dampening
R3: ip bgp 130
    bpg dampening route-map as150-damp
step4:测试
R3: sh ip bgp dampened-paths
R3: sh ip bgp flap-statistics
R5上shutdown LP2, 然后在R3上调用sh ip bgp flap-statistics然后观察
R5: sh ip bgp
出现*d标示, 说明已经dampening
clear ip bgp dampening
clear ip bgp flap-statistics
迅速清除dampening, 无需等待释放

```

~~~~~  
~~~~~  
LAB7: 基于AS-PATH的过滤, 如图:



情况一: 需求, AS130的路由不发送给R4, 齐其他正常
STEP1:
R2: ip as-path access-list 10 deny ^130\$

```
ip as-path access-list 10 permit .*
STEP2: 在BGP进程中对特定的路由进行调用
R2: nei 24.0.0.4 filter-list 10 out
情况二: 需求, 将起源于AS150的路由deny, 其他正常
STEP1:
R2: ip as-path access-list 10 deny _150$
     ip as-path access-list 10 permit .*
     nei 24.0.0.4 filter-list 10 out
情况三: 需求, 凡是穿越AS130的路由都deny掉, 其他正常
STEP1:
R2: ip as-path access-list 10 deny _130_
     ip as-path access-list 10 permit .*
     nei 24.0.0.4 filter-list 10 out
```

BGP属性

BGP Attributes (通过BGP的属性, 控制BGP路由选择)

BGP Route selection \BGP的选路原则 (1-12, 注意5以上的是越大越好, 5以下的越小越好)

- 1:next hop
- 2:synchronizde
- 3:highest weight (local to router)
- 4:highest local preference (global within as) (只影响本AS)
- 5:route originated by the local router (next hop=0.0.0.0)
- 6:shortest AS path
- 7:lowest origin code (IGP<EGP<incomplete)
- 8:lowest MED (from other AS) (MED即metric)
- 9:EBGP path over IBGP path (EBGP AD=20, IBGP AD=200)
- 10:the path through the closest IGP neighbor
- 11:oldest route for EBGP networks
- 12:the path with the lowest neighbor BGP router id (取router-id最小)

以上对比, 用sh ip bgp 可以查看到前几条

- 1: 在BGP路由器的BGP表中, 可能存在到达某个特定目标网络的, 多条路径 (图001)
- 2: BGP默认是不执行负载均衡, 而是严格按照网管的策略\意志, 进行BGP选路。网管是通过BGP属性, 去表达其策略\意志的, 实现BGP路由选择的控制。网管可以通过 “maximum-path (1-6)” 命令, 实现BGP的负载均衡。

令，实现BGP的负载均衡。

3:BGP是依靠BGP属性，优选出到达目标网络的最佳的那一条BGP路由，网管就是依靠控制BGP属性，达到对BGP路由进行操纵\选择对目的。

4:BGP所提交给路由表选择对路由，会和别的路由协议所生成的路由进行AD的比较。

5:AD最小的那个路由协议所生成的路由，将被注入\优选进路由表，成为达到目标网络的路由。（图002）

The routing protocol with the lowest administrative distance will be installed in the routing table.

BGP属性的分类：

well-known attributes（公认属性）

well-known mandatory 公认，强制的

well-known discretionary 公认，自决的

optional attributes（可选属性）

optional transitive attributes 可选，可传递，partial

optional nontransitive attributes 可选，非传递的

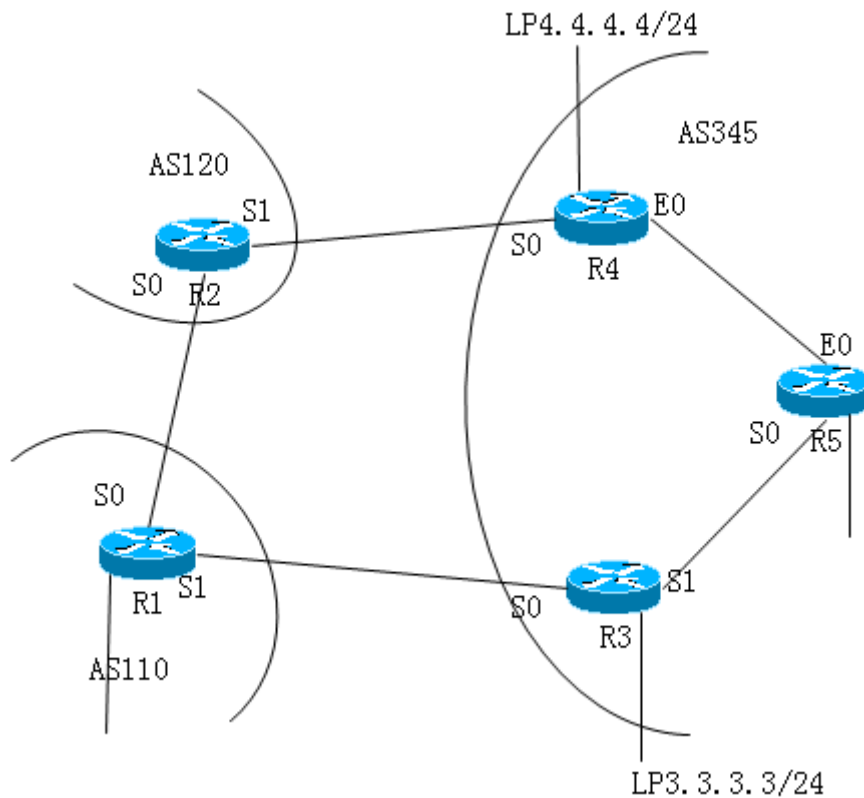
well-known, mandatory & transitive attribute

AS path

next-hop

origin

LAB1：1—12规则中判断标准演示



step1: 按图配置, AS345运行EIGRP, 注意, 环回口也宣告进EIGRP
 step2: EBGp: 2和4, 1和2, 1和3 IBGP: 4和5, 3和5
 step3: BGP路由宣告, 在R1上建立LP1和LP2, 并宣告进BGP路由中
 在R5上观察路由, R4和R3传给R5的路由, R5上都没优化
 step4: 对BGP路由对进行控制
 首先让R5满足同步\下一跳的条件, R3\4 :neighbor 5.5.5.5 next-hop-self
 观察后, 通过shortest as path 第六条, 选择了R3

LAB2: BGP路由操纵 (需求: 整个AS345的路由, 都从5-3-1传送)

用MED操纵, 1-12规则中第6条

特征: 1、越小越好 2、只发送给EBGP邻居, 用于建议对方, 如何离开对方的AS, 来访问我方的AS 3、MED attribute is optional and nontransitive 4、MED的默认值是0

step1: 通过ACL/prefix-list, 定义需要控制的路由

注意: 是在R1上发给对方的, 是R1希望R5通过R5-3-1到达R1, 而不要从R5-4-2-1到达, 所以根据BGP路由的方向, 设定MED:

R1: ip prefix-list P-1 permit 110.1.0.0/16

route-map med-t-3 permit 10

match ip address prefix-list P-1

set metric(med) 10

route-map med-t-3 permit 20

```
ip prefix-list P-1 permit 110.1.0.0/16
route-map med-t-2 permit 10
match ip add prefix-list P-1
set metric(med) 50
route-map med-t-2 permit 20
```

step3: 将不同的route-map发送给不同的neighbor

```
R1: neighbor 13.0.0.3 route-map med-t-3 out
    neighbor 12.0.0.3 route-map med-t-2 out
clear ip bgp * soft out
```

R5: bgp always-compare-med

默认情况下，BGP路由只比较来自同一个AS的MED值。可以通过此命令，来实现不同AS的MED的比较。

或者：step4: 把上个实验的指令：bgp always-compare-med，删除掉，然后在R4和R3间构建IBGP，也就是说在AS345里，是FULL-mesh连接。因为R4仍然从R2传送。

1、R3和R4构建邻居

2、R4关闭同步

实现！

LAB3:通过local preference，操纵BGP路由选择（需求：整个全部AS345的路由从5-4-2-1传送）

特征：

1、越大越好

2、其默认值是100

3、只发送给本AS中的IBGP邻居，用于影响他们如何离开本AS，去访问外AS。

4、local preference attribute is well-known, discretionary, and is passed only within the AS

step1: 在本AS的边缘路由器上定义路由

定义：根上个LAB的定义一样:ip prefix-list, R3和R4上定义

```
R4: route-map fr-r4 permit 10
    match ip add prefix-list p-1
    set local-preference 200
```

```
R3          fr-r3
```

step2: 在R3/4上的入口调用

```
R3: neighbor 13.0.0.1 route-map fr-r3 in
```

```
R4: neighbor 24.0.0.2 route-map fr-r4 in
```

step3: R3不能从5-4-2-1传送，因为下一跳不可达
在R4上做R3的下一跳为自己，然后R3关闭同步

LAB4: 用WEIGHT控制路由选择 需求：通过WEIGHT控制，从5-3-1传

传送

特征:

- 1、weight是CISCO私有的属性
- 2、越大越好
- 3、默认值是0
- 4、不发送给任何BGP邻居，只影响本路由器的选路

step1: 定义路由

```
R5: ip prefix-list P-1 permit 110.2.0.0/16
```

step2:

```
R5: route-map wei permit 10
```

```
    match ip ad prefix-list p-1
```

```
    set weight 2000
```

```
    route-map wei permit 20
```

```
router bgp 345
```

```
nei 3.3.3.3 route-map wei in
```

在R1上虚拟加上个AS10，那么第6条不能判断，7条MED，也叫METRIC，通过sh ip bgp 可以看到也是相同的，那么8条和9条也相同，不能判断，那么第十条，可以判断走R4，因为4-5为E路

NP-SWITCH

VLAN

VLAN:

低成本地实现网络的分片、分段

1 a broadcast domain 广播流量不能穿越VLAN的边界

2 Logical network (subnet) 为了实现VLAN之间的数据通信，需要依靠不同VLAN的不同子网实现VLAN路由

3 VLAN: 限制了广播域的范围，VLAN间路由: 又允许了VLAN间的正常数据包的通信

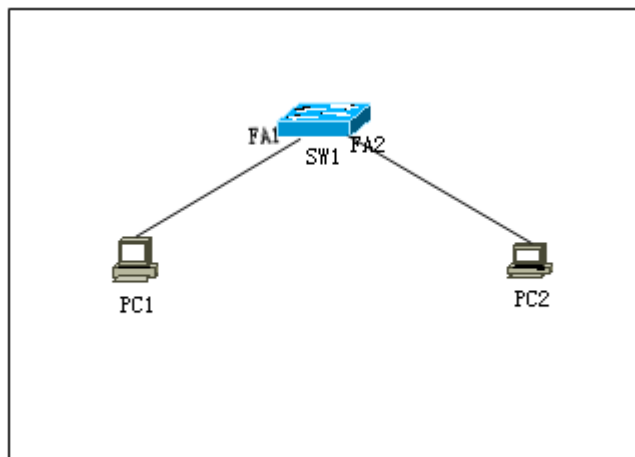
~~~~~

VLAN的分类

静态VLAN: 基于端口的

动态VLAN: 基于MAC的

LAB1: VLAN的创建、修改和删除



创建VLAN:

方法一 SW1: VLAN 10 (新式)

方法二 SW1: VLAN database (老式)

全局下做: VLAN 30

apply (保存)

sh vlan brief (查看摘要信息)

VLAN命名:

方法一 sw1:vlan 10 (新式)

name \*\*\*

方法二 SW1: vlan database

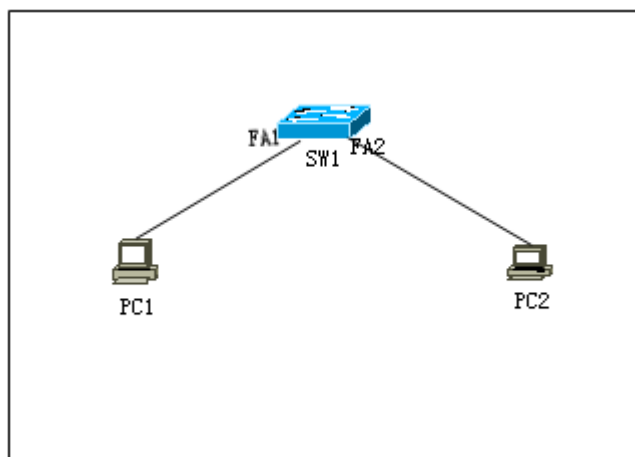
vlan 20 name \*\*

apply

注意老式的做完修改后一定要保存

(apply), 否则无效

LAB3:如图, 将端口放入VLAN中



SW1: in fa 0/2

switchport access vlan 10

此时, 两台PC测试PING, 不通, 因为不在一个VLAN中

注意：如果将一批端口，一次性放入VLAN中，（2, 3, 4, 8）

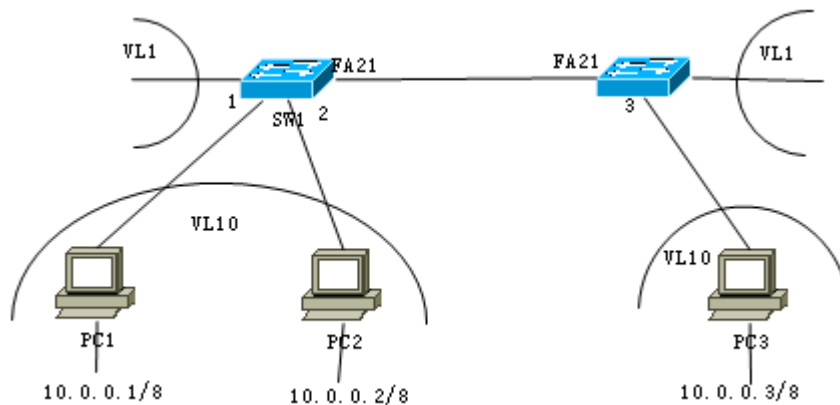
```
SW1: in range fa 0/2 -4 , fa 0/8
      switchport access vlan 10
```

注意：0/2 -4 代表端口2到端口4的全部端口，即2, 3, 4。—代表连续的，, 代表不连续的

~~~~~

VLAN trunk（跨交换机的，同一VLAN的数据通信）

trunk:在一条网络介质上，同时传输多个VLAN数据，这种技术，CISCO称之为trunk



trunk分为两种协议：ISL和802.1Q

第一种trunk, ISL: 是CISCO的私有协议，在原始数据帧外，生成一个26bytes的ISL头，还有4个bytes的FSC尾，在ISL头中包含了15bits的VID
如图：将一个端口配置成TRUNK端口

```
SW2: no ip routing
      in fa 0/3
      swit access vlan 10
      swit mode access (注意：连接PC的用指令access, 连接switch的用trunk)
```

SW1同理，将两个FA口配置为trunk端口

因为SW1和SW2的FA21口默认都在VL1中，不能传递VL10的数据帧，所以要配置trunk, 使之能够传递多个VL信息

第二种trunk, 802.1Q, 是开放型标准，兼容老的设备（不能支持trunk的设备，802.1Q都可以支持）

交换机检测到数据的目标不在本交换机时，当其检测到应当发到某条trunk链路时，先进行封装，在MAC地址后面，添加上4bytes的tag，完成封装后，就可以从trunk链路上发给对方交换机了

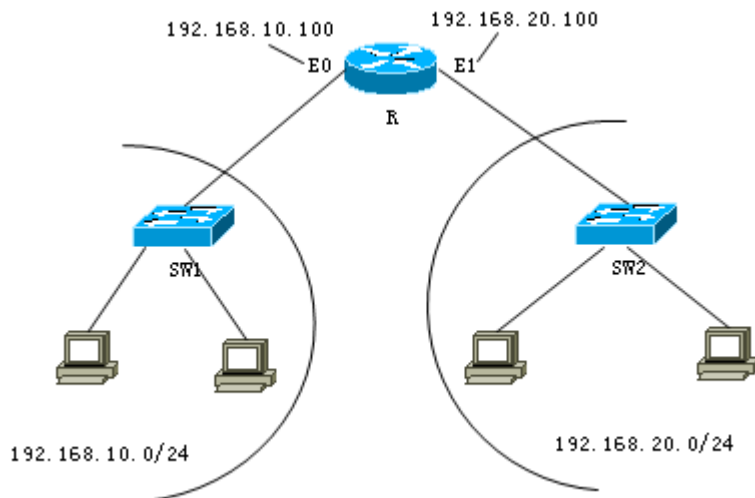
LAB4: 用802.1Q做trunk

```
SW2: in fa 0/21
      switchport trunk encap dot1q
      swit mode trunk
```

~~~~~

## VLAN间路由

第一代LAN间的数据通信，见图004



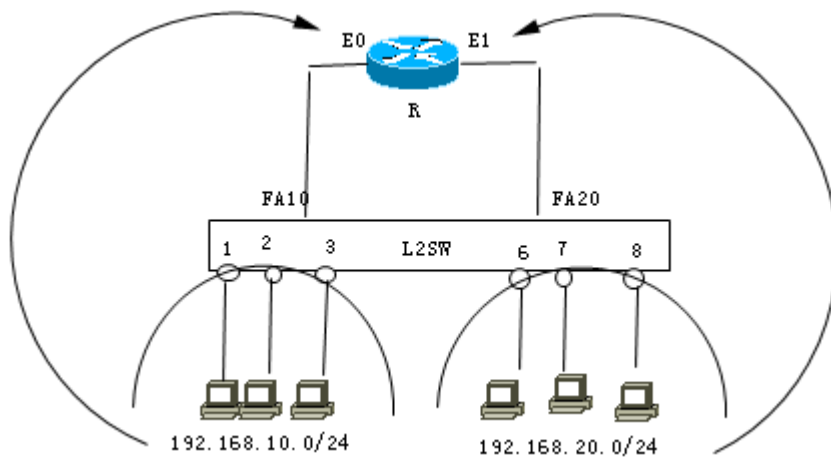
一个路由器，加上几个交换机，每个交换机的所有端口，都通属于一个LAN/网段，在路由器上，又几个网段，就有几个与之对应的直链路由，在那个时代，还没有VLAN这种技术

缺点：交换机的所有端口都同属于一个LAN，需要路由器的端口，与LAN的个数，一一对应，导致建网成本很高

优点：易于理解，便于实施

## 第二代VLAN间的数据通信

在这个时代，出现了能够支持VLAN的交换机。005



在路由器和交换机之间的链路，的那个交换机端口，是完全属于所对应的那个VLAN的

优点：可以将原属于每一个LAN的交换机，减少到只需要一台交换机，大大降低成本

缺点：路由器上，还是一个端口对应的一个VLAN，成本还是很高

### 第三代VLAN间的数据通信：

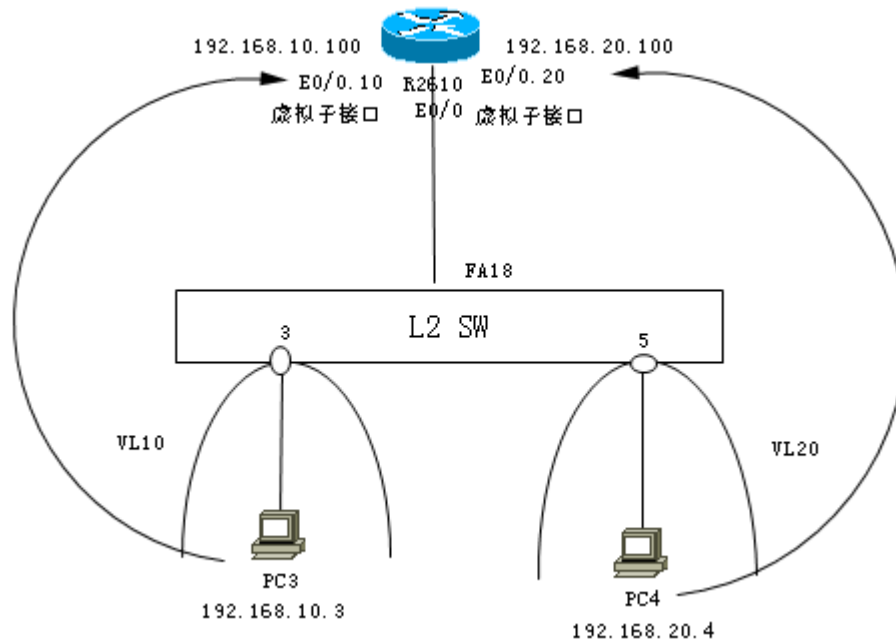
在这个时代，出现了TRUNK技术

优点：减少了连接交换机/路由器的端口个数，降低成本

缺点：从成本考虑，仍然需要外置路由器，从性能考虑，受TRUNK链路的带宽，路由器的查询路由表的能力限制

TRUNK：在一条物理介质，传输多个（网线/光纤）VLAN的信息

单臂路由：router on a stick 见图006



STEP1：在交换机端，配置TRUNK链路

in fa 0/18

switchport mode trunk

指定TRUNK的封装类型

switchport trunk encapsulation dot1q

或者switch port trunk encapsulation ISL

STEP2：在路由器端，配置TRUNK链路和子接口：

R2610: in E0/0 (主接口)

no shutdown

配置子接口

in E0/0.10 (为VLAN10工作的子接口)

encap dot1q 10 (VLAN10)

ip ad 192.168.10.100 255.255.255.0

in E0/0.20 (为VLAN20工作的子接口)

encap dot1q 20 (VLAN20)

ip ad 192.168.20.100 255.255.255.0

sh ip route 可以看到两条直链路由，对应的都是子接口

```
sh ip in E0/0.10 查看三层信息
no ip proxy-arp 关闭代理ARP
STEP3: 确认每个VLAN中的PC, 都能访问到本VLAN所对应的子接口
PC3上PING E0/0.10, 同理PC4PING, 可以PING通
PC4上SH ARP 查看IP和MAC对应关系
PC3上SH ARP, 发现两个子接口对应的MAC地址都相同, MAC地址:
0009.b7e0.d280
PC3 PING PC4不通
打开DEBUG IP PACKET查找原因
STEP4: 为每个VLAN中的用户确定默认网关
PC3: IP DEFAULT-GATEWAY 192.168.10.100
PC4: IP DEFAULT-GATEWAY 192.168.20.100
DEBUG ARP
PC3 PING PC4
所有数据包发送到其他VLAN的数据包, 发送给网关, 让网关转发
```

802.1q的VLAN的数据包, 在TRUNK链路中, 是无需打上TAG的标记的, 也就是说, 无需进行TRUNK的封装, 对于这个VLAN, 可以将其单独标记为NATIVE VLAN/本征VLAN (VLAN1), 目的在于兼容较为落后的设备, 实现与标准的以太网帧兼容。

ISL有没有本征VLAN呢?

没有的

所以在ISL的TRUNK链路上, 无需配置NATIVE

对网络设备的网管:

STEP1: 为交换机配置网管IP地址 (普通的L2交换机, 只能配置一个网管IP):

```
SW2: in vlan 1
      ip address 192.168.1.29
      no sh
```

STEP2: 在路由器上为VLAN1配置用于VLAN路由的子接口

方法一: R2610: in e 0/0 (在物理接口上配置)

```
ip address 192.168.1.100 255.255.255.0
no sh
```

PING VLAN1 192.168.1.29可以通 (图007)

如果VLAN1 (192.168.1.0) 要访问外网, 那么就要指一个网管E0

```
SW2 ip default-gateway 192.168.1.100
```

这样三个VLAN可以正常互访

PC3 telnet 192.168.1.29 注意SW2上要设定EN密码和VTY密码 (见NA笔记, 考!), 否则登录不上

方法二: in E 0/0.1

```
encap dot1q 1 native (本征)
ip address 192.168.1.100 255.255.255.0
```

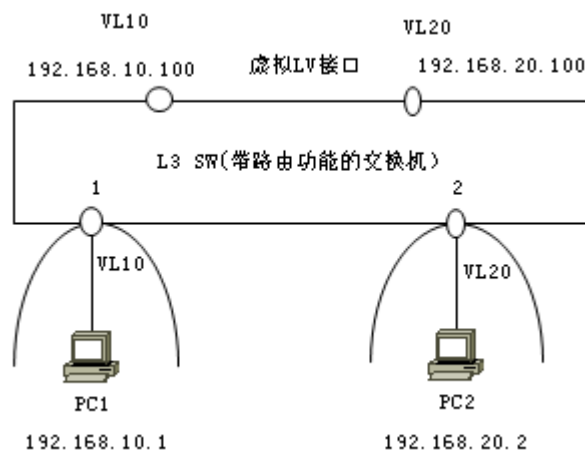
ip proxy-arp/代理ARP

作用: 代替默认网关

在VLAN, 或者网段中的用户无需配置网关

```
PC3/4: no ip default-gateway
在路由器上，确认代理ARP已经启动
R2610: in E 0/0.10
      ip proxy-arp
      sh ip in e/0.10 查看
      clear arp-cache
PC3   debug arp
      ping 192.168.20.4
STEP1:creating incomplete entry for ip address:192.168.20.4 in
E0
STEP2:sent req
      src 192.168.10.3 0060.5cf4.42e2
      dst 192.168.20.4 0000.0000.0000 E0
STEP3: revd req
      src 192.168.20.4 0009.b7e0.d280 (注意该MAC地址是路由器E0
的MAC地址)
      dst 192.168.10.3 E0
```

#### 第四代VLAN间的数据通信(图008)



将简化的路由器，和交换机合二为一，成为带有路由功能的交换机（L3 SW）

优点：从成本考虑，不需要外置路由器，可以降低成本

缺点：受路由器的CPU查询路由器的查询路由表的能力限制（多次路由多次交换）

```
SW1: dir flash: （查看FLASH中的文件）
      delete flash:vlan.dat
      delete flash:config.text
      获得空白交换机以满足实验就必须删除此两个文件
```

STEP1:

首先关闭路由功能以模拟PC

PC1/PC2配置IP地址

R2610的E0 shutdown

PC1/PC2指定网关10.100和20.100

STEP2:在交换机上创建两个VLAN，把端口划分到VLAN中去

VLAN 10

swit access vlan 10

swit mode access

在交换机上，为每个VLAN，创建一个与之对应的VLAN接口（逻辑接口）

in vlan 10

ip ad 192.168.10.100

in vlan 20

ip ad 192.168.20.100

注意查看IP PROXY-ARP是否启动，如没启动，配置默认网关

STEP4: 启动L3的路由功能

ip routing

sh ip routing

STEP5:测试

PC1 PING PC2可以通过

第五代VLAN间的数据通信

一次路由，多次交换 multilayer switching 第一次在三层查和发送，

以后就直接在2层发送了

流掩码/netflow mask

记录了源IP/port, 目标IP/PORT, 协议号

第六代VLAN间的数据通信:

基于CEF的多层交换（三张表）

FIB表（转发信息数据库）

邻接关系表

流掩码表

没有路由，多次交换

启动CEF

SW3550: ip cef

## VTP

### VTP (vlan trunk protocol)

将VLAN信息通过TRUNK链路发送到客户端，VLAN信息，只能在TRUNK链路上传递，是CISCO私有的协议。

VTP用于通告/同步VLAN的配置信息，简化关于VLAN的配置。VTP信息是触发更新，或者每5分钟周期性通告一次。

VTP的server/client的相同点：都可以转发VTP通告信息，而且都会向最新版本的VTP信息同步，

不同点：VTP的server可以创建，修改，删除VLAN，可以保存这些信息在NVRAM中

VTP的client不能创建，修改，删除VLAN，不能保存这些信息在NVRAM中

VTP的 transparent/透明模式：拥有独立的一套VLAN数据库，不会向server端发出的VTP信息同步，但是可以转发SERVER端发出的VTP信息，而且自己的独立数据库是可以保存的

### LAB1: VTP 图009

STEP1: 创建TRUNK链路，确认TRUNK链路已经成功建立  
sh in trunk 查看

STEP2: 确定一个VTP域名，和确认VTP server/client  
一般以网络中，最高端/中心的交换机做SERVER  
SW1/SW2/SW3: vtp domain ccnp

```
SW1: vtp mode server
SW2: vtp mode client
SW3: vtp mode client
```

sh vtp status (查看VTP信息, 特别注意, sh run 是看不到VTP信息的, 因为VLAN信息是保存在VLAN.DAT文件中的)

STEP3:通过在SERVER端, 增加, 修改, 删除VLAN, 查看VLAN信息同步, 观察VTP修订版本号的变化

```
SW1:VLAN 10
```

```
SH VTP status
```

```
sh vlan brief
```

```
SW2:sh vlan brief 观察, 已经学到VLAN信息
```

```
SW2同理
```

```
vlan 10
```

```
name mark
```

```
sh vtp status (版本号已经改变为2)
```

```
SW2/SW3也可看到此变化
```

VLAN信息每改变一次, 版本号就增加1 (sh vtp status查看)

STEP4:VTP的安全性, 配置VTP的password

不安全性: 假设SW3先与SW1/SW2断开, 再SW3上创建VLAN20/30/40信息, 且SW3设置为SERVER, 版本号高于SW1/SW2, 那么再接上SW1/SW2, 此时由于SW3的版本号高于SW2/SW1, 所以SW1/SW2会和SW3同步, 形成VLAN信息的错乱

```
SW1/SW2 :vtp password 123
```

```
sh vtp password (查看密码)
```

此时, SW3再更改VLAN信息, SW1/SW2都不会和SW同步

STEP5:VTP pruning (vtp的修剪)

减少不必要的带宽利用

```
SW2: in range fa 0/1 -10
```

```
swit access vlan 10
```

```
sh in trunk
```

```
sh vlan brief
```

```
SW1: in range fa 0/1 - 10
```

```
swit access vlan 10
```

需求: 把多余的VLAN 修剪掉

VTP server上:

```
SW1: vtp pruning
```

STEP6: VTP的transparent/透明模式 图010

可以通过transparent交换机传输VLAN信息, 但transparent交换机的VLAN数据库不会改变, 且版本号不会改变, 用sh vrp status 查看将SW3配置成VTP transparent

SW3:VTP MODE transparent

删除VTP信息 delete flash: vlan.dat (做实验时必须删除, 否则实验无法进行)

## STP

spanning tree protocol /生成树(避免环路的有效方法)

技术背景:

当两个segment之间, 只有一个物理连接的时候, 就有可能出现“单点实效”

segment的3种概念:

- 1 在STP领域: 一段网络介质(网线/光纤)
- 2 在数据封装领域: 携带了4层包头的用户数据
- 3 在路由领域: 被三层设备(路由器)所分割的逻辑子网

避免单点实效性的方法: 构建冗余网络

冗余交换网络避免了网络的单点实效性/single points of failure

但是同时带来了3大致命问题/核心问题: 网络环路(危险等级轻微)

- 1 多帧复制/ multiple frame copies
- 2 MAC地址数据库的不稳定(危险等级中等)/mac database instability/或者叫“端口漂移”
- 3 广播风暴/broadcast storms (严重)

解决方案:

STP核心:

provides a loop-free redundant network topology, by placing certain ports in the blocking state

在冗余网络中, 通过STP算法, 将特定的端口置于阻塞状态, 来实现既没有环路, 也可以冗余的网络

BPDU: (bridge protocol data unit) 桥接协议数据单元

在交换网络中, 由根桥, 每2秒发一次, 用于STP的计算和收敛, 交换机是通过BPDU来实现既没有环路, 也可以冗余的目的

STP的四大原则(越小越好)

- 1 lowest root bid
- 2 lowest path cost to root bridge
- 3 lowest sender bid
- 4 lowest port id

STP的四大结论

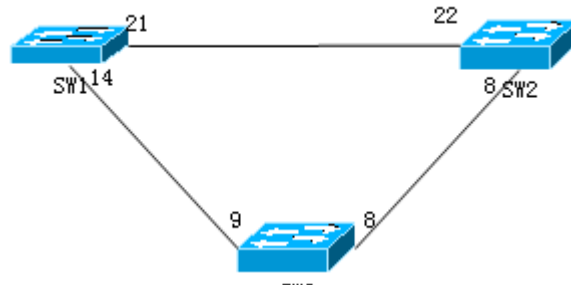
1:one root bridge per network 每一个交换(STP) one root port per nonroot bridge

LAB1: 根桥的选举

引用原则1:

lowest bid 那个交换机, 就是root bridge

BID:由本交换机的网关MAC地址, 和交换机的优先级组成, 一共8个字节



STEP1: 查看交换机的网管MAC地址

SW1:sh version 查看base ethernet mac address

SW2:sh version 查看MAC ad

SW3: . . .

注意: 交换机的STP的优先级, 默认都是0X8000 (化成十进制: 32768)

STEP2:sh spanning-tree 查看到SW3是根桥, 注意此时是默认情况下的, 也就是没有改变优先级的情况下 (MAC地址最小, 优先级最高)

用sh spanning-tree查看桥信息

root id/bridge id

root id 是指在本STP网络中, 根桥的BID

如果root id=bridge id, 说明本交换机就是根桥

STEP3: 人为控制, 根桥和后备根桥的选举

SW1(config): spanning-tree vlan 1-10 (选择1到10的VLAN) root primary (24576-0X6000)

SW2 : spanning-tree vlan 1-10 root secondary (28672-0X7000)

此时, SW1为根桥, SW2为备用根桥

sh spanning-tree查看

STEP4: 通过直接控制交换机的优先级, 来控制根桥

SW1(config): spanning-tree vlan 1-10 (选择1到10的VLAN) priority 0

SW2 : spanning-tree vlan 1-10 root priority 4096 (这个必须是4096的倍数)

2:one designated port per segment 除了根桥以外的所有的交换机, 都是非根桥

LAB2: 非根桥交换机的根端口的选择

的交换机，都是非根桥

**LAB2: 非根桥交换机的根端口的选择**

引用原则2: lowest path cost to root bridge

根端口: 在非根桥交换机上, 到达根桥所需的路径开销, 最小的那个端口

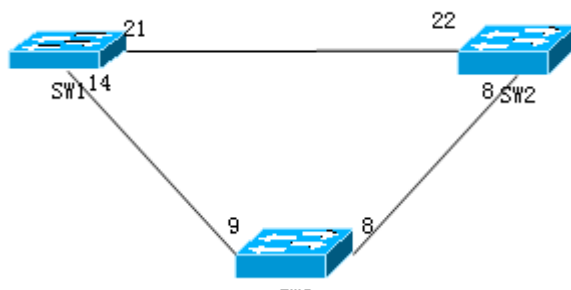
下图是带宽和开销对应表 (spanning-tree path cost)

| link speed | cost |
|------------|------|
| 10G        | 2    |
| 1G         | 4    |
| 100M       | 19   |
| 10M        | 100  |

sh spanning-tree 查看根端口

结论: 在根桥上, 没有根端口, 所有端口都是“指定端口” 所有端口

是指: 参与STP运算的端口, 通常是交换机与交换机相连的端口, 每个非根桥, 都只有一个根端口



STEP1: sh spanning-tree 查看根端口去往根桥的开销cost值

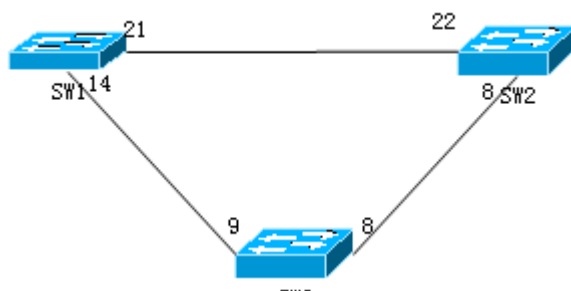
SW3: config-if-fa0/9 :speed 10 (mbps)

把9号口带宽改为10M, 那么开销为100, 那么端口8变为根端口, 那么到根桥的开销为38, 因为SW2到根桥的也是19 (见图009)

SW3: sh spanning-tree detail 可以查看到开销具体信息

**3: one designated port per segment**

LAB3: 每条segment上的, designated port 的选择 (图009)



引用原则3: lowest sender bid

指定端口的选择，可以手工指定 “指定端口” 的选择，通过修改优先级：

SW2: spanning-tree vlan 1-10 root priority 36846 (0X9000) 改大了

#### 4:nondesigned ports are blocked

LAB4: sender bid相同的情况下，designated port 的选择

如果sender bid 相同，则引用原则4 lowest port id (图011)

如果要想让SW2的21口作为根端口，那么更改port id, 更改SW1的22口调小

SW1:sh spanning-tree in fa 0/21 detail

观察到port identifier 128.21

SW1 (config-if22): spanning-tree port-priority 32

SW1:spanning-tree in fa 0/22 detail

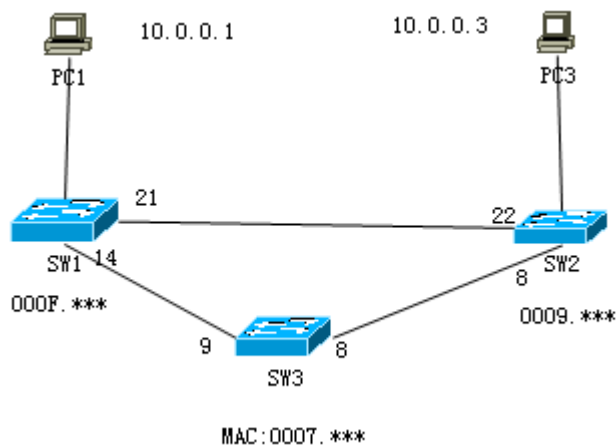
观察到port priority 32,port identifier 32.22

~~~~~  
~~~~~

Enhanced stp 增强型STP

LAB1: 观察STP的转发延迟计时器

图012



STEP1: 直链检测错误，经历35秒时间

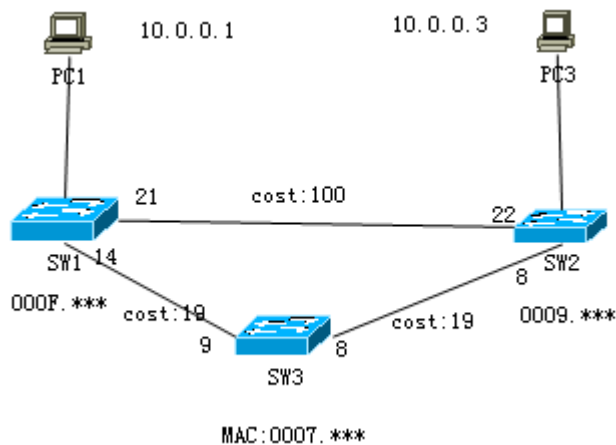
让SW1改为根桥，即把优先级改为最低

SW1: spanning-tree vlan 1 priority 0

然后PC3pingPC1的同时，SHDOWN掉SW2的FA22，因为SW2的FA8是阻塞的，他不是指定端口，所以DOWN之前是从SW2的 FA22口发送数据到PC1的

STEP2: 非直链检测错误，经历50秒时间

图013



FA22是阻塞的，因为SW1-SW2的链路带宽由原来的100M改为10M，那么SW1到SW2的链路cost值变为100，SW2到SW3的改为100M，SW1到SW3链路改为100M，那么cost值变为19，所以SW22口不是根端口，所以是阻塞状态PING的同时DOWN掉SW2的FA8后观察时间

## Enhanced stp

1 portfast 目的：主机断开后重新接上，交换机上能迅速ping到主机，无需等待50秒

只适用于，交换机与主机相连的端口，不应该在交换机与交换机，路由器，HUB互联的网络设备的端口使用

sh in status

in fa 0/1（交换机上，在连接主机的端口上配置）图014

switchport mode access

spanning-tree portfast



对于portfast端口，如果其接收到的BPDU包，意味这个本该连接主机的端口，确不能正确的连接到了交换机网络，意味这很可能已经形成了STP环路

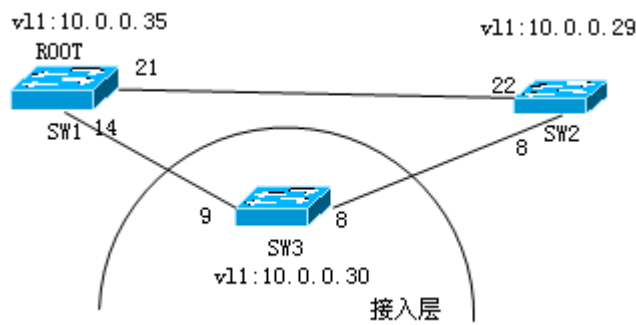
在STP算法判断出环路，并且将特定端口堵塞之前，已经立刻形成广播风暴，交换网络立刻瘫痪

结论：此方法不是最佳

2 uplink-fast （图015）

在所有接入层交换机上，配置uplink-fast，用于加速因为直链故障，所引起的STP网络变化的收敛速度

所引起的STP网络变化的收敛速度



配置后，DOWN掉SW1-SW3之间的链路后用扩展PING，在SW3PING 10.0.0.35，此时是没有启动uplink-fast的，观察重新通需要时间50秒，如果要减少这个等待时间，那么需要以下操作

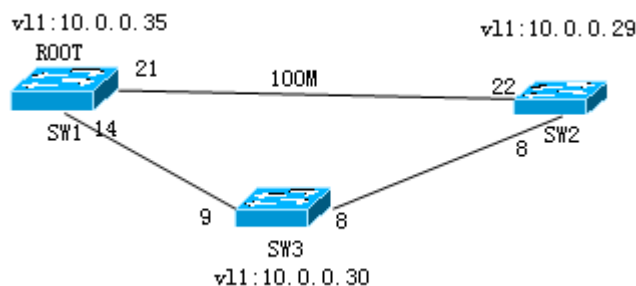
SW3: spanning-tree uplinkfast

然后再重复以上操作，观察等待时间5秒，其实就是跳过了 listening/learning 所用去的时间

3 backbone-fast 图016

在所有交换机上，配置backbone-fast，以利于全网的交换机，当遇到非直链测试错误时，加速其收敛速度

正常情况下，SW2 ping SW1时是从FA22出去，但如果SW1-SW2中断，那么就从SW3发送数据



测试：中断的同时，SW2 ping SW1，会中断50秒后才会PING通，解决此问题，如下操作

SW2/SW1/SW3: spanning-tree backbonefast

配置完成后重复以上操作，中断时间为35秒

4 portfast BPDUGuard

交换机端口的portfast BPDUGuard，是指：

在端口一旦接收到BPDU包时，立刻关闭端口，避免了更大范围的广播风暴，比1效果更好

SW3: in FA 0/9

spanning-tree bpduguard enable

sh in status

查看后，发现9号口已经DOWN掉，不会因为延迟而形成环路

SW3: spanning-tree portfast bpduguard default

此命令会在所有已经启动了portfast端口中，启动bgduguard, 见1

5 portfast bpdudfilter

全局的portfast端口上，都生效

SW3: spanning-tree portfast bpdudfilter default

bpdudfilter:don't send or receive BPDU on this interface

在特定的portfast端口上配置，那么，如下操作

SW3:conf-t-if spanning-tree bpdudfilter enable

以上的几种特性，都是CISCO私有的

## 802.1w Rapid STP (RSTP) 业界的开放性标准

RSTP的四种端口角色

1 Root Port (forwarding)

2 Designated port (forwarding)

3 Alternate port:收到别的交换机发来的BPDU，此端口是blocking

4 Backup port:收到本交换机发来的BPDU，此端口是blocking的

RSTP的3种端口状态

1 Discarding

2 Forwarding

3 Blocking

RSTP链路分类:

全双工链路来说为: point to point 链路

半双工链路来说为: share link

启动RSTP:

SW config spanning-tree mode rapid-pvst

其操作与STP大多是相同的

## Standard 802.1Q是CST (单生成树)

缺点:

所有的VLAN都是按照同一个STP来运行

所有的交换机都只维护一个STP实体

所有的VLAN的数据流量，都压向一个根桥的交换机

无法做基于VLAN的L2的负载均衡

优点:

对交换网络的开销较小

因为所有的VLAN都共享一个STP实体

那么与之对照的，是CISCO私有的STP MODE PVST (per vlan stp)

优点:

可以为每个VLAN，配置一个STP

不同的交换机，可以是不同VLAN的根桥

可以实现基于VLAN的，L2的负载均衡

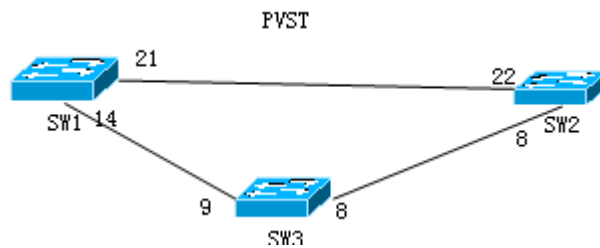
缺点:

交换机在维护很多的STP实体

对交换网络的开销比较大

一旦有任意一个VLAN的拓扑发生变化，都可能波及的交换机很多

**LAB3：基于以上对照，通过PVST，实现L2的VLAN间的负载均衡（图017）**



STEP1: 配置VTP

SW1/2:vtp mode server (SW1/SW2都设为SERVER, 互为备份)

SW1/2/3:vtp domain ccnp

SW3:vtp mode client

STEP2:在VTP server 上, 增加VLAN (vlan11-15)

SW1:VLAN 11.....

STEP3:SW1成为VLAN11-15的根桥, SW2成为VLAN21-25的根桥, 同时, 两个交换机要互为备份, 互为对方的备份根

确定两组VLAN的根:

SW1: spanning-tree vlan 11-15 root primary (0X6000)

SW2: spanning-tree vlan 21-25 root primary

两个交换机互为备份:

**SW1**: spanning-tree vlan **21-25** root secondary (0X7000)

SW2: spanning-tree vlan 11-15 root secondary

STEP4:在SW3上, 观察对于特定VLAN, 哪个是根端口、

SW2: sh spanning-tree

SW3: sh spanning-tree vlan 11 (9口转发, 8口堵塞)

21 (8口转发, 9口堵塞)

**802.1S Multiple spanning tree (MST) 业界的开放性标准**

MST就是对Standard 802.1Q/PVST (基于VLAN的生成树) 的折衷方案

MST就是基于 组/instance的STP (基于VLAN组的生成树)

MST的主要配置步骤:

1 MST首先对VLAN进行分组 (instance)

2 每个组都可以有独立的STP, 根桥, 实现了L2负载均衡, 冗余, 互为备份

**LA4: 通过MST, 实现L2的VLAN间的负载均衡。需求: 将两组VLAN化为两个组, SW1成为组1的根桥, SW2成为组2的根桥, 且互为备份 (图017)**

STEP1: 选定交换机的STP模式: MST

SW1/2/3: spanning-tree mode mst

STEP2:将VLAN划分到不同的组/instance中:

SW1/2/3:spaning-tree mst configuration

SW1/2/3:spanning-tree mst configuration

```
SW1/2/3 (conf-mst) :   instance 1 vlan 11-15
                        instance 2 vlan 21-25
```

STEP3:在核心交换机上，为不同的组，配置不同的优先级，实现两个组的互为备份/负载均衡

确定每个组的根桥：

SW1: spanning-tree mst 1 root primary

SW2: spanning-tree mst 2 root primary

互为备份：

SW1: spanning-tree mst 2 root secondary

SW2: spanning-tree mst 1 root secondary

STEP4:

```
SW1: sh spanning-tree mst configuration
      sh spanning-tree
```

FEC (fast ether channel) 捆绑带宽，使之变成几条带宽之和

SW1: in fa 0/21

```
      channel-group 1 mode on
```

SW2: in fa 0/21,22,23,24

```
      channel-group 1 mode on
```

所有参加channel-group的配置参数，必须一致，譬如：速率，双工模式  
配置完成后，交换机会自动生成一个虚拟接口，查看：

```
interface port-channel 1
```

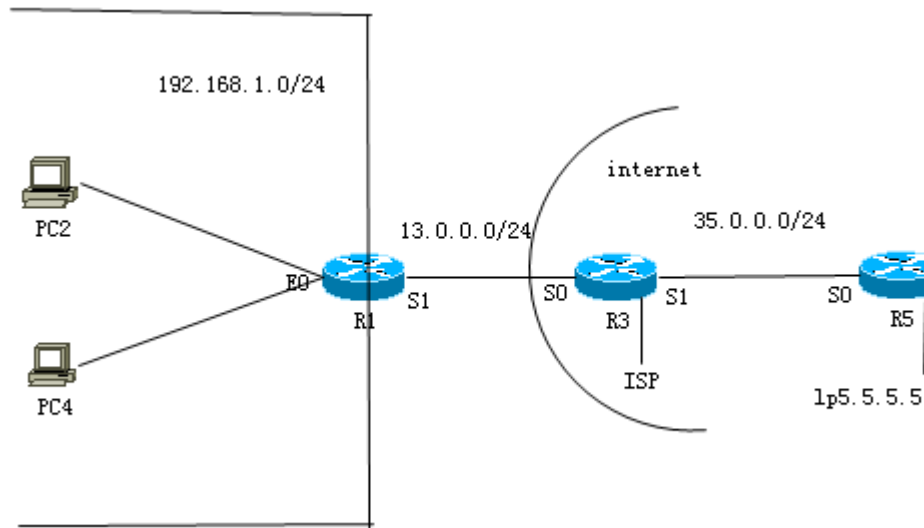
```
sh spanning mst 2(1)
```

变为了400M接口

## NAT

NAT (network address translation)

LAB1:NAT的基本配置（原理性的NAT）图018



STEP1: 网络背景: 使用R3模拟ISP, R3将与用户相邻的13.0.0.0网段重分布到IGP中, R5模拟公网上的路由器, 必须学到13.0.0.0的路由  
R1指一条默认路由到ISP (R3), 然后按图配置

R3/R5:RIPV2

R3: router rip

redistributed connected metric 1

把R1-R3直连路由重分布到RIP中

R1: ip route 0.0.0.0 0.0.0.0 13.0.0.3 (否则R1没有到达公网的路径)

R1 ping 5.5.5.5 OK!

STEP2:

R1 (conf t-e0) :no ip proxy-arp 关闭代理ARP, 否则PC是直接可以访问公网的, 达不到实验的目的

PC2: ip default-gate 192.168.1.1

PC4: ip default-gate 192.168.1.1 指定默认网关

此时, PC2/4 ping 5.5.5.5 不通的

STEP3: 定义NAT的外口和内口

R1:in e0

ip nat inside

in s1

ip nat outside

STEP4:静态的NAT转换

R1: ip nat inside (我方的) source static 192.168.1.2 13.0.0.10  
(在ISP那里买的公网地址, 我方的公网地址)

我方

所发起的

私网地址

公网地

址

STEP5: R1: sh ip nat translations (查看NAT的转换表)

R1: debug ip nat 查看

PC2再PING 5.5.5.5 OK!

然后在R1上debug ip nat的结果

发送: s=192.168.1.2->13.0.0.10 d=5.5.5.5

接收: s=5.5.5.5 d=13.0.0.10->192.168.1.2

结论: 静态NAT不实用, PC4也要做NAT, 如果用户多的话, 要购买相当多的NAT公网地址

LAB2:图018, 动态NAT, 也称为: PAT/NAPT/Overload

需求: 让所有用户通过R1的13.0.0.1做NAT

在中小企业, 通常使用基于接口的端口复用:

STEP1: 定义NAT的内口/外口 (通LAB1)

STEP2: 通过ACL, 定义准备进行NAT的用户群

R1: access-list 1 permit 192.168.1.0 0.0.0.255

STEP3:进行基于接口的NAT复用

R1: ip nat inside source list 1 interface s1 overload  
内网用户 NAT的外口

注意: 删掉LAB1中的静态NAT才能继续

测试, OK!

LAB3:图018, 使用route-map, 调用内网用户, 进行NAT转换, 如果用LAB2失败, 尝试用此实验

STEP1/2通LAB2

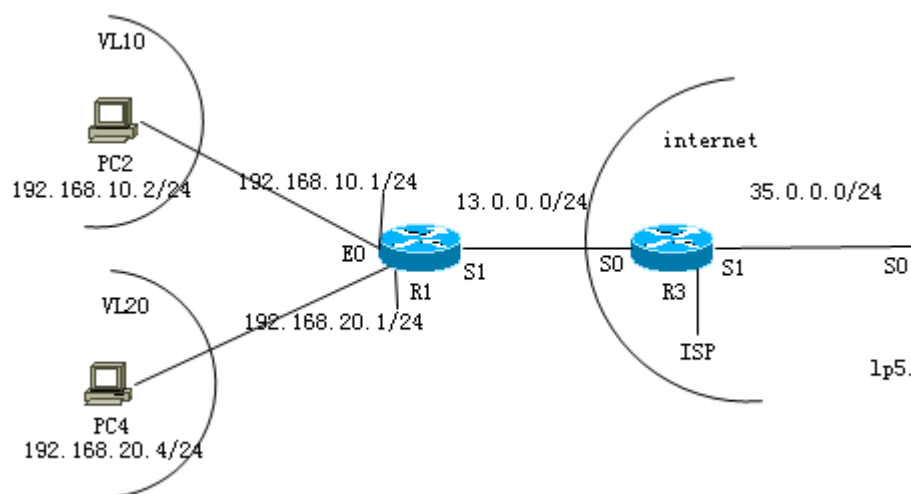
STEP3: 用route-map, 调用ACL1

R1:route-map NAT (自定义) permit 10  
match ip add 1

STEP4: 在NAT转换中, 调用route-map 所定义的用户群

R1: ip nat inside source route-map NAT interface s1 overload  
(注意, 先删除LAB2中的此语句)

LAB4:图019, 大型企业, 向ISP购买较多的公网IP后, 将其分别放入不同NAT-POOL, 以供不同部门进行NAT转换



STEP1: 定义NAT的外口/内口 (同LAB1)

STEP2: 在R1上模拟两个网段的网关:

```
R1-E0: in ad 192.168.20.1 secondary
        192.168.10.1 secondary
```

STEP3: 在两个网段/VLAN中的用户, 分别设定自己的网关

PC4-E0: ip ad 192.168.20.4

PC4: ip default-gateway 192.168.20.1

PC2-E0 : ip ad 192.168.10.2

PC2: ip default-gateway 192.168.10.1

STEP4: 通过ACL, 定义准备进行NAT的用户群

R1: access-list 10 permit 192.168.10.0 0.0.0.255

R1: access-list 20 permit 192.168.20.0 0.0.0.255

STEP5: 将从ISP处购买的公网IP, 放入地址池 (13.0.0.4---13.0.0.7给VLAN10用的, 13.0.0.8---13.0.0.11给VLAN20用的)

R1: ip nat pool pl-a 13.0.0.4 13.0.0.7 netmask 255.255.255.252  
(或者: prefix-length 30)

R1: ip nat pool pl-b 13.0.0.8 13.0.0.11 netmask  
255.255.255.252 (或者: prefix-length 30)

STEP6: 为不同的部门, 进行基于不同NAT-POOL的转换

R1: ip nat inside source list 10 pool pl-a overload

R1: ip nat inside source list 20 pool pl-b overload

debug ip nat 查看

STEP7: 将公网地址池与直连链路不一致的网段, pl-a的公网的地址池更改为: 203.13.5.0/29

R1: ip nat pool pl-a 203.13.5.1 203.13.5.6 prefix-length 29

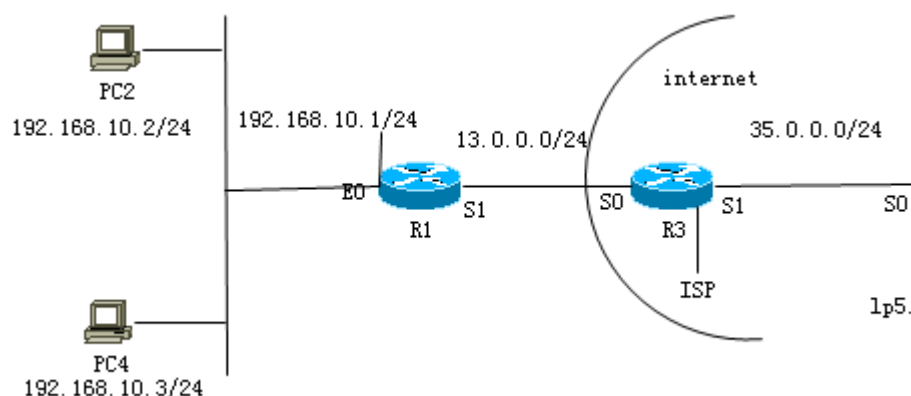
STEP8: 在ISP (R3) 上, 将公网地址池所在的网段通告到公网中

R3: ip route 203.13.5.0 255.255.255.248 13.0.0.1

R3: router rip

redis static metric 1 注意, 如果在ISP上指向用户一条静态路由, 那么必须把此条路由重分布到RIP里, 否则实验失败

LAB5: 图020通过NAT, 实现镜像服务器的负载均衡



需求：公网上能访问这两台服务器，R5访问这两台服务器时，负载均衡地到PC2/4

STEP1：定义NAT外口/内口（同LAB1）

STEP2：通过ACL定义“我方的公网地址”，做为来自外网访问的目标地址

R1: `access-list 10 permit 13.0.0.10 0.0.0.0`（精确匹配，只转换一个地址）

STEP3：定义内网的服务器群的地址池，一定要加上“轮转类型”

`ip nat pool pl 192.168.10.2 192.168.10.3 prefix-length 24 type rotary`

STEP4：定义

`ip nat inside(我方的) destination list 10 pool pl`(自定义)

STEP5：为了实验目的：

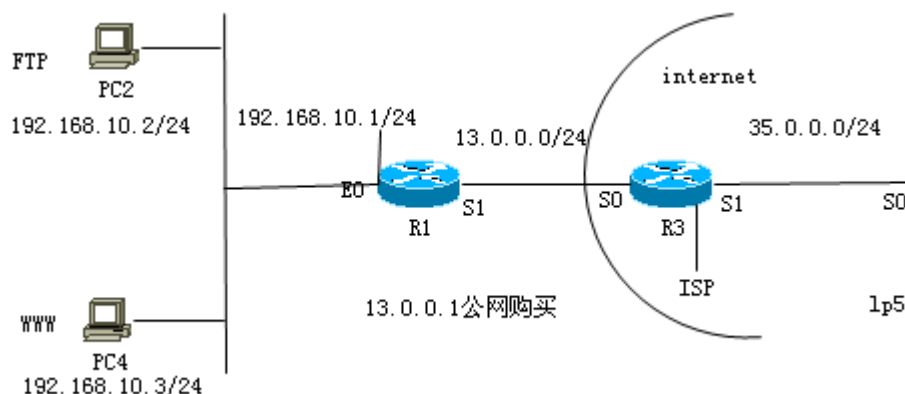
PC4: `line vty 0 4`

`no logging`（不需要密码，就可以被TELNET）

PC2同理

R5: `telnet 13.0.0.10` 可以看到是轮流

LAB6：图021使用一个接口的公网IP，对外网提供多个不同的服务器：如果你向访问我的FTP（TCP10000号端口或21口），WWW服务器（20000号端口或80口）



STEP1:

FTP服务器

R1: `ip nat inside source static tcp 192.168.10.2 21(端口) 13.0.0.1 21`

WWW服务器

R1: `ip nat inside source static tcp 192.168.10.3 80(端口) 13.0.0.1 80`

私网

公网

R1：为了实验，把私网端口改为23号端口，公网端口不要改

R1: 为了实验，把私网端口改为23号端口，公网端口不要改

R1: line vty 0 4

no logging

R5: TELNET 13.0.0.1 （无端口号）进入了R1

telnet 13.0.0.1 21 进入了FTP服务器

telnet 13.0.0.1 80 进入了WWW服务器

## 交换冗余

冗余网络

First hop routers on the LAN Redundancy Network/冗余网络

建立: Fault-tolerant/容错网络

避免: Single Points of Failure/单点失效

硬件冗余:

1 拓扑冗余

2 交换引擎冗余, 电源冗余, 线卡冗余, 风扇冗余, 线路冗余

软件冗余:

HSRP (RFC2281)

VRRP (RFC2383)

GLBP (GATEway load balancing protocol)

IRDP (RFC1256)

SRM (single router mode)

SLB (server load balancing)

Proxy arp (路由器使用代理ARP实现冗余): The client uses address resolution protocol (ARP) to get the destination it wants to reach, and a router will respond to the ARP request with its own MAC address

Routing protocol (主机使用路由协议实现冗余): The client listens to dynamic routing protocol updates (for example, from IGP (rip)) and forms its own routing table

IRDP (ICMP router discovery protocol) (主机使用IRDP实现冗余)

:client-the client runs an internet control message protocol (ICMP) router discovery client (网络的收敛性较慢, 而且受限于主机的操作系统)

重点HSRP: (Host standby router protocol)cisco私有  
the host standby router protocol (HSRP) is a first-hop redundancy protocol (FHRP) designed to allow for transparent fail-over of the first-hop IP router

HSRP provides high network availability by providing first-hop routing redundancy for IP hosts on ethernet, with a default gateway IP address

1 配置虚拟路由器

2 配置优先级

3 配置抢占路由器

## HSRP

### LAB1: 图032, HSRP

STEP1:在R1/2/3上, 运行RIPv2

R2/3上宣告192.168.1.0网络

注意no auto-summary

STEP2:在R2/3上, 配置NAT

2-1 使用ACL, 定义能够进行NAT的内网的用户群

R2/3:access-list 10 permit 192.168.1.0 0.0.0.255

2-2 定义路由器的内口, 外口

in e0

ip nat inside

in s0

ip nat outside

2-3 进行基于出接口的, 端口复用的NAT的转换:

R2/3:ip nat inside source list 10 interface s0 overload

STEP3:观察没有HSRP的网络实效的现象 (没有做冗余的网络实效的现象)

在PC4/5上:

PC4: ip default-gateway 192.168.1.3

PC5: ip default-gateway 192.168.1.2

扩展PING测试

模拟内网SHUT R2的E0口

没有冗余现象

STEP4:HSRP

4-1 在R2/3的内口:

R2/3:in e0

standby 1 ip 192.168.1.100 (定义虚拟路由器的IP)

standby 1 name G-1

standby 1 preempt (抢占)

R2: in e0

standby 1 priority 100 (standby) (默认为100, 越大越优先)

R3: in e0

standby 1 priority 103 (act)

4-2 让所有的用户, 都以虚拟路由器为默认网关

PC4/5: ip default-gateway 192.168.1.100

测试PING虚拟网关, OK!

4-3:

R2/3: debug standby (每3秒一个周期发送HSRP的HELLO包)

R2/3: show standby brief (查看HSRP的运行信息)

测试, R3 (active)的内口E0口断开, 观察active/standby的状态切换, 数据传送不受影响, 实现冗余

测试, R1 的外口s1断开, 那么不通, 那么:

测试，R1 的外口s1断开，那么不通，那么：

STEP5:跟踪外口

HSRP的路由器跟踪本机的外口，如果外口DOWN，那么就自动将优先级减

10

R3:in e0

standby 1 track s0 20 (外接口实效时，其优先级减少值定义为20，默认是10)

STEP6: 验证：

R3: in e0

standby 1 authentication cisco (HSRP认证)

standby 1 timers 5 15 (修改HSRP计时器，指定多长时间切换)

standby 1 mac-address \*\*\*\*\* (指定虚拟路由器的MAC)

### LAB2:图033，同个子网（VLAN）内的负载均衡

注意IP NAT定义外口和内口

STEP1: 定义standby

R3:in e0

ip ad 192.168.1.3 255.255.255.0

standby use-bia (在R2500上，启动多个HSRP组)

standby 2 name VR-A

standby 2 ip 192.168.1.100

standby 2 priority 100

standby 2 preempt

standby 2 track s0

standby 3 name VR-B

standby 3 ip 192.168.1.200

standby 3 priority 103

standby 3 preempt

standby 3 track s0

R2:in e0

ip ad 192.168.1.2 255.255.255.0

standby use-bia (在R2500上，启动多个HSRP组)

standby 2 name VR-A

standby 2 ip 192.168.1.100

standby 2 priority 102

standby 2 preempt

standby 2 track s0

standby 3 name VR-B

standby 3 ip 192.168.1.200

standby 3 priority 100

standby 3 preempt

```
standby 3 preempt
standby 3 track s0
STEP2:针对不同的用户群，定义不同的网关
192.168.1.150之前的用户，以VR-B为网关
192.168.1.150之后的用户，以VR-A为网关
PC4: 192.168.1.40 网关: 192.168.1.200
PC5: 192.168.1.250网关: 192.168.1.100
show standby brief 查看状态
```

### LAB3:不同子网（vlan）间的负载均衡（R2600）

```
RA:no sh
in e0.10
encapsulation dot1q 10
ip ad 192.168.10.1
standby 10 ip 192.168.10.100
standby 10 priority 105
standby 10 preempt
standby 10 name VL-10
standby 10 strack s0
access-list
ip nat inside...

in e0.20
encapsulation dot1q 20
ip ad 192.168.20.1
standby 20 ip 192.168.20.100
standby 20 priority 100
standby 20 preempt
standby 20 name VL-20
standby 20 strack s0

RB:no sh
in e0.10
encapsulation dot1q 10
ip ad 192.168.10.2
standby 10 ip 192.168.10.100
standby 10 priority 100
standby 10 preempt
standby 10 name VL-10
standby 10 strack s0
access-list
ip nat inside...

in e0.20
```

```
encapsulation dot1q 20
ip ad 192.168.20.2
standby 20 ip 192.168.20.100
standby 20 priority 106
standby 20 preempt
standby 20 name VL-20
standby 20 strack s0
```

#### LAB4:不同子网（VLAN）间的负载均衡（R3550）

3550A:

```
in vlan 10
ip ad 192.168.10.1 255.255.255.0
standby 10 ip 192.168.10.100
    priority 105
    preempt
    name VL-10
```

```
in vlan 20
ip ad 192.168.20.1 255.255.255.0
standby 20 ip 192.168.20.100
    priority 100
    preempt
    name VL-20
```

3550B:

```
in vlan 10
ip ad 192.168.10.2 255.255.255.0
standby 10 ip 192.168.10.100
    priority 100
    preempt
    name VL-10

in vlan 20
ip ad 192.168.10.1 255.255.255.0
standby 20 ip 192.168.20.100
    priority 105
    preempt
    name VL-20
```

#### VRRP

VRRP是业界的标准协议

使用224.0.0.18地址进行组播更新

LAB: 图035, vrrp冗余

R3:in e0

```
ip ad 1.3
```

```
vrrp 1 description VL-1
```

```
priority 105
```

```
preempt
```

```
ip 192.168.1.100
```

```
vrrp 1 authentication cisco
```

```
vrrp 1 timers advertise 2
```

## NP-REMOTE ACCESS

### RA(远程访问)

RA概述

Remote access

广域网的远程连接, 按L1分类:

1 通过电路交换网络实现的专线

1-1 通过真实的专用物理线路, 实现的专线: 一般是通过同步串行链路 (V.35), 其支持的二层封装协议主要包括: HDLC/PPP/SLIP

1-2 通过TDM网络(时分多路复用)实现的专线: 典型有: E1, DDN, (DDN, 可以直接在接口中配置IP地址即可) 其支持的二层封装协议与物理专线一样

2 按需拨号的电路交换网络 (on-demand circuit switched), 一般用于网络的链路备份

常见业务包括: ISDN/PSTN(其中PSTN通常是异步串行连接), ISDN也可以通过异步串行连接

3 包交换/分组交换 (packet switching)

在开始传输数据之前, 必需事先建立一条VC(虚电路), 用于数据包的传输, 通过同步串行链路连接ISP

常见业务: X.25/FR/ATM

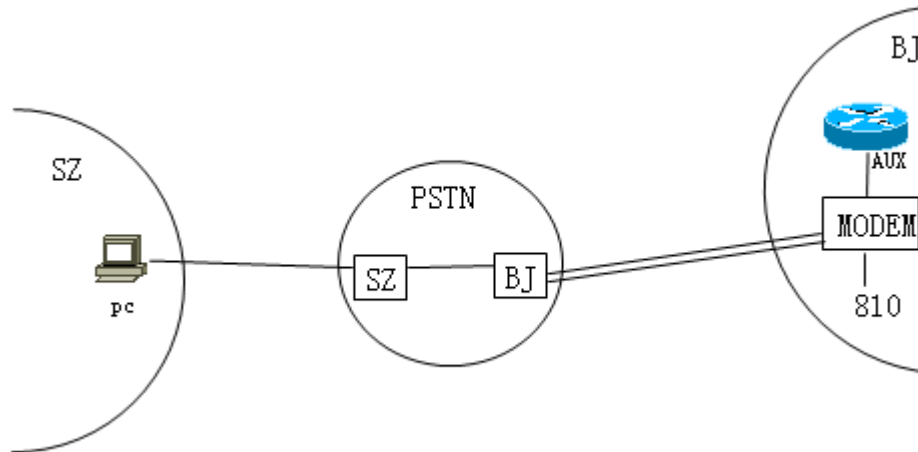
4 宽带接入/broadband access (未来的主流方向)

**xDSL:** 主要是传统的电信运营商所经营的网络, 主要使用传统的固化网(双绞线), 做为最后一公里的末端接入(语音+数据)

**Cable:** 主要是传统的有线电视运营商所经营的网络, 主要使用传统的

CATV网络，经过线路的双向改造，做为最后一公里的末端接入（视频+数据）

**LAB1：图022，通过异步MODEM/（PSTN 程控交换机/语音交换网/7号信令网/SS7），远程控制路由器的AUX接口**



STEP1：当人还在北京的机房中时，在路由器的AUX口进行基本配置，使用Console线控制Console口，目的：用于控制路由器的AUX口和MODEM之间的通信

1-1 RBJ: sh line

1-2 进入AUX口

RBJ: line 1 （绝对线路号）或者：

line aux 0（相对线路号）

1-3：配置AUX口

flowcontrol hardware 硬件流控

transport input all 在接口中传输所有协议

mode inout 可以进行入/出方向的呼叫

password 123 从AUX口进入路由器的密码

login 登录

1-4：配置进入特权模式的密码

RBJ: enable password wolf

STEP2：使用AT命令集，配置MODEM

笔记本的COM口，通过console线，RJ45专DB25的转换器，连接到MODEM的DB-25接口）

每个MODEM的厂商的AT命令集可能不一样

at 查看连接是否正常

at&v 查看MODEM的当前配置

at&f 或者：at&f1恢复出厂设置

at&s0=3 将MODEM的自动应答的响铃次数，设置为3，默认是0，不  
应答的

at&w 保存配置

at&w            保存配置

自动应答 (autoanswer/AA)

ate0            关闭字符回显

atel            打开字符回显

STEP3: 按图连好, 路由器AUX-反转线-MODEM-电话线-PSTN网

STEP4: 回到深圳, 使用深圳本机的MODEM, 拨北京的MODEM

建议使用超级终端, 通过长途电话, 拨北京的电话号码, 控制路由器

## HDLC:

HDLC一般不推荐, 原因有两个:

1 CISCO的HDLC帧头格式, 携带了一个CISCO的私有位

其好处: 实现在HDLC的环境中, 支持多协议: IP/IPX/AT/

其缺点: 只能跟CISCO的设备互通, 不能兼容各厂商设备, 原因: 标准的

HDLC只支持单协议

2 HDLC协议, 本身不支持认证, 无法保证安全性

建议使用PPP, PPP有多种可选模块, 可以提高网络安全性, 提升性能,

PPP可支持认证

SLIP, 相当于PPP的前身, 功能单一, 趋向淘汰

在CISCO的设备上, 串行链路默认使用HDLC

在华为的设备上, 默认使用PPP

## PPP (point-to-point protocol)

PPP是业界开放性的标准, 支持多协议环境, 所有的厂商都可以支持

HDLC不支持多协议和认证, PPP支持多协议和认证

LCP (link control protocol)

负责对L1的物理层的链路, 进行链路的建立、控制、维护

NCP (network control protocol)

负责对L3的网络层, 向下提供无差别的接口

async mode dedicated    通过路由器访问网页的封装

async mode interactive    访问改路由器的封装

PPP的四大模块

authentication

callback

compression

multilink

LAB1:图023, Encapsulation ppp    (从HDLC到PPP的迁移)

STEP1: 按图配置 (HDLC/被路由协议: IP, 寻路协议: RIP)

L1: V.35的同步串行传输

L2: HDLC

L3: IP/RIP

封装之前最好SHUT接口,封装后再NO SHUT

R1/R3: in s1

encapsulation ppp

debug ppp negotiation (PPP的协商)

出现如下提示说明正常:

1: interface s1, changed state to up

2: LCP: state is open

3: PPP的认证时可选项目, 只有PPP认证成功, 才有NCP的工作

4-1 sel ipcp : state is open

4-2 sel cdpcp: state is open

5: line protocol on interface s1, changed state to up

R3: sh in s0 可查看到

encapsulation ppp

LCP open

open: IPCP, CDPCP

测试: R2 ping R3是OK的, 说明封装不同 (PPP/HDLC), 也是可以互通的

### PPP authentication, 两种认证方式

PAP/CHAP (ppp的认证, 是链路的认证)

PAP: password authentication protocol

两次握手, 建议在网络工程中都使用双向认证

两次握手:

1 被认证方, 将对方所定义的账号密码以明文方式, 发送给主认证方

2 主认证方, 把收到的账号密码, 与自己的数据库进行核对后, 发回认证成功与否的信息

缺点: 账号密码是以明文方式在链路上传输, 不安全

### LAB2: PAP认证 (R1-R3)

STEP1: 确认链路已经封装为PPP链路

R1/R3 in s0

encapsulation ppp

STEP2: 在本路由器的数据库中, 为对方构建账号密码

R1: username BJ password BJ (对方的)

R3: username SH password SH

STEP3: 选定PPP的认证方式为PAP

R1 config-s1: PPP authentication PAP

R3 config-s0: ppp authentication pap

STEP4: 将“自己在对方数据库中的”账号密码, 发送给对方, 进行校验

R1 config-s1: PPP PAP sent-username SH password SH

R3 config-s0: PPP PAP sent-username BJ password BJ

debug ppp authentication (观察PPP认证过程)

PPP CHAP 认证 (challenge handshake authentication protocol)

三次握手:

1 主认证方的路由器, 发出的随机数

protocol)

三次握手:

- 1 主认证方的路由器, 发出的随机数
- 2 被认证方的路由器, 将接收到的随机数, 和事先定义好的密码, 一起放入MD5加密器, 进行HASH算法的加密计算, 把得到的数值, 以Response的形式, 发送给主认证方
- 3 主认证方, 同样进行第二步相同的操作, 将得到的数值Y', 与从被认证方发来的Y, 进行比较, 如果一致, 发出认证成功信息, 如果步一致, 发送认证失败信息

CHAP的优点: 不会在链路上传送密码, Challenge 和Response都是随机数, 这两者间是不可逆转运算的, 可以确保密码不被破译, 保证网络的安全性

### LAB3: 基于主机名的CHAP认证:

R1-R2启用CHAP认证

STEP1: 确认链路已经是封装为PPP链路

```
R1/R2 in s0
      encapsulation ppp
```

STEP2: 为对方创建账号密码

R1: username SZ password SS(对方的)

R2: username SH password SS

STEP3:

R1 config-s1: PPP authentication CHAP

R2 config-s0: ppp authentication chap

STEP4:将选定某一组账号密码进行CHAP认证

```
R2: config-s0 ppp chap hostname SZ
      ppp chap password SS
```

```
R1: config-s0 ppp chap hostname SH
      ppp chap password SS
```

STEP4:打开接口后观察

R2: debug ppp authentication

STEP5: 在CHAP认证中, 密码必需一致, 这是区别与PAP认证的

STEP6: 另外一种方法, 不使用特定的账号, 而直接使用路由器的主机名, 进行CHAP认证, 注意密码要一致

R2:username R1 password SS

R1 username R2 password SS

之前在接口模式下只需要以下操作:

```
in s0
encapsulation ppp
ppp authentication chap
```

PPP/MLP Multilink protocol(多链路捆绑) (2层冗余)

对比实验(3层冗余)(RIP), 因为此收敛速度受路由协议的收敛速率影响, 通常收敛比较慢

所以通过Multilink protocol实现2层冗余:

LAB5: 图023

STEP1: 将冗余的物理链路上的接口, 原有配置都删除, 但注意要在R3上的DCE端配置同步时钟

STEP2: 物理接口都封装PPP, 并且运行Multilink (无需配置IP地址)

R3/R5: in s0/s1

encapsulation ppp

ppp multilink

STEP3:在双方路由器上, 创建虚拟模板接口, 配置地址, 指定MLP

R5: in virtual-template 1

ip ad 35.0.0.2 255.255.255.252

ppp multilink

R3: in virtual-template 1

ip ad 35.0.0.1 255.255.255.252

ppp multilink

STEP4:在MLP中, 调用虚拟模板

R3/R5: multilink virtual-template 1

查看带宽:

R3:sh in virtual-access 1

internet address is 35.0.0.1/30

BW 3088 Kbit

R3/R5:RIP

sh ip route

PPP compression PPP压缩

PPP压缩可以提高PPP链路的带宽利用率

支持3种压缩算法:

1 mppc

2 predictor

3 stac

LAB6:PPP compression 图023

STEP1: 按图配置好IP, 确认通达

STEP2: 将R2-4链路改为PPP链路

STEP3: 在R2-4的PPP链路, 的接口种, 启用PPP压缩(3选1, 两端必需一致)

R2:in s1

compress mppc

R4:in s1

compress mppc

3-1 tcp头压缩: 语音的数据包, 其数据包的净荷很小, 头大身小的情况, 这种情况下使用以下压缩:

接口模式下做:

接口模式下做:

```
ip tcp header-compression
```

### ISDN(CCIE不考) 图RACK01图025

STEP1:

```
R3:isdn switch-type basic-ni
```

```
in bri 0
```

```
no sh
```

sh isdn status 查看到以下信息说明正常建立

```
layer 1 status
```

```
ACTIVE
```

```
layer 2 status
```

```
state=MULTIPLE_FRAME_ESTABLISHED(多帧已建立)
```

L2测试: 拨通对方路由器的电话号码

```
R3: isdn (testt) call interface bri 0 810888
```

```
R3:sh isdn active
```

查看ISDN当前连接状态

接状态

```
R5: isdn (test) disconnect interface bri 0 all (断开ISDN的连接)
```

### CISCO新版本种要用TEST

STEP2: L3通

所有的在ISDN接口种配置

逻辑命令:

2-4

```
R3: ip ad 35.0.0.3 255.255.255.0
```

```
dialer map ip 35.0.0.5 (broadcast) (需要穿越ISDN时才加)
```

```
810888 对方的L3地址与对方的L2地址映射
```

```
R5: ip ad 35.0.0.5 255.255.255.0
```

```
dialer map ip 35.0.0.3 810777 对方的L3地址与对方的L2地址映射
```

射

2-5 定义能够触发ISDN起拨的数据流

在全局:

```
R3: dialer-list 3 protocol ip permit (定义凡是IP数据包, 都能够触发ISDN起拨)
```

2-6 在接口中调用

```
R3: in bri 0
```

```
dialer-group 3
```

R5同理

测试ping

```
sh isdn active
```

STEP3:DDR, 按需拨号 (dial on demand route)

3-1 确保两端路由, 都有来去正确都路由

```
R3:ip route 5.0.0.0 255.0.0.0 35.0.0.5
```

```
R5:ip route 10.0.0.0 255.0.0.0 35.0.0.3
```

R5:ip route 20.0.0.0 255.0.0.0 35.0.0.3

3-2 确认上述路由的下一跳的可达性

确认有到达“路由下一跳”的映射

R3: dialer map ip 35.0.0.5 810888

R5: dialer map ip 35.0.0.3 810888

show dialer map 查看“路由下一跳”的映射

3-3 修改“能够触发ISDN起拨”的数据流（定义“感兴趣流”）

收窄“能够触发ISDN起拨的数据包的”范围

3-3-1: 通过ACL，定义能够触发ISDN起拨的数据包

access-list 10 permit 10.0.0.0 0.255.255.255

3-3-2: 通过dialer-list, 调用ACL10

dialer-list 3 protocol ip list 10

3-3-3: 在接口中，调用dialer-list 3

in bri 0

dialer-group 3

ISDN的。。。没记完，考

6226 2206 0012 3169

## ISDN

ISDN

ISDN的增强配置:

LAB 4: Dial Profile (多个拨号的配置文件) 图1 图2 图 3

Step1: 将物理接口,放入虚拟的dialer pool中:

R3(config)#in b0 (物理接口只配置物理命令)

R3(config-if)#dialer pool-member 10

step2 根据备份链路的需要,创建虚拟的dial口(dial Profile)

R3(config)#interface dialer 5 (逻辑命令都配置在dialer口)

R3(config-if)#dialer pool 10 (让if-dial 5 到pool 10 中,寻找物理接口)

Step3: 将原来在物理接口中的所有配置命令,  
进行基于物理命令/逻辑命令的分类:

物理命令:

Encapsulation ppp

Isdn switch-type basic-ni

逻辑命令:

Encapsulation ppp

Ip address 35.0.0.3 255.0.0.0

Dialer-group 3

Ppp authentication chap

Step4: 在物理接口中,调用物理命令  
在逻辑接口中,调用逻辑命令

Interface Dial 5

无法在dial 接口中,中应该换成:

```
Dialer string 810888
```

ISDN 的的备份: 图4

LAB1: 浮动静态路由的备份:

Step0: 感兴趣流: dialer-list 3 protocol ip permit

设置是永不断线:

```
R5(config)# In bri 0
Dialer idle-timeout 0
```

Step1: 在所有主干链路上运行ICP (RIP/EIGRP/OSPF)

Step2: 在ISDN链路上, 不运行动态路由协议,  
而是通过静态的浮动路由, 实现链路备份:

注意路由长度问题:

只有备份链路的静态路由长度,  
比主链路的尝到的动态路由要短,  
那么肯定是可行的!

```
R3(config)# ip route 5.0.0.0 255.0.0.0 35.0.0.5 125 (OK)
R5(config)# ip route 34.0.0.0 255.0.0.0 35.0.0.3 125
R3(config)#ip route 5.0.0.0 255.0.0.0 35.0.0.0 50 (OK)
```

只有备份链路的静态路由长度,  
比主链路的尝到的动态路由要长,  
那么肯定是错误的!

```
R3(config)#ip route 5.5.5.0 255.255.255.0 35.0.0.0 50 (NO OK)
B-这是错误的
R3(config)#ip route 5.5.5.0 255.255.255.0 35.0.0.0 125 (no OK)
B-这是错误的
```

注意路由AD问题:

只能在备份/主链路的路由长度相同的情况下, 才需要考虑AD:

```
R3(config)#ip route 5.5.0.0 255.255.0.0 35.0.0.5 50 (NO OK)
B-这是错误的
R3(config)#ip route 5.5.0.0 255.255.0.0 35.0.0.5 125 (OK)
```

LAB2: 通过" Backup-if" 的技术, 实现基于主链路接口物理层检测的备份:  
图5

(" Backup-if" , 不管是否有感兴趣流, 不管有没有路由,  
只要主链路接口的物理层down , ISDN就立刻起拨)

Step0: 感兴趣流: dialer-list 3 protocol ip permit

Step1: 在R3/R5之间, 不需要静态路由,  
可以

在ISDN链路上运行动态路由协议 (IGP):

```
Router rip
  Network 35.0.0.0 (ISDN)
```

Step2:

物理接口:

```
R3(int-bri 0)#dialer map ip 35.0.0.5 broadcast 810888
```

逻辑接口:

```
Interface dialer 5 (默认就支持广播通过)
Dialer string 810888
```

Step3:

```
R3(config)#in s 1 (在主链路接口中, DDN链路)
```

```
R3(config-if)#backup interface bri 0
                (ISDN链路作为DDN的备份链路)
```

```
Interface serial 0
  Description T0 BJ
  Backup interface dialer 200
```

```
Interface serial 1
  Description T0 SZ
  Backup interface Dialer 5
```

```
R3#show interface dialer 5
Dialer 5 is standby mod (spoofing)
```

为什么感兴趣流可以是:

Dialer-list 3 protocol ip permit ?

一旦ISDN作为DDN的备份接口,

ISDN接口就会down 下去, 不会因为感兴趣流而起拨.

Step4:

```
R3(config-if)#backup delay 5 15
```

当主链路断开5秒后, 才开始拨号;

当主链路恢复15秒后, 才断开(shutdown) ISDN

```
R3(config-if)#backup load 50 20
```

当主链路的带宽利用率高于50%, 启动ISDN进行负载均衡;  
当主链路+备份链路的总带宽利用率低于20%, 断开ISDN.

注意: 在真实情况下, 应该备份链路的两端都” backup-if”

LAB 3: 通过 “dial-watch-list”, 监控某条路由, 实现备份: 图5  
不依靠对主链路接口物理检测

Step1: 在ISDN链路上, 启用IGP

Step2: 修改感兴趣流:  
因为RIP/EIGRP的组播包不断通过ISDN,  
所以ISDN永远无法断开

2-1:

RIP修改为: (R5同样也执行)

```
R3(config)#access-list 100 deny udp any any eq 520
```

```
R3(config)#access-list 100 deny udp any eq 520 any
```

```
R3(config)#access-list 100 permit ip any any
```

EIGRP修改为:

```
Access-list 101 deny digrp any any
```

```
Access-list 101 permit ip any any
```

2-2: dialer-list 3 protocol ip list 100 (调用ACL 100)

2-3:

```
Interface bri 0
```

```
Dialer-group 3
```

Step3: 使用” dial-watch-list”, 监控某条路由:  
一旦这条被监控路由消失, 立刻起拨ISDN

```
R3(config)#dialer watch-list 55 ip 5.5.0.0 255.255.0.0  
(R3监控5.5.0.0 /16 这条路由)
```

Step4: 在R3的ISDN接口中调用” dialer watch-list 55”

```
R3(config)#in dial 5
```

```
R3(config-if)#dialer watch-group 55
```

Step5: R5所监控的那条路由, 也要映射到对方的ISDN电话号码:

```
R3(config-if-bri 0)#dialer map ip 5.5.0.0 810888
```

LAB4: OSPF 的Demand-Circuit (ISDN链路也在运行OSPF)

Step1: 感兴趣流是所有的IP包:

```
R3/5(config)#dialer-list 3 protocol ip permit
```

Step2: 在所有链路上, 都运行OSPF

Step3: 在ISDN链路的两端, 都配置

```
R3(config)#in b0
```

```
R3(config-if)#ip ospf demand-circuit
```

OSPF的demand-circuit 是在OSPF路由出现抖动时,  
ISDN会自动拨号, 如果没有后续的数据包通过, ISDN超时后, 会断开

## FR

### FR(帧中继), 帧中继是纯2层协议

帧中继特征:

1 面向连接的服务:

必须在通信开始之前, 通过VC(虚电路)构建好, 通信双方之间的虚拟通道

PVC: 永久虚电路, 长期连接不中断的

SVC: 交换式虚电路, 在用户需要连接的时候, 才临时构建起来的

2 帧中继用户, 是充当FR的DTE端, 而ISP是充当FR的DCE端 (FR Cloud)

NNI (network network interface) 图026

3 Fram Relay Signalling/帧中继的信令:

FR用户通过LMI (Local Management Interface)本地管理接口, 跟FR SW进行协商, 获得PVC的相关信息

LMI的3种标准

1 ANSI

2 Q933a

3 CISCO兼容的

标准不同也可以互通, LMI只是本地意义

4 通过LMI的状态, 判断FR PVC的故障点, 图027

5 FR的L3的IP地址, 与L2的DLCI地址的映射关系:

对方的L3的IP地址, 与本地的L2的DLCI进行映射

FR实验:

**LAB1: 图028, 基本的帧中继交换机, 以及基于每个物理连接, 一条PVC的Hub&Spoke架构:**

STEP1: 按图配置, 注意R2/R3关闭路由功能和启用帧中继功能

STEP2: 为接口配置同步时钟

STEP3: FR接口的基本配置

```
SW2/SW3 IN S0/S1:en fram-relay
```

```
SW2/SW3 IN S0/S1:en fram-relay
                        fram lmi-type cisco
                        fram intf-type dce
```

STEP4:配置FR路由

```
SW2:in s1 (在PVC的入口配置)
      fram route 401 in s0 104
      in s0
      fram route 104 in s1 401
```

STEP5:在用户端封装FR，然后把用户和ISP的端口都打开

```
R1/R4:in s0
      encapsulation fram
```

查看信息

```
sh ip int brief
serial0 L1:UP L2:UP
```

```
R1:show fram pvc
```

STEP6:L3测试

在一条PVC上的用户接口种，配置同个网段的IP地址

按图配置IP地址

show fram map有如下信息

ip 100.0.0.2 dlci 104 对方的IP地址和本方的DLCI的映射

dynamic:通过FR的自动反向ARP学习到的

broadcast:L2的FR PVC允许广播或组播流量通过

active:PVC工作状态是正常的

同理配置SW3/R1/R3

图028可以理解为图029

此时，在R1/4/5上建立LP，运行RIPV2

可以看到，路由学习和运行都是正常的

STEP7:结论，在此实验中不存在水平分割所导致的路由问题（无3层问题），因为，不是同个网段，同时，也不存在有路由而没有FR映射导致的L2映射问题，但是最大的问题是设备成本太高，FR需要两个，本地端口需要两个

**LAB2:图031，3端口的FR（比LAB1成本降低），FULL-MESH**

STEP1:按图配置

STEP2/3/4:同LAB1

STEP5:由于以太网不运行帧中继，所以在SW2/3之间的E口构建虚拟管道

tunnel（加上以太网包头），以实现实验目的

在SW2和SW3之间的以太口，配置IP，然后构建tunnle管道

STEP6:配置R4到R5的PVC的FR路由

首先对接tunnel管道

```
SW2:in s1
      fram route 405 in tunnel 1 1000
```

```
SW3:in s1
      fram route 504 in tunnel 1 1000
```

```
SW2:in s0
    fram route 105 in tunnel 1 1001
SW3:in s1
    fram route 501 in tunnel 1 1001
同理：配置104/401PVC
ISP(SW)上查看FR路由
sh fram-relay route 信息如下：
tunnel 1 1000 s1 504 active
serial 1 501 tunnel ...
测试，互通！！！！！！！！
```

### LAB3:图031，IGP在FR上的运行

通过FR的自动反向ARP，FULL-MESH的PVC，IPG网络（RIP/EIGRP/OSPF）

FULLMESH：一般在核心网中用FULLMESH  $n*(n-1)/2$

接LAB2

STEP1:

1-1 RIP over fullmesh FR PVC 网络（RIP V2 no-autosummary）

R1/4/5运行RIP，注意LP口也宣告进RIP

因为FR-SW允许广播通过，所以R1/4/5都可以相互学到路由

但R1/4/5ping自己的IP是不通的

debug ip packet

看不到自己的映射，所以PING不通，如果需要PING通自己的IP，那么需要手工做一条静态映射，如下：

```
R5:S0:fram map ip 100.0.0.5 501(504)
```

1-2 EIGRP over fullmesh FR PVC 网络

R1/4/5宣告EIGRP，注意no auto-summary

测试：OK！

1-3 OSPF over fullmesh FR PVC 网络

如果想看到LP准确长度

```
in loop 0
```

```
ip ospf network point-to-point
```

观察路由都没有互相学习到，那么

```
sh ip ospf nei （没有邻居）
```

```
sh ip ospf in
```

观察接口的OSPF运行模式，为NBMA，默认是不主动发送组播的HELLO包的，但L2FR是允许通过广播包的，只是在FR中OSPF接口默认模式是NBMA的，所以不能互相学习到路由，且互相不能建立邻居关系

解决方法：将OSPF的接口运行模式，从NBMA改为Broadcast

R1/4/5

```
in s0
```

```
ip ospf network broadcast
```

此时观察邻居，可以成功建立

```
show ip route ospf
```

```
show ip route ospf
show ip ospf database查看
测试: OK!
```

#### LAB4:使用物理接口, 通过静态映射构建IGP网络

STEP1:关闭FR的自动反向映射 (在CCIE考试中, 必须关闭)

在较早的IOS版本中, 只在主接口(物理接口)中, 有自动反向映射功能  
在较新的IOS版本中, 主接口和子接口, 都有自动反向映射功能, 也就是说都要关闭反向映射

```
R1(config-s0): no fram-relay inverse-arp
```

```
R1:clear fram-relay inarp (清除FR MAP的Cache)
```

STEP2: 在接口中, 根据题目允许使用的PVC, 建立FR静态手工映射

```
R1:in s0
```

```
    fram map ip 100.0.0.4 104 broadcast
```

```
    fram map ip 100.0.0.5 105 broadcast
```

STEP3: IGP都实验一下, 与LAB4相同

FR的子接口:

子接口的判断原则:

1:判断创建几个子接口:(一个子接口对应一个子网)

在路由器的一个物理的FR接口中, 如果对应着几个子网就应该创建几个子接口

2:判断子接口的类型:(P2P/P2MP)

如果一个子接口中, 对方是只有一个点, 那么接口类型就是P2P

建议:(一条PVC对应一个IP子网)

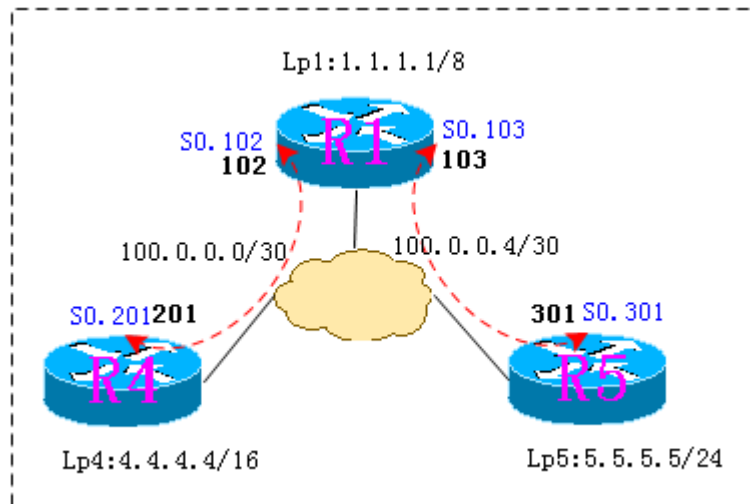
Hub&Spoke/星型网络拓扑, 有多少个分支点/分公司, 就应该创建多少个点对点子接口

如果一个子接口中, 对方是多于一个点, 那么接口类型就是P2MP

建议:多点子接口只是用于FullMesh网络拓扑

#### LAB5:通过将Hub&Spoke的FR网络每个分支点的PVC作为独立的一个子网/30

构建P2P的子接口(作为Hub&Spoke拓扑的最佳解决方案)



step1:配置FR的主接口

```
in s0
no ip address(无需配置IP)
encapsulation frama-relay
no frama-relay inverse-arp
no shutdown
```

step2:配置FR的子接口

```
R1(config)#interface serial 0.102 point-to-point
R1(config-subif)#ip address 100.0.0.1 255.255.255.252
R1(config-subif)#frama-relay interface-dlci 102(该点对点子接口
所使用的PVC本地DLCI号)
```

```
R1(config)#interface serial 0.103 point-to-point
R1(config-subif)#ip address 100.0.0.5 255.255.255.252
R1(config-subif)#frama-relay interface-dlci 103
```

R4/R5同理

step3:观察FR的映射

```
R1#show frame-relay map(P2P默认是允许广播通过)broadcast
serial0.102 (up): point-to-point dlci, dlci 102
在点对点子接口中,对本子接口的IP无需映射也可以通达
(主接口/多点子接口都需要对自己的IP进行映射)
```

step4:IGP over FR Hub&spoke 点对点网络

4-1:RIP over FR Hub&spoke 点对点子接口,FR网络

有全路由(Full Route),能够全面通达

关于DV协议的水平分割问题:(RIP/IGRP)

在FR的子接口中,不论点对点还是点对多点子接口其水平分割都是启  
动/Enable的

```
R1#show ip interface brief s0.102/103
Split horizon is enable
(而FR物理接口/主接口的水平分割默认是关闭的)
```

#### 4-2:EIGRP over FR Hub&spoke 点对点网络

检查

```
1:show ip eigrp interface(查看接口是否在运行EIGRP)
2:show ip eigrp neighbors(查看EIGRP的邻居是否已经)
3:show ip eigrp topology(查看eigrp的拓扑表是否已经收敛)
4:show ip route eigrp(查看从eigrp学到的路由)
```

#### 4-3:OSPF over FR Hub&spoke 点对点网络

show ip ospf interface S0.102/103 (观察当前运行ospf的接口的运行模式/Network Type是Point-To-Point)  
FR P2P子接口的默认运行模式是Point-To-Point, 允许广播通过  
∴ospf邻居可以建立

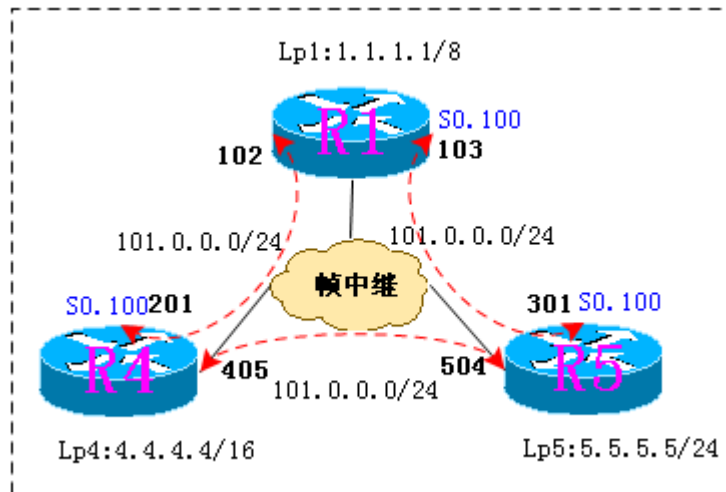
step5:关于本实验中的所有的分支点的下一跳

```
R4# any route via 100.0.0.1
```

```
R5# any route via 100.0.0.1
```

∴所有分支点之间的路由的下一跳都是可达的.

LAB6:使用多点子接口(MultiPoint Sub-if)构建FullMesh的FR PVC网络  
通过静态映射构建IGP网络(RIP/EIGRP/OSPF)



step1:主接口

R1:

in s0

no ip address(无需配置IP)

encapsulation frama-relay

```
encapsulation frama-relay
no frama-relay inverse-arp
no shutdown
```

step2:创建子接口(R1)

```
R1(config)#in s0.100 multipoint
R1(config)#ip address 101.0.0.1/24
R1(config)#no fram-relay inverse-arp
R1(config)#frama-relay map ip 101.0.0.4 102 broadcast
R1(config)#frama-relay map ip 101.0.0.5 103 broadcast
R4/R5同理做映射
R4(config)#in s0.100 multipoint
R4(config)#ip address 101.0.0.4/24
R4(config)#no frama-relay inverse-arp
R4(config)#frama-relay map ip 101.0.0.1 201 broadcast
R4(config)#frama-relay map ip 101.0.0.5 405 broadcast
```

```
R5(config)#in s0.100 multipoint
R5(config)#ip address 101.0.0.5/24
R5(config)#no frama-relay inverse-arp
R5(config)#frama-relay map ip 101.0.0.1 301 broadcast
R5(config)#frama-relay map ip 101.0.0.4 504 broadcast
```

step3:RIP Over FullMesh FR

虽然每个子接口的水平分割都是enable的,

但∴全网都是FullMesh

∴每个路由器都有全路由, 而且路由的下一跳都是可达的所以可以全网通达

step4:EIGRP Over FullMesh FR

情况与RIP相同

step5:OSPF Over FullMesh FR

特别注意:

在OSPF中, FR物理接口和多点子接口其OSPF的运行模式默认是NBMA

```
R5#show ip ospf interface serial 0.100
```

```
Network Type NON_BROADCAST
```

∴无法建立邻居!

step6:更改OSPF的运行模式为:**Broadcast**

```
R1/4/5(config)#interface s0.100
```

```
R1/4/5(config-subif)#ip ospf network broadcast
```

step7:查看OSPF邻居/DR/路由/下一跳:

L2 FR允许广播流量通过, 而L3的OSPF又主动发送组播的Hello包

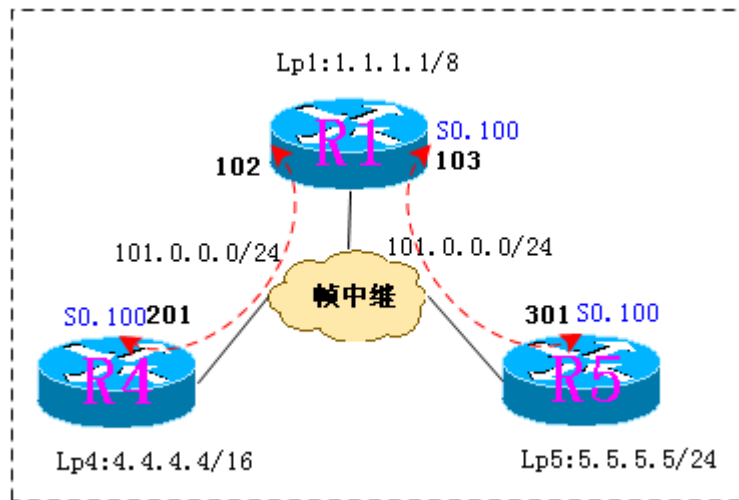
所以OSPF邻居能够自动建立.

OSPF协议能够自动正确选出DR/BDR

路由的下一跳都是可达的

LAB7:使用多点子接口(MultiPoint Sub-if)构建

Hub&Spoke的FR网络, 通过静态映射构建RIP网络



step1:

R1#show ip interface brief s0.100

Split horizon is enable

step2:关闭水平分割

R1(config-if-s0.100)#no ip Split-horizon

step3:

R5# 4.4.0.0 via 101.0.0.4

R4# 5.5.0.0 via 101.0.0.5

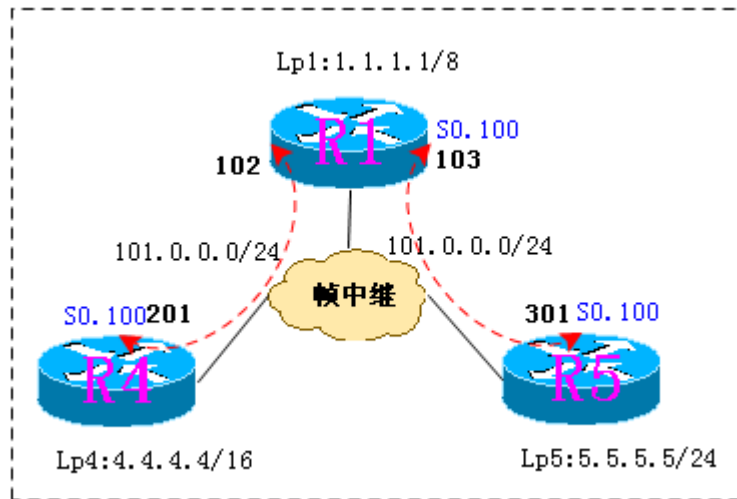
step4:为了解决路由的下一跳问题, 在分支点上做别的分支点的映射

R4(config-subif)#frame-relay map ip 101.0.0.5 201

R5(config-subif)#frame-relay map ip 101.0.0.4 301

LAB8:使用多点子接口(MultiPoint Sub-if)构建

Hub&Spoke的FR网络, 通过静态映射构建EIGRP网络



step1: EIGRP的水平分割与普通DV协议是不同的:

R1(config-if-s0.100)#no ip Split-horizon eigrp 90

step2:

R4/5两分支点间的用户之间的通信无需做映射:

R5# 4.4.0.0 via 101.0.0.1

R4# 5.5.0.0 via 101.0.0.1

在老马的印象中不需要做映射, 但ping不通! 需要自己去验证!

R4#ping 5.5.5.5 source 4.4.4.4 !!!!!

R4#ping 5.5.5.5 .....

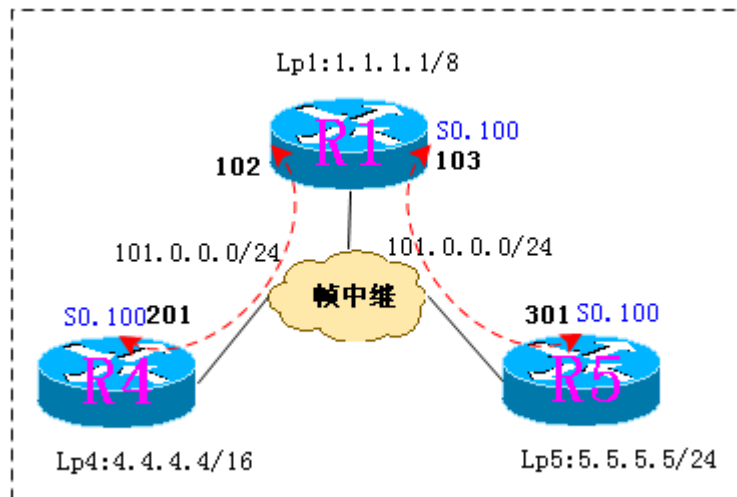
访问分支点的外口还是需要做映射的:

R4(config-subif)#frame-relay map ip 101.0.0.5 201

R5(config-subif)#frame-relay map ip 101.0.0.4 301

LAB9: 使用多点子接口(MultiPoint Sub-if)构建

Hub&Spoke的FR网络, 通过NBMA模式构建OSPF网络



step1: 邻居问题

1-1: L2 FR要 可以不允许广播流量通过

```
frama-relay map ip 101.0.0.4 102(broadcast)
```

1-2:在中心点单播建邻居

```
R1(config)#router ospf 110
```

```
neighbor 101.0.0.4
```

```
neighbor 101.0.0.5
```

step2:DR问题

R4上: 自己是DR R1是BDR

R5上: 自己是DR R1是BDR

R1上: 自己是BDR, R5是DR, R4是DROTHER

解决:通过OSPF的接口优先级控制中心点一定成为DR, 没有BDR

```
R1(config-subif)#ip ospf priority 10
```

```
R4/5(config-subif)#ip ospf priority 0
```

step3:下一跳问题

为了解决路由的下一跳问题,在分支点上做别的分支点的映射

```
R4(config-subif)#frama-relay map ip 101.0.0.5 201
```

```
R5(config-subif)#frama-relay map ip 101.0.0.4 301
```



## NP+

### RIP

内网RIP使用默认版本，不可用静态

- 1 R3不向R4通告更新（20分）
- 2 R1是连接到INTERNET的路由器，R1向内网通告一条默认路由，以上INTERNET的配置不在RIP范围，不用配置（10分）
- 3 timer在R4上: update 5s, invaldid 10s, holdown 20s, 没收到update 15s后被删除（flash:15s）20分
- 4 在删除R1上发出的缺省路由后，R1与R4之间分别连接主机H1, H2不可达，请解决
- 5 R1的s0口上级联一个RIP邻居后，要求仅在s0口上发送和接收RIPv2更新，并使用MD5加密，密码wolf
- 6 R1向R2发送10.0网段明细，要求R3、R4可PING通R2上的地址（5分）
- 7 不可修改R2任何配置，要求在R1上只允许R2成功直接telnet到R1，并直接可到enable模式（无需密码，无需输入“en”命令）R2使用2.2.2.2发出telnet，其他路由器访问被拒绝（10分）
- 8 要求R3收到R1发过来的199.172.\*.0路由，奇数metric为10

默认路由

第二地址

times的修改:update invalid holdown flash

端口发送和接收V1或V2

加密MD5

TELNET访问无需输入密码直接可以TELNET (line vty 0 4 /no login)

### EIGRP

EIGRP的STUB路由器

STUB后面的5个参数

connected

redistributed

receive-only

summary (注意只发送汇总路由的概念, 是在STUB路由器做出接口的汇总, 从而影响到邻居路由器的路由表)

static

不等价负载均衡:

show ip eigrp topology

P 1.1.1.0/24, 1 successors, FD is 409600

via 100.1.1.1 (409600/128256), Ethernet0

via 12.1.1.1 (2297856/128256), Serial0

$2297856/409600=5.61$ , 那么:

router eigrp 100

variance 6

来实现不等价负载均衡

show ip route 1.1.1.0 查看两条路径包走向的个数, 旧版本按上面的操作那么包比例为6:1, 新版本的是240: 43

EIGRP的抑制时间更改命令:

对应所有接口上 ip hold-time eigrp \*

实验列表

- 1 和IGRP的重新分配
- 2 关闭自动路由汇总
- 3 对K值的设置
- 4 EIGRP的 TTL值的研究
- 5 HUB/SPOKE模式下启用EIGRP
- 6 匹配EIGRP的所有流量
- 7 不等价负载均衡
- 8 修改EIGRP的计时器
- 9 EIGRP占用带宽
- 10 EIGRP stub
- 11 distr-list pre-list, defau-metric等命令的应用
- 12 passive接口对EIGRP的影响, 单播更新需要注意的地方
- 13 EIGRP域内公布默认路由
- 14 EIGRP的第二地址与建立邻居学路由
- 15 traffic-share和variance
- 16 认证 (多加验证) 只支持MD5
- 17 EIGRP的AD

## 17 EIGRP的AD

### OSPF

- 1 链路状态协议及OSPF三张表
- 2 网络分层，各种区域，各种LSA类型
- 3 OSPF各种包类型
- 4 OSPF建立邻居的必要条件及各种过程
- 5 OSPF各种网络类型
- 6 OSPF的验证
- 7 汇总
- 8 重分布

#### 一 影响OSPF建立邻居关系的参数

router-id  
认证  
stub/nssa  
hello interval, dead interval

疑问：

1 影响OSPF建立邻居关系的参数中，network mask(始发路由器接口的地址掩码)怎么会影响呢？

3 X.25，令牌环，FDDI，ATM

4 公布默认路由时的参数route-map的问题

```
route-map C  
match address lo 1 lo 2 ***
```

5 虚链路和tunnel通道和多进程重分布都可以连接被分割的area0

6 default-information originate(aways)作用是什么

## 二 LSA的类

# 型

- 1 router lsa 由每个路由器产生，用来描述产生它的路由器，该路由器的直连链路和状态，以及该路由器的邻居
- 2 network lsa 由多点接入链路上的指定路由器（DR）产生，描述了该链路以及所有相连的邻居
- 3 network summary lsa 由区域边界路由器（ABR）产生，描述了区域间的目标网络
- 4 ASBR summary lsa 由区域边界路由器（ABR）产生，描述了区域外的自主系统边界路由器（ASBR）
- 5 AS external lsa 由自主系统边界路由器（ASBR）产生，描述OSPF域外的目标网络
- 7 NSSA external lsa 由非末梢区域内的自主系统边界路由器产生

## 三 OSPF特殊区域

OSPF汇总，1、减小路由表，2、使路由发生变化的时候，只影响小的区域，3、减小CPU资源

OSPF里面没有自动汇总

1、在进程下用 `area * range *` 会抑制汇总范围内的明细路由，而不会影响其他路由的传送，另外，`area range` 只能对LSA1和LSA2生效，在哪个ABR上做，就只能对哪个ABR的邻接区域有效果

2、`summary address`汇总，只能在ASBR上做，把范围内的明细抑制掉，其他路由可以传送进来

`default-information originate(aways)`，命令：这个命令必须在路由表里事先有一条默认路由存在才可以生效例如：`ip route 0.0.0.0 0.0.0.0 null 0`，注意，如果没有事先的静态或者默认路由，那么`default-information originate`命令后面必须加`aways`

OSPF在区域内是链路状态，而在区域间是距离矢量

area 0 负责传递其他区域的LSA，也就意味backbone拥有LSA1-5  
非backbone拥有LSA1-5

STUB区域，只接受OSPF域内的LSA1/2/3，做完STUB区后，ABR会自动产生一条默认路由指向STUB区外，产生的这条默认路由COST是1，再加上链路的COST值，就是总COST

total stub区域，只接受本area的LSA，只有LSA1/2  
area \* stub no-summary，只在ABR上做就可以，而stub区，需要全部做成stub，过滤了LSA3、4、5，但是默认路由是LSA3类的，这条默认路由COST也是1，再加上链路COST，就是总COST

STUB和TOTAL STUB不允许有ASBR，而且不能有多个ABR，否则会产生次优路径

NSSA: LSA7 (NSSA-external), 是NSSA区域内部的外部LSA area \* nssa ,  
stub区域的全部路由器都需要做  
NSSA区域的ABR不会自动产生默认路由

NSSA区域，首先是一个STUB区域，即deny了LSA4/5，但是这个区域可以存在外部的LSA（本区域内部ASBR自己产生的）  
可以有LSA1/2/3/7

TOTAL NSSA 区域，可以存在LSA1/2/7, area \* nssa no summary，只需要在NSSA的ABR上做就可以，会自动产生默认路由，这条默认路由是LSA3的

在NSSA的ABR上产生默认路由的命令：

```
area * nssa default-information originate
```

更改特殊区域的COST值：

```
ospf 110  
area 2 default-cost *
```

当NSSA区的其中一个路由器既是ABR又是ASBR时，需要增加这个参数：

```
area * nssa no-redirect no-summary
```

增加这个参数的目的：

- 1 已经有了默认路由到ABR了，不需要明细路由了
- 2 明细路由进来会影响NSSA区域链路的资源

小结论：特殊区域中，只有NSSA区中不会自动产生默认路由，其他都可以，stub, total stub, total nssa

只有area0 可以传递其他区域的LSA，那么建立了虚链路后可以传递，说明虚链路是属于area0的  
用以下命令建立虚链路，必须双方都要做

用以下命令建立虚链路，必须双方都要做

```
area * vir 对方路由器ID
```

注意不能在特殊区域内部建立虚链路，因为只有规则区域和area0一样拥有全部的LSA

虚链路的作用：

- 1 将没有和area0直连的路由器，直接拉入area0
- 2 连接被分割的area0
- 3 area0的冗余

虚链路和tunnel通道都可以连接被分割的area0

```
sh ip os da
```

do not age (DNA):不会老化

通过tunnel建立虚链路步骤

```
tunnel:
```

```
in tunnel 0
```

```
tunnel source
```

```
tunnel des...
```

接口：IP unnumbered lo 0 借用lo 0的地址，注意借用必须是非穿过区域的接口地址（area 0的地址），否则路由学不到，tunnel接口的cost是11111

进程下default-metric \*修改OE2的COST值

# OSPF认证

链路认证

STEP1：在链路上启用认证类型，明文IF#：ip ospf authentication \

或者密文IF#：ip ospf authentication message-digest

STEP1: 在链路上启用认证类型, 明文IF#: ip ospf authentication \

或者密文IF#: ip ospf authentication message-digest

STEP2: 在链路上设置密码

IF#:ip os authentication-key \* (明文)

IF#:ip os message-digest-key 1 md5 \*(密文)

区域认证

STEP1:在进程中启用认证类型

#area \* authentication (明文)

#area 0 authentication message-digest(密文)

STEP2:在链路上设置密码

if)#ip ospf authentication-key \* (明文)

if)#ip ospf message-digest-key 1 md5 \*(密文)

区域认证其实就是为了减少命令

## OSPF2

1、OSPF运行SPF算法后, 从根部到叶节点特点:最短、无环

)当ASBR重分布外部路由时, 满足以下条件Forwarding address字段不为0:Z;noR

- A. OSPF进程将其外部路由的下一跳接口地址公告(network) j>
- B. 该外部路由的下一跳接口没有被passive(no passive interface )8
- C. 该外部路由的下一跳接口不为p-t-p的OSPF网络类型m
- D. 该外部路由的下一跳接口不为p-t-m的OSPF网络类型)SA| {j

2)当ASBR中的区域为NSSA区域, 同样会导致外部路由Forwarding address字段不为0:

注意在NSSA的选择Forwarding address地址的规则, 选择最高的lookback接口地址--->选择最高的可用物理接口地址^dhax

还有一点要提的, NSSA ABR产生的LSA7转成LSA5时不会继承LSA7 forwarding address, 将设置为0.0.0.0。而NSSA ASBR产生的LSA7转成LSA5时继承其LSA7 forwarding address地址。

## OSPF三张表

邻居、数据库、路由

EIGRP三张表

EIGRP三张表  
邻居、拓扑、路由

## 网络分层、各种区域及各种类型

为什么要分层：1减少路由条目2稳定行

六种区域：backbone, 普通区域（规则区域），NSSA，STUB，TOTAL NSSA，TOTAL STUB

LSA类型：LSA1/2/3/4/5/7

LSA1 (show ip ospf database route)：是路由器本身产生的，描述的是本路由器直连的链路，（在MA网络类型里，描述的是DR的地址，没有描述在MA链路上的其他邻居，在其他网络类型，描述的是自己的邻居，描述了中间链路：a stub network）在本区域泛宏的

OSPF组步调机制更改：进程下：timers pacing lsa-group \*

show ip ospf database router 1.1.1.1(link id)的应用，查看网络类型，及DR/BDR查看

在多访问网络里，都和邻里建立邻居关系，不一定建立邻接关系

LSA2 (show ip ospf database network):由DR产生，描述了链路的子网掩码，首先比较网络号，还描述了这条链路上由哪些路由器，

LSA3(show ip ospf database summary)：由ABR产生，LSA3其实就是将一个区域的LSA1/2转换成LSA3来传递到其他区域，还包括在本区域内泛宏的LSA3。每经过一个ABR,这时advertise-router就会变成本区域的ABR的router-id。

LSA4(asbr-summary):其实是路由，由ABR产生，每经过一个ABR，advertise-router就会变成本区域的ABR的router-id，描述的是ASBR的router-id，描述的是其他区域的ASBR的router-id。

LSA5:由ASBR产生，会在OSPF域内泛宏，其实是路由

查看信息构建拓扑

show ip ospf database

show ip ospf database router 1.1.1.1 (link-id)

LSA7 (nssa-external):其实是路由，由NSSA区域的ASBR产生的,经过NSSA区域的ABR时，会转变成LSA5，在其他区域泛宏

BACKBONE区域：可以有LSA1/2/3/4/5，可以传递一个区域的LSA到其他区域

规则区域：可以有LSA1/2/3/4/5不能传递LSA

STUB区：可以有LSA1/2/3，deny了LSA4/5，STUB区域内，不能有ASBR，ABR会产生LSA3的默认路由在STUB区域内泛宏

TOTAL STUB区：可以有LSA1/2，不能有ASBR，ASBR会产生LSA3默认路由在STUB区内泛宏

NSSA区：允许有ASBR，但不能产生默认路由，如果向产生默认路由，需要：ip default-network orgi....产生LSA7泛宏到其他区域

TOTAL NSSA:ABR会产生默认路由LSA3泛宏到其他区域

TOTAL NSSA:ABR会产生默认路由LSA3泛宏到其他区域

### 3 OSPF的各种包

HELLO

DBD 数据库描述报文

LSR 请求LSA (不需要LSA头部, 依靠以下3个参数来请求: LSA类型,

LINK id, ADV router)

LSU 更新LSA (有LSA头部和其他信息, K值、AS号、验证)

LSACK 确认LSA (显式确认, 隐式确认)

显式: 发出去的包必须收到确认包

隐式: 用DBD来确认

### 4 OSPF接口的各种网络类型

show ip ospf interface \*

|              | hello                      | 手工建立nei | 32位 |
|--------------|----------------------------|---------|-----|
| P2P          | 10                         | NO      | 没   |
| P2MP         | 30                         | NO      | 有   |
| P2MP-NON     | 30                         | Y       | 有   |
| NBMA         | 30                         | Y       | 没   |
| BROADCAST    | 10                         | N       | 没   |
| LOOP         | N                          | N       | 有   |
| VIRTUAL-LINK | 10(虚链路建立好后 N<br>就不发送HELLO) |         | 没   |

|           | P2P             | P2MP                | P2MP-NON            | BROADCAST     |
|-----------|-----------------|---------------------|---------------------|---------------|
| P2P       | Y               | 修改TIME后可以建立, 有路由    | 修改T后可以建立, 有路由       | 可以建立, 有路由     |
| P2MP      | 修改T后可以建立, 有路由   | Y                   | 可以建立, 有路由 (32位主机路由) | 可以建立, 有路由     |
| P2MP-NON  | 修改T后可以建立, 有路由   | 可以建立, 有路由 (32位主机路由) | Y                   | 修改T后可以建立, 有路由 |
| BROADCAST | 修改T后可以建立, 但没有路由 | 可以建立, 但没有路由         | nei可以建立, 但没有路由      | Y             |
| NBMA      | 修改T后可以建立, 但没有路由 | 可以建立, 但没有路由         | nei可以建立, 但没有路由      | 修改T后可以建立, 有路由 |

### 5 OSPF建立邻居的必要条件和各种过程

Q1: hello包里有哪些字段对建立邻居有作用?

hello/dead

末节区域标示

area-id

认证 (认证类型和认证密码)

Q2: 是不是每种网络类型建立邻居时经过的各种过程都一样

TYPE字段里包含的5种包

HELLO

DBD

LSR

LSU

LSACK

debug ip packet

debug ip ospf packet

debug ip ospf event

建立邻居过程中, 假如MTU不匹配, 怎么办

1 改MTU, 接口下, MTU \*, 可以更改, 如果不匹配, 会停留在extra状态

2 接口下, ip ospf mtu-ignore, 在建立邻居过程中, 忽略MTU检测, 注意在MTU值小的接口定义

3 接口下, ip mtu \*, 注意这里的\*最大只能是在接口下mtu \*的值, 也就是说1包含3

不会把学到的LSA放入路由表的情况

1 对DR认识不同

2 对中间链路认识不同

接口优先及为0时, 输入NEI, 尝试建立邻居, 不会成功, 在SHUN RUN里也不会显示, 但show ip ospf nei里会显示, 是attmep状态

注意: 邻居表里看到有尝试建立邻居的提示attmep, 但shun run里没有, 那么就是这个原因, 是接口优先级为0了, 解决: 把接口优先级改为>0即可, 然后再删除那条没用的NEI

EIGRP只支持密文, 不支持明文

OSPF支持明文和密文

RIPV1支持

## OSPF汇总

area range 只能对LSA1/2有作用

summary adress只能在ASBR上做, 且只能汇总引入外部路由的那个ASBR的路由器

nssa区, 在ABR上用summary adress做也可以汇总, 因为重分布进NSSA的LSA7经过ABR会转化为LSA5

使用以下命令的条件:

are \* nssa no-redistribution default-information-originate

1 必须是NSSA区的

- 2 既是ABR又是ASBR时候
- 3 往NSSA公布默认路由

### 重分布

default-metric \* , 这个命令不能匹配非直连重分布

既重分布了直连链路和协议路由, 那么只有直连链路的路由, 因为优先级高于协议, 解决办法, 把协议那个接口匹配直连重分布

### 更改COST值的方法。3个

- 1 特殊区域的默认路由cost值的更改  
are \* default-cost \*
- 2 修改参考带宽  
auto-cost reference-bandwidth 10000 (M)
- 3 修改链路带宽  
bandwidth \*

### 虚链路作用

将没有于BACKBONE相连的区域拉进BACKBONE

冗余

区域0分割

虚链路只能做在规则区域, 因为只有规则区域才有全部的LSA, 特殊区域没有全部的

注意: 在区域认证必须在虚链路做完之后才能做

### 解决区域分割办法:

- 1 VL
- 2 建立tunnel (注意用unnumbered借用地址时, 地址必须是ARE0中的地址, 可以是任意地址, 如果是手工指定地址, 那么两个地址必须在同一网段)

具体过程:

```
R1#in tunnel *
    tunnel source 13.1.1.1
    tunnel source 13.1.1.3
    ip unnumbered s0(注意这里的地址是ARE0的地址)
R3#in tunnel *
    tunnel source 13.1.1.3
    tunnel source 13.1.1.1
    ip unnumbered s1
```

### 3 多进程重分布

在多进程里, 不能共享同一条链路, 第一个宣告的进程, 首先会往对方发送HELLO包, 就会首先和第一个宣告的进程建立邻居, 例如线宣告了OSPF2, 那么邻居就会优先和OSPF2的接口建立邻居)

### OSPF过滤

1 过滤路由，只能在本路由器生效

istribute-list ....in 接口

istribute-list ....out 协议号 在重分布时过滤掉

2 area \* range ...not-advertise... 本区域的ABR上做，过滤LSA1/2

3 summary-address...not-advertise... 本路由器自己产生的ASBR上做, 过滤LSA5/7

4 are \* filter-list prefix ... 过滤LSA3, ABR

5 接口下过滤全部LSA:ip ospf database-filter all out (注意做完清进程)

6 进程下对特定邻居做全部LSA的过滤: nei (address) database-filter all out (注意在NBMA和P2MP才能出现在sh run里，在其他网络类型下此命令起作用，但不会出现在sh run里，这里是攻击点)

### OSPF实验

1 基本配置

2 末梢区域 (两个ABR时候，COST对默认路由的影响)

3

### 路由控制

1 标准访问控制列表access-list只能匹配数值，不能匹配掩码 (1-99)

2 前缀列表

3 扩展访问列表 (100-199)

匹配协议、源地址目标地址、路由

## ISIS

<file:///D:/cisco/cisco/isisnote.pdf>

Modify Time : 2005年7月11日 19:43:30 File Size : 373547

## BGP

### BGP

1 BGP特性及3张表

2 EBGp和IBGP的比较

3 NEI和network语句的含义，以及邻居的flaping, 下一跳在多访问网络中的实现

3 NEI和network语句的含义，以及邻居的flapping, 下一跳在多访问网络中的实现

4 BGP的各种包，以及建立邻居的各种状态

5 BGP聚合和路由及各参数

6 BGP的各种属性，BGP选路原则，以及路由最优的必要条件

7 damping、联邦、反射器、对等体组

BGP特性：

1 可靠传输，在TCP的179端口

2 触发、增量更新

3 keepalive维持邻居关系

4 丰富的度量值

5 可组建可扩展的巨大网络

3张表

1 邻居表

2 转发表

3 路由表

同步

IBGP学到的路由也能从IGP学到，就满足同步，否则不能同步

两种情况关闭同步

1 BGP路由重分布到IGP里

2 包的走向上面所有运行BGP

从一各IBGP学到的路由不会传递给另一个IBGP（类似于水平分割）

将IGP学来的路由引入BGP时，下一跳是IGP的下一跳

nei和network语句的含义

nei:用我的更新源去访问这个地址的179端口，也允许这个地址来访问我的179端口

BGP的auto-summary

作用：

1 在重分布时将明细变为主类

2 既能将明细引入BGP，也能将此明细的主类也引入BGP

改变路由下一跳的方法

1 nei 3.3.3.3 next-hop-unchanged

只能在EBGP多跳时才能使用此命令

2 set ip next-hop 2.2.2.2

set in s0

指EBGP邻居时强制更改下一跳

BGP的各种包

1 open  
2 keepalive  
3 update  
4 notifaction

BGP聚合及各种参数

1 汇总再引入  
2 aggregate 聚合

继承属性:

as-path

local precence

所有明细的community属性: 1 local- 2 no-export 3 no adver

查看团体的属性:

sh ip bgp community no-export

sh ip bgp community no-adver

属性的方向设置

local-preference in方向

med in/out

as-path in/out

origin code set origin ? in/out

filter-list 与as-path合用 in/out

BGP注意事项

1 BGP链路认证: nei \* password 1 cisco \*(65512-65535)

2 删除AS号, 两种方法

nei \* remove-private-as \*

联盟

3 RR, 关闭反射 (只在IBGP使用), 条件: 所有客户是全互联

no bgp client-to-client refcet

4 影响选路, 优选IGP或IBGP路由, 而不优选EBGP路由

net \* backdoor

5 允许经过多个AS

bgp maxas-limit \*

6 过滤列表

ip as-path access-list 1

nei \* filter-list 1

7 本地用新AS和你建立邻居

nei \* local-as \*

nei \* local-as \* no-prepend (收AS时把local-as号去掉)

8 条件路由

nei \* advertise-map aaa non-exist-map bbb

当bbb不存在时, 就发aaa

- . 匹配任何单个的字符，包括空格
- ^ 匹配一个字符串的开始字符
- \$ 匹配一个字符串的结束字符
- \_ 下划线，匹配一个逗号，大括号，一个输入字符串的开始，一个输入字符串的结尾或一个空格
- | 管道符，具有逻辑或 (or) 的含义，意思是可以匹配两个字符串中的一个
- \ 转义字符，用来紧跟其后的控制字符转变为常规字符
- \* 匹配前面字符的任何序列 (0次或多此出现)
- + 匹配前面字符的一个或多个序列 (1次或多次出现)
- ? 匹配前面字符的0次或1次出现
- [] 表示一个范围

### aggregate-address中的访问控制列表

aggregate-address \* \* suppress-map

例如：现有以下路由，需要用此命令来抑制明细，只允许聚合发送给对方

192.168.192.0

192.168.193.0

192.168.194.0

192.168.195.0

那么：用扩展访问列表来匹配明细

access-list 101 deny ip 192.168.192.0 0.0.3.0 host  
255.255.255.0

注意黄色部分的通配符：用来说明一个范围，即192.168.192(193、194、195).0

用以下来计算范围：

在第三个段中的BIT位，192.168.192/192/194/195.0, 用二进制表示如下：

192: 1100 0000

193: 1100 0001

194: 1100 0010

195: 1100 0011

如上：黄色部分位完全匹配的，那么后面的就是一个范围，用最大的表示即：192.168.192.0 0.0.3.0)

如果是192-199, 那么：

192: 1100 0000

193: 1100 0001

194: 1100 0010

```
195: 1100 0011
196: 1100 0100
197: 1100 0101
198: 1100 0110
199: 1100 0111
```

那么ACL的范围就是:

```
access-list 101 deny ip 192.168.192.0 0.0.7.0 host
255.255.255.0
切记!
```

再举个例子:

用ACL写出199.172.\*.0/25, 那么:

```
access-list 101 permit ip 199.172.0.0 0.0.255.0
255.255.255.128 0.0.0.0
```

注意黄色部分, 0.0.255.0, 说明第三个段的范围是全部, 红色部分是准确的子网掩码

## BGP不考虑属性的方法

1 如果CISCO运行BGP的路由器与另外一个厂商的路由器不考虑as-path长度的路由器对等, 这种情况会产生潜在的路由决定不一致, 可以通过以下命令忽略as-path属性

```
bgp bestpath as-path ignore
```

## BGP概念类

originator\_id和cluster\_list属性

两者都是可选非传递属性, 由路由反射器使用, 以防止环路,

originator\_id是由路由发起者产生的一个32比特的值, 该值是本地AS里路由发起者的路由器ID。如果路由发起者在接收到路由的originator\_id中发现自己的RID, 就知道产生了路由环路, 于是该路由就被忽略掉

cluster\_list是路由经过的路由反射器簇ID的一个序号, 如果路由反射器在接收到路由的cluster\_list中发现了自己的本地簇ID, 就知道产生了环路, 于是该路由就被忽略掉

## IPV6

**IPV4:32位, 42亿地址, 一个映射消耗内存64K**

IPV4包头有20个字节, 有4个段, 每个段有8

IPV6包头有40个字节, 有8个段, 每个段有16

**为什么V6中端到端有安全性? 因为扩展包头里有ESP和AH字段**

**V6书写规则:**

## V6书写规则:

- 1 前导0省略，一个段里前面0可省略
- 2 连续0省略，如果有两个段全是0，则可省略位::，注意双冒号在一个地址当中只能用一次

## IPv6编址及寻址

- 1 未指定 00...0(128bit) ::/128
- 2 环回 00...1(128bit) ::1/128
- 3 全局单播 0010 2000/3  
范围：第一个字段的前3位（2000~3FFF）
- 4 组播地址 1111 1111 FF00::/8  
范围：前8位
- 5 链路本地地址 1111 1110 10 FE80::/10  
范围：前10位  
链路本地地址：link-local地址，用来标示一条链路上的唯一设备，而不是一个设备上的唯一链路
- 6 站点本地： 1111 1110 11  
站点本地相当于IPv4中的私有地址（旧的私有地址）
- 7 unique local unicast address 1111 1100 FC00::/8  
1111 1101 FD00::/8  
unique local unicast address:新的IPv6私有地址

## IPv6基本配置:

自动生成本地链路（link-local）IPv6地址  
接口下 ipv6 enable  
更改IPv6 link-local地址：  
接口下 ipv6 address fe80:: link-local  
启用IPv6路由功能  
ipv6 unicast-routing  
不发送链路RA公告  
ipv6 nd suppress-ra

## MAC地址转化为IPv6地址规则:

MAC地址是48位的，从中间断开各24位，在中间加上FF FE，再把加上后地址的第7位0改成1，如果是1则改成0

## 全球和本地唯一:

MAC地址：第2位如果是0，则全球唯一，如果是1，则本地唯一  
IPv6地址：第7位规则和MAC相反，0本地唯一，1全球唯一

## eui/64规则:

eui/64地址是link-local地址的后64位

例如：

eui/96, 那么只填充后32位，因为 $128-96=32$

eui/64, 那么只填充后64位，因为 $128-64=64$

### 组播地址：

FFFF::/8 匹配所有组播地址

|           |      |      |                      |          |
|-----------|------|------|----------------------|----------|
| 8bit      | 4bit | 4bit | 88bit                | 32bit    |
| 1111 1111 | flgs | scop | reseved must be zero | group id |

flags:

0000 永久地址

0001 临时地址

scope:

0001 本地接口

0010 本地链路

0011 本地子网

0100 本地管理

0101 本地站点

1000 组织机构

1110 全球

### 被请求节点组播地址：

此地址会为每个IPV6地址匹配一个，匹配规则：

FF02::1:FF00:0000/104

前104位不变，只替换后24位（黄色部分）

只将后24位变为IPV6地址的后24位即可

作用：替代ARP和DAD（重复地址检测）时用的

IPV6的播方式：

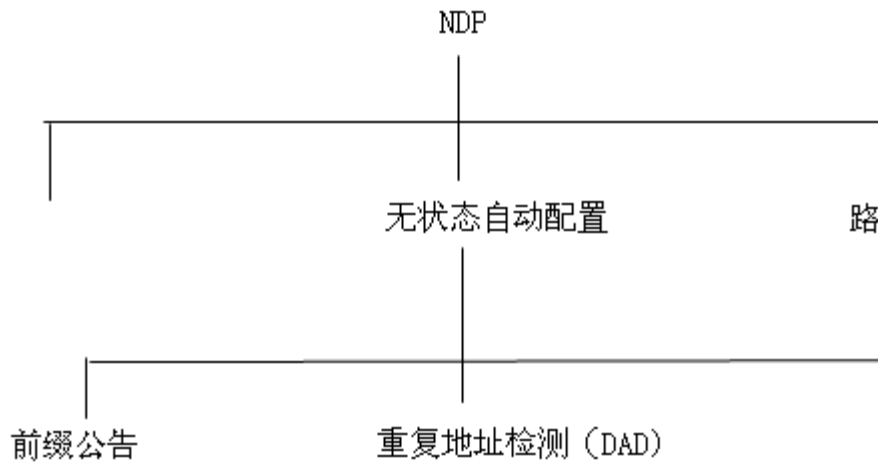
anycast:任意播（找最近的设备或节点）

unicast:单播

mutilcast: 组播

泛播地址规则：前缀+64个0

### IPV6邻居发现协议（NDP）



## 什么时候进行DAD（重复地址检测）

- 1 在接口NO SH 时
- 2 自动配置地址时

## NDP用到的几种ICMP V6的消息

**ICMP:** 向源节点公告关于向目的地址传输数据的错误和信息

RS:路由请求 ICMP TYPE: 133

RA:路由应答 ICMP TYPE: 134

注意RA:只有在启用ipv6 unicasting(启用V6路由功能)才会发送RA (RA默认200秒发一次,存活时间是1800秒)

NS:邻居请求 ICMP TYPE: 135

NA:邻居公告 ICMP TYPE: 136

redirect:重定向

## IPV6 debug工具

debug ipv6 pa

debug ipv6 nd

debug ipv6 icmp

## 修改IPV6 RA发送时间（200秒）间隔和存活（1800秒）间隔

IPV6 nd ra-interval \*

ipv6 nd ra-lifetime \*

修改IPV6 DAD（重复地址检测）次数

IPV6 nd dad attempts \*

## 前缀公告

路由器接口会自动配置且发送RA给链路设备，但不会发前缀，那么此时这条链路设备不会因收到路由器的RA而自动配置地址，因为没有收到前缀

## 无状态自动配置

ipv6 nd prefix 2001::/64 at \* (\*: 日期间隔，如：此条前缀用到\*月\*日终止使用)

## ARP协议的替代协议（替代ARP）

## IPV6支持的RIP

IPV6的RIP特性

- 1 IPV6所有协议都在接口下激活的
- 2 RIPNG与RIPV2采用相同算法
- 3 协议通讯基于UDP521端口
- 4 组播更新地址FF02::9
- 5 采用本地链路地址FE80::1/10作为源地址发送更新信息

基本配置：

```
IPV6 unicastng
ipv6 route rip *
sh ipv rip
ipvr rip * default-informition (产生默认路由)
```

## IPV6支持的BGP PLUS

IPV6的BGP特性：

- 1 BGP update中只有以下字段与IPV4有相关性

next-hop  
aggregator  
NLRI

- 2 BGP-4 PLUS 增加的两个属性

MP\_REACH\_NLRI  
MP\_UNRE

基本配置

IPV6 BGP的基本配置同IPV4

```
router bgp 1
bgp router-id *
no bgp default ipv4_unicast (此命令的目的是让所有IPV4的BGP邻居都必须在IPV4地址簇里激活，以实现一对一激活)
nei * remot 2
```

```
nei * lop0
address-family ipv4 unicast (一对一激活, 必须激活该簇才能确立邻居关系)
nei * active
address-family ipv6 unicast (IPv6的簇激活)
```

## IPv6支持的OSPF V3

IPv6的OSPF特性

1 增加两个LSA

intra-area-prefix-lsa及link-lsa

2 原来的2个LSA被改名

summary-lsa改为: intra-area-prefix-lsa

ASBR-summary-lsa改为: inter-router-prefix-lsa

组播地址变化

ALLSPFRouters FF02::5

DR/BDR FF02::6

基本配置

```
ipv6 router ospf *
```

接口下:

```
router-id *
```

```
ipv6 ospf * area 1
```

```
ipv6 add 2001::1
```

~~~~~  
~~~~~

IPv6建立静态tunnel的顺序:ipv6、mode、默认路由

IPv6建立动态tunnel (6to4动态tunnel)

1 前缀必须用2002::/16

2 必须改造自己的网络特性

这种改造是根据source地址的改造, 具体: 把十进制源地址转化为十六进制, 然后用转化后的十六进制地址替换2002后的32位即可 2002:\*\*\*:3:3

基本配置

```
in tunnel 0
```

```
tunnel source lo 0(1.1.1.1)
```

```
ipv6 unnumber e0 (2002:101:101::1/64
```

```
tunnle mode ipv6ip 6to4
```

```
ipv6 route 2002::/16 tunnel 0
```

## 综合练习

### EIGRP

#### 一 桥接

帧中继R1的s0口只用于子接口（仅使用图中所提供的DLCI）

#### 二 RIP

R3的lo1、lo2运行RIPv2，RIP和OSPF做双向重分布

#### 三 EIGRP

EIGRP 1

R1上起4个环回口

lo1: 167.1.32.1/24

lo2: 167.1.33.1/24

lo2: 167.1.34.1/24

lo2: 167.1.35.1/24

宣告在EIGRP 1中

EIGRP 100

R1/R4/R5在EIGRP 100中，R1/4/5的环回口以及VLAN\_A为EIGRP 100域内路由

R4上看R1/5的环回口

D 10.1.1.1/32 [90/256000] via 10.1.145.1 s0

D 10.1.5.5/32 [90/256000] via 10.1.145.5 s0

R5上看R1/4的环回口

D 10.1.1.4/32 [90/256000] via 10.1.145.4 s0

D 10.1.5.1/32 [90/256000] via 10.1.145.1 s0

R1上看R4/5的环回口

D 10.1.5.5/32 [90/256000] via 10.1.145.5 s0.1

D 10.1.4.4/32 [90/256000] via 10.1.145.4 s0.1

R4/5收到167.1.32.0/22的一条汇总路由，不允许在EIGRP 100内做手工汇

D 10.1.4.4/32 [90/256000] via 10.1.145.4 s0.1

R4/5收到167.1.32.0/22的一条汇总路由，不允许在EIGRP 100内做手工汇总。EIGRP 1 向lo1发送一条10.1.0.0/16的路由不发送其他10.1.0.0/16包括的明细路由，向其他接口公告明细的路由信息。

EIGRP 2

R2的lo1/lo2口在EIGRP 2中，EIGRP 2 与ospf双向重分布

四 OSPF

R2/3/4/5运行，R3/4/5之间都只能形成two-way状态，当VLAN\_A中再加入一台新设备时，R2还是DR

R2/3只看到167网段和10.1.145.0网段负载均衡，R2访问R1的环回口正常情况下走R4，当R2/4间链路断了后走R5，R3访问R1的环回口走R5，当R3/5之间断了后，就丢包

五 过滤

EIGRP 1 和EIGRP 2 不接受RIP的路由，RIP也不接受他们的路由

注：要求全网互通，所有loopback0接口的地址为10.1.x.x/32(x为路由器号)，本实验主网段为10.1.0.0/16,不允许出现静态路由

```
~~~~~  
distance 180 100.1.1.5 0.0.0.0 1
access-list 1 permit 167.1.32.0 0.0.3.255 (修改AD，选择路径)
```

## OSPF

IGP小节练习

一 桥接

帧中继中R1的S0口只用于子接口，要求PING通所有接口（仅使用图中所提供的DLCI）

二 OSPF

1 R1/2/3的S0口在OSPF的ARE 1中，R2/3的E0口在OSPF的ARE 0中，R1的LOOP1在OSPF的ARE2中，R3的LOOP1在OSPF的ARE3中

2 OSPF的帧中继中不允许使用NBMA和广播模式

3 ARE2只接受OSPF的LSA1/2路由

4 R3在日后会从ARE3中接收到一些LSA7的路由以及ARE3中会有一条默认路由

5 ARE0使用明文认证，ARE1使用更安全的认证方式，验证密码cisco

6 所有LOOP0接口均在OSPF域内

7 R1的LOOP3到LOOP10接口不允许直接宣告进OSPF域内

8 R1的LP2宣告进EIGRP 90中，做双向重分布，不接收RIP域内的路由（重分布直连的优先级高于协议，那么要把和协议直连的那个接口也匹配进去

三 EIGRP和OSPF在R2上做双向重分布，EIGRP只向OSPF

1 发送一条路由（不允许是167.1.0.0/16）

1 发送一条路由（不允许是167. 1. 0. 0/16）

四 RIP

1 R3的LOOP2到LOOP5在RIPV2中

2 RIP和OSPF在R3上做双向重分布

3 在R1和R2上只能看到RIP域过来的一条汇总路由（不能是166. 1. 0. 0/16），要和EIGRP的汇总用不通的方法

五

1 R1的LOOP2接口不允许宣告在任何路由协议中

六 过滤

1 在R2上可看见这样的一些路由168. 1. X. 0（X为奇数） distribute-list

1 在R3上可看见这样的一些路由168. 1. X. 0（X为偶数）

区域和虚链路都需要认证，虚链路需要做3条，其中一条是冗余，在R2/3上做，穿越ARE1，虚链路指定的是对方的router-id

## BGP1

需求：

1 不允许用静态路由

IGP需求：

1 为各路由器建立LOOP0口（IP:135. y. x. x/24, y为RACK号，x为路由器号，发布在适当的IGP路由进程中，在IGP配置后，在每台都能看到所有路由器的LOOP0路由

2 R4上LOOP8-11，IP:4. 4. 8. 4-4. 4. 11. 4/24, 重分布到OSPF中，并汇总

3 汇总路由在OSPF中传递时，COST值要发生改变

4 如果以后在运行OSPF的路由器上，在Gibyte以太网时，其COST值为5，其他接口COST值也要改变

5 ARE2的路由条目要尽量优化，并且LSDB中要求存在7类默认路由

6 ARE3中不能存在外部路由，但应该存在域间路由

7 R3的E0口永远不能成为DR或BDR

BGP需求：

1 所有BGP邻居关系均使用LOOP口为更新源

2 EBGp: R1-2 , R2-4, R3-5

3 IBGP: R1-R3

4 R4公布LOOP20, IP:200. 200. 4. 4/24, R2传递给R1时要去掉私有AS信息

5 R1上公布LOOP20, IP:200. 200. 1. 1/24, R2公布LOOP20, IP:200. 200. 2. 2/24

6 R3上只能将200. 200. 4. 0/24和200. 200. 1. 0/24传递给R5

7 R5公布LOOP10-17, IP:199. Y. 0. 5-7. 5/24

8 R3对其进行聚合，只将聚合路由发给邻居，不能使用summary-only, 不能使用路由过滤

9 R1/2/3/4都可以PING到R5公告的明细路由

8 R3对其进行聚合，只将聚合路由发给邻居，不能使用summary-only, 不能使用路由过滤

9 R1/2/3/4都可以PING到R5公告的明细路由

10 R2监控R1的L0P20，震荡一次就给1000惩罚，当惩罚值超过2500时，改路由被抑制，每30分钟惩罚值降到原来的一般，当惩罚值降到1200，改路由被重新启用

题解

```
6 distribute-list permit *
```

```
8 aggregate * * suppress-map *
```

```
9 set ip next-hop
```

## BGP2

BGP实验二（图3-1）

IGP全网互通，请用OSPF作为IGP协议，主网段为170.16.0.0/16，不能配置BACKBONE

IBGP

R1/2/3/4在AS1234里，R1/2, R1/3, R2/4, R2/3, R3/4互为IBGP邻居关系，R2是RR，R3和R4是它的客户，IBGP邻居关系要尽可能的稳定

EBGP

R3/4分别和BACKBONE建立EBGP邻居关系，BACKBONE属于AS2，BACKBONE只与AS1建立邻居

R4与R5直连口的地址为110.100.2.1，R3与R5直连口的地址为110.100.1.1

策略

R3上有一个环回口，L020:200.200.3.3/24，宣告进BGP，只能在本AS内传递

R1/2分别有一个环回口，R1: L020: 200.200.1.1/27，

R2:L020:200.200.2.2/27，分别将其宣告进BGP，R3/4分别只发送

200.200.1.0/27和200.200.2.0/27的路由给BACKBONE，让BACKBONE访问R1的环回口走R3，访问R2的环回口走R4

聚合

在R4上面聚合，R4的BGP转发表如下

| network         | next hop      | weight |     |
|-----------------|---------------|--------|-----|
| path            |               |        |     |
| *>198.16.0.0/19 | 0.0.0.0       | 32768  | 2 i |
| S>198.16.1.0    | 110.100.2.254 | 1000   | 2 i |
| *>198.16.2.0    | 110.100.2.254 | 1000   | 2 i |
| *>198.16.3.0    | 110.100.2.254 | 1000   | 2 i |
| S>198.16.5.0    | 110.100.2.254 | 1000   | 2 i |
| *>198.16.22.0   | 110.100.2.254 | 1000   | 2 i |
| *>198.16.23.0   | 110.100.2.254 | 1000   | 2 i |

在R3上做策略，当R3与BACKBONE之间链路正常的时候，R1访问BACKBONE走R3，当R3与BACKBONE失去连接时，R1访问BACKBONE走R4，R3从BACKBONE只接

|               |               |      |     |
|---------------|---------------|------|-----|
| S>198.16.5.0  | 110.100.2.254 | 1000 | 2 i |
| *>198.16.22.0 | 110.100.2.254 | 1000 | 2 i |
| *>198.16.23.0 | 110.100.2.254 | 1000 | 2 i |

在R3上做策略，当R3与BACKBONE之间链路正常的时候，R1访问BACKBONE走R3，当R3与BACKBONE失去连接时，R1访问BACKBONE走R4，R3从BACKBONE只接受源自于AS2或者AS2直连AS的路由

R4上监控198.16.3.0这条路由，如果FLAPPING，就对其进行惩罚，当198.16.3.0FLAPPING一次后，经过20分钟，惩罚降到500，当惩罚值超过2500时对其抑制，当惩罚值降到700时，将重新启用这条路由

## 杂七杂八

### 异步传输

AUX密码

```
R1(config)#line aux 0
```

```
R1(config-line)#flowcontrol hardware
```

```
R1(config-line)#transport input all
```

```
R1(config-line)#modem inOut
```

```
R1(config-line)#password cisco
```

```
R1(config)#enable password cisco
```

```
R1(config)#logging
```

### 新建信息项目...

BGP路由重分发时的过滤用access-list \* 匹配，然后用neighbor distribute-list \* 来调用

### BGP

1 设置BGP对等体验证密码,对等体都要设置，并且密码要相同,设置后自动是密文

```
nei * password 7 *
```

2 设置邻居描述信息，可以产生一个最大80字符的描述信息

```
nei * description *
```

3 限制一个路由器可以从一个邻居接收的前缀（路由）数量,如果收到前缀大于指定数值，则自动中断邻居关系

```
nei * maximum-prefix *
```

如果只让路由器通告超过了前缀而不中断邻居关系，此时只会产生一个日志信息，则：

```
nei * maximum-prefix * warning-only
```

4 暂时中断一个邻居关系，但不删除邻居关系配置，则：

```
nei * shutdown
```

5 过滤命令注意

过滤列表：

nei \* shutdown

5 过滤命令注意

过滤列表:

nei \* filter-list \* in(out)与以下列表方式合用, 正则表达式

ip as-path access-list \* permit ^\_

分发列表:

nei \* distribtue-list \* in(out)与以下列表方式合用, 访问(扩展)控制列表

access-list \*

6 在route-map里用set local-preferece \*定义优先级, 然后用neighbor调用, 注意local-preferece只具有本地属性

7 在route-map里用set metric \* 定义MED, 然后用neighbor调用, 注意MED只适用于相邻的AS之间

8 在route-map里用set as-path prepend \*定义需要添加的AS, 然后用neighbor调用

9 IGP重分布进BGP时, 不会附带AS-PATH属性, 解决办法就是在route-map里用set as-path tag 来标记这条路由, 然后IGP就会继承AS属性

## WOLF-3小黑

### 路由

#### 01晚-路由原理与静态路由

(已完成实验)

CCNP BSCI

CCNP routing 课程时间规划: (15-17)

路由原理 / 静态路由: 1

RIP: 1

EIGRP: 2

网络分层:

Access layer:终端用户的接入

Distribution Layer: 实现基本网络策略控制, 和网络服务。

Core Layer:高速地传送数据包。

网络划分的种类:

- 1: 可以按照网络的不同的功能部门进行划分。
- 2: 可以按照网络所分布的地域范围进行划分。

网络拓扑:

Full Meshed / 全互连

任意两点（节点）之间，都有直接相连的连接。

Full Meshed连接，所需连接个数的公式：

公式： $C_2^n$ （上标）n（下标）

∴：在核心网中，构建双冗余的Full Mesh是较为昂贵， $N*(N-1)$

∴：可以考虑构建双Hub&Spoke。  $2*(N-1)$

既可以保持冗余性，也可以降低建网成本。

一个规划良好的网络中，通常考虑到以下3点：

Scalability / 可扩展性

Predictability / 可预测性

Flexibility/灵活性

Benefits of Hierarchical Addressing/网络地址层次化的划分：

- 1：在路由器上，有更高的路由转发效率，减少路由表的路由条目。
- 2：提高地址的利用率。

VLSM: Variable-Length Subnet Mask

VLSM

可以实现根据不同网络需求，拆分 / 划分不同大小的子网，

特别注意：

VLSM不能凭空额外的创造出更多的IP地址来。

Step1:将所需要划分的网络，自大而小地排列出来。

195.15.20.0 / 22 （有效IP： $1024-2=1022$ ）主机位有10位  $2$ 的10次方=  
1024

500 / 200 / 50 / 10 / 2 （Serial Link）

图01—01-VLSM

500IP:9个主机位 / Host Bits

200IP: 8个主机位

50IP:6个主机位

10IP: 4个主机位

Serial Link:2个主机位

step3: 根据每个子网的主机位, 算出子网的网络位:

建议: 在选择网络位时, 先取0, 再取1.

500IP: ∴有23个网络位 ( $32-9=23$ )

200IP: ∴有24

Step4: 计算每个子网的:

网络号, 广播地址, 首个有效地址, 最后一个的有效地址

原则:

如果主机位为全0, 那么这个地址为这个子网的网络号。

如果主机位为全1, 那么这个地址为这个子网的广播地址。

能够容纳500个主机的子网:

网络号: 195. 15. 20. 0 / 23

第一个有效IP: 195. 15. 20. 1 / 23

最后一个有效IP: 195. 15. 21. 254 / 23

广播地址: 195. 15. 21. 255 / 23

能够容纳200个主机的子网:

网络号: 195. 15. 22. 0 / 24

第一个有效IP: 195. 15. 22. 1/24

最后一个有效的IP: 195. 15. 22. 254 / 24

子网的广播地址: 195. 15. 22. 255 / 24

能够容纳50个主机的子网:

网络号: 195. 15. 23. 0 / 26

第一有效IP: 195. 15. 23. 1 / 26

最后一个有效IP: 195. 15. 23. 62 / 26

子网的广播地址: 195. 15. 23. 63 / 26

能够容纳10个主机的子网：

网络号：195. 15. 23. 64 / 28

第一个有效的IP：195. 15. 23. 65 / 28

最后一个有效的IP：195. 15. 23. 78 / 28

子网的广播地址：195. 15. 23. 79 / 28

N0.1 Serial Link:

网络号：195. 15. 23. 80 / 30 有效IP：81 / 82

N0.2 Serial Link

网络号：195. 15. 23. 84 / 30 有效IP:85 / 86

N0.3 Serial Link

网络号：195. 15. 23. 88 / 30 有效IP：89 / 90

Route Summarization/路由汇总：

在需要进行汇总的明细路由中，

寻找前部相同的位数，以这一点为分界点，将明细路由汇总到这一位。

Step1:按照IP路由的4个字段，寻找相同 / 不同点的分界。

Step2:将第一个不同的字段，展开为2进制数，进一步寻找相同 / 不同点的分界点。

Setp3: 将不同的位数，置为全0

以相同的位数作为汇总路由的路由长度。

172. 16. 12. 0 / 22

CIDR: (Classless Interdomain Routing) / 无类域间路由

可以突破网络的边界。

CIDR是可以突破网络主类边界，实现网络汇总。

CIDR:

Block addresses can be summarized into single entries without regard to the classful boundary.

Routing Principles/路由原理

IOS语法:

正体字: 表示需要准确的IOS命令 (不变)

斜体字: 根据实际情况, 输入的参数 (变化)

{        }: 必选项

[        ]: 可选项

| :        OR, 或关系, 多个元素中, 选择一个

LAB1: 静态路由的原理性实验:

图01-02-static route

NO. 1: 关于直链路由:

如果物理接口的链路状态: L1 / L2都能够UP / UP

那么接口所在的网段, 就以直链的形式出现在路由表中。

Step1: 按图配置IP, 进行链路测试:

R1#show ip int brief (察看接口简要IP信息。)

R2#debug ip packet (察看IP数据包)

特别注意：

路由器不是基于接口考虑数据包的路由，而是基于整个路由的路由表，进行数据包的路由。

NO. 2: IP路由的核心原理（路由对联）：（路由的单向性）

上联：“去往目标网络的路径上的”所有路由器，都要有，去往目标网络的路由。（保证数据包能够送到目标网络）

下联：：“在返回源网络的路径上的”所有路由器都要有，返回源网络的路由。（保证数据包能够从目标网络返回源地址）

横批：有去有回，肯定能通！

NO. 3: 路由的下一跳的可达性问题：

每条路由，都必须检查其路由的下一是否可达，如果下一跳不可达，那么这条路由会被路由器从路由表中删除。

R2#clear ip route \* (reset/复位路由表)

路由表的递归查询：

NO. 4: 路由的单向性：

IP路由是单向的。

IP数据包所经过的路径，有可能往返是重叠的，也有可能是不同路径的，取决于路由的指向。（图）01—03—单向性

## 02晚-动态路由协议-RIP

（已完成实验）

Dynamic Routing:动态路由协议：

现代IP网络中，主要的动态路由协议：

AD/管理距离：

- 1：DV/距离向量协议：RIP(120)/IGRP(100)
- 2：LS/链路状态协议：OSPF(110) /IS-IS(115)
- 3：DV-LS/混合协议：EIGRP(90)
- 4：ODR(160)

LAB1：

ODR (On-Demand Routing)

CISCO私有的协议，基于CDP协议的运行。

（CDP只能发现直接相连的Cisco设备）

适用于：

在全网都是CISCO设备的网络环境中，

而且网络拓扑是HUB&SPOKE架构的简单网络中，

（所有的分支点都是直接与Hub相连）

ODR用于使网络配置最简化的操作。

CDP的Hello周期是60S, Hold-time: 3\*60=180.

图02-01-ODR

Step1:确认设备的连接(察看CDP的运行状态)

1-1:

```
show cdp interface
```

(察看正在运行CDP的接口, 默认所有接口都运行CDP)

1-2:控制CDP的运行范围:

```
R5(config)#cdp run
```

(在全局启动CDP, 默认已经启动, 所有接口都运行CDP)

```
R3(config)#in serial 1
```

```
R3(config-if)#cdp enable (对特定接口, 启动CDP)
```

1-3:

```
R5#show cdp neighbors
```

| Device ID | Local Intrfce | Holdtme | capability | Platform | Port  |
|-----------|---------------|---------|------------|----------|-------|
| R3        | ser 0         | 120~179 | R          | 2500     | ser 1 |
|           | 接口            |         |            |          | 对方    |

注意:

1:在ISP的角度考虑, ISP的PE是不会与用户的CE运行CDP的.

2:注意以太网交换机的对CDP的影响. (如果两个路由器上中间有个CISCO的交换机, 那么这台交换机将会阻止ODR的运行)

step2:在中心路由 (HUB) R1上, 启动ODR:

```
R1(config)#router odr
```

实现了内网的全网通信, 但与HUB是非直链的R2, 没有路由.

step3:外网的互通:

CE指向ISP的默认路由:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 13.0.0.3
```

在ISP PE 指向 用户的静态路由:

```
R3(config)#ip route 175.100.0.0 255.255.0.0 serial 0
```

Step4: ODR 的路由观察:(在R2建立一个环回口)

第个分支点, 通过ODR HUB 发来的ODR默认路由, 访问外网:

```
R2#0* 0.0.0.0/0[160/1]via 175.100.12.1,
```

在中心点R1#上, 也会自动获得每个分支点下面所直链的明细路由

```
0 175.100.24.0 [160/1] via 175.100.12.2,
```

```
0 175.100.22.0 [160/1] via 175.100.12.2,
```

Step5:ODR路由的局限性:

1:全网设备必须都是CISCO设备.

2:分支点的路由器, 必须跟中心点直接相连.

## Classful Routing & Classless Routing

落后的淘汰的      先进的, 正在运行的

~~~~~

Classful Routing/有类路由协议:

IGRP /RIP V1

1. 在发送路由更新信息时, 不携带子网掩码, 无法描述路由条目的路由长度.
2. 在主类的网络边界上, 自动发生路由汇总, 汇总到主类的边界, (自动汇总是无法关闭的)
3. 由于上述原因, 有类路由协议会产生“不连续子网”的路由通达性问题.

## Classless Routing/无类路由协议

~~~~~

RIP V2 的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后, 再通告出去.
3. 把收到的明细路由放进路由表中, 但会对明细路由进行汇总后再通告出去!

EIGRP的automatic summary

1. 不会对收到的明细路由进行汇总,
2. 对自己直连的路由进行汇总后, 再通告出去
3. 把收到的明细路由放进路由表中, 并且把收到的明细路由通告出去!

IP Classless(在IOS为12.0以后的版本中, 默认启动无类路由.)

RIPV2/EIGRP/OSPF/IS-IS/BGPv4

1. 在发送路由更新信息时, 已经携带子网掩码.
2. 支持VLSM, 路由的手工/自动汇总, (可以关闭自动汇总)
3. 在部分先进的路由协议中, 支持CIDR(超网)

在网络边界:

DV协议默认会执行自动汇总, (RIP/IGRP/EIGRP)

LS协议默认不执行自动汇总. (OSPF/ISIS)

LAB2:RIP的基本配置:

~~~~~

R3#

router rip

network 176.16.0.0 (宣告与本路由直接相连的网络, 只需要宣告其主类网络)

show ip route rip

debug ip rip

un all

LAB3:RIP的版本控制:

~~~~~

show ip protocols (察看RIP的版本控制)

在未指定版本时, RIP的默认版本是: 发1, 收1/2

指定V1: 发1收1

指定V2: 发2收2.

在全局配置, 对所有接口生效:

R1(config)#router rip

R1(config-router)#version 1

R1(config-router)#version 2

对特定接口, 指定RIP版本, (接口的优先级一般比全局要高)

```
interface serial 0
```

```
ip rip send version 1/2
ip rip receive version 1/2
```

V1与V2不兼容~!

V1:不携带子网掩码, 不携带下一跳信息, 向广播地址发送路由更新.

V2:携带子网掩码, 携带下一跳信息, 向组播地址(224. 0. 0. 9)发送路由更新.

LAB4:RIP的单播更新:(Unicast-Update)/被动接口

(适用于物理链路上, 无法支持广播/组播流量的情况)

~~~~~

Step1:让需要进行单播更新的接口, 不再发送广播/组播更新.

1-1:需要PASS的接口很少时:

```
R5(config-router)#passive-interface serial 0
```

(让特定一个接口不再发送广播/组播的路由更新.)

1-2:需要pass的接口很多时:

```
R2(config-router)#passive-interface default
```

(让所有接口都不发送广播/组播的路由更新.=(passive 所有的接口)

```
R3(config-router)#no passive-interface serial 0
```

(单独让一个接口可以发送广播/组播更新)

Step2:让已经Pass的接口, 发送单播的路由更新.

```
R3(config-router)#neighbor 35.0.0.5
```

```
R3(config-router)#neighbor 35.0.0.5
R5(config-router)#neighbor 35.0.0.3(对方的IP)
```

debug ip rip (察看RIP路由的收发情况)

LAB5:

RIPv2可以关闭自动汇总,

手工汇总

在需要进行路由汇总的出接口:

```
R3(config-if-S1)#IP summary-address rip 175.100.0.0 255.255.192.0
```

在运行RIP的接口中进行手工汇总,把汇总后的结果从接口通告出去

```
R1(config-if)#ip summary-address eigrp 90 172.16.0.0 255.255.248.0
```

在运行EIGRP的接口中进行手工汇总,把汇总后的结果从接口通告出去

```
R5#
R 175.100.0.0/18
```

### 03晚-动态路由协议-RIP-EIGRP

LAB6:Offset-list/偏置列表,实现RIP路由的控制:

~~~~~

图03-01-RIP offset-list

step0:观察RIP的等价的负载均衡:

R4:

```
R 12.0.0.0/8 [120/1] Via
 []
```

step1:通过offset-list,实现RIP中的不等价网络拓扑中的,等价负载均衡:

step1:通过offset-list, 实现RIP中的不等价网络拓扑中的, 等价负载均衡:

(R4观察2.2.2.0/24)

出方向的控制(R2当前是汇总的):

~~~~~

p

Step2:通过ACL定义需要控制的路由:

R2(config)#access-list 2 permit 2.0.0.0 0.0.0.0

Step3:在R2的S1口做路由偏置(+1):

R2(config-router)#offset-list 2 out 1 serial 1  
(出口的偏置, 只影响下游路由器, 不影响本机)

R4#

R 2.0.0.0/8 [120/2]VIA 14.0.0.1, e0  
[120/2]VIA 24.0.0.2, Serial0

入方向的控制:(no auto-summary)

Step4:通过ACL定义需要控制的路由:

R4(config)#access-list 20 permit 2.2.2.0 0.0.0.0

step5:

R4(config-router)#offset-list 20 in 1 serial 0  
(入口的偏置, 本机和下游路由器都受影响)

```
R 2.2.2.0/24 [120/2]via 14.0.0.1, e0
 [120/2]via 24.0.0.2, s0
```

LAB7:RIP authentication/RIP 认证:(保证RIP的网络安全性)

~~~~~

Step1:在全局模式,配置KEY-CHAIN:

R4/R3#

```
key chain ccnp
```

```
key 1
```

```
key-string cisco1
```

step2:在接口中,调用key chain:

```
R5/R3(config-if)#ip rip authentication key-chain CCNP
```

Step3:在接口中,选择认证类型:(明文/密文)

```
R1/3 (config-if)#ip rip authentication mode text (明文) (默认的,可以不打这条命令)
```

```
R1/3 (config-if)#ip rip authentication mode md5 (密文)
```

LAB8:default-network:

~~~~~

图03-02-RIP-default-network

Step0:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial 1
```

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial 13.0.0.3
```

step1:

```
R1(config)#interface loopback 192
```

```
R1:ip add 192.168.1.1 255.255.255.0
```

step2:

```
R1(config)#router rip
```

```
R1(config-router)#network 192.168.1.0
```

Step3:

```
R1(config)#IP default-network 192.168.1.0
```

```
R* 0.0.0.0/0
```

```
R*192.168.1.0
```

LAB10:不连续子网(Discontinued Network)

图03-03-Discontinued Network

V1:

第二地址的虚拟管道,

要求一:与不连续子网同在一个主类网络(172.10.\*.\*)

要求二:与不连续子网的网络长度相同(/26)

Step1:配置第二地址:

Step2:

在R3增加172.10.0.0的网络宣告

```
R3(config)#router rip
```

```
R3(config-router)#network 172.10.0.0
```

V2:不需要第二地址:

Step1:

Router rip

Version 2

auto-summary(默认)

R3(config-router)#no network 172.10.0.0

Step2:在全网的RIP路由器上,自动关闭汇总

EIGRP(Enhanced IGRP)

EIGRP的特:

IGRP/EIGRP都是CISCO的私有协议.

1:是唯一的一种LS/DV的混合协议.

2:Rapid convergence

EIGRP拥有目前最快的网络路由收敛性.(依靠后备路由器/FS)

3. 配置简单,能够支持中型到大型网络.

4:Incremental updates

增量/触发更新.

5:EIGRP可以支持等价/不等价的负载均衡,

默认是支持等价负载均衡,

可以通过调整Variance,来实现不等价的负载均衡.

EIGRP到达同一个目标网络,可以同时有4条路径,最多6条.

6:EIGRP默认使用组播(224.0.0.10)进行路由更新.

也可以支持单播更新.

IGRP:广播:225.255.255.255

7:EIGRP可以支持VLSM,  
支持汇总:默认有自动汇总/手工汇总.  
EIGRP可以汇总到超网,CIDR

8:EIGRP可以支持多种网络协议:

IP/IPX/AT(AppleTalk)

EIGRP的3张表:

~~~~~  
NO.1:EIGRP的邻居表:

本路由器的接口,所直接相连的EIGRP邻居的信息.

NO.2:EIGRP的拓扑表:(详细的拓扑表)

本路由器,从自己的邻居那里,得到去往特定目标网络的(一切)可能的路径,  
都承载/存在于拓扑表中.

NO.3:从EIGRP形成的路由表:

是EIGRP路由器,从拓扑表中,择优将"去往特定目标网络开销最小  
的"路由,放入了路由表.

EIGRP和IGRP在AS号相同的情况下,可以实现自动重分布.

EIGRP的Metric值是IGRP的256倍.

EIGRP Packets

~~~~~

1:Hello:用于建立/维护EIGRP邻居关系.

2:Update:更新包:发送路由更新信息.

3:Query:查询包:当路由器丢失了原有的路由后, 会向邻居发送“查询请求”.

4:Reply:当被查询路由器, 收到“查询请求”后, 将自己知道的路由信息回应给发起查询路由器.

5. Ack:用于对EIGRP的可靠传输报文的进行确认.

## 04晚-动态路由协议-EIGRP

EIGRP的Hello包:

1:EIGRP路由器, 向224. 0. 0. 10, 发送Hello包, 同时也监听这个组播地址.

2:Hello包中, 包含了EIGRP的K值, 两个路由器的K值必需匹配, 如果不匹配无法建立邻居.

K值的默认值:K1=K3=1, K2=K4=K5=0

3. Hello包中, 包含了AS号, 两个路由器的AS号必需相同, 如果不相同, 无法建立邻居

Autonomous-System/自治系统.

4:两个建立EIGRP邻居关系的路由器的直链接口, 其IP地址必需在同一个IP子网, 否则无法建立邻居.

5. 即使EIGRP的Hello/hold计时器不完全匹配, 只要自己Hello的间隔不超过对方的Hold间隔, 邻居是可以建立的. 但默认情况下, 建议不要修改这两个计时器...

EIGRP计时器:

Hello Timer:

在大于T1链路, 点对点链路上, 默认5秒发送一次Hello包

在小于/等于T1的多点链路上, 默认60秒发送一次Hello包.

Hold Timer:

Hold timer默认是Hello Timer的3倍,

如果Hold Timer所定义的时间内,收不到对方的Hello包,邻居关系就会Reset.

在本路由器上设置hello time和hold time是告诉邻居的!!也就是说在本路由器上设置的hello time和hold time是影响邻居的!!

只要在本路由器上设置的hello time 小于hold time,那么邻居还是可以建立起来的,而不管邻居的hello time是多少!!

EIGRP的可靠传输报文:

Update/Query/Reply, 收到此包后, 需要发送ACK进行确认.

EIGRP的非可靠传输报文:

Hello/Ack, 收到此包后, 不需要进行确认.

EIGRP Retransmission Policy/重传机制:

EIGRP对于可靠传输报文, 有必需的确认机制,

如果没有收到对方的确认, 那么可靠传输包就会发生重传, 极限是16次,

如果达到极限, 都没有收到对方的确认, 那么就RTO.

RTO:Retransmission TimeOut

因为EIGRP的是windows size of one (stop-and-wait mechanism)  
所以如果在对一组邻居, 进行组播的路由更新时,  
有个别路由器响应特别慢,  
可能导致整个EIGRP网络的收敛效率低下.

解决方案是:

对正常的大部分路由器做组播更新,  
对特别慢的路由器, 单独进行单播更新.

EIGRP的DUAL算法: (EIGRP Diffusing Update Algorithm)



auto-summary

(系统默认的,路由在穿越主类网络边界时,会发生自动汇总,可以关闭)

Step2:

R1#show ip eigrp interfaces(察看当前正在运行EIGRP的接口)

R1#show ip eigrp neighbors (察看EIGRP的邻居表)

show ip eigrp topology detail-links(详细的拓扑表)

show ip eigrp topology(拓扑表)

show ip route eigrp(从EIGRP学到的路由表)

LAB2:EIGRP的路由的Metric值的准确计算:

~~~~~  
~~~~~

Step1:确定路由的入口:

(也就是访问该目标网络的数据包的出口)

Step2:收集各个路由的入口的BW/DL信息:

R2#show interfaces serial 1

        BW 1544 Kbit        DLY 20000 (这两个参数是用来控制三层协议的  
选路,而clock rate是反应链路上真实的速度,这两个参数可以在接口上更  
改,但仅仅是影响路由协议的选路)

Step3:

$Metric = \{(10,000,000/BW) + (DL/10)\} * 256$

BW:单位是:Kbps,多个路由入口中的带宽最小值.

DL:单位是:us(微秒),多个路由入口的延迟之和.

## 05晚-动态路由协议-EIGRP

### LAB3:Wildcard Mask in EIGRP

(通过反掩码,控制运行EIGRP的接口的范围

有哪些接口在运行EIGRP)

~~~~~  
~~~

Wildcard Mask/反掩码的匹配原则:

0:表示准确匹配

1:表示忽略不计

当EIGRP passive接口时, 不会向外路由, 也不会接收邻居发过来的路由.

(在所有路由器上, 关闭自动汇总)

Step0:

network 0.0.0.0 255.255.255.255 (路由器的所有接口都运行EIGRP)

Step1:要求以155.\*.\*.\*的所有接口都运行EIGRP:

network 155.0.0.0 0.255.255.255

Step2:要求以155.2.\*.\*的所有接口都运行EIGRP:

network 155.2.0.0 0.0.255.255

(反掩码不表示网络长度!!)

Step3:要求以155.2.3.\*的所有接口都在运行EIGRP:

155.2.3.0 0.0.0.255

Step4:

对于A类接口, 不写反掩码时, 其默认的反掩码是0.255.255.255

对于B类接口, 不写反掩码时, 其默认的反掩码是0.0.255.255

对于C类接口, 不写反掩码时, 其默认的反掩码是0.0.0.255

Step5:要求以215.5.5.5的这个特定接口运行EIGRP:

R5(config-router)#network 215.5.5.5 0.0.0.0

215.5.5.0 0.0.0.0 (这个写法是错误的, 因为路由器上根本找不到这样的接口)

结论:

EIGRP的反掩码

不控制EIGRP的接口在路由条目中, 表现出来的路由长度.

EIGRP/OSPF都是通过这种反掩码的方式, 只控制运行协议的接口范围.

LAB4:Automatic Summarization:

EIGRP在主类网络的边界/(On major network boundaries),  
会发生自动汇总, 是默认产生的.

自动汇总会将该子网的路由, 汇总到该主类的网络边界:

LAB5:Summarization: Manual (手工汇总, 是基于接口的)

step0:

```
R3(config)#router eigrp 100
```

```
R3(config-router)#no auto-summary
```

Step1:在汇总路由的出接口中, 手工汇总:

```
R3(config-if-S0)#ip summary-address eigrp 100 175.10.0.0
255.255.192.0 (5)
```

```
show ip route 172.16.0.0 255.255.248.0
```

察看某条路由的详细信息.

Step2:在生成汇总的EIGRP路由器上, 既有明细路由, 也有汇总:

```
D 175.10.0.0/18 (AD:5) is a summary, Null0
```

```
D 175.10.12.0/24 (AD:90), Ethernet0
```

但汇总路由一定比明细路由“短”,  
按照“最长配置原则”, 一定不会发到空接口,  
而是按照特定明细路由所指示的接口, 发送出去.,

路由条目的比较步骤:

1:首先按照“最长配置原则”, 优先选择路由长度最长的路由.

- 1:首先按照“最长配置原则”, 优先选择路由长度最长的路由.
- 2:假如, 有多条长度相同的路由, 才按照AD最小进行比较.
- 3:如果, 连AD也相同, 才比较每条路由的Metric值.

Step3:

在生成汇总路由的R3上,

```
R3#show ip route 175.10.0.0 255.255.192.0
```

LAB6:Default-network for EIGRP(在用户连接ISP的边缘路由器上/R3)

~~~~~  
~~~~~

Step0:网络背景:

通常ISP与用户之间是不运行IGP的:

0-1:

在ISP R3上做去往用户的静态路由:

```
R3(config)#ip route 0.0.0.0 0.0.0.0 35.0.0.5
```

0-2:

```
R5(config)#ip route 175.10.0.0 255.255.0.0 serial 0
```

R3#

```
router eigrp 90
```

```
redistribut static metric 1544 100 100 1 1500
```

Step3:内部的EIGRP路由器上察看由EIGRP生成的默认路由:

R1/2/4#

```
D*EX 0.0.0.0/0 [170/.....]via R1
```

Step4:

内网的EIGRP路由器上察看同EIGRP生成的默认路由:

R1/2/4#

D\*EX 0.0.0.0/0 [170/.....]Via R1

EIGRP产生默认路由的方法B:

~~~~~

Step1:同方法1.

Step2:创建一个私网地址,提供默认网络:

R3#

Step3:将私网宣告到EIGRP中:

R3#

router eigrp 90

network 192.168.1.0

Step4:指定这个网络是默认网络,它可以在EIGRP域中,以EIGRP路由的形式存在和传播:

R3(config)#ip default-network 192.168.1.0

Step5:R1/2/4上察看路由:

D\* 192.168.1.0/24 [90/

LAB7:EIGRP Equal-cost Load Balancing /等价负载均衡:

~~~~~

R2#show ip protocols

EIGRP maximum metric variance 1

show ip eigrp topology detail-linkks

show ip eigrp topology

show ip route ieigrp

LAB8:EIGRP Unequal-cost Load Balancing/不等价

R4观察175.10.12.0/24

P 175.10.12.0/24, 1 successors, FD is 2195456

R4(config)#router eigrp 90

R4(config-router)#variance 2

LAB9:EIGRP Route Authentication(路由协议的, 链路级别的, 链路认证)

~~~~~  
~~~~~

Step1:定义Key-chain

R1/R3#

key chain CCNP

key 1

key-string CISCO

Step2:

R1/R3#

interface s0/s1

ip authentication mode eigrp 100 md5

ip authentication key-chain eigrp 100 CCNP

如果EIGRP的认证失效, 连EIGRP邻居都无法建立.

LAB10:Adjusting the EIGRP Metric Weights(调整EIGRP的K值)

~~~~~

R3#show ip protocols

EIGRP metric weight k1=1, k2=0, k3=1, k4=0, k5=0

Router(config-router)#metric weights tos k1 k2 k3 k4 k5

在全AS的EIGRP路由器, 都必须有相同的K值, 建议不要轻易更改:

R5/R3(config-router)#metric weights 0 1 2 3 4 5

实际上, K值是EIGRP协议衡量:

BW/DL/Reliability/Loading/MTU, 这5个衡量路径优劣的不同参数的权重默认只考虑BW/DL,  
∴K1=K3=1 K2=K4=K5=0

## 06晚-动态路由协议-OSPF

### OSPF

理论:

OSPF三张表 (OSPF AD:110)

1. Neighbor table (列出了所有和本路由器直接相连的OSPF邻居)

2. Topology table (LSDB链路状态数据库)

列举了所有从自己的邻居那得到的LSA, 在同一个OSPF区域中的路由器, 都有完全一致的OSPF Database。一个OSPF区域, 就对应着一个OSPF Database。

3. Routing table (从OSPF这个路由协议, 学到的路由。)

在OSPF的数据库中, 通过SPF算法, 计算得到了路由。也称为: Forwarding Database

区域划分, 及划分的目的:

划分的目的:

1. 提高路由效率:

缩减部分路由器的OSPF的路由条目。

对某些特定的LSA, 可以在区域边界 (ABR) 上, 实现汇总/控制/过滤。

(通过OSPF的汇总路由/默认路由实现OSPF区域之间的全网互通)

2. 提高网络稳定性:

当某个区域内的一条OSPF路由出现抖动时, 可以有效控制受影响的波及面。

(对于大型的路由协议来说, 稳定是很重要的一个因素。)

3. OSPF VS. IS-IS的路由可扩展性的对比:

OSPF: 以Area0为BackBone

ISIS: 以Level2的链路为BackBone

OSPF对不同物理链路的处理分类:

OSPF Routing updates and topology information are only passed between FULL adjacent routers.

1. P2P (HDLC/PPP Serial/ Point2Point Sub-if) 一定要求是Full状态. (没有DR / BDR的选举的, 代表是E1, 两台路由器直连)

(没有DR / BDR的选举的, 代表是E1, 两台路由器直连)

2. BMA (EtherNet/TR/FDDI) (代表是局域网)  
Broadcast Multi-Access (有DR/BDR的选举的)

3. NBMA (FR/ATM/X.25) (代表是广域网, 如帧中继)  
Non-Broadcast Multi-Access (有DR/BDR的选举的)

关于MA (Multi-Access) 网络的DR/BDR的选举:

Step1. 根据OSPF路由器的OSPF接口的优先级选举DR/BDR:

每个接口默认的优先级都是: 1。

其中优先级最大的成为DR, 次大的成为BDR, 其它的都是DR-Other。

如果有路由器的Pri:0, 放弃DR/BDR的选举, 成为DR-Other。

(OSPF Priority:0~255)

Step2. 如果接口的优先级相同, 将使用Router-ID来进行DR/BDR的选举:

其中Router-ID最大的成为DR, 次大的成为BDR, 其它的都是DR-Other。

在选出DR/BDR后, 如果有新的优先级更高的路由器加入, 那么新加入的路由器并不会成为DR/BDR, 需要在下次选举中才能生效。

OSPF的SPF算法:

1. 每一个OSPF区域, 就对应着一个独立的OSPF Database (LSDB)
2. 每一个OSPF路由器, 都生成了以自己为根的, 一棵SPF树。
3. 从本路由器出发, 到特定目标网络的整体开销最小的那个路径, 成为最佳路径。
4. 那么这条最佳路径, 就成为OSPF这个协议, 提交给路由表的, 到达这个目标网络的路由。

最长匹配, AD比较, Metric比较

先比较最长匹配如 (10.2.2.0/24, 10.0.0.0/8, 将选10.2.2.0/24, 而不管路由协议的AD), 然后比较AD, 最后比较Metric。

LSA的传播更新规律 (OSPF是LS协议, 无需遵循水平分割, DV协议才遵循水平分割。)

Step1. 如果本路由器从来没有收到过此LSA, 那么路由器就将其加入LSDB,

LSDB，并且转发/泛洪此LSA，同时继续SPF计算，得出达到此目标的最佳路由。

Step2. 如果本路由器，曾经受到过描述同一个网络的LSA：

- 2-1. 如果新收到的LSA序号与自己已有的相同，则丢弃此LSA。
- 2-2. 如果新收到的LSA序号比自己已有的更新，则同Step1，去计算最佳路由。
- 2-3. 如果新收到的LSA的序号，比自己的更旧，就将自己较新的LSA，发送给源。

OSPF的5种数据包

1. Hello（建立和维护邻居（Neighbor）关系，路由器发送Hello包的缺省时间间隔是10秒）
2. DBD(Database Description)
3. LSR(LinkStatus Request)
4. LSU(LinkStatus Update) (LSA是包含在LSU中的)
5. LSAck

OSPF的Protocol ID:89（EIGRP:88）

在OSPF的Hello包中，影响建立邻居关系的4个关键因素：

- Hello/dead interval
- Area-ID（链路所在的Area ID）
- Authentication password（OSPF认证的密码）
- Stub area flag（NSSA标示位）

这四个信息必须匹配才能建立邻居

在BMA网络和点对点网络上，默认的Hello Interval值是10秒，Dead Interval值是40秒。在NBMA网络上，默认的Hello Interval值是30秒，Dead Interval值是120秒。

修改Hello Interval和Dead Interval的值：（在接口上修改）

R1(config-if)#ip ospf hello-interval time(time的取值为1-65535秒)

R1(config-if)#ip ospf dead-interval time(time的取值为1-65535秒)

OSPF邻接关系的建立过程

1. Down（路由器A从运行OSPF的接口以组播地址224. 0. 0. 4发送Hello数据包）
2. Init（所有收到从路由器A发送来的Hello数据包的路由器，都把路由器A添加到自己的邻居Neighbor列表中）
3. Two Way（所有收到路由器A的Hello包的路由器都向其发送一个单点传送的回复Hello包，其中包含有它们的信息。路由器A收到这些信息后，检查这些数据包，把哪些Hello包的邻居域中有自己ID的路由器也加入自己的邻居列表中。在这个过程中同时选举出DR和BDR）
4. Exstart（DR和BDR与其他的路由器建立相邻关系（Adjacency））。

的邻居列表中。在这个过程中同时选举出DR和BDR)

4. Exstart (DR和BDR与其他的路由器建立相邻关系 (Adjacency)。

5. Exchange (由DR向其他路由器发送数据库描述数据包 (DBD, Database Description)。DBD有序号, 由DR决定DBD的序号)

6. Loading (发送链路状态请求包的过程)

7. Full (路由器及哪个新的链路状态条目添加到它们的链路状态数据库中。

当所有的LSR都得到答复时, 相邻的路由器就被认为达到了同步并处于

“Full”状态了。路由器必须在达到Full状态后才能正常转发数据。此时区域内的每个链路应该都有相同的数据链路状态数据库。)

OSPF数据包的发送地址

DR/BDR notifies LSU on 224.0.0.5 (映射到二层MAC地址:

010005e00000005)

DR-Other notifies LSU to OSPF DR on 224.0.0.6 (映射到二层MAC地址:

010005e00000006)

DR负责宣告整个网络的路由更新, BDR或DR-Other只能先把路由更新先发给DR, 然后再由DR发给BDR和DR-Other

每次收到LSU, 路由器在重新计算路由表之前等待一段时间, 默认是5秒。每个LSA都有一个老化 (Aging) 计时器, 到期时由产生该LSA的路由器再发送一个有关该网络的LSU以证实该链路仍然是活跃的, 这个Aging时间默认是1800秒。

DR和BDR是在交换Hello数据包的过程中选举出来的, 然后其他路由器都与DR和BDR建立相邻关系。每台DR-other路由器都只与DR和BDR建立相邻关系 (Adjacency), 交换链路状态信息。

LAB1. OSPF的Router-ID (要求全网唯一)

一旦启动OSPF, 立刻确定Router-ID

通过此命令察看Router-ID:

R1#show ip ospf

在OSPF路由器上, 确定Router\_ID的3个优先级别:

Step1. (建议使用router-id命令来确定Router-ID)

通过router-id命令, 修改Router-ID, 其优先级别最高, 也是建议的。

R1(config)#router ospf 110

R1(config-router)#router-id 100.0.0.1

Step2. 假如没有通过router-id命令指定router-id, 那么路由器会自动的将自己的环回口的IP, 作为router-id. 如果存在多个环回口, 那么路由器会自动的悬着一个IP地址最大的那个IP作为自己的Router-ID。

Step3.

如果路由器上，连一个环回口都没有，那么路由器会自动从当前是Active（激活状态下：UP/UP）的物理接口中，选择IP地址最大的那个接口的IP作为自己的Router-ID。这是很不稳定的，不建议的方法。

LAB2. 通过反掩码控制有哪些接口，在运行OSPF（在OSPF/EIGRP中，network命令中携带的反掩码不表示接口网络长度，而表示运行路由协议的接口的范围，即有哪些接口在运行OSPF）

反掩码/通配符:wild card bits

反掩码原则：

0:表示准确匹配

1:表示忽略不计

LAB3.

show ip ospf neighbor (detail)（查看路由器的OSPF邻居表，当前有哪些OSPF的邻居, DR/BDR/DR-other状态）

show ip ospf interface（查看有哪些接口在运行OSPF，本路由器是DR，或者BDR，还是DR-other，还有优先级）

show ip ospf database

show ip route ospf

LAB4. DR/BDR的选举：（只发生在多路访问网络/Multi-Access Network，BMA和NBMA）

1. 在点对点链路，是没有DR/BDR的选举

2. 在BMA网络中：

2-1. OSPF首先通过优先级，控制DR/BDR的选举：

优先级越大，越可能成为DR。OSPF路由器的优先级，默认是1。

如果需要进行DR的人为控制，应该建议，通过OSPF的接口优先级进行控制。

修改特定接口的优先级

R1(config)#int s0

R1(config-if)#ip ospf priority 10

R1#clear ip ospf process（清OSPF进程）

特别注明：OSPF的优先级是针对某个特定的MA接口而言的，不是针对整个路由器的。

2-2. 如果OSPF路由器的优先级，全部都是默认值1，路由器默认通过Router-ID, 选举DR/BDR，Router-ID最大的成为DR，次大的成为BDR。其余的统统都是DR-other。

3. 在Hub&Spoke的NBMA网络中，中心点（HUB）应该成为DR。

结论:

1. 同一个路由器的不同MA接口，可能在不同的MA网络中，充当不同的DR/BDR/DR-other.
2. 在一个MA网络中:  
DR/BDR与所有的邻居都是Full状态，DR-Other与DR/BDR是Full的，但与别的DR-Other是2way状态。

特别注意:

只有Full状态才能交换路由信息。

## 07晚-动态路由协议-OSPF

Advance OSPF Controlling

OSPF的基本实验配置

建议宣告每一个Router-ID的网络，便于网管Telnet。如Router-ID是195.100.0.1，则宣告网络network 195.100.0.0 0.0.0.255 area 0。

影响OSPF Metric计算的3种操纵方法

在路由协议计算Metric/Cost值时，只计算路由的入口

OSPF计算Cost的公式:

$Cost = 100,000,000 / BW$  ( $10^8 / BW$ , BW的单位是bps)

R1访问35.0.0.0网络的Cost是 $10^8 / BW + 10^8 / BW$  (BW是bps, 即1544Kbps=1,544,000)

Show interfaces serial 1 (BW 1544 Kbit)

show interfaces ethernet 0 (BW 10,000 Kbit)

show interfaces fast-ethernet 0 (BW 100,000 Kbit)

show interfaces loopback 5 (BW 8,000,000 Kbit, 环回口的OSPF Cost固定为1，而不管参考带宽的值是多少)

方法1: 直接修改接口Cost值

R2(config)#int s0

R2(config-if)#ip ospf cost 100 (用这个命令修改后，就不再用 $10^8 / BW$ 这个公式计算Cost了)

方法2: 修改接口的带宽BW

R3(config)#int s0

R3(config-if)#bandwidth 2048 (单位是Kbps, 2.048Mbps/E1)

方法3:修改参考带宽（分子）

工程/题目需求:

考虑到为了的网络发展，要求将来再10Gbps链路上，其OSPF Cost为：10，所以我们要把分子更改为100G, 即 $10^{11}$ 。

```
R1(config)#router ospf 110
```

```
R1(config-router)#auto-cost reference-bandwidth 100000 （单位是Mbps）
```

要求：一旦准备更改OSPF的参考带宽，就必须在所有的OSPF路由器（包含不同区域）上一起更改。

“ip ospf cost”设置的值要优先于 “auto-cost reference-bandwidth”命令计算出来的值。

特别注意:

1. 路由入口的定义。

2. 同步串行的同步时钟速率: clock rate 2400，只会影响链路的真实物理速率（带宽），不会影响OSPF的Cost的计算。

但接口种配置的 “Bandwidth 2048”，只影响OSPF的Cost的计算，影响OSPF选路，不影响真实物理速率

3. 对于以太网，其速率就等于其接口的带宽。

改接口速率:

```
int fastethernet 0/1
```

```
speed 10(100)
```

OSPF的路由汇总

RIP和EIGRP的路由汇总是设置在接口上的，它们是DV协议。

链路状态路由协议的路由汇总需要在路由进程中设置，链路状态协议没有自动汇总的特性。

因为在何种路由协议的路由汇总中:

生产的汇总路由包含范围过大，则很可能形成路由黑洞。

生成的汇总路由包含范围过小，则很可能丢失部分明细路由。

所以，默认情况下，在工程/题目没有指定汇总的长度的时候，应该进行最精准的汇总。

1. OSPF的域间汇总，发生在连接不同OSPF区域的ABR上。

```
0 IA (IA:inter-area)
```

ABR: (互联了2个 , 或者2个以上的OSPF区域的路由器。)

2. OSPF的域外汇总, 发生在OSPF与别的路由协议相连的ASBR上。

0 E1 (OSPF external type 1)

0 E2 (OSPF external type 2)

ASBR: (在OSPF区域的边界上, 互连了OSPF和别的其它路由协议的路由器。)

LAB3. OSPF的域间路由的汇总

Step1. 在Area24中的路由器R4上, 模拟4条/26的路由:

175. 14. 14. 1/26

175. 14. 14. 65/26

175. 14. 14. 129/26

175. 14. 14. 193/26

Step2. 在R4上, 将明细路由宣告到OSPF中

R4(config)#router ospf 110

R4(config-router)#network 175. 14. 14. 0 0. 0. 0. 255 area 24

Step3

Step4. 在明细路由所在的区域Area 24的ABR上, 进行OSPF域间路由汇总:

R2(config)#router ospf 110

R2(config-router)#area 24 range 175. 14. 14. 0 255. 255. 255. 0

原明细路由所在区域

Step5.

R1/R3, 只有汇总路由

0 IA 175. 14. 14. 0/24

R2, 即有明细路由, 也有汇总路由

R4, 没有汇总, 只有明细。R4代表整个Area24中的所有路由器。

路由汇总黑洞 (null0), 只会出现在做路由汇总的路由器上。

在OSPF协议下向区域内产生一条默认路由的语法:

R1 (config-router)#default-information originate [always]

使用Default-information originate命令产生缺省路由的前提是, 使用该命令的路由器必须存在一条默认路由。

如果不使用参数(always), 那么路由器上必须存在一条0/0的默认路由, 它把默认路由通过到整个区域, 否则该命令不起作用。但使用参数(always)时, 无论路由器上是否存在0/0的默认路由, 使用该命令的路由器总会向区域内注入一条默认路由。

LAB4. OSPF的域外路由的汇总。

R3/R5上运行EIGRP，

R5上，

178.15.15.177/28

178.15.15.161/28

178.15.15.129/28

177:10110001

161:10100001

129:10000001

在R3（ASBR）上将域外的EIGRP路由，重分布（Redistribute）到OSPF中。

R3(config)#router ospf 110

R3(config-router) redistribute eigrp 90 subnets

在R3（ASBR）上将域外的路由，做OSPF的域外汇总：

R3(config)#router ospf 110

R3(config-router)#summary-address 178.15.15.128 255.255.255.192  
（路由长度，即/26）

E1 - OSPF external type 1,

（在路由传播的路径上，OSPF的Cost会随着路径的远进叠加链路的开销，其Cost会发生改变）

LAB5. OSPF Virtual Links(虚链路)

Step1. 在Area0和非连续区域（Area35）的ABR上（R2/R3）

在virtual link穿越的区域内（area 123）的ABR上，互相指向对方的Router-ID

R2的Router-ID是192.100.0.2

R3的Router-ID是192.100.0.3

R2(config-router)#area 123 virtual-link 192.100.0.3

R3(config-router)#area 123 virtual-link 192.100.0.2

Step2. 检查OSPF的virtual link的连接状态:

```
process 110,
nbr 192.100.0.2 on ospf_vl1 from loading to full, loading done
```

R2#show ip ospf virtual-links

```
virtual link ospf_vl0 to router 192.100.0.2 is up Adjacency State
Full(检查Full, 不能看UP)
```

Step3. 在骨干区域的路由器, 就可以接受到非连续区域的路由。全网路由正常。

Step4. 每个路由器的数据库的个数:

```
R4:1个 (Area 0)
R5:1个 (Area 35)
R1:1个 (Area 123)
R2:2个 (Area 0, Area 123)
R3:3个 (Area0, Area 35, Area 123)
```

Step5. OSPF virtual link的应用要点:

5-0. By default, all areas must connect to area 0

5-1. 在大型网络工程中, 由于历史原因, 导致网络扩展不佳, 迫于网络扩容的原因, 被迫新建OSPF区域, 使用OSPF虚链路。

5-2. 在大型网络中, 处于网络冗余考虑, 选择合适的ABR做OSPF-VL, 避免因个别物理链路的中断, 导致整个OSPF区域的全网中断。

5-3. 导致OSPF的Area0区域出现双BackBone

## 09晚-动态路由协议-OSPF

LAB1. 3口的FR-SW

Step1. 在R2上关闭路由功能, 启用FrameRelay交换功能。

Step2.

配置接口时钟

```
interface serial 1/0
clock rate 2000000
```

```
encapsulation frame-relay
frame-relay lmi-type cisco
```

```
frame-relay lmi-type cisco
frame-relay lmi-type dce
```

Step3. 配置FR的路由：（PVC401/PVC104）

在FR-SW的角度观察：从S1，以PVC401进入交换机的数据，将以PVC104，从S0离开交换机

```
FR-SW2(config-if-S1)#frame-relay route 401 interface serial 0 104
```

从S0，以PVC104进入交换机的数据，将以PVC401，从S1离开交换机

```
FR-SW2(config-if-s0)#frame-relay route 104 interface serial 1 401
```

两端口的FR-SW配置完成。

在FR交换机上检查FR路由：

show frame-relay route（status必须是active才是正确的。）

Step4. 开始配置FR-SW的第3个口，构建R2-R3之间的Tunnel.

```
FR-SW3(config-if-e0)#ip ad 23.0.0.3 255.255.255.0
```

```
FR-SW2(config-if-e0)#ip ad 23.0.0.2 255.255.255.0
```

```
FR-SW3(config)#
interface tunnel 3
tunnel source 23.0.0.3
tunnel destination 23.0.0.2
```

```
FR-SW2(config)#
interface tunnel 2
tunnel source 23.0.0.2
tunnel destination 23.0.0.3
```

Step5. 在FR-SW3的S1口，也要有接口的配置（同Step2）

Step6. PVC405/PVC504

```
FR-SW2(config-if-s1)#Frame-relay route 405 interface tunnel 2 1000
```

```
FR-SW3(config-if-s1)#Frame-relay route 504 interface tunnel 3 1000
```

PVC105/PVC501

```
FR-SW3(config-if-s1)#Frame-relay route 501 interface tunnel 3 1001
```

```
FR-SW2(config-if-s0)#Frame-relay route 105 interface tunnel 2 1001
```

3口FR SW配置完成

```
FR-SW2(config-if-s0)#Frame-relay route 105 interface tunnel 2 1001
```

3口FR SW配置完成

FR-SW的基本检查:

1. show frame-relay route
2. 在FR的用户端R1/R4/R5上, show frame-relay pvc, PVC Status必须是Active
3. 在FR的用户端配置IP地址, 进行ping的L3测试。show frame-relay map (观察FR的自动反向ARP是否能够成功建立, L3的IP地址, 与L2的DLCI的映射)

OSPF Over FR (NBMA网络)

LAB4. Broadcast: (Over FULL Mesh 的NBMA网络中: FR)

Full Mesh: 在N节点的网络中, 每个节点都有到别的所有节点的, 直接相连的连接/PVC.

在FR网络里面, 动态映射dynamic默认允许广播数据通过, 手动/静态映射(static) 需要添加broadcast来允许广播数据通过。

Step1. Show frame-relay pvc

Step2. 确认FR支持广播数据流通过

在接口中的映射语句中:

```
frame-relay map ip 100.0.0.5 105 broadcast
```

Step3. 启动OSPF, 指定运行模式是Broadcast:

```
int s0
```

```
ip ospf network broadcast
```

注意: Serial接口如果封装了FR, 其OSPF的运行模式, 默认是NBMA

(NON\_BROADCAST), 默认情况下, 不发送OSPF的组播包的, 所以无法自动建立邻居。

如果Serial接口封装的是HDLC/PPP, 其OSPF的运行模式是: P2P。

Step4. 邻居问题:

```
show ip ospf neighbor 察看OSPF邻居
```

Step5. DR问题

确认DR/BDR的正确选择。

因为PVC是Full mesh的, 所以每个路由器都能收到所有别的路由器的Hello包, 所有能够正确的选举出DR和BDR

Hello包，所有能够正确的选举出DR和BDR

Step6. 下一跳问题：

手工next-hop：

（通过手工做映射或者自动FR-inverARP，得到OSPF的路由的下一跳，的可达性）

LAB5. RunMode:NBMA (NON-BROADCAST)

(Full Mesh或Hub&Spoke的NBMA网络中，不支持广播流量通过)

Hub&Spoke

Step1. 邻居问题：

当前无法建立邻居，解决办法：OSPF的单播更新（通过单播建立OSPF邻居）

在中心点R4上：

R4#

```
router ospf 110
```

```
neighbor 100.0.0.1
```

```
neighbor 100.0.0.5
```

无需在对端进行同样的nei命令

虽然邻居已经成功建立，但出现了DR/BDR选举错误，很可能导致OSPF数据库混乱。

{

单播方法：

RIP: 1.passive interface 2.neighbor xxxxx（两端都要做）

EIGRP: 直接neighbor xxxxx(两端都要做)

OSPF:直接neighbor xxxxx(在中心，单边做)

}

Step2. DR问题

在Hub&Spoke网络拓扑中：的DR选举原则：

一定要让Hub成为DR, 其余的Spoke成为DR-Other, 无BDR。

在中心点R4#

```
interface Serial 0
```

```
ip ospf priority 10（通过OSPF的接口优先级，控制DR的选举）
```

在其余的Spoke, R1/R5

```
Interface Serial 0
```

```
ip ospf priority 0
```

DR/BDR选举正常了，OSPF数据库出现了能够学到OSPF路由，在中心点访问所有的分支点都是可通的。但是在分支点访问别的分支点时，路由的下一跳不

所有的分支点都是可通的。但是在分支点访问别的分支点时，路由的下一跳不可达。所有在Spoke之间，不能互通。

### Step3. Next-Hop下一跳问题

对于分支点R1/R5，其下一跳不可达问题，不是没有路由，而是在进行二层封装时，没有合适的FR的2层封装。

解决办法：

应该通过FR map这个命令，完成L3:IP L2:DLCI进行映射，来解决

到达别的分支点的IP数据包，统统都发往去中心点的PVC

```
R1#frame-relay map ip 100.0.0.5 104
```

```
R5#frame-relay map ip 100.0.0.1 504
```

LAB6. OSPF Run Mode:Point-to-Multipoint Non-broadcast（适合Hub&Spoke的NBMA网络，不支持广播流量通过的）

Step1: 在每个分支点，都只做到达中心点的映射，无需做分支点之间的映射。

Point-to-Multipoint Non-Broadcast的优点之一

Step2: 指定OSPF的运行模式

```
int s0
```

```
ip ospf network point-to-multipoint non-broadcast(不发组播)
```

邻居问题：

解决方法同LAB5.

DR问题：

在P2MP No-BR0，是没有DR/BDR的概念的，根本不存在DR问题。

下一跳问题：

由于P2MP的自动建立下一跳特征，导致分支点间无需做映射，

LAB7. OSPF RunMode: Poinit-to-Multipoint

（Hub&Spoke的NBMA网络中，要求支持广播流量通过）

指定OSPF的运行模式：P2MP

```
int s0
ip ospf network point-to-multipoint
```

邻居问题:

当前模式是P2MP, OSPF会自动发送组播, 去尝试建立邻居, 同时L2 FR也允许广播, 所有邻居自动建立。

DR: 问题

同LAB6

下一跳问题:

同LAB6

## 10晚-动态路由协议-OSPF

帧中继的子接口选用原则:

0. 在一个封装FR的物理接口上, 可以同时承载多条PVC。

为了网络的可扩展性, 建议不论在考试环境还是在工程环境中, 都应该优先考虑使用子接口

1. 应该创建几个子接口: 在一个物理接口中, 对应着几个网络, 就应该建几个子接口。一个IP子网对应着一个子接口。

2. 选用哪种子接口? (点对点/多点)

在一个子接口中: 如果对应着一个点, 那么子接口类型应该是P2P。默认情况下, 其OSPFRunning Mode是Point-to-Point。OSPF对待这种子接口就像对待点对点串行链路一样。

如果对应着多个点, 那么子接口类型应该是Multi Point。默认情况下, 其OSPF的Running Mode是NBMA。OSPF对待这种子接口就像对待FR的主接口/物理接口一样。

LAB8. 此实验是将主接口配置, 全部迁移到多点子接口中:

在同一个网络的NBMA网络中, 其主接口或多个子接口的配置, 是完全一样的。

LAB9. 通过P2P子接口, 实现NBMA网络的优化/简化配置:

Point-to-Point子接口:

默认就有传输广播流量的能力。

默认就可以Ping通自己的接口。

不能使用Frame-relay map，应该使用frame-relay interface-dlci的命令。

Step1. 为每一条PVC，规划一个不同的IP子网：  
有多少条PVC，就有多少个点对点子接口，多少个子网。

Step2. 在每一条PVC的两端，都构建一个P2P子接口。  
封装主接口（物理接口），然后关闭RARP，不需要配IP地址。  
encapsulation frame-relay  
no frame-relay revert-arp

配置子接口

```
R4(config-if)#interface s0.401 point-to-point
R4(config-subif)#ip address 100.0.0.2 255.255.255.252
R4(config-subif)#frame-relay interface-dlci 401 （不能用frame-relay map命令）
```

Step3. 运行OSPF

Step4. 邻居问题：  
因为，只要是P2P子接口，其就具备让广播或组播通过的能力（L2）。  
只要是P2P子接口，其L3 OSPF的运行模式：默认就是P2P。  
所有，OSPF邻居一定可以建立。

Step5. DR问题  
在P2P链路上，运行P2P模式，根本没有DR, 所有无DR问题。

Step6. 下一跳问题  
因为，每个分支点收到的OSPF路由的下一跳，都是本分公司的哪条PVC的对端，（也就是中心点/总公司）  
所有，下一条不在成为问题。

LAB10. OSPF认证（保证网络安全）

按照参与认证的成员，进行分类：

1. 链路认证

（只能确保两路由设备之间的链路是安全的）

链路：两个路由器之间的物理链路。

2. 区域认证

（能够确保整个OSPF区域的所有路由器的安全性）

### 3. 虚链路认证 (Virtual Link)

(能保证跨越多个路由器的, Virtual Link的连接安全性)

按照认证的加密方式, 进行分类:

1. 明文认证
2. 密文认证

步骤:

Step1. 在接口配置密码

Step2. 在进程 (区域认证) 启用认证。或: 在接口 (链路认证) 启用认证。

#### 1-1:链路的明文认证

(int S 1)ip ospf authentication-key cisco1(配置认证密码)

(int S 1)ip ospf authentication (启动明文认证)

#### 1-2. 链路的密文认证

(int S 1)ip ospf message-digest-key 1 md5 cisco1(配置认证密码)

(int S 1)ip ospf authentication message-digest (启动链路认证)

#### 2-1:区域的明文认证

对于区域认证: 凡是Area0的路由器, 都必需参与认证, 如果不参与认证/认证不成功, 将无法交换OSPF路由信息。

(int s 0)ip ospf authentication-key cisco2

(router ospf 110)area 0 authentication

如果两台路由都没有在接口上打上密码, 而且启用了区域或链路的认证, 则也可以正常地建立起邻居。

存在问题:

Area0中的路由器, 没有Area35的路由

原因是: R1-R3之间的VL, 出现中断,

R3相当于是Area0的路由器, 但是没有参加Area0的区域认证。

解决方法: 在建立VL的R3上, 只要宣告:

R3#

router ospf 110

area 0 authentication

即使R3没有配置密码, 也可以成功与BackBone进行认证。

```
router ospf 110
```

```
area 0 authentication
```

即使R3没有配置密码，也可以成功与BackBone进行认证。

又带来了VL的安全问题。

## 2-2: 区域的密文认证

```
(int s 0)ip ospf message-digest-key 3 md5 cisco2
```

```
(router ospf 110)area 0 authentication message-digest
```

(明文认证对应明文的密码!!!密文认证对应密文的密码!!!)

3:key ID 两端的key ID不一样,即使密码一样也会认证失败!!所以,key ID和密码要两端都一样!!

存在问题:

同: 区域的明文认证

解决VL安全性的办法: 单独进行基于VL的认证。

## 3-1. VL的明文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication
```

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication-
key cisco3
```

## 3-2. V1的密文认证

```
(router ospf 110)area 123 virtual-link 100.0.0.3 authentication
message-digest
```

```
(router ospf 110)area 123 virtual-link 100.0.0.3 message-digest-
key 1 md5 cisco3
```

查看验证类型:

R1#

```
00:18:30: OSPF: Rcv pkt from 192.168.10.65, FastEthernet0/1
: Mismatch Authentication type. Input packet specified type 1, we
use type 0
```

type0:不验证,默认状态

type1:明文验证

type2:密文验证

IS-IS

能同时支持IP和CLNP网络

Link-state routing protocol

SPF algorithm

Level1:local area route( system ID)

Level2:the route between areas (Area ID)

1.L1 Router

(OSPF internal nonbackbone router): Intra-area routing, L1 LSDB

2.L1/L2 Router (OSPF BackBone ABR)

Inter-area & Inter-area routing

separate L1 and L2 LSDBs

advertises default route to L1 routers in the Area.

3.L2 Router(OSPF backbone routers):

Inter-area routing.

L2 area LSDB.

LAB1. 使用IS-IS协议构建集成的IP网络:

Step1. 规划IS-IS区域, 规划每个路由器的NET (network entity title) / 网络实体标识。

NET: -----|-----|-----

Area-ID, System-ID, NSEL

1-13byte, 6byte, 1byte(路由器固定是00)

```
router isis
```

```
net 49.0035.0000.0000.0002.00
```

Step2. 配置IP地址:

Step3. 在接口中激活IS-IS的运行 (让IS-IS为这个接口所在的网段进行路由)

```
int s0
```

```
ip router isis
```

Step4. 察看IS-IS的邻居建立:

```
show clns neighbors
show ip route
```

Step5. 控制接口的IS-IS的Level类型 (默认是L1L2)

```
int s0
isis circuit-type level-2/Level-1
```

## 11晚-路由控制

(已完成实验)

Route-Controlling

Redistributing/重分布、重分发

将A的路由, 重分布到B。

使A路由协议中的路由, 以外部路由的形式, 进入B路由协议。

default-metric/默认的度量值 (种子度):

A协议中的路由, 进入B路由协议后, 在B协议中表现出来的, 默认的Metric值 (度量值)

default-metric在不同的路由协议中的默认取值:

如果有外部路由 (不管此路由源自哪种协议), 进入以下的路由协议, 默认在这些协议中表再找 Metric是:

1:DV协议 (RIP/IGRP/EIGRP), 其默认Metric是无穷大 (不可达不可用)

### Default Seed Metrics

| Protocol   | Default Seed Metric                    |
|------------|----------------------------------------|
| RIP        | infinity                               |
| IGRP/EIGRP | infinity                               |
| OSPF       | 20 for all execept BGP, which is 1     |
| IS-IS      | 0                                      |
| BGP        | BGP metric is set to IGP metric value. |

在两种路由协议进行重分布时:

要特别注意路由的控制/过滤, 避免路由环路的出现。

如果控制不慎, 可能出现路由环路。

如果控制不慎，可能出现路由环路。

一般不建议使用双向出口的重分布

三种建议解决办法：

- 1) 在一个方向上做重分布，另外一个方向做静态、默认路由。
- 2) 在一个方向上做重分布，反方向做带过滤的重分布。
- 3) 在一个方向上做重分布，反方向做带修改AD的重分布。

LAB2：将外部路由，重分布EIGRP中：

=====

Step1: 在路由协议的边界路由器R1:

```
R1 (config) #router eigrp 90
```

```
R1 (config-router) #redistribute rip
```

没有指定Metric, 默认就是无穷大

EIGRP 区域中的R3/R5都没有RIP的路由。

Step2: 添加Metric参数(把RIP重分布到EIGRP里面)

```
R1 (config-router) #redistribute rip metric 1544 2000 255
```

```
1 1500
```

BW DL Reliability

Loading MTU

1544(Bandwidth metric in Kbits per second)

2000(IGRP delay metric, in 10 microsecond units)

255(IGRP reliability metric where 255 is 100% reliable)

1 (IGRP Effective bandwidth metric (Loading) where 255 is 100% loaded)

1500 (IGRP MTU of the path)

```
R4#ping 35.0.0.5 !!!!!
```

```
R4#ping 5.5.5.5 !!!!!
```

把EIGRP重分布到RIP中：

```
redistribute eigrp 90 metric 3
```

90: Autonomous system number

3: Default metric,,, (本路由器会把Metric为3的外部路由发出去, 也就是说, 下一跳收到重分布过来的路由是3跳, 而下一跳的下一跳收到的就是4跳!)

**LAB3:** 将外部路由, 重分布OSPF中:

=====

**Step1:**

R1 (config) #router ospf 110

R1 (config-router) #redistribute rip

**"Only classful networks will be redistributed"**

只能让主类的路由重分布到OSPF, 子网的路由将不被重分布到OSPF中

R5#

0 E2 14.0.0.0/8

**Step2:**

R1 (config-router) #redistribute rip subnets

所有路由, 包括子网路由都被重分布到OSPF中

R5#

0 E2 25.0.0.0/16

0 E2 24.0.0.0/24

0 E2 12.0.0.0/24

0 E2 14.0.0.0/8

**Step3:进入OSPF后的外部路由的类型: E1/E2:**

默认是E2型, 其OSPF的cost/Metric值, 不会随着路径远近的变化而变化。  
(无法反映路径的远近)

R3# 0 E2 14.0.0.0/8 [110/20]

R3# 0 E2 14.0.0.0/8 [110/20]

R1 (config-router) #redistribute EIGRP 90 Subnets metric-type 1

OSPF E1, 会随着路径的远近, 其Cost会累加:

R3# 0 E1 14.0.0.0/8 [110/84]

R3# 0 E1 14.0.0.0/8 [110/148]

```
R3# 0 E1 14.0.0.0/8 [110/84]
R3# 0 E1 14.0.0.0/8 [110/148]
```

Step4:进入OSPF后的路由的Metric值: (默认是20)

```
R1 (config-router) #redistribute EIGRP 90 Subnets metric-type 1
metric 100
```

**LAB 4:** 将外部路由, 重分布IS-IS中:

=====

Step1: 将rip路由重分布到ISIS中

```
R2 (config) #router isis
R2 (config-router) #redistribute rip
```

Step2:将ISIS路由, 重分布到RIP:

```
R2 (config) #router rip
R2 (config-router) #redistribute isis metric 2
```

Step3:IS-IS路由重分布到别的协议时, 特殊情况:

在边缘路由器R3上, 没有将直链的34.0.0.0路由重分布到外部路由协议RIP中

解决方案: 进行重分布直链路由

```
R4 (config) #router rip
R4 (config-router) #redistribute connected metric 1
```

**LAB5:** 重分布直链路由:

=====

```
R4(config)router A
R4 (config-router) #redistribute connected metric 1
```

在RIP中, 如果重分布直链路由的话, 在进程中

```
Redistribute connected = redistribute connected 1
```

这时, RIP会把直链路由为1跳发给对方, 也就是说, 在对方的路由表里面显示的是1跳!!

```
redistribute connected 2
```

这时, RIP会把直链路由为1跳发给对方, 也就是说, 在对方的路由表里面显示的是2跳!!

LAB6: 重分布静态/默认路由:

=====

R4#

```
router rip
 redistribute static metric 1
ip route 0.0.0.0 0.0.0.0 34.0.0.3
or
ip route 4.0.0.0 255.0.0.0 34.0.0.3

ip route 35.0.0.0 255.0.0.0 34.0.0.3
ip route 34.0.0.0 255.0.0.0 34.0.0.3
```

## 12晚-路由控制

(已完成实验)

LAB7: RIP中的被动接口 (passive)

=====

被动接口:

只接收RIP路由, 而不发送RIP路由

R5

```
router rip
 passive-interface s0
```

R5#

```
router rip
 passive-interface default
 no passive-interface s0
```

除了s0外, 所有接口都不往外发送路由

单播更新: (R1-R2)

```
router rip
 passive-interface s0
 neighbor 12.0.0.2 (链路对方的RIP)
```

LAB8: 在DV协议 (RIP/EIGRP) 中, 实现基于接口的路由过滤:

=====

```
EIGRP 90 ---- RIP V2
(4) - (2) - (1) - (3) - (5)
```

8-1: Distribute-list调用ACL, 比较落后:

Step1:通过ACL, 定义所需要过滤的路由:

R3(config)#Access-list 3 deny 5.5.5.0(这里的deny跟数据包过滤是一样的意思, 都是拒绝的意思)

R3(config)#Access-list 3 permit any(这里的permit跟数据包过滤是一样的意思, 都是允许的意思)

Step2:

R3(config)#router rip

```
R3(config-router)#distribute-list 3
out serial 0
 IP access list number in
Filter incoming routing updates
 out
Filter outgoing routing updates
```

-----

8-2:比较先进, 通过Distribute-list调用前缀列表

(用RIP举例, 也可用于EIGRP) ACL的缺陷, 无法定义路由的长度

Step1: 使用prefix-list定义所需要过滤的路由: (SEQ5为首, 预留空间方便编辑)

ip prefix-list R-10 seq 5 permit 10.1.0.0/24(跟过滤数据包一样的permit表示允许, deny表示拒绝)

ip prefix-list R-10 seq 10 deny 10.1.0.0/16

ip prefix-list R-10 seq 15 permit 0.0.0.0/0 le 32 (any)

以下是错误的说法:

```
access-list 10.deny 10.1.0.0
 wildcard
```

```
ip prefix-list R-10 seq 10 deny 10.1.0.0/16
```

```
ip prefix-list R-10
```

Step2:

```
R3(config)#router rip
```

```
R3(config-router)#distribute-list prefix R-10 in serial 1(调用了整个R-10的访问列表, 往下匹配)
```

distribute-list(可以调用访问列表, 也可以调用前缀列表!!)

out方向的控制:

只在路由出口方向进行控制, 影响路由下游方向的路由器, 而不影响本路由器.

In方向的控制:

在路由器的入口方向进行控制, 直接影响到本路由器.

```
distribute-list 3 out serial 0
```

Step 3:

R3:

```
R3(config)#router rip
```

```
R3(config-router)#distribute-list 3 in serial 1
```

```
R1# clear ip route * (清除/复位路由表)
```

LAB9:对用户数据包的过滤:

在R3上, 对来自本机S0口的, 访问目标网络是15.0.0.0/8的数据包, 进行过滤:

Step1:通过ACL定义需要进行过滤的数据包.

```
R3 access-list 100 deny ip any 15.0.0.0 0.255.255.255
```

LAB10:在路由协议之间的重分布中,实现路由重分布时的基于协议的路由过滤:

~~~~~

在EIGRP和RIP之间做双向重分布:

```
R1(config)#router eigrp 90
R1(config-router)#redistribute rip metric 1544 2000 255 1 1500

R1(config)#router rip

R1(config-router)#redistribute eigrp 90 metric 1
```

10-1:通过ACL,实现路由控制:

Step1:通过ACL,定义所需要控制的路由:

```
R1(config)#access-list 5 deny 5.5.5.0 0.0.0.255
R1(config)#access-list 5 permit any
```

Step2:在EIGRP进程中:

```
R1(config)#router eigrp 90
R1(config)#distribute-list 5 out rip (理解成rip out)
```

意思是:禁止把rip中的5.5.5.5路由重分布到EIGRP里面,也就是说除了R1之外,其它的EIGRP的路由器都没有5.5.5.5的路由!!!!

distribute-list: Filter networks in routing updates

15: IP access list number

out:

- in Filter incoming routing updates
- out Filter outgoing routing updates

10-2:通过prefix-list ,实现路由控制:

Step1:通过Prefix-list, 定义所需要控制的路由:

```
R1(config)#ip prefix-list DN-15 deny 15.0.0.0/8
R1(config)#ip prifix-list DN-15 permit 0.0.0.0/0 le 32
```

Step2:

```
R1(config)#router eigrp 90
R1(config-router)#distribute-list prefix DN-15 out rip
```

调用一个空的, 不存在的prefix-list, 相当是permit any:

```
R1(config-router)#distribute-list prefix D-15 out rip
```

prefix-list 命名, 都是大小写敏感的:

```
R1(config-router)#distribute-list prefix dn-15 out rip
```

```
~~~~~
~~~~~
```

route-map

如果没有set, 则表示set nothing  
如果没有match, 则表示match any

route-map的特征:

1. route-map 由一组statements/声明, 所组成,  
每句声明中, 可以包含了Match set 语句.

每句statements有个编号, 10/20/30.....  
路由器就按照statements的编号, 按自小到大(自上而下)顺序执行.

2.  
match commands specify criteria to be matched

set commands modify matching routes.

一旦匹配/符合特定某些条件

就立刻执行由set所定义的参数/操作, 并且离开route-map. (而不再往下执

行)

3:

the first match found for a route is applied.

once there is a match, leave the route map.

路由器在向下检查匹配条件时,不断地与每个statements中的匹配条件进行比较,

如果不匹配,则向下继续检查下一个statements;

如果匹配,则按照set所定义的参数,进行操作,然后立刻离开route-map (即使下面还有statements没有进行匹配检查,也不往下检查了)

4:route-,map 的逻辑关系:

in the same line uses a logical OR(水平方向:或关系)

vertical match uses a logical AND (垂直方向:与关系)

5:在route-map中:

如果没有match,表示:match any (上面statements剩余的any)

如果没有set,表示: set nothing !! (保留其原始的默认值)

LAB11:通过route-map,实现路由协议之间的路由过滤/控制:

Step1:通过前缀列表定义所需控制的路由:(通过ACL也可以,但比较落后)

```
R1(config)#ip prefix-list R-15 permit 15.0.0.0/8
```

```
R1(config)#ip prefix-list R-25 permit 25.5.0.0/16
```

```
R1(config)#ip prefix-list R-35 permit 35.0.0.0/24
```

注意:这里的permit是表示:"匹配",而不是"允许".

Step2:创建用于重分布的路由控制的route-map:

```
R1(config)#route-map RP-SPF permit 10(permit:允许)
```

```
R1(config)#match ip address prefix-list R-15 R-25
```

```
R1(config-route-map)#set metric 100
R1(config-route-map)#set metric-type type-1
```

```
R1(config)#route-map RP-SPF deny 20 (deny 不允许重分布)
R1(config-router-map)#match ip address prefix-list R-35 没有
Set:Set nothing!)
R1(config)#router-map RP-SPF permit 30
```

step3:在重分布的协议进程中,调用route-map RP-SPF:

```
R1(config)#router ospf 110
R1(config-routiter)#redistribute rip route-map RP-SPF subnets
```

Step 4 :如果没有最后的这句空的route-map,将有什么效果?

```
R1(config)#route-map RP-SPF permit 30
```

剩余的部分路由,都将被Deny.

Step5:在调用时出错:

```
router ospf 110
redistribute rip route-map RP-SP subnets
```

空的route-map,意味着deny any.

### 13晚-路由控制与BGP开头

(已完成实验)

administrative distance is a way of ranking the trustworthiness of routing information.administrative distance is expressed as an integer,from 0 to 255.Lower administrative distance is more trustworthy.

administrative distance 会影响路由的选路。

路由选择:

- 1.longest match(最长匹配)
- 2.lowest administrative distance(不同协议之间的)
- 3.lowest metric(rip:hop)(同一种协议)

LAB11:双向/双出口的重分布(在“我的文档”里面有详细的Word文档)

通过调整AD,防止次优路径的产生:

~~~~~

step0:

R1/R2/R4运行OSPF

R1/R4/R3/R5运行RIP

Step1:在RIP/OSPF间, 做双向/双出口的重分布:

R1/4#

```
router ospf 110
```

```
redistribute rip subnets
```

```
router rip
```

```
redistribute ospf 110 metric 1
```

Step2:在观察路由:(次优路径)

R1#

```
0 E2 34.0.0.0 [110/20] via 12.0.0.2
```

```
0 E2 35.0.0.0/8 [110/20] via 12.0.0.2
```

R4#

```
0 E2 13.0.0.0/8 [110/20] via 24.0.0.2
```

step3:通过ACL, 定义出需要更改AD的路由(RIP路由)(不支持前缀列表)

定义RIP路由:

R1/R4#

```
access-list 10 permit 13.0.0.0
```

```
access-list 10 permit 34.0.0.0
```

```
access-list 10 permit 35.0.0.0
```

Step4:针对源自RIP, 进入OSPF的路由, 修改AD, 比RIP的AD:120大. (125)

R1/R4#

(AD的修改仅对本路由器生效)

```
router ospf 110
```

```
distance 125 0.0.0.0 255.255.255.255 10
```

```
AD:125 路由的源:any ACL:10
```

~~~~~

PBR

Policy-Based Routing/策略路由

PBR allows you to implement policies that selectively cause packets to take different

PBR的核心目的:实现网管的人为的网络操纵意图/策略.

PBR has the following benefits:

策略路由是基于源的.

普通正常IP路由是基于目标的.

PBR的应用:

Source-based      SP 提供区分服务

Qos

Load sharing

Defining Policies Using a Route map

PBR只应用于路由器入口的数据包.  
所有的PBR都应该在入口实现.

PBR大量使用Route-map这个工具.

set ip next-hop ip-address.....  
(have an explicit route to the destination)

优于路由表

set ip default next-hop ip-addredss.....

```
set ip default next-hop ip-addresss.....
(have no explicit route to the destination)
```

次于路由表

LAB12:PBR-(R3#)

~~~~~

Step1:通过ACL, 定义用户群:

```
access-list 10 permit 10.0.0.0 0.255.255.255(permit:匹配)
access-list 20 permit 20.0.0.0 0.255.255.255
```

Step2:创建route-map, 对不同的用户, 进行不同的策略.

```
route-map PBR permit 10
match ip address 10
set ip next-hop 13.0.0.1
route-map PBR permit 20
match ip address 20
set ip next-hop 34.0.0.4
route-map PBR permit 30
```

Step3:在数据包的入口, 调用route-map PBR

```
R3(config)#inter s1
```

```
R3(config-if)#ip policy route-map PBR
```

Step4:

```
R3#debug ip policy
```

如果匹配的statements中是permit, 意味着进行策略路由

如果匹配的statements中是Deny, 意味着不进行策略(要进行正常路由)

如果连一个statements都匹配不上, 意味着不进行策略路由(正常路由)

BGP:

1:BGP基本理论

2:BGP邻居建立(物理接口)

- 3: BGP路由宣告
- 4: BGP路由的优化基本条件(下一跳/同步)
- 5: .....

AS: 自治系统

An AS is a collection of networks under a single technical administration. (独立的一个技术/管理域)

IGP:

包括RIP/IGRP/EIGRP/ODR/OSPF/ISIS  
运行在同一个AS中.

IGP是通过Cost/Metric判断路由的优劣, 越小越好.

IGP的主要任务:

更快/更好的正确描述路由信息, 尽快地将数据包送到目的地.  
IGP强调收敛速率.

BGP: (v4版本)

运行在不同的AS之间, 用于对BGP路由进行控制/策略,

BGP的主要任务: 网管的人为意图的集中体现, 强调策略控制, 不强调收敛.

BGP是通过BGP的属性/Attributes, 判断多条路径的优劣, 从中进行择优选取.

BGP可以通过网管定义的策略/Policies, 实现数据或路由的控制/操纵.

AS: (autonomous Systems/自治系统) (大)

独立的技术/管理域, 通常是一个大型公司, 或者组织, 或者国家.

这是区别于IGRP/EIGRP的AS的概念. (小)

BGP本身就是一种策略路由/PBR(Policy-Based Routing)  
实现网管的人为意图的集中体现.

实现网管的人为意图的集中体现.

BGP是一种AS 之间的AS的高级距离向量/DV协议.

BGP认为, 每经过一个AS是一跳

RIP认为, 每经过一个router是一跳

应该使用BGP的情况:

1:ISP:

当允许AS 1 的数据包穿越AS 2,

但是不允许AS 1的用户访问AS2内部的时候:

2:Multihome/多宿主:

对于一个用户的AS, 如果他同时连接到多个AS/ISP

3:PBR:

当需要对BGP路由/数据, 进行人为控制/操纵的时候.

不该使用BGP的情况:

1. 与ISP只有单连接, 没有同时连接到多个ISP

2. 如果网络硬件设备的档次不够(内存/CPU).

3:对BGP路由操纵理解有限, 无法预计BGP的后果.

4. 链路带宽不足.

BGP特征:

1:BGP是高级DV协议.

2:BGP工作在TCP/IP协议栈的4层:TCP179端口.

3:BGP是触发/增量更新的协议

4:BGP通过周期性的发送Keepalive信息, 保证TCP连接的可靠性.

5:BGP使用多种"BGP属性/Attribut", 来衡量路径的优劣.

6:BGP是为巨型网络设计的, 意味着这种路由协议, 可能带来海量的路由和数据.

BGP协议的三张表:

1:BGP的邻居表(BGP neighbor table)  
BGP的邻居可以直接相连,也可以凌空建立.

2:BGP表(BGP forwarding table/database)

3:IP路由表:

上面BGP提交的“优化/最佳”路由,在经过“AD的竞争/竞选”之后,如果获胜,才能成功的送进路由表.

BGP的4种信息包类型:

1:Open  
2:Keepalive  
3:Update  
4:Notification

Peers=neighbors

Any two routers that have formed a TCP connection, to exchange BGP routing information, are called BGP peers or neighbors.

1:

EBGP Neighbor  
两个BGP Neighbor分别属于两个不同的AS.

EBGP Neighbor通常是直接相连的.

2:

IBGP Neighbor  
两个BGP Neighbor同时属于一个相同的AS.  
IBGP Neighbor可以是直连的,也可以是非直连的.

**14晚-BGP**

## 14晚-BGP

(已完成实验)

LAB1:BGP的基本配置:

~~~~~  
~~

Step1:确认L1/L2的连通性.  
可以通过Ping, 进行链路测试.

Step2:确保L3的IP路由的通达(是通过IGP实现):

(一般都是在同一个AS中完成)a

(两个AS之间是不运行IGP的)

=====

在AS123中的所有路由器(R2/R3), 运行IGP:RIP V2

```
router rip
version 2
network 23.0.0.0
no auto-summary
```

测试L3的IGP网络(ping)

```
show ip route
```

step3:启动BGP, 并且立刻指定全局唯一的bgp router-id.

```
R2(config)#router bgp 123
R2(config-router)#bgp router-id 123.0.0.2
```

Step4:构建BGP Neighbor

~~~~~

R5-R3 EBGp

```
R5(config-router)#neighbor 35.0.0.3 remote-as 123
R3(config-router)#neighbor 35.0.0.5 remote-as 150
```

R3-R2 IBGP:

```
R3(config-router)#neighbor 23.0.0.2 remote-as 123
```

```
R2(config-router)#neighbor 23.0.0.3 remote-as 123
```

R2-R4 EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
```

```
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

Step5:

R4#debug ip bgp 观察BGP邻居建立的过程, 及其经历的5个状态:

1-1:Idle:router is searching

1-2:connect:Completed three-way TCP 179 handshake.

1-3:Open sent: Open message sent

1-4:Open confirm:router received agreement

1-5:Established: Peering is established;BGP Routing begins!!

R5#show tcp brief(察看TCP连接的摘要信息)

| Local Address  | Foreign Address | (state) |
|----------------|-----------------|---------|
| state/PfxRcd   |                 |         |
| 35.0.0.5.11001 | 35.0.0.3.179    | ESTAB   |

收到的路由条目

R5#show ip bgp summary(察看BGP邻居的简要信息)

BGP router identifier 150.0.0.5, local AS number 150

| Neighbor                          | V | AS       | ***** |
|-----------------------------------|---|----------|-------|
| State                             |   | / PfxRcd |       |
| 35.0.0.3                          | 4 | 123      | * * * |
| (如果邻居已经建立好了, 这里必需空白) / 0(收到的路由条目) |   |          |       |
| 18.0.0.8                          | 4 | 100      | * * * |
| / 3                               |   |          |       |



## Step7: IBGP路由的优化

6-1:

BGP路由器, 把自己学到的BGP路由, 转发给别的BGP邻居的必要条件: (R3)

每个BGP路由器, 对于特定的某条BGP路由,  
必须是自己已经优化的路由, 才具备转发给别的BGP邻居的能力.

注意:

这是必要条件, 不是充分条件!!!

这意味着: 即使自己已经优化, 但此路由器, 可能转发, 也可能不转发.

察看R3向R2发送了哪些BGP路由条目:

```
R3#show ip bgp neighbors 23.0.0.2 advertised-routes
```

7-2: (在R2上, 观察从R3这个IBGP邻居获得的BGP路由)

IBGP路由的必要优化条件: (以下是必要条件, 而不是充分条件)

1: 下一跳可达

2: 路由的同步

6-3: 察看R2上, 在R3没有作任何更改前, 当前的BGP路由:

注意观察:

1: 本路由器R2, 是否能够到达这条BGP路由的下一跳?

```
R2#show ip bgp
```

| Network         | Next Hop  |
|-----------------|-----------|
| * i105.5.0.0/16 | 35.0.0.5  |
|                 | 当前下一跳它不可达 |

解决方案:

```
R3(config)#router bgp 123
```

```
R3(config-router)#neighbor 23.0.0.2 next-hop-self
```

```
R3#clear ip bgp *soft out
```

```
R2#show ip bgp
```

| Network         | Next Hop | Metric | LocPrf | Weight | Path |
|-----------------|----------|--------|--------|--------|------|
| *>i105.5.0.0/16 | 23.0.0.3 | 0      | 100    | 0      | 150  |

当前的下一跳是可达的, 所以是最优的

7-4: 同步问题 (在新版的IOS里面, 默认是关闭同步的)  
(本质: R2是否能够通过IGP, 得到这条105.1.0.0/16路由)

所谓同步, 是指:

如果R2能够通过IGP (RIP), 学到这条路由 (105.5.0.0/16), 那么就同步.  
(也就是: 满足同步要求).

如果R2不能够通过IGP, 学到这条路由 (105.5.0.0/16), 那就不满足同步, (也就是: 不满足同步要求).

BGP路由器, 对于从IBGP邻居学到的路由, 默认要求同步.

但是,

∴ 实际上R2是不可能通过IGP, 学到另外的一个AS中的路由.

∴ 只能关闭同步规则, 也就是不遵循同步规则.

解决方案: 关闭同步规则:

```
R2#
router bgp 123
no synch
```

step8: EBGP邻居的路由优化问题:

~~~~~

1: 下一跳问题:

1: 下一跳问题:

∴ BGP路由的下一跳是其直连路由,

∴ 不存在下一跳不可达的问题.

(整个AS140中的100个路由器, 察看到这条BGP路由的下一跳都是: 24. 0. 0. 2)

(整个AS123中的100个路由器, 察看到这条BGP路由的下一跳都是: 35. 0. 0. 5)

2: 同步问题:

对于EBGP邻居,

∴ 无需遵循同步规则,

∴ 没有同步问题!!

结论: 对于EBGP, 只要是从EBGP邻居那里传来的路由,

在不考虑BGP属性的情况下, 是肯定可以优化的.

```
R4#show ip bgp
*>105.5.0.0/16 24.0.0.2
```

Step9:

EBGP

~~~~~  
~~

```
R3#show ip bgp
```

|    | Network      | Next Hop |
|----|--------------|----------|
| *> | 105.5.0.0/16 | 35.0.0.5 |

```
R3#show ip route
B 105.5.0.0 [20/0] via 35.0.0.5,
```

IBGP:

~~~~~

```
R2#show ip bgp
```

```
R2#show ip bgp
```

| Network        | Next Hop |
|----------------|----------|
| *>105.5.0.0/16 | 23.0.0.3 |

```
R2#show ip route
```

```
B 105.5.0.0 [200/0] via 23.0.0.3
```

BGP的更新源:(BGP Neighbor Update Source Address)

原则1:

在默认情况下,

BGP路由器以自己路由表中, 到达对方BGP邻居的地址的那条路由所指示的出接口的地址, 作为自己的BGP更新源(源地址)

原则2:

当BGP路由器, 收到邻居发来的BGP信息时, 会检查其源地址, 然后和自己宣告的Neighbor的目标地址进行比较, 如果一致, 这个BGP Session才可建立起来.

LAB2:验证通过物理接口, 构建IBGP邻居的不稳定性:

~~~~~

Step1:确认L1/L2通达

Step2:确认L3的IGP通达(RIP), 在AS123内

Step3:通过物理接口, 在R2/R3之间构建IBGP邻居

```
show run | begin router bgp
```

```
R2#neighbor 23.0.0.3 remote-as 123
```

```
R3#neighbor 23.0.0.2 remote-as 123
```

结论:

在IBGP中, 如果使用物理接口构建邻居, 是很不稳定的.

很可能因为某条物理链路的抖动, 导致IBGP邻居的Flapping/抖动:

建议:

使用环回口/Loopback接口, 构建IBGP邻居.

LAB3:以Loopback接口作为BGP更新源, 构建稳定的IBGP Session

Step1:为每个IBGP路由器, 构建一个环回口:

Step2:把此Loopback接口, 宣告到IGP(RIP)中.

R2/R3#

```
router rip
```

```
network 2.0.0.0
```

or

```
network 3.0.0.0
```

step3:

在R2/R3上, 删除原来的, 通过物理接口构建的邻居.

Step4:通过环回口构建IBGP邻居:

4-1:

以对方的环回口, 作为IBGP的目标地址:

```
R2#neighbor 3.3.3.3 remote-as 123
```

```
R3#neighbor 2.2.2.2 remote-as 123
```

4-2:

以自己的环回口, 作为IBGP连接的源地址:

```
R2#neighbor 3.3.3.3 update-source loopback 2
```

```
R3#neighbor 2.2.2.2 update-source loopback 3
```

step5:

任意切断本AS123 中的物理链路,

只要两个IBGP路由器R2/R3之间, 还有最后一条能够到达, 对方的环回口的路由, IBGP邻居都不会中断.

建议:

凡是构建IBGP, 默认都使用环回口做更新新源, 以构建稳定的IBGP.

## 15晚-BGP

(已完成实验)

LAB4:在两AS间, 存在多条冗余链路的网络环境中:  
以LoopBack接口作为EBGP更新源, 构建稳定的EBGP Session.

~~~~~

Step1:在两AS之间回口.  
, 构建多条冗余链路.

如果两AS间, 没有多条冗余链路, 就使用物理接口构建EBGPP即可.

Step2:为两AS间的EBGP路由器, 构建环

Step3:在各自路由器上, 指定到达对方环回口静态路由:

∴有两条冗余链路,  
∴要有两条到达对方环口的静态路由

```
R5:(config)#
ip route 3.3.3.3 255.255.255.255 35.0.0.3
ip route 3.3.3.3 255.255.255.255 100.0.0.3
```

测试:

```
R3#ping 5.5.5.5 source 3.3.3.3 !!!!!!!!!!!!!!!
```

Step4:建立邻居EBGP邻居:

```
R3#router bgp 123
neighbor 5.5.5.5 remote-as 150
R5#router bgp 150
neighbor 3.3.3.3 remote-as 123
```

step5:告知对方, 自己的更新源:

```
R3#neighbor 5.5.5.5 update-source loopback 3
```

```
R3#neighbor 3.3.3.3 update-source loopback 5
```

Step6:更改EBGP的TTL值(Time to Live)

∴EBGP TTL值默认是1,

∴EBGP的TTL值最少要设为2

而实际上EBGP多跳这个命令,在不指定其取值时,会自动默认指定为255

```
R5#neighbor 3.3.3.3 EBGP-multihop 2
```

```
R3#neighbor 5.5.5.5 EBGP-multihop(255)
```

TTL:time to live是L3的IP包头中的一个特定字段, IP包每经过一个路由设备,其TTL会自动减1.

如果TTL减到为0,即使路由器有去往目标的路由,也不会继续转发这个IP包.

Step7:测试

```
R3#ping 105.5.5.5 source 3.3.3.3 repeat 1000000 size 15000
```

结论:

在一般情况下,EBGP的邻居关系,是不需要使用环回口构建邻居的.

默认都直接使用物理接口,

在只有单链路的时候,都是使用物理接口构建邻居.

只有在两AS之间,存在多条冗余链路的时候,才需要考虑使用环回口构建EBGP邻居,以确保其EBGP的稳定性.

BGP路由黑洞的解决方案:

解决BGP路由黑洞,可供选择的解决方案/Solution:

1:Redistribute Selected BGP Route into IGP

2:Full-mesh IBGP

3:Part-mesh IBGP+Reflector

4:Confederation

5:MPLS

LAB5:

LAB5:

part-mesh IBGP, Redistribute Selected BGP Route into IGP, with sync  
(同步/synchronization)

~~~~~  
~~~~~

Step1: 定义需要重分布到IGP中的, BGP的路由:

R3(config)#ip prefix-list B-105 permit 105.1.0.0/16

R2(config)#ip prefix-list B-104 permit 104.1.0.0/24

Step2: 通过Route-map, 控制重分布到IGP的范围,

route-map R3-BGP-RP permit 10

match ip address prefix-list B-105

set metric 1

R2#

route-map R2-BGP-PR permit 10

match ip address prefix-list B-104

set metric 1

小提醒:

不要配置: "route-map R3-BGP\_RP permit 20"

一旦配置, 意味着所有一切BGP路由都进入IGP!

Step3: 按照route-map所定义的条件, 将BGP路由注入RIP:

R3#

router rip

redistribute bgp 123 route-map R3-BGP-RP

Step4: 在R2上观察, 105.5.0.0/16,

R2: 同时从RIP和BGP, 都能学到路由, 但因为AD竞争原因,

从RIP所获得的路由, 成功进入路由表,

而从BGP所获得的路由, 不能进入路由表.

R2#show ip route

R 105.5.0.0[120/2]

R2#show ip bgp

```
r>i105.5.0.0/16 3.3.3.3
```

Step5:在R2观察,如果R2,此时启动了BGP的“同步”,是否还能优化?

结果:可以优化~~~!!!

因为:R2此时通过RIP,学到105.5.0.0/16,  
结果:可以优化!!!!

Step6:在R1观察两条路由:

```
R 104.4.4.0[120/1] via 12.0.0.2
R 105.5.0.0[120/1] via 13.0.0.3
```

R1不再是黑洞!!

Step7:在R2上观察路由的递归查询:

```
R 105.5.0.0[120/2] via 12.0.0.1
C 12.0.0.0 is directly connected, serial0
```

Step8:测试:

```
R4#ping 105.5.5.5 source 104.4.4.4 !!!!!!!!!!!!!
```

LAB6:

Full-mesh IBGP with No-Sync/(Peer Groups)

~~~~~

Step0:启动BGP进程

```
R1(config)#router bgp 123
```

Step1:在R1上,使用Peer-Group(一个模版),对R2/R3建IGBP邻居:

1-1:定义peer-group:

```
neighbor R1-PG peer-group
neighbor R1-PG remote-as 123
neighbor R1-PG update-source loopback 1
```

1-2:对不同的IBGP邻居,调用peer-group:

```
neighbor 2.2.2.2 peer-group R1-PG
neighbor 3.3.3.3 peer-group R1-PG
```

peer-group 只是一种模版,只影响本路由器的,邻居建立的方法.

在R2/R3上,与R1的邻居建立,仍然可以使用普通方法建立.

Step2:确保整个AS123中的所有BGP路由器的,下一跳,同步问题能够解决:

2-1:

R1/R2/R3#关闭同步

2-2:

R2上,对R1/R3 Say next-hop-self

R3上,对R1/R3 Say next-hop-self

Step3:在R1上,观察所有BGP路由:

```
*>i103.3.3.0/24 3.3.3.3
*>i104.4.4.0/24 2.2.2.2
*>i105.5.5.0/16 3.3.3.3
```

Step4:观察在R2上的BGP路由的递归查询:

R2#

B 105.5.0.0[/0]via 3.3.3.3

R 3.3.3.3[120200/2]via 12.0.0.1

C 12.0.0.0 is directly connected, serial0

Step5:测试:

```
R4#ping 105.5.5.5 source 104.4.4.4
```

LAB7:

part-mesh IBGP+"RP"(BGP Route-Reflector)

~~~~~  
~~~~~

Step1:

删除R2-R3IBGP Session

删除之前:R2有105.5.0.0/16路由

```
R3(config-router)#no neighbor 2.2.2.2
```

```
R2(config-router)#no neighbor 3.3.3.3
```

删除之后

IBGP Split Horizon Rule:/IBGP的水平分割原则:

(已完成实验)

IBGP的水平分割原则,

by default, routes learned via IBGP are never propagated to other  
IBGP peers.

默认情况下:

对于一个BGP路由器R1来说, 从一个IBGP邻居R3那里学到的BGP路由,  
是不是传递给另外一个IBGP邻居R2的

提醒:EBGP是没有这种规则的!!

Step2:解决方法:BGP Route Reflector /RR

在R1上, 定义R2/R3为自己的“路由反射器的客户端”

2-1:

如果使用Peer-group:

```
R1#neighbor R1-PG route-reflector-client
```

2-2:

如果没有使用peer-group:

```
R1(router bgp 123)#
```

```
neighbor 2.2.2.2 route-reflector-client
```

```
neighbor 3.3.3.3 route-reflector-client
```

Step3:

问题:在R1上, 定义R2为自己的“路由反射器的客户端”, 但R3不是.  
R4能收到105的路由吗?

## 21晚-BGP

LAB8:

Confederation(联邦) / (community)

~~~~~

假定:子AS65003/AS65012之间, 是不运行IGP的.

AS65012内部运行的IGP是RIP.

Step1:子AS内的IGP是RIP

```
router rip
version 2
network 1.0.0.0
net 12.0.0.0
no auto-summary
```

step2:

构建R3—R5之间的EBGP:

```
R3(config-router)#neighbor 35.0.0.5 remote-as 150
R5(config-router)#neighbor 35.0.0.3 remote-as 123
```

构建R2-R4之间的EBGP:

```
R2(config-router)#neighbor 24.0.0.4 remote-as 140
R4(config-router)#neighbor 24.0.0.2 remote-as 123
```

提醒:

在联邦以外的EBGP邻居，他们能查看到的是联邦的大AS号，而不是子AS号。

Step3:在联邦内部的路由上，都要标识他们同属于AS123

```
R1/2/3#bgp confederation identifier 123
```

Step4:在子AS的边界路由器R1/R3上，互相指定对方的子AS号:

```
R1(config-router)#bgp confederation peers 65003
R3(config-router)#bgp confederation peers 65012
```

Step5:构建联邦子AS中的联邦IBGP:

R1/R2的联邦IBGP:

```
R2#neighbor 1.1.1.1 remote-as 65012
R2#neighbor 1.1.1.1 update-source loopback 2
```

```
R1#neighbor 2.2.2.2 remote-as 65012
R1#neighbor 2.2.2.2 update-source loopback 1
```

Step6:构建联邦EBGP:

```
R1#(config-router)#neighbor 13.0.0.3 remote-as 65003
R3#(config-router)#neighbor 13.0.0.1 remote 65012
```

以上建立BGP邻居的步骤

~~~~~

以下是观察BGP的路由的传递:

Step7:

```
R3# *> 105.5.0.0/16 35.0.0.5
```

但是由于R1无法通过IGP, 获得到达35.0.0.0这个网络的路由,

但是由于R1无法通过IGP, 获得到达35. 0. 0. 0这个网络的路由,  
∴R1# \* 105. 5. 0. 0/16 35. 0. 0. 5

解决办法:

R3(config-router)#neighbor 13. 0. 0. 1 next-hop-self

R1# \*>105. 5. 0. 0/16 13. 0. 0. 3

Step8:子AS内部到IBGP路由器:

R2# \*i105. 5. 0. 0/16 13. 0. 0. 3

R1#(config-router)#neighbor 2. 2. 2. 2 next-hop-self

R2

\*>i105. 5. 5. 0/24 12. 0. 0. 1 0 100 0  
(65003) 150 i

R4#\*> 105. 5. 0. 0/16 24. 0. 0. 2

结论:

联邦子AS之间的EBGP的下一跳, 不像普通EBGP那样每经过一个AS, 都发生改变.

而保留原始的BGP下一跳.

Step9:联邦内的同步问题:

R2/R3# no sy

R1(config-router-65012)#sy

R1#show ip bgp (当R1启动"同步")

\* i104. 4. 4. 0/24 2. 2. 2. 2 (来自联邦IBGP)

\* >105. 5. 0. 0/16 13. 0. 0. 3(来自联邦EBGP)

R2#sh ip bgp

R2#(config-router)#sync

结论:

联邦子AS之间的同步问题:

如果路由来自联邦IBGP, 则需要审查同步条件.

如果路由来自联邦EBGP, 则不需要审查同步条件.

Step10:

假定:子AS65003/AS65012之间, 是运行可以互通的IGP的.

在R1-R3上, 运行IGP:RIP V2

```
R3(config)#router rip
```

```
R3(config-router)#version 2
```

```
R3(config-router)#no auto-summary
```

```
R3(config-router)#network 13.0.0.0
```

查看RIP路由:

```
R2# R 13.0.0.0/24
```

```
R3# R 2.2.2.2/32
```

删除原R1/R3的更改下一跳命令:

```
R1(config-router)#no neighbor 13.0.0.3 next-hop-self
```

```
R3(config-router)#no neighbor 13.0.0.1 next-hop-self
```

```
R3# *>104.4.4.0/24 2.2.2.2
```

LAB9:

community/团体

(相当于一种BGP路由的标识位, 常用于标识这条BGP路由应该传播的范围)

Step1:通过前缀列表, 定义BGP路由:

```
ip prefix-list B-1 seq 5 permit 40.1.0.0/16(permit:表示匹配)
```

```
ip prefix-list B-2 seq 5 permit 40.2.0.0/16
```

```
ip prefix-list B-3 seq 5 permit 40.3.0.0/16
```

Step2:通过调用前缀列表.

为每条路由, 指定其Community:

```
router-map CMM permit 10
```

```
match ip address prefix-list B-1
```

```
set community no-advertise(do not advertise to any peer)
R4通知R2, 不要发给任何BGP邻居
```

```
router-map CMM permit 20
```

```
match ip address prefix-list B-2
set community local-AS (Do not send outside local AS)
联邦的子AS/小AS
```

```
route-map CMM permit 30
match ip address prefix-list B-3
set community no-export (do not export to next AS)
联邦的AS
```

```
route-map CMM permit 40
(match any, set Nothing!)
```

(如果没有这句话, 除了这三条路由以外, 其它的路由都不发给R2!!)

Step3:在BGP进程中, 对R2调用route-map CMM:  
R4:neighbor 24.0.0.2 route-map CMM out

in:入方向的控制, 影响本路由器.  
out:路由的出方向的控制, 影响对方路由器

Step4:  
将community这些标签, 发送给对方,  
每向前走一个BGP Router, 就要"Send-community"推一下.

```
R4#neighbor 24.0.0.2 send-community
```

```
R2#neighbor 1.1.1.1 send-community
```

```
R1#neighbor 13.0.0.3 send-community
```

```
clear ip bgp *
```

```
R2#show ip bgp community
```

```
R5#show ip bgp community
```

```
show ip bgp community no-advertise
show ip bgp community local-AS
show ip bgp community no-export
```

```
show ip bgp 40.3.0.0/16
```

Step5:

如果Route-map中, 没有最后的那句空的route-map:

```
"no route-map CMM permit 40"
```

R4向R2通告的bgp路由只有3条:

```
R4#show ip bgp neighbor 24.0.0.2 advertisedp-routes
```

```
*> 40.1.0.0/16
```

```
*> 40.2.0.0/16
```

```
*> 40.3.0.0/16
```

BGP路由过滤的方法之一.

BGP-Summarization-15"

LAB1:非专业汇总(network命令, 是不需要宣告明细路由的.)

Step1:在BGP进程中, 使用network命令, 宣告所需要汇总的路由

```
R1(config-router)#router bgp 140
```

```
R4(config-router)#network 40.0.0.0 mask 255.252.0.0
```

Step2:手工配置静态的汇总路由, 指向空接口,

∴BGP进程, 在路由宣告前,  
会检查路由表, 如果能够查到汇总路由,  
才会将此汇总路由宣告到BGP进程中.

∴

```
R4(config)#ip route 40.0.0.0 255.252.0.0 null 0(空接口)
```

在R5上, 可以查看到明细路由/汇总路由,  
实际上, 明细路由是不需要的:

Step3:删除原明细BGP路由的宣告:

```
R4(config-router)#router bgp 140
```

```
R4(config-router)#no network 40.0.0.0 mask 255.255.0.0
```

```
R4(config-router)#no network 40.1.0.0 mask 255.255.0.0
```

```
R4(config-router)#no network 40.2.0.0 mask 255.255.0.0
```

```
R4(config-router)#no network 40.3.0.0 mask 255.255.0.0
```

```
R5#show ip bgp
```

```
*>40.0.0.0/14
```

LAB2:专业汇总(推荐方法)

~~~~~

Step1:宣告准确的明细路由:

```
R4(config)#router bgp 140
```

```
R4(config-router)#network 40.0.0.0 mask 255.255.0.0
```

```
R4(config-router)#network 40.1.0.0 mask 255.255.0.0
```

```
R4(config-router)#network 40.2.0.0 mask 255.255.0.0
```

```
R4(config-router)#network 40.3.0.0 mask 255.255.0.0
```

不要使用network命令,宣告汇总路由:

```
R4(config-router)#no network 40.0.0.0 mask 255.255.0.0
```

Step2:aggregate-address命令是不需要事先配置汇总路由的.

```
R4(config-router)#aggregate-address 40.0.0.0 255.252.0.0
```

明细路由/汇总路由,都传播出去了

Step3:为了不让明细路由传播出去,调用“summary-only”:

```
R4(config-router)#
```

```
aggregate-address 40.0.0.0 255.252.0.0 summary-only
```

只有汇总路由传播出去,而明细路由就被抑制了,不往外发送了.

## 22晚-BGP

BGP基本配置:

Step1. 确认L1/L2的连通性(ping直连链路的接口)

Step2:确认L3 IGP的路由的通达

Step3:启动BGP,用物理接口建EBGP,用环回口建IBGP

Step4:宣告BGP路由.

Step5:对BGP路由的进行控制

BGP中的路由控制/过滤:

LAB1:Distribute-list调用ACL(较落后)

Step1:通过ACL定义BGP路由:

如果不打算建立route-map

那么Access-list里面的

(Deny:不允许)

(permit:允许)

```
R3(config)#access-list 15 permit 115.1.0.0
```

```
R3(config)#access-list 15 permit 115.2.0.0
```

(ACL的最后,有隐患的Deny Any.)

Step2:通过distribute-list/分布列表

```
R3(config)#router bgp 350
```

```
R3(config-router)#neighbor 13.0.0.1 distribute-list 15 out
```

Redistribute:重分布,将一种路由注入到另外一种路由中.

软清:

```
R3#clear ip bgp * soft out (使用在:出方向的策略发生改变时)
```

Step3:观察R3对R1发送了哪些BGP路由:

```
R3#show ip bgp neighbor 13.0.0.1 advertised-routes
```

LAB2:通过neighbor命令,直接调用prefix-list(较先进,简洁)

~~~~~

Step1:通过前缀列表/prefix-list,定义需要控制的BGP路由:

(Deny:不允许)

(Permit:允许)

```
R3(config)#ip prefix-list T-R1 (seq 5) permit 115.2.0.0/16
```

```
R3(config)#ip prefix-list T-R1 (seq 10) permit 115.1.0.0/16
```

如果Prefix-list直接被BGP进程调用,则应该严格匹配路由表的路由长度!!

IP prefix-list的最后, 有隐患的Deny Any.

Step2: 在R5的出方向, 调用prefix-list T-AS130, 进行路由过滤:

R3(config-router)#neighbor 13.0.0.1 prefix-list T-R out  
在BGP进程中, 不能对同一个邻居同时调用access-list和prefix-list

如果只打上:

neighbor 13.0.0.1 prefix-list maple out

而没有建立一个名为maple的prefix-list, 则全部的路由都放行, 即邻居把所有的路由都可以收到!!

LAB3: neighbor+route-map+ACL  
(使用route-map, 功能强大, 实现复杂功能)

Step1: 通过ACL定义BGP路由:  
(permit: 匹配)

R3(config)#access-list 1 permit 115.1.0.0  
R3(config)#access-list 3 permit 115.3.0.0

Step2: 通过route-map, 调用ACL,  
让Route-map决定是否允许这些定义好的路由穿过本AS:

(permit: 允许), (Deny: 不允许)!!!

route-map T-AS110 permit 10  
match ip address 1

route-map T-AS110 deny 20  
match ip address 3  
route-map T-AS110 permit 30

(是否有这一句空的route-map, 决定了:  
route-map中, 没有明细定义的路由, 将是否可以传到别的AS中)  
(115.3.3.0/24)

Step3: 在BGP进程中, 对AS110中的用户, 调用路由控制策略.  
R3(config-router)#neighbor 13.0.0.1 route-map T-AS110 out

LAB4:neighbor+Route-map+Prefix-list

Step1:通过前缀列表/prefix-list, 定义需要控制的BGP路由:  
(permit:匹配)

```
R3(config)#ip prefix-list B-1 permit 115.1.0.0/16
R3(config)#ip prefix-list B-3 permit 115.3.0.0/16
```

Step2:通过Route-map, 决定是否允许特定路由穿越:  
(permit:允许), (Deny:不允许)!~!!!

```
route-map T-AS110 permit 10
match ip address prefix-list B-1
```

```
route-map T-AS110 deny 20
match ip address prefix-list B-3
```

```
route-map T-AS110 deny 30
```

Step3:R3对R1的BGP路由, 出口策略发生改变:

```
R3(config-router)#neighbor 13.0.0.1 route-map T-AS110 out
```

对于上述4个实验, 如果在R3上做in方向的路由过滤, 则需要硬清除.  
Clear ip bgp \* (会Reset TCP 的连接)

LAB5:BGP BackDoor/后门

~~~~~

用于EBGP与IGP之间的AD竞选时, 人为的让IGP优先进入路由表的一种操作.  
20 120

R2#做后门之前的路由

```
B 115.3.0.0 [20/0] via 12.0.0.1,
B 115.2.0.0 [20/0] via 12.0.0.1
B 115.1.0.0 [20/0] via 12.0.0.1
```

后门的操作:

```
R2(config-router)#network 115.1.0.0 mask 255.255.0.0 backdoor
R2(config-router)#network 115.2.0.0 mask 255.255.0.0 backdoor
```

```
B 115.3.0.0 [20/0] via 12.0.0.1
```



```
reusing a route
2000:suppressing /开始进行抑制的阈值 penalty to start
suppressing a route
60:(分钟) Maximum duration to suppress (设置为半衰期的4倍)
Maximum duration to suppress a stable route
```

Step1:通过前缀列表,定义需要进行Damp的BGP路由:  
R1(config)#ip prefix-list R-123 permit 123.0.0.0/8

Step2:创建Route-map,对特定路由进行Damping,并且可以设定Damping的参数.

```
route-map DAMP permit 10
match ip address prefix-list R-123
set dampening 15 750 2000 60
```

Step3:在BGP进程中调用Route-map,进行BGP dampening

```
router bgp 110
Bgp dampening route-map DAMP
```

旧版本:  
R3#show ip bgp flap-statistics

R3#show ip bgp dampened-paths

新版本:

```
R3#show ip bgp dampening flap-statistics
R3#show ip bgp dampening dampened-paths
```

## 23晚-BGP

BGP Route Selection/BGP的选路原则:

1:  
The BGP forwarding table usually has multiple pathways from which to choose for each network.

在BGP路由器的BGP表中,可以存在到达某个特定目标网络的,都能满足“同步”和“下一跳可达”的,多条路径.

2:

BGP is not designed to perform load balancing:

Paths are chosen because of policy.

Paths are not chosen based upon Metric/bandwidth.

BGP默认不执行负载均衡, 而是严格按照网管的策略/意志, 进行BGP选路.

网管是通过BGP属性/Attribute, 去表达其策略/意志的, 实现BGP路由选择的控制.

而IGP是通过最小的Metric, 实现路由选择的.

3:

The BGP selection process eliminates any multiple pathways through attrition, until a single best pathway is left.

BGP是按照BGP属性, 自上而下, 依次剔除不是最佳的路由:  
直到优选出, 到达目标风格的最佳的那一条BGP路由.

4:

That best pathway is submitted to the routing table manager process and evaluated against the methods of other routing protocols for reaching that network (administrative distance).

BGP所提交给路由表选择的路由, 会和别的路由协议所生成的路由进行AD比较.

5:

The routing protocol with the lowest administrative distance will be installed in the routing table.

AD最小的那个路由协议所生成的路由, 将被注入/优先进路由表, 成为达到该目标网络的路由.

BGP属性的分类:

Well-known attributes:

Well-known mandatory (公认, 强制的)

Well-known discretionary (公认, 自决的)

Optional attributes:

Optional transitive attributes (可选, 可传递的, partial)

Optional nontransitive attributes (可选, 非传递的)

Well-known, mandatory & transitive attribute:

AS path

next-hop

origin

IGP (i) (RIP/IGRP/EIGRP/OSPF/IS-IS)

通过BGP中的network command, 宣告进BGP的.

R2#router bgp 12

network 120.1.0.0 mask 255.255.0.0

EGP (e)

Redistributed from EGP

Incomplete (?)

Redistributed from IGP or static

BGP路由优化的前提条件:

A:Sync;B:next-Hop

1:Weight (LAB6)

2:L-P (LAB5)

3:AS Path (LAB4)

4:MED (LAB3)

5:EBGP VS IBGP的对比 (其实是AD的对比) (LAB2)

6:Closest IBGP Neighbor (LAB1)

BGP配置的基本步骤:

~~~~~

Step1:确认L1/L2的连通性

Step2:确认L3 IGP的路由的通达 (AS120:OSPF; AS345:RIP)  
Step3:启动BGP, 用物理接口建EBGP, 用环回口建IBGP.  
Step4:宣告BGP路由.  
Step5:对BGP路由的进行控制.

LAB1:Closet IBGP Neighbor

~~~~~

Step1:将AS345中的IGP更改为:EIGRP

```
R5#show ip route
D 3.3.3.3 [90/2297856]
D 4.4.4.4 [90/409600]
```

```
R5#show ip BGP
```

```
*>i110.0.0.0 4.4.4.4
*i 3.3.3.3
```

LAB2:EBGP VS IBGP

~~~~~

在R4上观察:

Step1:如果从R2和R3的两个方向上, 都收到BGP路由:

```
R4#show ip bgp
* i110.0.0.0 3.3.3.3 (来自IBGP)
*> 24.0.0.2 (来自EBGP)
```

Step2:切断R2/R4的链路:

```
R4#
*>i110.0.0.0 3.3.3.3
```

LAB3:MED(Multi-exit discriminator)(also called the metric)

~~~~~

MED的特征:

- 1:MED的取值是越小越好
- 2:MED只发送给EBGP邻居, 用于建议对方如何离开对方的AS, 来访问本AS中的网络
- 3:MED是一种可选的, 非传递的属性.
- 4:MED的默认值:0.

关于“传递”的属性:

1. 不论“可传递”还是“非可传递”, 如果设备支持这种属性, 那么都会传递.

如果设备不支持:

2-1:对于“可传递”,那么,,会被标识为“partial”,来进行继续传递.  
2-2:对于“非可传递”,那么这种属性被丢弃,但BGP这条路由条目,还是正常传递. 9

Step1:定义BGP路由:

```
R1/R2(config)#ip prefix-list B-110 permit 110.0.0.0/8
```

Step2:在Route-map中, 设定路由的MED值:

```
R1(config)#route-map T-AS345
```

```
R1(config-router-map)#match ip address prefix-list B-110
```

```
R1(config-router-map)#set metric 150
```

```
R2(config)#route-map T-AS345
```

```
R2(config-route-map)#match ip address prefix-list B-110
```

```
R2(config-route-map)#set metric 200
```

Step3:

```
R1(config-router)#neighbor 13.0.0.3 route-map T-AS345 out
```

```
R2(config-router)#neighbor 24.0.0.4 route-map T-AS345 out
```

```
clear ip bgp * soft out
```

Step4:

```
R4#show ip bgp
```

| network     | next hop | metric |
|-------------|----------|--------|
| * 110.0.0.0 |          |        |

```
R5#
```

```
*>i110.0.0.0 3.3.3.3
```

LAB4:AS path

~~~~~

实验需求:

整个AS345的所有路由器, 都从R2走

在R1与R3之间, 虚拟一个AS100

Step1:通过Prefix-list, 定义BGP路由:

Step2:在Route-map中, 为此路由添加一个虚拟的AS100

```
route-map T-AS345 permit 10
match ip address prefix-list B-110
set metric 150
set as-path prepend 100
```

step3:在BGP进程中, 调用route-map T-AS345, 发给R3

Step4:

R3#

| Network      | Next-hop | metric | LocPrf | Weight | Path |
|--------------|----------|--------|--------|--------|------|
| *>i110.0.0.0 | 4.4.4.4  | 2000   | 100    | 0      | 120i |
| *            | 13.0.0.1 | 150    |        | 0      | 120i |
| 100i         |          |        |        |        |      |

LAB5:Local Preference

~~~~~

LP的特征:

- 1:LP的取值越大越好, 默认值是100.
- 2:LP只发送给本AS内的IBGP邻居, 用于建议他们如何离开本AS, 去访问外网.
- 3:LP的属性是公认的, 自决的, 只会在本AS传递的. !!!

R3#

Step1:通过Prefix-list, 定义BGP路由:

Step2:作用Route-map, 为特定路由设定LP值:

```
route-map LP-3 permit 10
match ip address prefix-list B-110
set local-preference 130
```

Step3:在R3的BGP进程中, 对来自R1对路由, 调用route-map LP-3:

R3(config-router)#neighbor 13.0.0.1 route-map LP-3 in

Step4:观察:

R3#

| Network | next Hop | Metric | LocPrf | Weight |
|---------|----------|--------|--------|--------|
| Path    |          |        |        |        |

| Path | Network      | next Hop | Metric | LocPrf | Weight |
|------|--------------|----------|--------|--------|--------|
|      | *>110.0.0.0  |          |        |        |        |
| R4#  |              |          |        |        |        |
|      | *>i110.0.0.0 |          |        |        |        |
| R5#  |              |          |        |        |        |
|      | *>i110.0.0.0 |          |        |        |        |

LAB6:Weight  
~~~~~

Weight的特征:

- 1:cisco私有的属性.
- 2:默认值为0, 越大越好.
- 3:不发送给任何BGP邻居, 只影响本路由器的选路.

方法1:

```
R4:(config-router)#neighbor 24.0.0.2 weight 10
```

方法2:

```
R4:(config-router)#network 110.0.0.0 mask 255.0.0.0 weight 10
```

方法3:

3-1:定义路由

3-2:

```
route-map WEI permit 10
```

```
match ip address prefix-list B-110
```

```
set weight 10
```

3-3:

```
R4#(config-router)#neighbor 24.0.0.2 route-map WEI in
```

交换

## 交换

### 16晚-VLAN

CISCO的AVVID架构是集成了语音, 视频和数据的体系架构

大致可以包括三个层次.

园区网架构/企业边界/服务提供商边界

网络体系结构:

访问层 (access)

企业园区网如何来提供性能, 可扩展性和可用性

在访问层: 到桌面的性能是关键

提供端口密度来提供扩展性

提供冗余来增加可用性

在分布层: 园区网的性能是关键  
通过交换机的模块化提供可扩展性

Layer 2 Switching

- Hardware-based bridging
- Wire-speed performance
- High-speed Scalability
- Low Latency

- MAC address

在交换机上更改MTU和MAC地址:

```
switch(config)#system mtu 1520
```

保存重启才能生效!!!!

在路由器上更改MTU:

```
interface E0
```

```
mtu 1500 (不能更改以太口的MTU)
```

```
ip mtu 900 (但数据包达到900的时候就分段)
```

```
service timestamps debug uptime/datetime (显示时间)
```

uptime是开机到现在的时间

datetime是绝对时间

### VLAN Overview

一个VLAN对应一个广播域, VLAN不受地理限制的一个广播域

A VLAN= A Broadcast Domain

VLAN的分类:

1. 静态的VLAN(基于接口的VLAN)
2. 动态的VLAN(基于MAC的VLAN)

VLAN, 限制了广播域的范围, VLAN间的路由, 又允许了VLAN间的正常数据包的通信.

LAB1:VLAN的创建, 修改, 删除.

~~~~~

1.

2.

```
SW1#VLAN database(老式)
SW1(Vlan)#vlan20 name CC
SW1(vlan)#apply (Exit是应用加退出)
```

删除VLAN,

LAB2:添加端口到VLAN中

~~~~~

将特定的一个端口

将一组

```
interface range f0/13 -15 , f0/18 -19 , f0/22
```

二. 动态

基于MAC

1. 配置VMPS服务器在哪里

```
#vmps server X.X.X.X
```

2. 将接口划进动态VLAN

```
if#sw mo ac
```

```
switch acc vlan dynamic(把端口变为动态的接口)
```

1. the pc send a frame to the switch

2. The VMPS client learns the PC MAC address on the dynamic port

3. the VMPS client sends a VQP request to the VMPS. the request:contains the VMPS client:IP address, the PC MAC address,the PC port unumber,and the VTP Domain.

4. the VMPS parses its database file for PC VLAN assignment

5. the VMPS sends a VQP response to the VMPS client.

6. if the VQP response contains a VLAN assignment, the VMPS client assigns it to the VLAN. Otherwise, it denies the PC access.

MAC

```
show mac-address-table
```

```
mac-address static
```

静态配置MAC地址表

```
Switch(config)#mac-address-table static aaaa.bbbb.cccc vlan 3
interface f0/10
```

Trunk

~~~~~

在一条网络介质上,同时传输多个VLAN的数据,这种技术,在CISCO称为Trunk

cisco交换机支持两种Trunk:

1. 802.1q

cisco交换机支持两种Trunk:

1. 802.1q

业界标准, 开放性的标准, 能够兼容老式设备(不能支持Trunk的设备)

交换机检测到数据的目标地址不在本交换机时,

当其检测到应当发到某条Trunk链路时,

先进行Trunk封装:

在MAC地址后面, 添加上4Byte的tag

二层接口有5种模式:

- access 相当于trunk的OFF状态
- trunk 相当于trunk的ON状态, 主动发DTP包
- nonegotiate 认为自己是trunk但不会发DTP包(与动态的模式互相排斥)

if#sw mo trunk

if#

- dynamic desirable (动态协商) 相当于cisco交换机接口的默认状态, 主动发DTP包
- dynamic auto 不会主动发DTP包, 但是会主动响应!!

switchport trunk encapsulation dot1q

更改trunk的封装模式

ISL的封装比Dot1q多了30个字节

802.1QFrame

VTP:VLAN Trunk Protocol

VTP Protocol Features

- Advertises VLAN configuration information
- Maintains VLAN configuration consistency throughout a common administrative domain

common administrative domain

- Sends advertisements

```
interface FastEthernet0/10
switchport trunk encapsulation dot1q
switchport mode trunk
switchport trunk allowed vlan all(默认) (1-4094)
switchport trunk allowed vlan remove 100(1-99, 101-409)
switchport trunk allowed vlan except 100(1-99, 101-4094)
```

VTP:

VLAN Trunk Protocol

VTP信息, 只能在Trunk链路上传递.  
是CISCO私有的协议.

VTP用于通告/同步VLAN的配置信息.

VTP信息是触发更新, 或者每5分钟周期性通告一次

VTP 3 种模式, server. client 透明

VTP的Server/Client的相同点

都可以转发VTP通告信息, 而且都会向最新版本在NVRAM中

VTP的Client:

不能.....

运行VTP的条件.

1. domain一致
2. Trunk端口
3. C/S模式(server会与client同步VLAN信息.)

假如VTP信息两边不一样, 这时Trunk能否形成???????????

vtp有3种报文类型.  
summary:由server发来的  
subset:子设置报文,

## 远程

LAB3:通过FR的自动反向ARP, FullMesh的PVC, IGP网络(RIP/EIGRP/OSPF)

Full Mesh:在N个节点的网络中,任何两个节点,都有直接

Step3:启动IGP(RIP/EIGRP/OSPF)

3-1:RIP Over FullMesh FR PVC网络

3-2:EIGRP Over FullMesh FR PVC网络

3-3:OSPF Over FullMesh FR PVC 网络

R1#

```
router ospf 110
```

```
router-id 100.0.0.X
```

```
network 0.0.0.0 255.255.255.255 (所有接口都运行OSPF)
```

观察:

```
1:show ip ospf interface
```

(观察当前运行OSPF的接口有哪些,

并且特别注意:接口的OSPF运行模式/Network Type是哪种?

FR主接口默认是NBMA, 默认不主动发送组播的hello包)

(默认情况下,FR的主接口/Multipoint Subinterface,其OSPF的运行模式都是NBMA)

Point to Point Subinterface:的运行模式都是P2P.

```
2:show ip ospf neighbor
```

∴OSPF的运行模式是NON\_Broadcast (NBMA)

∴OSPF无法建立邻居

解决方案:将OSPF的接口的运行模式,从NBMA改为Broadcast:

R1/R4/R5#

```
conf t
```

```
interface serial 0
```

```
ip ospf network broadcast
```

OSPF邻居,可以成功建立.

```
4:show ip route ospf
```

测试:

```
R1#ping 5.5.5.5!!!!!!!!!!
```

LAB4:使用物理接口,通过静态映射构建IGP网络:(RIP/EIGRP/OSPF)

Step1:关闭FR的自动反向映射:(在CCIE考试中,必须关闭)

在较早的IOS版本中,只在主接口(物理接口)中,有自动反向映射功能

在较新的IOS版本中,主接口和子接口,都有自动反向映射功能.

```
R1(config-if-s0)#no frame-relay inverse-arp
R1#clear frame-relay inarp
```

Step2:在接口中,根据题目中,明确允许使用的PVC,建立FR静态/手工映射:

```
R1#
interface Serial 0
frame-relay map ip 100.0.0.4 104 broadcast
frame-relay map ip 100.0.0.5 105 broadcast
 对方IP 本地的DLCI
```

Step3:IGP网络(RIP/EIGRP/OSPF)的运行,与LAB3完成一致.

## 24晚-PPP认证

RA概述:

remote access:

广域网的远程连接,按L1分类:

1:通过电路交换网络实现的专线:(circuit switching)

1.1:通过真实的专用物理链路,实现的专线:

一般是通过同步串行链路(V.35)连接的,其支持的二层封装协议主要包括:  
HDLC/PPP/SLIP

1.2:通过TDM网络(时分多路复用)实现的专线:

典型有:E1

其支持的二层封装协议与物理专线一样

2:按需拨号的电路交换网络(On-Demand circuit Switched)

一般用于网络的链路备份.

常见的业务包括:ISDN/PSTN(其中PSTN通常是以异步串行连接)

(ISDN也可以通过异步串行连接)

ISDN分为两种业务:(在中国ISDN交换机类型:Basic-net3)(全数字信道)

BRI:2B+D

PRI:30B+D

**B信道:**是用户用于传输数据的信道(用户数据信道),其带宽为64KBPS

**D信道:**是信令信道,不传输用户数据.

**B信道:** 是用户用于传输数据的信道(用户数据信道), 其带宽为64KBPS

**D信道:** 是信令信道, 不传输用户数据.

PRI:D带宽是64kbps; BRI:D带宽是16kbps

PSTN: 56kbps (48~52kbps) (模拟信道)

### 3: 包交换/分组交换(packet switching)

在开始传输数据之前, 必须事先建立一条VC(虚电路), 用于数据包的传输.  
通过同步串行链路连接ISP.

常见业务: X.25/FR/ATM

#### 4: 宽带接入/Broadband Access

xDSL: 主要是传统的电信运营商所经营的网络, 主要使用传统的固话网(双绞线), 作为最后一公里的末端输入.

(语音+data)

Cable: 主要是传统的有线电视运营商所经营的网络, 主要使用传统的CATV网络, 通过线路的双向改造, 作为最后一公里的末端接入.

(视频+data)

无线宽带:

CDMA/GPRS/PHS/3G

powerLine Modem:

(电能+Data)

HDLC:

HDLC一般不推荐, 原因有两个:

1: Cisco的HDLC帧头格式, 携带了一个cisco的私有位:

其好处: 实现HDLC的环境中, 支持多协议: IP/IPX/AT

(appletalk)

其缺点: 只能跟CISCO的设备互通, 不能兼容各厂商设备。

(原因是: 标准的HDLC只支持单协议: IP)

CISCO默认在串口中, 以HDLC为2层封装协议。

2: HDLC协议, 本身不支持认证, 无法保证安全性

建议使用PPP, PPP有多种可选模块, 可以提高网络安全性, 提升性能。

(PPP可支持认证)

(PPP可支持认证)

SLIP, 相当于是PPP前身, 功能单一, 趋向淘汰

在CISCO设备上, 串行链路默认使用CISCO HDLC, 在华为的设备上, 默认使用PPP

PPP (Point-to-point protocol)

~~~~~  
~~~~~  
PPP是业界开放性的标准, 支持多协议环境, 所有的厂商都可以支持。

HDLC/PPP的对比:

HDLC不支持多协议, PPP支持多协议

HDLC不支持认证, PPP支持认证

LCP (link control Protocol)

负责对L1的物理层链路, 进行链路的建立, 控制, 维护,

NCP (network control protocol)

负责对L3的网络层, 向下提供无差别的接口 (\*CP)

LCP包含了4大网络模块:

1: 认证 (authentication:PAP/CHAP)

2: 压缩 (compress)

3: 回拨 (callback)

4: 多链路捆绑 (mulit-link)

LAB1: Encapsulation PPP (从HDLC到PPP的迁移)

~~~~~  
~~  
step1: 确认L1/L2/L3通达

(L2: HDLC) / (L3: 网络协议/被路由协议: IP, 寻路/路由协议: RIP)

routed

routing

L1: V.35 sync serial1/的同步串行链路

L2: HDLC/PPP

L3: IP/RIP

step2: 在R1-R3之间的链路改为PPP

R1/R3(config)#in s0

enca ppp

观察:

R1#debug ppp negotiation (PPP的协商)

1:Interface Serial0, changed state to up (L1 up)

2:LCP: State is Open

3:PPP的认证:

(这是可选项目, 如果进行认证, 就必须成功, 才有NCP的工作)

4-1: sel IPCP: State is Open (IP)

4-2: sel CDPCP: State is Open (cdp) (show cdp neighbor)

5:Line protocol on interface serial1, changed state to up (L2 up)

检查:

R3#show interfaces s0

serial0 is up, line protocol is up

enca ppp

LCP Open

open: IPCP, CDPCP

测试:

R2#ping 3.3.3.3!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

R1#PPP链路的对方接口的32位主机路由:

C 13.0.0.3/32

R1(config)#in serial 1

R1(config-if)#no peer neighbor-route (取消本路由表里面的32位明细路由, 仅对自己有效)

PPP authentication

PAP/CHAP (PPP的认证, 是链路的认证)

PAP (password authentication protocol)

两次握手, 建议在网络工程中使用双向认证。

两次握手:

1: 被认证方, 将对方所定义的账号/密码, 以明文方式, 发送给主认证方。

2: 主认证方, 把收到的账号/密码, 与自己的数据库进行核对后, 发回认证成功与否的信息。

PAP的缺点: 账号/密码以明文方式, 在链路上传输, 不安全

PAP的缺点：账号/密码以明文方式，在链路上传输，不安全

LAB2: PAP认证 (R1-R3)

~~~~~  
step1:确认链路已经是封装为PPP链路

step2:在本路由器的数据库中，为对方构建账号/密码:

R1 (config) #username xx password xx

R3(config)#username yy password yy

step3:选定PPP的认证方式为PAP

R1/3 (config-if)#PPP authentication PAP

step4:将“自己在对方数据库中的账号/密码，发送给对方，供对方进行校验

R1(config-if)#ppp pap sent-username XX password XX

R3(config-if)#ppp pap sent-username YY password YY

(密码用户名大小写敏感)

观察:

R3#debug ppp authentication (PPP的认证)

LAB3: 使用”主机名“作为”用户名“的简化的PAP认证:

~~~~~  
step1:

R1(config)#username R3 password R3

R3(config)#username R1 password R1

step2:

R1(config-if)#PPP pap sent-username R1 password R1

R3(config-if)#ppp pap sent-username R3 password R3

PPP chap认证 (challenge handshake authentication protocol)

~~~~~  
3次握手:

chap的优点:

从不在链路传送密码，challenge (X) 和response (Y) 都是随机数，这两者间是不可逆运算的，可以确保密码不被破译，保证网络的安全性。

1: 主认证方的路由器，发出随机数 (X=9)

2: 被认证方的路由器，将接收的随机数，和事先定义好的密码=7，一起放入MD5的加密器，进行HASH算法的计算加密，把得到的数值y=32，response的形式，发送给主认证方。

3: 主认证方，同样进行与第2步相同的操作，将得到的数值y'，与从被认证方发来的Y，进行比较:

如果一致，发出认证成功信息:

方发来的Y，进行比较：  
如果一致，发出认证成功信息：  
如果不一致，发送认证失败信息

#### LAB4: CHAP认证:

~~~~~

step1: 确认链路已经是封装为PPP链路  
step2: 为对方建账号/密码:  
R1(config)#username xx password xx  
R2(config)#username yy password yy  
step3: 选定认证方式是CHAP:  
R1/2(config-if)#ppp authentication chap

step4: 选定某组账号密码，进行CHAP认证  
R2(config-if)#ppp chap hostname yy  
R2(config-if)#ppp chap password yy

R1(config-if)#ppp chap hostname xx  
R1(config-if)#ppp chap password xx

打开接口，观察CHAP认证过程  
" authentication failed " 原因是双方密码不一致  
"MD/DES compare failed"

step5: 在CHAP认证中，密码必须一致:  
R1(config)#username xx password xx  
R2(config)#username xx password xx  
认证成功

#### LAB6: 使用主机名作为用户名，简化的CHAP认证

~~~~~

step1: 直接使用路由器的主机名，进行CHAP认证:  
R1(config)#username R2 password xx  
R2(config)#username R1 password xx

step2: 在PPP接口中，只需要以下命令:  
interface serial 0  
encapsulation ppp  
ppp authentication chap

PPP/compression PPP 压缩

~~~~~

PPP压缩可以提高PPP链路的带宽利用率。

支持3种压缩算法：

- 1: mppc
- 2: predictor
- 3: stac

step1: 按图配置好IP，确认L1/L2/L3 (RIP)

step2: 将R2-R4链路更改为PPP链路

step3: 在R2-R4的PPP链路的接口中，启动PPP压缩：（3选1）

3-1: (if-s0) compress mppc

3-2: (if-s0) compress predictor

3-3: (if-s0) compress stac

3-4: TCP头压缩：（语音的数据包，其数据净荷很小，头大身小情况）  
(config-if) #ip tcp header-compression

## 25晚-Frame-relay

FR（帧中继）（帧中继是纯2层协议）

帧中继的特征：

1: 面向连接的服务：

必须在通信开始之前，通过VC（虚电路）构建好  
（VC：通信双方之间的虚拟通道）

PVC：永久虚电路，长期连接不中断的。

SVC：交换式虚电路，在用户需要连接的时候，才临时构建起来的

2: 帧中继的用户，是充当FR的DTE端

而ISP是充当FR的DCE端（FR CLOUD）

NNI (network network interface)

3: Frame relay signaling/帧中继的信令：

FR用户通过LMI (Local management Interface)  
本地管理接口

跟FR SW进行协商，获得PVC的相关信息

LMI的标准3种：

（在IOS12.0以后的版本中，CISCO路由器可以自动检测到ISP所用的LMI类型）

- 1: ANSI
- 2: Q933A
- 3: CISCO兼容的

4: 通过LMI的状态，判断FR PVC的故障点。

5: FR的L3 IP地址, 与L2的DLCI地址的映射关系

对方的L3 IP与本地的L2的DLCI进行映射

```
FR MAP A B
L3 IP 10.1.1.1 10.1.1.2
L2 DLCI 500 600
```

LAB1: 基本的帧中继交换机 (2端口)

(基于每个物理连接, 一条PVC的HUB&SPOKE架构)

~~~~~

step1:

```
R2(config)#hostname FR-sw2
FR-SW2(config)#no ip routing
FR-SW2(config)#frame-relay switching
```

step2: 为V.35 DCE接口配置同步时钟

```
FR-SW2(config-if-s0/s1)#clock rate 2000000
```

step3: FR接口的基本配置:

```
interface s0/s1
 encapsulation frame-relay
 frame-relay lmi-type cisco
 frame-relay intf-type dce(FR DCE)
```

step4: 配置FR路由:

```
FR-SW2(config-if-s0) #在PVC的入口S0
frame-relay route 104 interface serial 1 401
 进入的PVC 出口 出去的PVC
```

FR-SW2(config-if-s1) #在PVC的入口: S1

```
frame-relay route 401 interface serial 0 104
```

step5: 在用户端封装FR, 然后将ISP和用户的端口都打开 (L2测试)

```
R4/R1#
interface serial 0
 encapsulation frame-relay
 no shutdown
R1/4#show ip int brief
serial0 L1:up L2:up
```

```
R1#show frame-relay pvc
```

DLCI=104,.....PVC STATUS=ACTIVE, INTERFACE=Serial0

step6:L3测试:

在PVC上，用户的接口中，配置同一个网段的IP地址

FR的自动反向ARP:

R1#show frame-relay map

serial0(up):ip 10.0.0.4 dlci 104,

对方的IP 本地的DLCI的映射

dynamic:通过FR的自动反向ARP学到的

broadcast:L2的FR PVC允许广播/组播的流量通过

active : PVC工作状态是OK的

R1#ping 100.0.0.4!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

step7:FR

LAB2:3端口的FR-SW的配置

~~~~~  
~~~~~

step1~~~3:与LAB1一致

step4:构建tunnel:

4-1:

确认FR-SW2，FR-SW3的以太网接口:

在以太网接口配置IP地址。

4-2:

FR-SW2 (config) #

interface tunnel 2

tunnel source 23.0.0.2

tunnel destination 23.0.0.3

FR-SW3(config)#

interface tunnel 3

tunnel source 23.0.0.3

tunnel destination 23.0.0.2

step5:R4~R5之间的PVC（405/504）的FR路由:

FR-SW2 (config-is-s1) #

frame route 405 int tunnel 1 1000

```
FR-SW3 (config-is-s1) #
frame route 504 int tunnel 1 1000
```

step6:R1~R5之间的PVC (105/501) 的FR路由:

```
FR-SW2 (config-is-s0) #
frame route 105 int tunnel 2 1001
```

```
FR-SW3 (config-is-s1) #
frame route 501 int tunnel 3 1001
```

FR-SW3#show frame-relay route (在帧中继交换机的路由状态)

LAB3: 通过FR的自动反向ARP, fullmesh的PVC, IGP网络  
(RIP/EIGRP/OSPF)

~~~~~  
~~~~~

fullmesh:在N个节点的网络中, 任何两个节点间, 都有直接相连的连接  
fullmesh连接, 所需要的PVC条数的计算公式:  $N * (N - 1) / 2$

step1:在FR的用户端 (CPE), 封装FR, 打开接口:

step2:在CPE端配置IP地址, 观察FR自动反向ARP的L2/L3的映射:

R1#show frame-relay map

step3: 启动IGP (RIP/EIGRP/OSPF)

3-1: R1over fullmesh FR PVC 网络

3-2:EIGRP over fullmesh FR PVC 网络

```
R1#router eigrp 100
network 0.0.0.0 (所有接口, 都运行EIGRP)
no auto-summary
```

3-3:OSPF over fullmesh FR PVC 网络

```
R1#
router ospf 100
router-id 100.0.0.x
network 0.0.0.0 255.255.25.255 area 0 (所有接口都运行OSPF)
```

观察:

1: show ip ospf interface

(观察当前运行OSPF的接口有那些, 并且特别注意: 接口的OSPF运行模式/network type是哪种)

FR主接口默认是NBMA, 默认不主动发送组播的hello包的)

(默认情况下, FR的主接口/multipoint subinterface, 其OSPF的运行模式都是NBMA)

point to point subinterface:的运行模式都是P2P  
2: show ip ospf neighbor  
因为OSPF的运行模式是non\_broadcast (nbma), 所以OSPF无法建立邻居

解决方案:

将OSPF的接口运行模式, 从NBMA改为BROADCAST:

R1/R4/R5#

conf t

interface serial0

ip ospf network broadcast

OSPF邻居建立成功

ip ospf network point-to-point

LAB4: 使用物理接口, 通过静态映射构建IGP网络: (RIP/EIGRP/OSPF)

~~~~~  
~~~~~

STEP1:关闭FR的自动反向映射: (在CCIE考试中, 必须关闭)

在较早的IOS版本中, 只在主接口(物理接口)中, 有自动反向映射功能  
在较新版本的IOS中, 主接口和子接口, 都有自动反向映射功能

R1 (config-if-s0) #no frame-relay inverse-arp

R1#clear frame-relay inarp 清除FR MAP 的CACHE

step2:在接口中, 根据题目中, 明确允许使用的PVC, 建立FR静态/手工映射:

R1#

interface serial0

frame-relay map ip 100.0.0.4 104 broadcast

frame-relay map ip 100.0.0.5 105 broadcast

对方的IP 本地的DLCI

Step3:IGP网络(RIP/EIGRP/OSPF)的运行, 与LAB3完全一样

## 26晚-Frame-relay

FR子接口:

子接口判断原则:

1. 判断创建几个子接口?(一个子接口, 对应一个ip子网)

在路由器的一个物理的FR接口中,  
如果对应着几个子网, 就应该创建几个子接口.

2. 判断子接口的类型: (P2P/MP)

如果一个子接口中, 对方是多于一个点, 那么接口类型是MP:

建议:多点子接口,只适用于Full Mesh的网络拓扑.

建议:

多点子接口,只适用于Full Mesh的网络拓扑.

(多条PVC,对应一个IP子网)

相当于一个Mult-Access接口.

如果一个子接口中,对方是只有一个点,那么接口类型是P2P.

建议:

在Hub&Spoke/星型网络拓扑中使用.

有多个分支点/分公司,就应该创建多少个点对点子接口.

(一条PVC,对应一个IP子网)

相当于一个点对点链路的接口.

LAB5:使用多点子接口,替代物理接口,构建IGP网络:(RIP/EIGRP/OSPF)  
(考虑到网络将来的发展,扩展性) (Full-Mesh)

~~~~~  
多点子接口和物理接口的配置方法完全一样!!

本实验与LAB3/LAB4完全一样.

LAB4:interface serial0(主接口)

LAB5:interface serial 0.100 multipoint

LAB6:通过将Hub&Spoke的FR网络,  
每个分支点的PVC作为独立的一个子网/30,构建P2P的子接口.  
(作为Hub&Spoke拓扑,最佳解决方案)

Step1:

配置FR的主接口:

```
interface Serial0
no ip address(无需配置IP)
encapsulation frame-relay
no frame-relay inver
no shutdown
```

Step2:配置FR的P2P子接口:

R4#

```
interface serial0.401 point-to-point
ip address 100.0.0.2 255.255.255.252
frame-relay interface-dlci 401
interface Serial0.405 point-to-point
ip address 100.0.0.6 255.255.255.252
frame-relay interface-dlci 405
R1#
interface Serial 0.104 point-to-point
ip address 100.0.0.1 255.255.255.252

R5#interface serial0.504 point-to-point
ip address 100.0.0.5 255.255.255.252
frame-relay
```

检查:

```
R4#show frame-relay map
```

Serial0.405 (up):

point-to-point dlci, dlci 405, broadcast, active

P2P子接口中, ping该子接口所在网段中的所有节点, 都能通(包括本机节点)

MP子接口中, ping该子接口所在网段中的所有节点, 都必须手工做映射  
(FrameRelay map) (包括本机节点)

Step3: 运行IGP (RIP/EIGRP/OSPF):

3-1: RIP:

中心点R4#

```
router rip
version 2
network 4.0.0.0
network 100.0.0.0
no auto-summary
```

3-2: EIGRP:

```
router eigrp 100
network 0.0.0.0
no auto-summary
```

3-3: OSPF:

```
router ospf 110
router-id 110.0.0.4
network 0.0.0.0 255.255.255.255 area 0
```

```
network 0.0.0.0 255.255.255.255 area 0
```

分支点到达汾河的一个网络的下一跳。都是中心点100.0.0.2 下一跳可达

LAB7:

使用多点子接口(Multipoint sub-is),

构建Hub&Spoke FR网络

通过静态映射构建RIP网络

~~~~~

R4#

```
interface serial0.100 multipoint
```

```
ip add 100.0.0.4 255.255.255.0
```

```
frame-relay map ip 100.0.0.1 401 broadcast
```

```
frame-relay map ip 100.0.0.5 405 broadcast
```

R1#

```
interface Serial0.100 multipoint
```

```
ip address 100.0.0.1 255.255.255.0
```

```
frame-relay map ip 100.00.4 broadcast
```

R5#

```
interface Serial0.100 multipoint
```

```
ip address 100.0.0.5 255.255.255.0
```

```
frame-relay map ip 100.0.0.4 504 broadcast
```

Step1:

3个路由器都运行RIP:

察看FR接口的水平分割特性:(debug ip rip)

R4#show ip interface s0.100

Split horizon is enabled(启动/生效)

在FR的子接口中,不论是点对点,还是多点子接口,其水平分割默认都是启动/Enable的.

而水平分割默认都是启动/Enable的.

而FR物理接口/主接口的水平分割,默认是关闭的

关于DV协议的水平分割问题:(RIP/IGRP)

现象:

分支点的路由器,只有中心点的路由,没有别的分支点的路由.

解决方案:

Step2:关闭中心点路由器的接口的水平分割:

```
R4(config)#in s0.100
```

```
R4(config-subif)#no ip split-horizon
```

Step3:

路由下一跳存在不可达问题:

```
R5# 1.0.0.0/8 [120/2] via 100.0.0.1
```

```
R1# 5.5.5.0 [120/2] via 100.0.0.5
```

Step4:为了解决路由的下一跳问题,

在分支点上,手工做别的分支点的静态映射:

```
R1(config-subif)#frame-relay map ip 100.0.0.5 104 (bro)
```

```
R5(config-subif)#frame-relay map ip 100.0.0.1 504
```

测试:

LAB8:

使用多点子接口(Multipoint Sub-if)

构建Hub&Spoke FR网络

通过静态映射构建EIGRP网络

~~~~~

Step1:EIGRP的水平分割,与普通DV协议的,是不同的:

```
R4(config-subif)#no ip split-horizon eigrp 90
```

Step2:

R1/R5两分支点间的用户之间的通信,无需做映射:

```
R1# 5.5.5.0 [90/2809856] via 100.0.0.4 (中心点,自动下一跳)
```

```
R5# 1.0.0.0/8 [90/2809856] via 100.0.0.4
```

```
R1#ping 5.5.5.5 source 1.1.1.1 !!!!!!!!!!!!!
```

Step3:访问分支点的外口,还是需要做相互映射:

```
R1#ping 5.5.5.5
```

```
R1(config-subif)#frame-relay map ip 100.0.0.5 104
R5(config-subif)#frame-relay map ip 100.0.0.1 504
```

```
R1#ping 5.5.5.5!!!!!!!
```

LAB9:

使用多点子接口(Multipoint Sub-if),  
构建Hub&Spoke FR网络  
通过NBMA模式, 构建OSPF网络:

~~~~~

Step0: 确认OSPF的接口的运行模式(Network Type)

Step1: 邻居问题:

```
R5#show ip ospf interface
Network Type Non_Broadcast (NBMA)
```

确认L2的FR的PVC是否允许广播流量或者主播流量通过:

```
R5#show frame-relay map
broadcast (是否有Broadcast对此LAB无影响)
无法建立邻居!
```

Step1: 邻居问题: (通过OSPF的单播更新)

```
router ospf 110
neighbor 100.0.0.1
neighbor 100.0.0.5
```

现象: DR/BDR/DR-Other混乱, 影响OSPF数据库不正常.

Step2: DR问题:

```
R4#
in s0.100
ip ospf priority 100
```

```
R1/R5#
in s0.100
ip ospf priority (放弃DR选举)
```

现象:

∴ OSPF是LS协议, ∴ 没有水平分割问题, 分支点之间都没有对方路由.  
如果R1/R5之间没有对方的下一跳映射, 是无法互通的.

Step3: 下一跳问题: (手工配置FR的下一跳映射:)

```
R1# o 5.5.5.5[110/65] via 100.0.0.5
R5# o 1.1.1.1[110/65] via 100.0.0.1
```

```
R1#frame-relay map ip 100.0.0.5 104
R5#frame-relay map ip 100.0.0.1 504
```

LAB10:

使用多点子接口 (Multipoint Sub-if),  
构建Hub&Spoke FR网络  
通过P2MP NON-Broadcast模式, 构建OSPF网络:  
~~~~~

Step1: R1/4/5# 选定运行模式:

```
in se0.100#
ip ospf network point-to-multipoint Non-broadcast
```

网络环境: 与LAB10完全一致.

Step2: 邻居问题: (与NBMA一样) (同LAB9)

Step3: DR问题: (P2MP运行模式中, 根本没有DR的概念, 无DR问题)  
无需使用接口中的 "ip ospf priority \*", 来控制DR选举.

Step4: 下一跳问题: (无下一跳问题)

所有分支点, 所收到的OSPF路由, 的下一跳都集中在中心点. (自动下一跳)

```
R1# 0 5.5.5.5 [110/129] via 100.0.0.4
R5# 0 1.1.1.1 [110/120] via 100.0.0.4
```

特别注意: P2MP模式会自动产生所有该FR网络的接口的/32主机路由, 其下一跳也中心点,  
∴

LAB11:

使用多点子接口 (Multipoint sub-if),

构建Hub&Spoke FR网络

通过P2MP (Broadcast模式, 构建OSPF网络:

~~~~~

网络环境:

L2 FR必须允许广播流量通过!!

Step1:R1/4/5选定运行模式:

in s 0.100

ip ospf network point-to-multipoint

Step2:邻居问题:(与Broadcast一样)

OSPF邻居能够自动建立, 不存在问题.

(不需要单播更新)

Step3:DR问题:(P2MP模式中, 根本没有DR的概念, 无DR问题)(同LAB10)

Step4:下一跳问题:(无下一跳问题)(LAB10)

所有分支点的路由的下一跳都集中在中心点.(自动下一跳)

## 27晚-ISDN

ISDN

~~~~~

2层电话号码:

3层IP:

1:ISDN主要有两种接口类型:BRI/PRI

2:BRI:是2B+D B信道:64Kpbs, D信道:16Kbps.

3:PRI:中国的PRI是基于E1, 30B+D, B信道:64Kbps.

ISDN的交换机类型:

isdn switch-type basic-net3(中国标准, 欧洲标准)

isdn switch-type basic-ni (Wolf实验室, 北美标准)

Step1:L1/L2通:



2-6: 定义“能够触发ISDN起拨的”数据流:

2-6-1: 使用ACL, 定义可以触发ISDN起拨的数据包:

```
R3(config)#access-list 1 permit any
```

2-6-2: 在Dialer-list 3中, 调用ACL:

```
R3(config)#dialer-list 3 protocol ip list 1
```

(任何数据包都可以触发ISDN的起拨)

2-6-3: 在接口中调用dialer-list 3

```
R3(config)#in bri 0
```

```
R3(config-if)#dialer-group 3
```

```
R4(config)#dialer-list 4 protocol ip permit
```

```
R4(config)#in bri 0
```

```
R4(config-if)#dialer-group 4
```

L3: 测试: Ping

```
R3#ping 34.0.0.4!!!!!!!!!!!!!!!!!!!!!!
```

Step3: DDR/按需拨号: (Dial on Demand Route)

~~~~~

3-1: 确保全网的路由器, 来去都有正确的路由:

```
R3(config)#ip route 4.4.0.0 255.255.0.0 34.0.0.4
```

```
R4(config)#ip route 10.0.0.0 255.0.0.0 34.0.0.3
```

```
R4(config)#ip route 20.0.0.0 255.0.0.0 34.0.0.3
```

3-2: 确认上述路由的下一跳的可达性: (R3/R4)

(确认有到达“路由的下一跳”的映射)

```
R3#dialer map ip 34.0.0.4 broadcast 810888
```

```
R4#dialer map ip 34.0.0.3 broadcast 810777
```

(察看“路由的下一跳”的映射:

```
show dialer maps
```

3-3:修改/收窄“能够触发ISDN起拨的”数据流:(定义“感兴趣流”)

修改2-6-1中的ACL:

```
R3(config)#access-list 1 permit 10.0.0.0 0.255.255.255
```

2-6-2, 2-6-3无须修改.

| DDR      | 之前     | 之后                                           |
|----------|--------|----------------------------------------------|
| 10.x.x.x | 不通的情况下 | 通的                                           |
|          | 通的情况下  | 把Idle Time复位了/Reset                          |
| 20.x.x.x | 不通的情况下 | 断                                            |
|          | 通的情况下  | 不会把Idle time Reset,当Idle time 到达120秒时,ISDN断开 |

修改Idle time

```
R3(config-if)#dialer idle-timeout 100
```

Step4:PPP通:

将ISDN链路的L2封装更改为PPP:

```
interface bri0
encapsulation ppp
```

Step5:认证通:(CHAP认证)

```
username R4 password R-34
```

```
interface bri0
ppp authentication chap
```

Step6:PPP Multilink(实验室里用模拟器,看不到现象)

(可以将两个B信道捆绑在一起,成为一个128Kbps的信道)

```
R3(config-if)#ppp multilink load-threshold 167
(链路负荷/利用率一个B信道的65%时,启动第二B信道)
```

Step7:PPP Compress:

在R3-R4的PPP链路的接口中, 启动PPP压缩: (3选一)

- 1: (if-b0)compress mppc
- 2: (if-b0)compress predictor
- 3: (if-b0)compress stac

4:TCP头压缩: (语音的数据包, 其数据净荷很小, 头大身小情况)  
(if-b0)#ip tcp header-compression

Step8:PPP call-Back/回拨

8-1:

R3(config-if)#ppp callback request

8-2:

R4(config-if)#ppp callback request  
R4(config-if)#dialer callback-secure

8-3:

R4(config)#map-class dialer CB  
R4(config-map-class)#Dialer callback-server username

8-4:

R4(config-if)#  
dialer map ip 34.0.0.3 name R3 class CB broadcast 810777

R3#

dialer map ip 34.0.0.4 name R4 broadcast 810888

R3#debug isdn Q931 (观察ISDN连接的过程)

LAB2:ISDN备份: (使用Dial-profile, 做Backup接口)

按照拓扑, 配置网络链路,, 并且测试接口链路.

Step1:

router eigrp 90

```
network 0.0.0.0
```

Step2:使用物理的ISDN接口做DDN主链路的备份:

```
R3/4(config-if-S1)#backup interface bri 0
```

定义ISDN是:主链路S1口的备份接口

ISDN接口成为主链路的备份接口后, down

Step3:使用虚拟的"Dial"接口, 做备份: (Dial-Profile)

```
interface BRI0
no ip address
encapsulation ppp
dialer pool-member 10(将物理接口, 放入Dial-Pool 10中)
```

```
isdn switch-type basic-ni
```

```
isdn sp11 31077701
```

```
isdn sp12 81077702
```

```
interface dialer4(逻辑接口)
ip address 34.0.0.3 255.255.255.0
encapsulation ppp
dialer pool 10(Dial4口在Dial-pool 10 中调用物理接口)
dialer idle-timeout 100
dialer string 810888
dialer-group 3
ppp authentication chap
multilink load-threshold 167 outbound
```

Step4:在DDN主链路接口中, 指定Dialer4为备份接口:

```
interface serial1
backup interface Dialer4
```

LAB3:使用浮动静态路由, 实现链路备份:

ISDN链路, 是不运行动态路由协议的, 也无须指定备份接口.  
(ISDN配置浮动静态路由)

Step1:

允许所有数据包触发ISDN起拨的:

```
access-list 1 permit any
dialer-list 3 protocol ip list 1
```

EIGRP是不会触发ISDN起拨的,  
•EIGRP是不network这个ISDN链路的.

Step2:配置浮动静态路由:

```
R3(config)#ip route 4.4.0.0 255.255.0.0 34.0.0.4 91
```

```
R4(config)#ip route 23.0.0.0 255.255.255.0 34.0.0.3 91
```

浮动:  
备份链路中的静态路由的AD:91,  
比主链路中从动态协议中获得的路由的AD:90, 要高.

## 28晚-NAT

NAT(Network Address Translation)

NAT的3组关键概念:

NAT的图:

LAB1:NAT的基本配置(原理性的NAT)

Step1:按图配置IP网络, 测试链路, R1-R3-R5之间链路运行IGP:RIPv2

Step2:在模拟PC2/PC4的路由器上, 关闭IP路由, 并指定网关:

```
R4(config)#hostname PC4
PC4(config)#no ip routing
PC4(config)#ip default-gateway 192.168.1.1
```

并确认, PC2/PC4都能访问到自己的网关:

Step2.5:指定NAT路由器的外口,内口:

```
interface Ethernet0
ip nat inside
interface Serial1
ip nat outside
```

Step3:静态NAT:

```
R1(config)#ip nat inside source static 192.168.1.2 13.0.0.10
 私网IP 公网IP
```

Step4:观察NAT转换的过程:

```
R1#debug ip nat
```

```
NAT*:s=192.168.1.2->13.0.0.10, d=5.5.5.5(去包)
```

```
NAT*: s=5.5.5.5, d=13.0.0.10->192.168.1.2(回包)
```

```
R1#show ip nat translations
```

LAB2:在中小型企业,通常使用基于接口的端口复用:(PAT/NAPT/Overload)

Step1:定义NAT的外口/内口:(同LAB1)

Step2:通过ACL,定义准备进行NAT的内网的用户群:

```
R1(config)#access-list 1 permit 192.168.1.0 0.0.0.255
 (192.168.1.*)
```

Step3:进行基于接口的NAT端口复用:

```
R1(config)#
ip nat inside source list 1 interface serial 1 overload(新IOS版本
里面默认启用overload)
 (内网用户) (NAT的外口) 端口复用
```

测试:

实现了192.168.1.\*整个网段的用户, 共享一个IP地址, 连接到Internet.

LAB4:大型企业, 向ISP购买较多的公网IP后,

将其分别放入不同NAT-Pool, 以供不同的部门进行NAT转换:

~~~~~  
~~~~~

Step1:定义NAT的外口/内口:(同LAB1)

Step2:在R1上模拟两个网段的网关:

```
R1#interface Ethernet0
ip address 192.168.20.1 255.255.255.0 secondary
ip address 192.168.10.1 255.255.255.0
```

Step3:在两个网段/VLAN中的用户, 分别设定自己的网关

```
PC2(config-if-e0)#
ip address 192.168.10.2 255.255.255.0
ip default-gateway 192.168.10.1
```

```
PC4(config-if-e0)#
ip address 192.168.20.4 255.255.255.0
ip default-gateway 192.168.20.1
```

确认每个内网用户, 都能访问到自己的网关.

Step4:通过ACL, 定义准备进行NAT的用户群:

```
R1(config)#access-list 1- permit 192.168.10.0 0.0.0.255
access-list 20 permit 192.168.20.0 0.0.0.255
```

Step5:将从ISP处买到的公网IP, 分别放入为不同VLAN准备的地址池中:

```
R1(config)#
ip nat pool PL-10 13.0.0.8 13.0.0.11 prefix-length 24
ip nat pool PL-20 13.0.0.12 13.0.0.15 prefix-length 24
ip nat pool PL-20 13.0.0.12 13.0.0.15 255.255.255.0
(两句一样的效果)
```

Step6:为不同的部门, 进行基于不同NAT-Pool的转换:

```
R1(config)#
ip nat inside source list 10 pool PL-10 Overload
```

```
ip nat inside source list 20 pool PL-20 Overload
```

LAB5:通过NAT,实现镜像服务器的负载均衡:

~~~~~

Step1:定义NAT外口/内口.(同LAB1)

Step2:通过ACL,定义“我方的,公网”地址,  
作为来自外网访问的目标地址.

```
R1(config)#access-list 30 permit host 13.0.0.100
```

Step3:定义内网的服务器群的地址池,轮转类型.

```
ip nat pool ser 192.168.10.2 192.168.10.3
prefix-length 24 type rotary
```

Step4:

```
R1(config)#ip nat inside destination list 30 pool ser
```

Step5:

```
PC-2/4(config)#line vty 0 4
```

```
PC-2/4(config-line)#no login(不需要密码,就可以被Telnet)
```

```
R5#Telnet 13.0.0.100
```

LAB6:使用一个接口的公网IP,对外网提供多个不同的服务器:

(静态的TCP端口映射,静态的端口复用)

~~~~~

~~~~~

Step1:定义NAT的外口/内口:(同LAB1)

Step2:

工程上的做法:

www server:

```
ip nat inside source static tcp 192.168.10.3 80 13.0.0.1 80
```

FTP server:

```
ip nat inside source static tcp 192.168.10.2 21 13.0.0.1 21
 (私网) (公网)
```

Wolf实验室的做法:

```
ip nat inside source static tcp 192.168.10.3 23 13.0.0.1 80
ip nat inside source static tcp 192.168.10.2 23 13.0.0.1 21
```

Modem connections:

LAB1:通过异步Modem/(PSTN程控交换机/语言交换网/7号信令网/ss7), 远程控制路由器的AUX口:  
(带外网管)

Step1:当人还在北京的机房中时, 在路由器的AUX进行基于配置:  
(使用Console线控制Console口)  
(目的:用于控制路由器的AUX口和Modem之间的通信)

1-1:  
(察看路由器的AUX口的绝对线路号)

```
BJ-2610#show line
Tty Typ
* 0 CTY(Console)
```

64(2个4\*8异步串口模块/32AS)

1-2:  
进入Aux口:  
BJ-2610(config)#line 65(绝对线路号)  
BJ-2610(config)#line aux 0(相对线路号)

1-3:配置AUX口:  
flowcontrol hardware(硬件流控)  
transport input all(在接口中传输所有协议)  
password 123(从AUX口进入路由器的密码)  
Login(登录)

1-4:配置进入特权模式的密码:

```
BJ(config)#enable password wolf
```

Step2:使用AT命令集,配置Modem:

(笔记本电脑的Com口,通过Console线,RJ-45转DB-25转换器,连接到Modem的DB-25接口)

每个的Modem厂商的AT命令集都可能不一样,需要查询文档.

然后通过虚拟终端软件(超级终端/Secu-CRT)

1:

```
at (输入)
```

```
OK (输出) 说明工作状态OK
```

2:

```
at&v (察看Modem的当前配置)
```

Active profile:相当于PC的内存

Stored Profile 0:

Stored Profile

3:

```
at&f (将Modem恢复出厂的默认设置)
```

```
at&f1 (每个Modem厂商的AT命令都有可能不一样.)
```

```
OK
```

4:自动应答 (AutoAnswer/AA)

(S0寄存器=0:表示不应答,是默认值)

```
ats0=3 (将Modem的自动应答的响铃次数,从0设置为3)
```

```
OK
```

5:

```
at&w (保存配置)
```

```
OK
```

6:

```
ate0 (关闭字符回显) 正在打出的命令不显示
```

```
ate1 (打开字符回显)
```

Step3:按图连接好, 路由器AUX-反转线-Modem-电话线-PSTN网

Step4:回到深圳, 使用深圳本机的Modem, 拨北京的Modem  
(电话号码:810)

## 29晚

LAB2:终端服务器:

~~~~~

通过Telnet到终端服务器, 从终端服务器的8异步口, 反向Telnet到多个网络设备的Console口:

固定异步串行线路:2509:8个异步口:2511:16个异步口

模块化的AS Network Module:26\*\*, 36\*\*,

Step1:

使用Console线, 配置2509的以太口, 并且配置好密码, 便于Telnet:

```
hostname R2509
```

```
!
```

```
enable password wolf
```

```
!
```

```
interface Ethernet0
```

```
ip address 192.168.1.103 255.255.255.0
```

```
no shutdown
```

```
!
```

```
line vty 0 100
```

```
no login(不要密码)
```

```
line vty 0 100
```

```
password 123
```

```
login in
```

```
line vty 0 100
```

```
login local(使用路由器中的本地的账号, 密码进行认证)
```

```
每个人都有自己的密码
```

Step2:在主机上, Telnet 192.168.1.103

Step3:

```
R2509(config)#line 1 8
R2509(config-line)#transport input all
```

Step4:配置用于反向Telnet的环回口:

```
R2509(config)#interface loopback 1
R2509(config-if)#ip address 1.1.1.1 255.255.255.255
```

Step5:

把八爪鱼线的第1根线, 连到Rack02号实验台的R1上的Console口, 并为此创建“快捷方式”:

```
R2509(config)#ip host rack02r1 2001 1.1.1.1
```

Step6:

键入“rack02r1”

就可以进行对本地的1.1.1.1这个设备的2008端口的反向Telnet.

```
R2509#telnet 1.1.1.1 2001
```

Step7:banner:

```
R2509(config)#banner motd $
```

E1线路知识要点:

在中国/欧洲使用E1, (欧标)

在北美/日本作用T1. (美标)

1. 一条E1是带宽为:2.048M的链路, 用PCM编码.
2. 一个E1的帧长为256个Bit, 分为32个时隙, 一个时隙为8个Bit.  
(timeslots:时隙)
3. 每秒有8K个E1的帧通过E1接口, 即 $8K \times 256 = 2048Kbps$
4. 因为每个时隙在E1帧中占8bit,  
∴一个时隙的带宽是 $8bit \times 8k = 64Kbit$ ,  
即一条E1中含有32个64K信道/时隙.

E1帧结构:

E1有不成帧, 成帧, 成复帧三种方式:

1:

在不成帧有E1中(透明模式):

所有32个时隙都可用于传输有效数据.

2:

在成帧的E1中:

第0时隙用于传输帧同步数据.

其余31个时隙可以用于传输有效数据.

3:

在成复帧的E1中

第0时隙用于传输帧同步数据,

第16时隙是用于传输信令,

只有1 到15,

使用E1线路实现多个64K专线连接:

以下例子为:

E1连接3条64K专线, 帧类型为NO-CRC4, 非平衡链路, 路由器具体设置如下:

```
hostname shanxi
```

```
Step1:controller E1 1 (进入E1控制器)
```

```
Step2:framing NO-CRC4
```

```
linecode hdb3
```

```
Step3:channel-group 1 timeslots 1
```

```
 channel-group 2 timeslots 2
```

```
 channel-group 3 timeslots 3
```

```
exit
```

```
Step4:
```

```
interface serial:1
ip address 222.119.96.1 255.255.255.252
interface serial1:2
ip address 222.119.96.5 255.255.255.252
interface serial1:3
ip address 222.119.96.9 255.255.255.252
```

Step5:

```
ip route 139.20.40.0 255.255.255.0 Serial1:1
ip route 139.20.41.0 255.255.255.0 serial1:2
ip route 139.20.42.0 255.255.255.0 serial1:3
```

PPPoE(ADSL) (router AS PPPoE client)

~~~~~  
~~~

ADSL Step by Step

Step1:VPDN:

(These VPDN commands are not needed with cisco IOS Software  
Release 12.2(13)T or later!!!)

```
VPDN enable
no vpdn logging
vpdn-group pppoe
request-dialin
protocol pppoe
```

you are the pppoeclient who requests to establish  
a session with the aggregation unit (6400 NRP)

Step2:Internal Ethernet network

~~~~~

```
interface FastEthernet0
ip address 192.168.1.1 255.255.255.0
ip nat inside
```

step3:XDSL interface(ADSL)

~~~~~

```
interface ATM0
no ip address
no atm ilmi-keepalive
```

```
dsl operating-mode auto
bundle-enable
hold-queue 224 in
-----all defaults.
```

PPPoE runs on top of AAL5SNAP  
However, the encaps aal5snap command is not used!!!

Step4:ATM Sub-interface  
~~~~~

```
interface ATM0.1 point-to-point
pvc 1/1(VPI/VCI)
pppoe-client dial-pool-number 1
(将这个虚拟的ATM子接口放入POOL, 1中)
```

PVC 1/1 is an example value, that must be changed  
to match the value used by the ISP!!!

深圳:  
VPI/VCI(8/35)  
深圳的是:8/32

Step5:if dial:  
(the pppoe client code ties into a dialer interface upon which a  
virtual-access interface is cloned.)  
~~~~~  
~~~~~

```
interface Dialer 1
ip address negotiated(与ISP协商IP地址)
ip mtu 1492
Ethernet MTU default=1500
(1492+PPPoE headers=1500)

ip nat outside
encapsulation ppp
dialer pool 1(去dialer pool1中, 调用刚放入的ATM的子接口)
```

CHAP OR PAP:  
Chap:-----  
ppp authentication chap callin  
ppp chap hostname <username>  
ppp chap password <password>

PAP:-----

```
ppp authentication pap callin
```

```
ppp pap sent-username <username> password <password>
```

The ISP instructs you about the type of authentication to use.

Step6:NAT:

~~~~~

~

```
access-list 1 permit 192.168.1.* 0.0.0.255
```

```
ip nat inside source list 1 interface dialer1 overload
```

```
ip route 0.0.0.0 0.0.0.0 dialer1
```

```
end
```

对比实验:3层冗余, (RIP)

PPP/MLP(m

通过Multilink Protocol实现2层冗余:

Step1:

将冗余的物理链路上的接口, 原有配置都删除, (Default)

DCE端, 都配置同步时钟.

Step2:4个物理接口都封装PPP, 并且运行Multilink. (无需配置IP地址)

```
int s0/s1
```

```
encapsulation ppp
```

```
ppp multilink
```

Step3:在双方路由器上, 创建“虚拟模板”接口, 配置IP地址, 指定MLP.

R1#

```
interface Virtual-Template 1
```

```
ip address 12.0.0.1 255.255.255.252
```

```
ppp multilink
```

R2#

```
interface Virtual-Template 1
```

```
R2#
interface Virtual-Template 1
ip address 12.0.0.2 255.255.255.252
ppp multilink
```

Step4:在MLP中,调用虚拟模板:

```
R1/2(config)#multilink virtual-template 1
```

Step5:

```
R2#show interfaces virtual-access 3
Internet address is 12.0.0.1/30
 BW 3088Kbit
```

Step6:IGP:(RIPV2)

IPv6

IPv4地址有32Bits,以“点分十进制”标示.

IPv6地址有128Bits,以16进制数表示.

每4位的16进制数,组成一个字段,

IPv6地址共有8个字段.

每个字段之间用冒号分割.

2031:0000:030f:0000:0000:0000:8760:130B

IPv6地址的简写:

对于全零字段“:0000:”,可以简写为“:0:”

对于多个全零字段“:0000:0000:0000:”,可以简写为“::”

“030F”可简写为“30F”

特别注意:

一个IPv6地址,只能出现一次::

2031:0:30F::8760:130B

网络协议:IPv4

寻路协议/路由协议:RIP/IGRP/EIGRP/OSPF (V2) /ISIS/BGP

网络协议:IPv6

寻路协议/路由协议:RIP/OSPF (V3) /ISIS/BGP

LAB1:在IPv4的海洋中, 将IPv6的孤岛实现网络互通.  
(通过Tunnel实现)

~~~~~

Step1:构建IPv4网络 (RIP v2), (R1/R2/R3#)

R1#

```
router rip
version 2
network 12.0.0.0
network 13.0.0.0
no auto-summary
```

Step2:在R3/R5;R2/R4间, 构建IPv6网络:

R4/5(config)#

no ip routing

ipv6 unicast-routing

双栈路由器:

R2/3(config)#

ip routing

ipv6 unicast-routing

在IPv6网络中, 配置IPv6的链路地址, 测试链路:

R3(config)#

R3(config-if)#no shutdown

为了实现IPv6孤岛的互通,  
在双栈(v4/v6)路由器R2/R3上,

在双栈(v4/v6)路由器R2/R3上,  
构建能承载IPv6通信流量的Tunnel:

Step3:在R2/R3, 为Tunnel新增一个环回口,  
注意此

将新建的环回口, 宣告到RIP中

Step4:R2/R3, 都以自己的环回口

```
R3#
interface tunnel 1
tunnel source 3.3.3.3
tunnel destination 2.2.2.2
```

Step5:在R4/R5上, 新增IPv6的环回口, 模拟IPv6的网段, 运行IPv6的路由协议

Step6:

在所有的IPv6网络 中, 启动IPv6的路由协议OSPFV3  
(包括R2/R3上的Tunnel接口)

6-1

```
R2/R3#
interface tunnel 1
ipv6 enable
tunnel mode ipv6ip
```

6-2:

在所有的IPv6路由器上 (R2/3/4/5)  
, 都启动IPv6的路由协议:"OSPF"

```
ipv6 router ospf 160
router-id 100.0.0.X
```

```
router-id 100.0.0.X
```

6-3:在所有需要运行IPv6的OSPF接口中, 激活IPv6的OSPF的运行:

```
R3(config)#
```

```
in s1
```

```
ip v6ospf 160 area 0
```

```
in tunnel 1
```

```
ipv6 ospf 160 area 0 (Tunnel跑OSPF v3, 不跑IPv4的RIP)
```

6-4:

```
R2#show ipv6 ospf interface
```

```
R2#show ipv6 ospf neighbor
```

```
R2#show ipv6
```

Step7:如果IPv6网络运行的路由协议是:RIP for IPv6

从Step1~6-1, 都是相同的.

```
no IPv6 router ospf 160
```

## NP加强(周涛主讲)

IGP选路原则:1. 下一跳可达 2. 最长匹配 3. AD小的(不过协议之间比AD) 4. cost 小的(相同的路由协议比Metric)

无量协议在主类边界会主动汇总

[www.cisco.com/univercd](http://www.cisco.com/univercd)

## OSPF

## OSPF

## OSPF知识点

### 1. 链路状态协议及OSPF的三张表

- Neighbor table: show ip ospf neibor
- Topology table: show ip ospf datebase
- Router table: show ip router ospf

### 2. 网络分层, 各种区域及各种LSA类型

#### OSPF的组播地址

- DR and BDR 在224. 0. 0. 6里
- DRother在224. 0. 0. 5里
- OSPF hello包全部发往224. 0. 0. 5

#### 分层:

**LSA1:** (Router Link) 0 本路由器产生 :产生的是Router-ID 直接网络 , 邻居的地址

**LSA2:** (Network link) DR产生, MA网络中. 只有DRother到DR的链路不包括.

**LSA3:** (Summary Net) ABR产生, 产生的是路由 OSPF域内是链路状态, 域间是距离矢量

**LSA4:** (Summary ASB) ABR产生, 产生的是路由. 描述的是ASBR的Router-ID, 告诉其他区域到本区域的ASBR的信息, 在本ASBR的区域是不产生LSA4信息的

**LSA5:** ASBR产生的 描述的是域外路由 (不属于任何一个区域的)

**LSA7:** ASBR产生 描述的是特殊区域的路由 (toto stub NSSA)

#### 各种区域拥哪些LSA类型:

**Area 0 和普通区** 域拥有12345全数据库, 区别在于: Area0可以传输其他区域的LSA

**Stub:** 末梢区域, 没有了域外路由 (LAS4, LSA5). 只有: LSA123 (有默认路由 LSA3的默认路由)

**total stub:** 没有了域外路由和其他区域的域间路由 (LSA3, LSA4, LSA5), 只有LSA12 (有默认路由 LSA3的默认路由)

**nssa:**

1237 (没有默认路由)

**total nssa:** 127 (有默认路由 LSA3的默认路由)

### 3. OSPF各种包类型

Hello (组播)

DBD: Database Description (单/组)

LSR: Link-State Request (单)

LSU: Link-State Update (?)

LSA: Link-State Acknowledgement (单播)

### 4. OSPF接口的各种网络类型

| 链路类型      | 32 位路由 | Hello 间隔                             | 手工 neighbor |
|-----------|--------|--------------------------------------|-------------|
| lo        | Y      | None                                 | N 可以打,但不看   |
| P2P       | N      | 10                                   | N 不能指定      |
| P2MP      | Y      | 30                                   | N 可以指定      |
| P2MP-Non  | ?      | 30                                   | Y           |
| NBMA      | N      | 10                                   | Y           |
| BROADCAST | N      | 10                                   | N N         |
| 虚         | N      | 10 当建立邻居后就不发 hello 包,但发生变化时会发 LSA 更新 | 单播          |

链路Metric计算: $10^8/BW$ (带宽)

各种接口的默认BW(Kbit): ethernet:1000; serial:1544; loopback: $8 \times 10^7$

修改带宽命令:(接口)ip ospf cost ?; bandwidth ?

## 5. OSPF建邻居的必要条件以及各种过程

- 1) **Area ID**:哪个进程先起与哪个路由器建邻居 ?(一条链路只能启用一条进程. 如果启动两条进程, 先启动的Area ID发Hello, 后启动的不发)
- 2) MTU值(可忽略): ip ospf mtu-ignore (接口) 忽略MTU值 在小的的一方打:因为,
- 3) **hello/dear间隔** hello时间为dear时间的四倍. 修改hello时间, Dear时间会跟着改变
- 4) **验证**
- 5) **末间区域标识**

过程:(不同网络类型建邻居的过程是不一样的)

MA:

Down:

I:我收到对方的hello后, 就把对方设置成此状态

Two:在对方发来的hello里看到自己的Router-ID时 . 选DR-BDR 当选DR-BDR时, 对网络掩码有严格要求

ES:主/从: router-ID MTU

EC:

L:

Full

debug ip ospf adj (查看停留在哪个状态)

关闭FR的广播功能(二层不支持broadcast): 1. 关闭自动ARP 2. 手工指定映射时不携带broadcast

|      | P2P      | P2MP     | Broadcast | NBMA   | P2MP-non |
|------|----------|----------|-----------|--------|----------|
| P2P  |          | Yes/ Yes | Yes/No    | Yes/No | Yes/Yes  |
| P2MP | Yes/ Yes |          | Yes/Yes   | Yes/No | Yes/Yes  |

|           |          |         |         |        |         |
|-----------|----------|---------|---------|--------|---------|
| P2MP      | Yes/ Yes |         | Yes/Yes | Yes/No | Yes/Yes |
| Broadcast | Yes/No   | Yes/No  |         | Yes/No | Yes/No  |
| NBMA      | Yes/No   | Yes/No  | Yes/Yes |        | Yes/No  |
| P2MP-non  | Yes/Yes  | Yes/Yes | Yes/No  | Yes/No |         |

## 6. OSPF的认证

类型：配置：作用  
 链路：接口:1.启用;2.密码 整个区域里面, 不同的链路可以启用不同的认证.  
 区域：接口:1.启用;2.密码 所有运行本区域的OSPF接口都运行认证:Simple password authentication enable  
 虚：接口:1.启用;2.密码 注意:虚链路的认证方式和区域认证是一样的, 例:区域的认证是明文认证, 密码是cisco. 则在虚链路认证时只需打2密码

## 7. OSPF的汇总

域间汇总：ABR上做:(进程):area \* range :只能把LSA1和LSA2路由汇总  
 域内汇总：ASBR上做:(进程)summary-add :只对路由器产生的LSA5路由起作用

## 8. OSPF的重分布

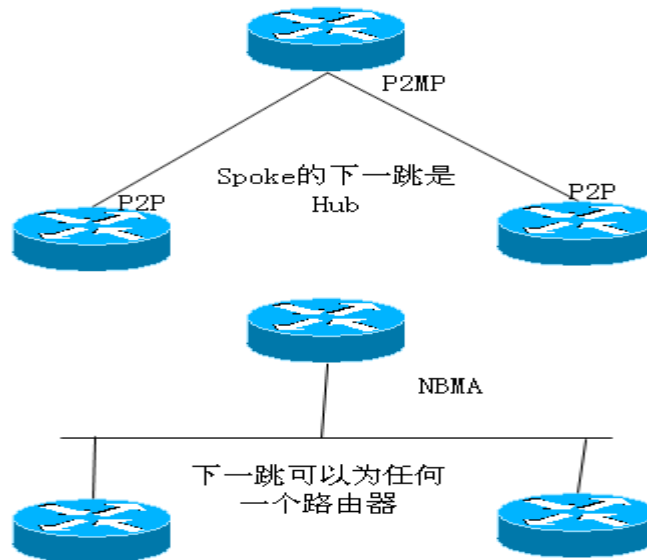
## 9. OSPF虚链路

作用:1. 把没有和Area0相直连的区域拉到Area0  
 2. 用与Area0的备份  
 3. Area0分割

### 难点:

- 1) 两台路由器是否形成邻接关系以后就能学到各自的路由? (不一定)
- 2) DR BDR 如何看LSA更新地址? :DRother ,BDR到DR发.6 ; 其他的互相发.5
- 3) 一个区域, 有两个ABR会如何选择要去往哪个ABR? (走Metric小的;Metric一样时, 负载均衡)
- 4) LSA与Hello发包类型是否一至(组播还是广播)? (不一定)
- 5) 为什么ip mtu 最小为68 而mtu 最小 64
- 6) 不同网络类型为什么不能学到路由: 当邻居对同一条链路描述的网络类型不一样时, 它不能计算, 并不能写入路由
- 7) Loopback口会不会发LSA? 只发hello包, LSA包未知(好像是不发)
- 8) 两台路由器同一子网但掩码不一样, 是否可以建立起邻居?(可以)
- 9) 两个ABR产生默认路由给R3, 会不会负载?
- 10) stub 两条

- 10) stub 两条
- 11) 在Hub&Spoke NBMA网络中 Spoke中,priority 0 (neighbor在show run中不出现)指定nei, 出现attemptp状态 ;priority > 1 neighbor才会在show run 中出现
- 12) 在Hub&Spoke NBMA网络中, neighbor无法删除,要把NBMA接口地址改为同网段(要删除的Neighbor地址的同网段),才能删除
- 13) P2MP 相当与多个点对点 ; NBMA 相当于以太网模型,但不能主动发hello包



- 12) 帧中继网络中P2MP优秀之处:1. 有32位主机路由(易于排错);2. 下一跳是Hub端
- 13) FR中主接口和多点子接口默认网络类型是NBMA;主接口和多点子接口,必须关闭发向ARP;P2P不能关闭inver arp
- 14) 当ASBR和同路由器不在同一个区域的时候必须有LSA4才能学到路由. LSA4是由ASBR所在区域的ABR产生的,LSA4经过ABR时会更改宣告路由器
- 15) 接口的第二地址和第一地址必须宣告在同个区域里面,才可以传出去
- 16) 重分布直连强烈建议挂Router-map,把需要的直接接口重分布进来.
- 17) 如果本路由器看到一条路由同时出现在LSA5和路由表中但没有出现在LSA4中,则这个路由为ASBR并且在本区域里
- 18) 当database看到LSA5,没LSA4,而且路由表没有,证明本区域内无ABR是和公告路由的ASBR在同一区域.
- 19) 在进程下:service password-encryption (把密码加密:show 的时候看不出密码是什么)
- 20) 两个区域,但没有Area0区域,只用做条VL,那这两个区域就能学到对方的路由.
- 21) 只有OSPF重分布到其他协议时,可以配  
match:external/internal/nssa-external
- 22) OSPF中,如果同时输入两个进程,则先输入的进程生效.
- 23) 重分布是基于路由表,和启用本协议的接口.

- 22) OSPF中, 如果同时输入两个进程, 则先输入的进程生效.
- 23) 重分布是基于路由表, 和启用本协议的接口.
- 24) RIP的在(接口的)手工汇总不能产生一条在路由表里不会产生一条标记为R的路由. EIGRP和OSPF可以
- 25) 自动汇总只能汇总A, B, C类的主类网络
- 26) ?Area0被分割, 一边的Area0如果没有直连邻居, 就学能学到对方Area0的路由??
- 27) 一个路由器最大运行31个OSPF进程
- 28) 点分十进制: 0. 0. 1. 0=256 255. 255. 255. 255=2<sup>32</sup>
- 29) tota stub 和stub的默认路由, 它是走stub  
tota nssa 和nssa的默认路由, 它是走tot nssa

## 命令解析

Ip ospf mtu-ignore:

--忽略MTU值 :停在extend所 (在交换机上修改:)

Debug ip ospf adj:

--查看邻居停留在哪个状态

System mtu:

--修改交换机以太口的的MTU(保存重起)

service password-encryption :

--把密码加密:show 的时候看不出密码是什么

Redistribute:

-- 其他协议重分布到OSPF 可以设置tag/metric

-- OSPF重分布到其他协议 重分布OSPF时可以Match到

Internal/External

Summary-address:

-- 只对本路由器产生的LSA起作用

Range:

-- 只对LSA1/LSA2起作用

Ip ospf database-filter all out [cost]:

-- 在接口中过滤所有LSA, 可以匹配过滤特定Metric的LSA

Neighbor x. x. x. x(接口地址) database-filter all out:

-- 过滤从邻居过来的所有LSA, 不过必须把接口类型改为P2MP; 在P2P里也能打进去, 虽然会提示出错, 但会起作用, 不过在running-config里

看不到.

Area x nssa default-information-or no-summary:

-- Total NSSA区域的ABR上产生一条LSA7的默认路由.

Ignore lsa mospf

-- 过滤LSA6(可能Cisoc路由器不支持)

Distance x x.x.x.x:

-- 对某个路由器公告的路由改AD(x.x.x.x是R-ID).

Fliter-list:

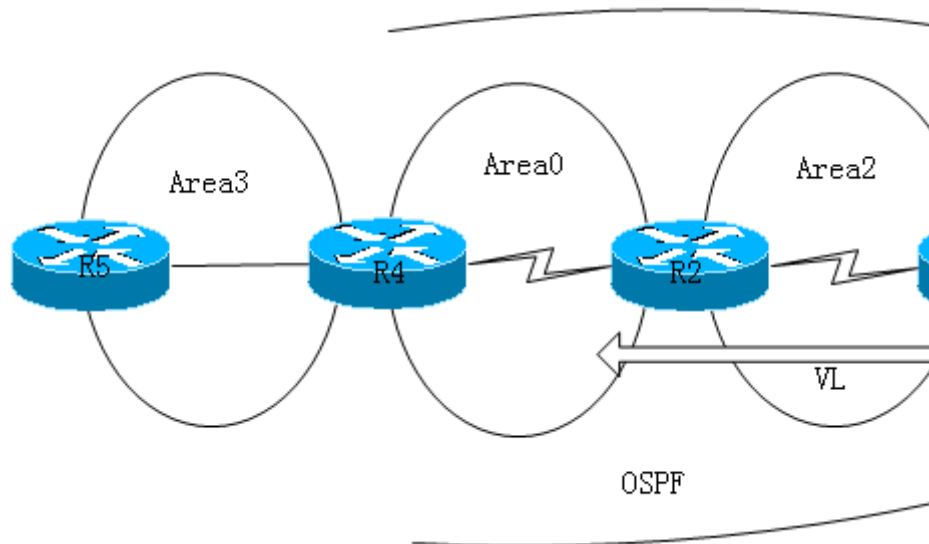
-- 过滤LSA(能过滤几号LSA?实验证明)

Area x default-cost:

-- 修改OSPF产生的默认路由的Metric值.

#### LAB:

1. NBMA上做不同Area ID邻居的建立:同一条链路宣告两个
- 2.



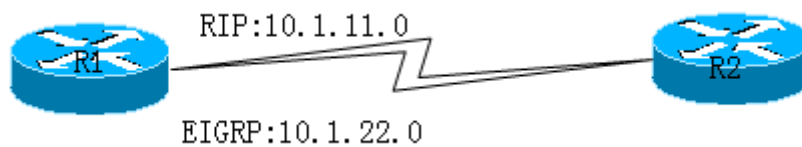
3.



1. 宣告第一地址能否建立邻居和学到路由

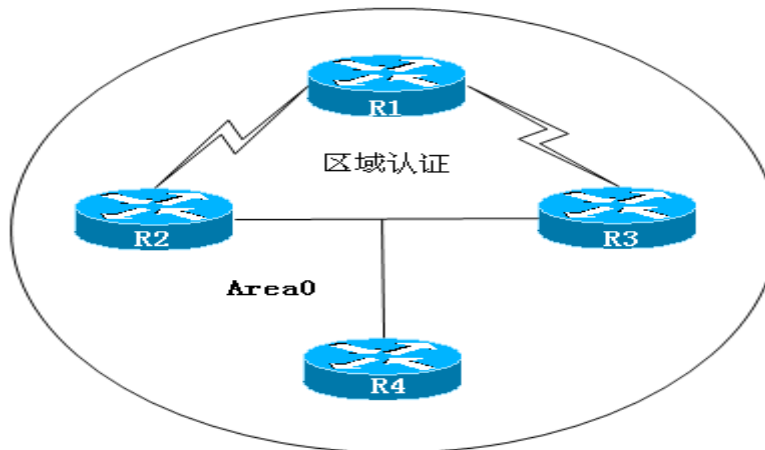
3. 宣告R1的第二地址和R2的第一地址能否建立邻居和学到路由

4.



5

LAB6:



**实验要求:**Area0运行区域认证, R1与R2, R1与R3配置接口密码, R2, R3, R4的E0口上不配置密码

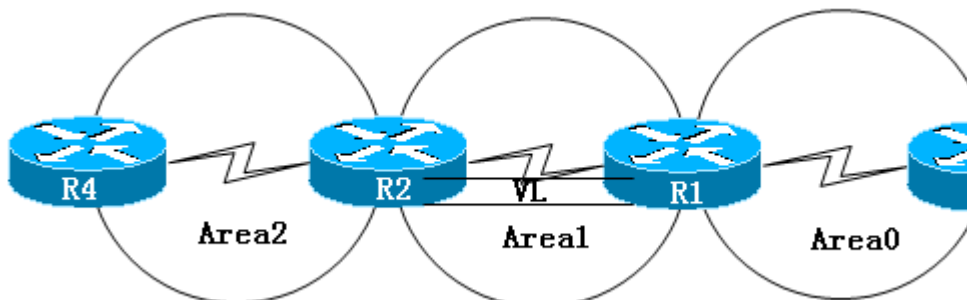
**实验过程中遇到的问题:**

把R4的E0口的网络类型改成了Point-to-Point, 这时, R4上出现了不稳定情况:不停的提示:与R2建立邻居, 与R3建立邻居

**原因:**Point-to-Point, 只能和一个路由器建立邻居, 所有出现这种不稳定情况

**实验结果:**R4能学到R1, R2, R3的全路由

LAB7:



**需求:**虚链路一边(area0)有Key, 另一边没打Key

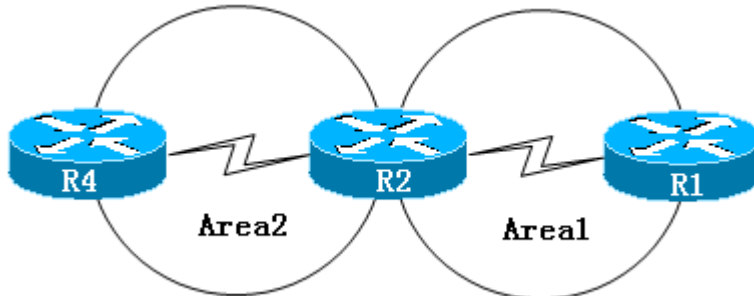
**实验发现:**VL没有建立成功, 在R2上出现了Area0区域(把R2看成了Area0).

Area0). 原因virtual-links相当于一个Area0区域

此时, Area2中只有R2在OSPF中network给R3的路由

**实验结果:**在Area0做了区域认证, 其他区域没有做认证, VL无法建立, 但如果已经建立了成功VL, 后来才启动认证, 则, 这条VL不会断, 只有当网络发生改变的时候才会断开, 原因: VL链路如果没有更新包通过, 只会发一次Hello包. 但当有更新包时就会重新发Hello包, VL断开

LAB8:



**需求:**没有Area0如何能让Area2与Area1正常学到各自的路由

**实验结果:**方法一: 在ABR (R2) 上宣告建立一条虚链路, 并把这条虚链路宣告到Area0中.

方法二: 在任意一个区域的两台路由器上建立virtual-links. 原理: virtual-links相当于一个Area0区域

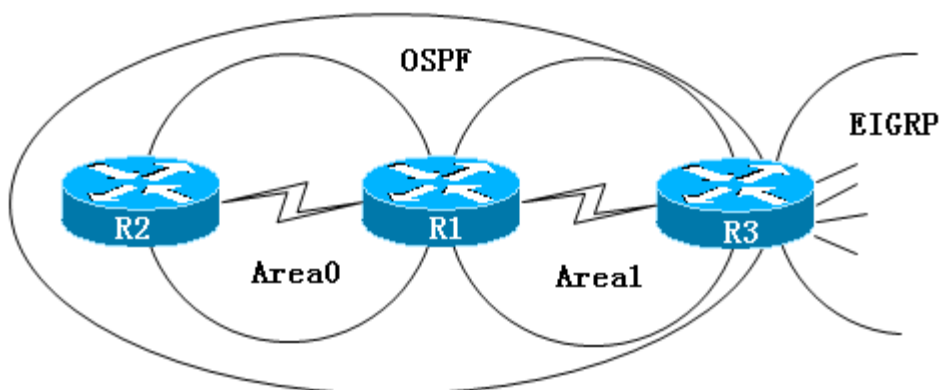
8:



R5上没有做汇总能不能在R3上做汇总

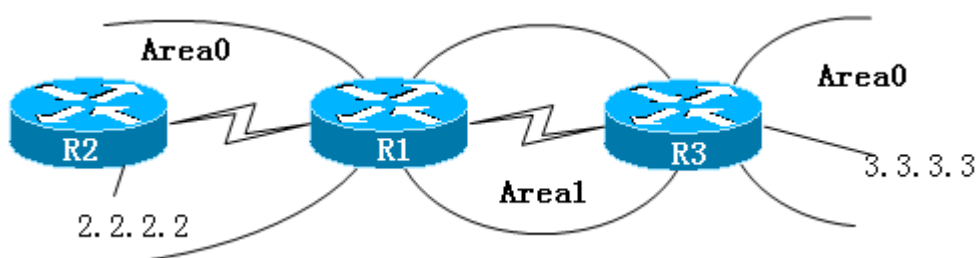
LAB12: Area2不能学到域间路由

9:



R3没有做EiGRP的汇总, 在R2上可不可以做汇总?  
在Area1上做NSSA

9.1



实验过程: 在R3上看到了1.1.1.1和2.2.2.2的路由, 因为: Area0被分割, 一边的Area0如果没有直连邻居, 就学不到对方Area0的路由??

实验需求, Area0路由正常

1. VL

2. 双向重分布

3. tunnel

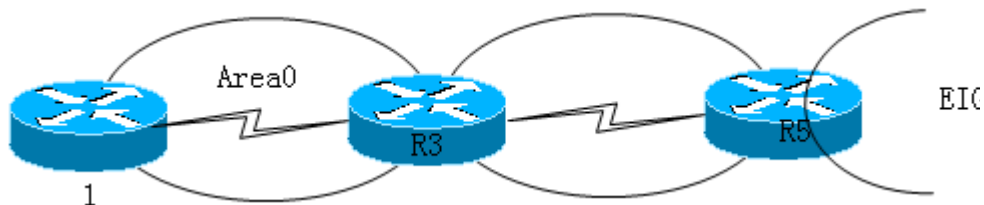
ip add (可以配普通的IP地址(需要宣告))(可以借地址: 借Area0(要相连区域)的地址(不需要宣告))

tun sou

tun des

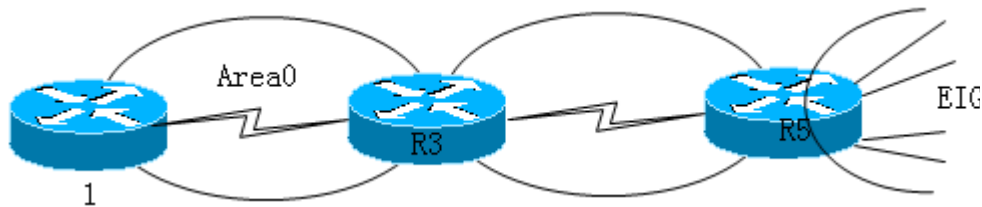
在进程把tunnel的地址宣告到Area0

10:



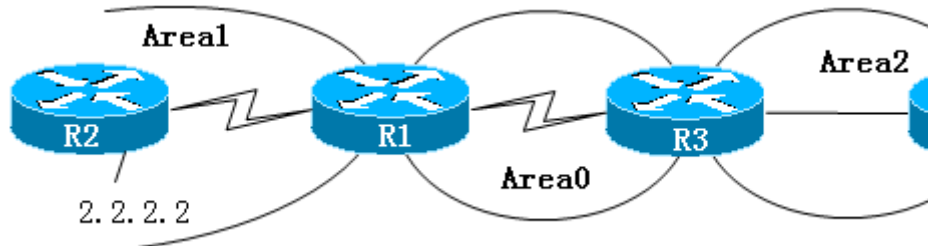
让Area0学不到域外路由.

11:



eigrp上有四条路由, 如何能让R3只收到一条eigrp的汇总路由  
13:RIP上有四条路由, 如何能让R3只收到一条eigrp的汇总路由

14:LSA过滤:



需求: 在R2看不到Area0和Area2的LSA3和LSA5

方法1:distribute-list (进程:in+入来接口) (允许)

LSA3和LSA5是路由, 所以能过滤, 但把本路由器(R1)的也过滤了

方法2:are rang .....no-adv.....

无效:因为are rang只能对LSA1, LSA2起作用.

方法3:if#ip ospf database-filter all out (过滤全部LSA)

方法4:进程:[nei \*.\*.\*.\* (对方接口地址, only allow P2MP (P2P也会起作用, 但在show run 中不显示出来) (建议用在HUB端)) database-filter all out cost 20]

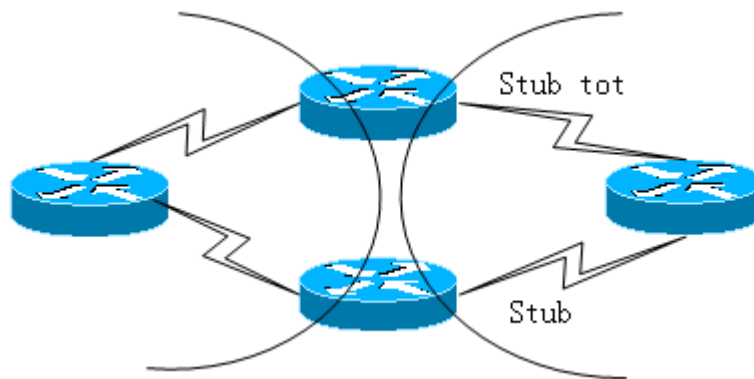
cost 20的含有:从别的邻居学过来的链路cost值改为20

方法5:NSSA区域, 把Area2改为NSSA, 在R3上用summary-add 过滤, 原因:在R3上, NSSA的LSA7=>LSA5, 相当于R3产生的, 而Summary-add 对自己产生的有效

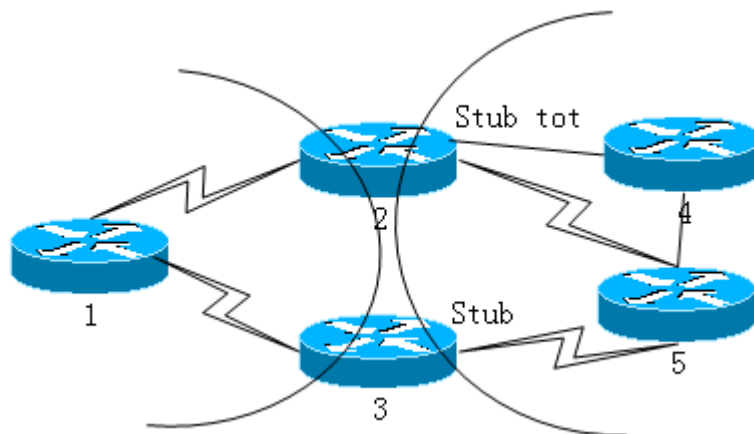
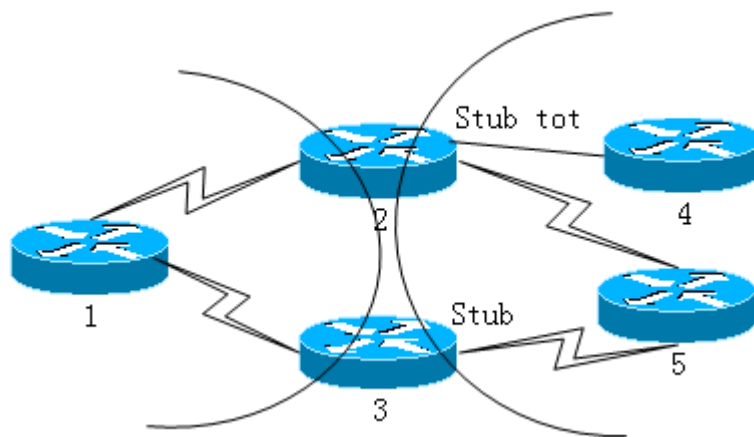
实验

LSA1 基本的OSPF配置 (Router-id)

LSA3 末梢区域 (两个ABR时, COST对默认路由的影响)



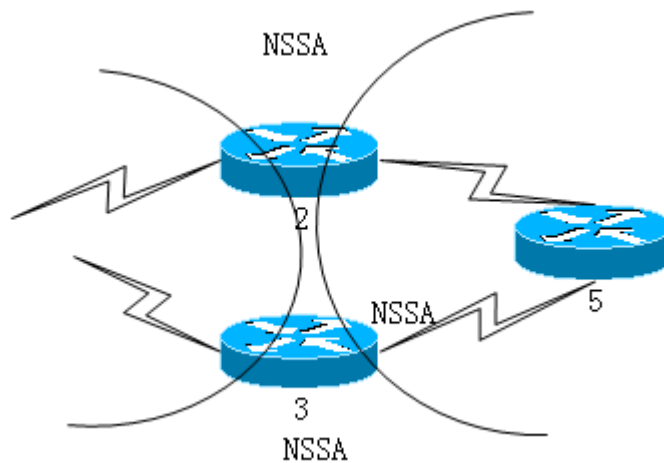
路由器会走Stub:原因:Stub有域间的明细路由



R4的路由这时会怎么走?(最长匹配)

LSA4 完全末梢

LSA5 NSSA区域(两个ABR时, COST对默认路由的影响)



NSSA区域的ABR既是ASBR时需要注意的地方, 默认路由的类型.

1个下一跳还

total stub 和stub的默认路由, 它是走stub

total nssa 和nssa的默认路由, 它是走tot nssa

NSSA区域的ABR既是ASBR是需要注意的地方, 默认路由的类型.

建议输入:Area x nssa no-redist

LAB 6 地址汇总 (ABR和ASBR的汇总, EIGRP向OSPF重分布)

LAB7 区域认证和链路认证

LAB9 虚链路的几种作用(认证, 一边起一边不起)

LAB10 NBMA网络中的OSPF

Broadcast网络中spoke中有多少个nei

LAB11 OSPF过滤

LAB12 TAG值METRIC-TYPE

OSPF的汇总, ABR汇总ASBR的路由

区域零分割的3种解决方法

OSPF的COST的修改方法(接口地址和进程下参考带宽的修改)

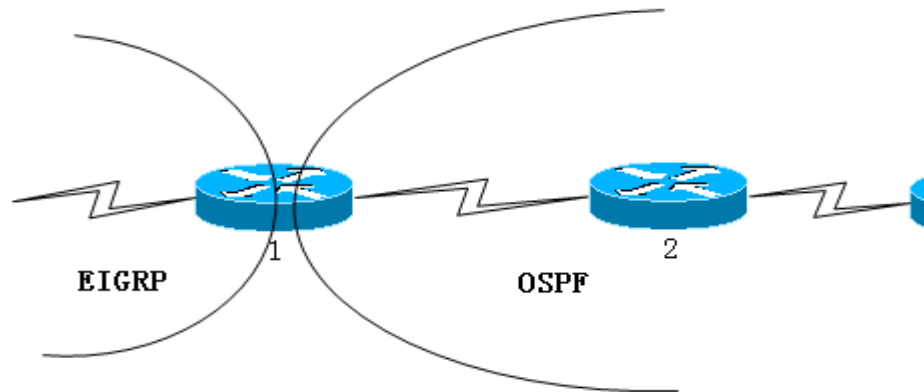
mtu值对OSPF建邻居的影响(影响入方向):  
如果是G网络, 那么, 在进程下打: no autocost

#### 4号lsa对路由的影响

deny ospf和deny lsa (deny所有lsa和在ABR上deny 3号lsa): auto-cost ref.... (过滤LSA6)

#### LSA13 NBMA网络和P2M的比较

#### LSA14 distr-list的工作机制



in 的方向在接口 (in方向只是阻止了Database表进入路由表, 但不影响database的传递), out 方向不能加接口, out方向在asbr上可用  
自己产生的LSA如果在路由表中没有的, 那么这条路由不会传给其他人,  
但不是自己产生的就会传给其他人.

#### OSPF网络类型与相互建邻居 (OSPF建立邻居的必要条件)

distance解决对双电双向重分布的危险  
distance ospf external 121

7转5抑制, 3号LSA抑制以及FILTER-LIST用法  
能过滤3号? 5号? 1号? 对in或out方向起不起作用

#### ospf默认路由类型及其METRIC

向这个区域公告路由默认是多少: area 1 filter-list prefix 1 in  
area 1 default -cost ?

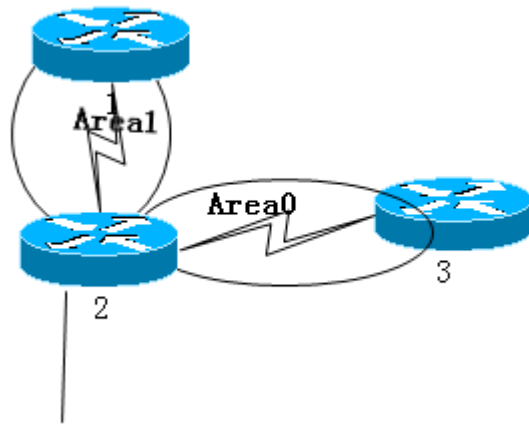
5号类型需要有4号类型才能放进路由表?

passive接口与nei

passive接口与nei

nei看不到

不宣告能学到对方的24位路由



LAB12 多访问网络下DR的选举

路由控制

路由控制:

192.172.12.0/24

1.先一刀切 $2^N$ 移到0看是否相同

|    |          |
|----|----------|
| 4  | 00000100 |
| 5  | 00000101 |
| 6  | 00000110 |
| 7  | 00000111 |
| 12 | 00001100 |

```
13 00001101
14 00001110
15 00001111

```

网络位: 00000100 (相同为几就为几,不同为0)  
掩码位: 00001011 (相同为0,不同为1)

连续子网的切割:

1. 条目要是2的N次方 2. 最小的除与条目数是2的N次方. 在把原理的掩码-N  
即得出匹配的掩码

例: 192.172.16.0/24

匹配199.172.X.0/24

**Access-list:**

access 1 per 199.172.0.0 0.0.255.0

标准的访问控制列表只能匹配数值

**Prefix-list**

ip pre 1 per 199.172.0.0/16 ge 24 le 24 (ge 24 le 24: 大于24小于24, 则24-16=8 既  $2^8=256$  则可用放入256条: 既0~255)

**扩展Access-list**

acc 101 per ip 199.172.0.0 0.0.255.0 (匹配数值) 255.255.255.0  
0.0.0.0 (匹配掩码(掩码通配符))

例: acc 101 per ip 199.172.16.0 0.0.0.3.0 255.255.255.0  
0.0.0.0

acc 101 per ip 199.172.16.0 0.0.0.3.0 255.255.255.128  
0.0.0.0

-----  
acc 101 per ip 199.172.16.0 0.0.0.3.0 255.255.255.0  
0.0.0.128

匹配: 199.172.16.0/24

199.172.16.0/24

199.172.16.0/24

只DENY DEFAULT

acce 1 deny 0.0.0.0

acce 1 per any

acce 100 deny ip 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

acce 100 per ip any any

ip pre xx per 0.0.0.0/0 ge 1 le 32

ip route 0.0.0.0 0.0.0.0 NU 0

所以的路由

ip per \*\* per 0.0.0.0/0 le 32

## 路由重分发

路由重分布

redistribute 协议

ospf注意加sub

eigrp注意加Metric参数 1000(带宽) 100(延迟) 255(可信度) 1(负载)

1500(MTU最大传输单元) (默认)

或者在路由协议中输入default-metric 输入

RIP注意加metric小于15 建议写1

或者在进程中输入default-metric

在重分发ospf进程其他协议的时候 后面多了个match的选项可用匹配ospf特定的路由

router rip

re os 1 me 1 match internal/external/nssa-external

OSPF宣告到EIGRP中

redistribute ospf 110 metric 1000 100 255 1 2500

## PER(策略路由)

policy based route

网络管理者来控制包的转发

#ip local policy route-map XXX

全局模式下输入这条命令, 匹配发自本路由器的包, 或者穿过本路由器的包

全局模式下输入这条命令, 匹配发自本路由器的包, 或者穿过本路由器的包

```
if#ip policy route-map XXX (不能匹配自己产生的)
```

**进程转发** :从转发层面入到控制层面(查路由表), 在从转发层出去

配置:接口:no ip route-cache

**快速转发** (低端cisco默认开启):一个小cach, 直接在转发层走, buffer小, 有时间限制:基于流:五元素:源;目的地址;原, 目的地址;端口号

配置:

**CEF**: (高度cisco默认开启) 基于拓扑, 不会消失

配置:全局:ip cef

接口:ip route-cache cef

察看:show ip int cef s0 de

**路由器转发机制**:首先走**CEF**, 如果没有走**route-cache**, 还没有走**路由表**, 再没有就**丢弃**

实验:cef和Policy Based Route 的优先级谁高?(Set interface/set ip next-hop)

1. 开启CEF ip local plicy au

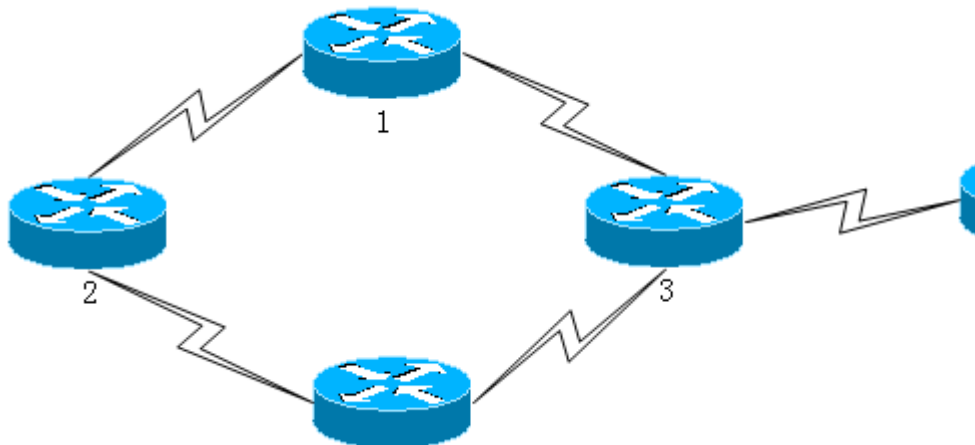
2. 开启CEF

```
router-map match ip add
set int s0/set ip next-hop XXXX
```

```
route-map
```

```
.....
```

```
...default int /set default
```



ip policy route0map wo(本地发起的包是不匹配的)

## Distribute-list

Distribute-list

在IGP协议中, 只能跟标准访问控制列表

distribute-list 列表 in/out 接口  
distribute-list 列表 out(外部) 协议  
distribute-list route-map WORD in (OSPF)

要跟access-list匹配

## router-map

route-map 一般可用挂在哪, 可用匹配哪些东西, 可用配置哪些东西

Match ip:

同一行:逻辑或

不同行:逻辑与

比如重分布后面

Router-map格式:

router-map WORD per 10 (序列号)

match XXX

set XXX

一般设置过以后, 最后还要写个空的(对一些特定的路由或包做过设置后, 还得让其他的路由或包走)

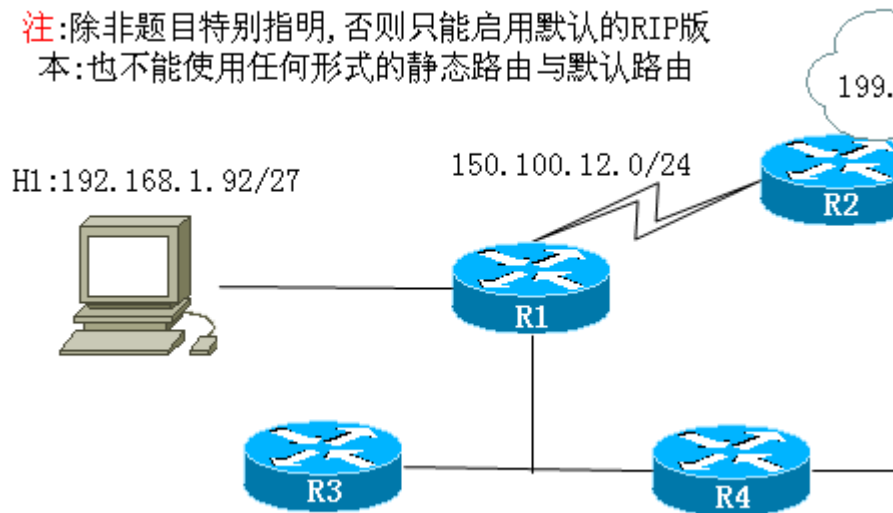
## IGP小结测试



WinRAR 压缩文件

## RIP

注:除非题目特别指明,否则只能启用默认的RIP版本:  
本:也不能使用任何形式的静态路由与默认路由



- 1) R3不能向R4通告路由更新;
- 2) R1是连接到internet的路由器,在R1上面通过RIP向内网注入一条缺省路由;(有关上Internet的配置不再RIP考试范围,可以不进行配置)(10分)???

default int oran...

- 1) 修改R4的RIP计时器时间Update Interval为5S;Invalid为10S;Holddown为20S;但要求路由在没有收到Update后的15s就被删除(20)
- 2) 在删除R1上发送出来的缺省路由后;R1与R4之间分别连接主机H1和H2(用环回口代替);发现不能互相访问对方,请解决(10)
- 3) R1的S0口上级联一个RIP邻居,要求仅在R1的S0口上面发送和接收RIPv2的更新,并且要求使用MD5加密验证;密码为"wolf";配置成功的话,你一个从R2上面收到一些路由(199.172.X.0 X为任意数)(10分);R1向R2发送10网段的明细路由;要求R1/R3/R4能ping通这些路由的IP例:199.172.1.254)(5分)
- 4) 你不能用任何手段登录到R2上查询和修改配置,但要求在R1上配置--能且只能允许R2成功直连TELNET到R1的"enable"模式(即无须输入密码和键入enable命令),R2将用2.2.2.2这个地址发出telnet (10分);其他路由器访问将被拒绝 (10分);  
access 101 per tcp host 2.2.2.2 host 150.100.12.1 eq tel(??)

- 1) R3收到R1发送过来的199.172.X.0/24网段路由,要求基数路由在路由表里面metric为10 (5分)

抓路由:access 10 per 199.172.1.0 0.0.254.255

RIP:offset-list 10 8 (增大8跳)

199.172.

199.172.  
off  
???

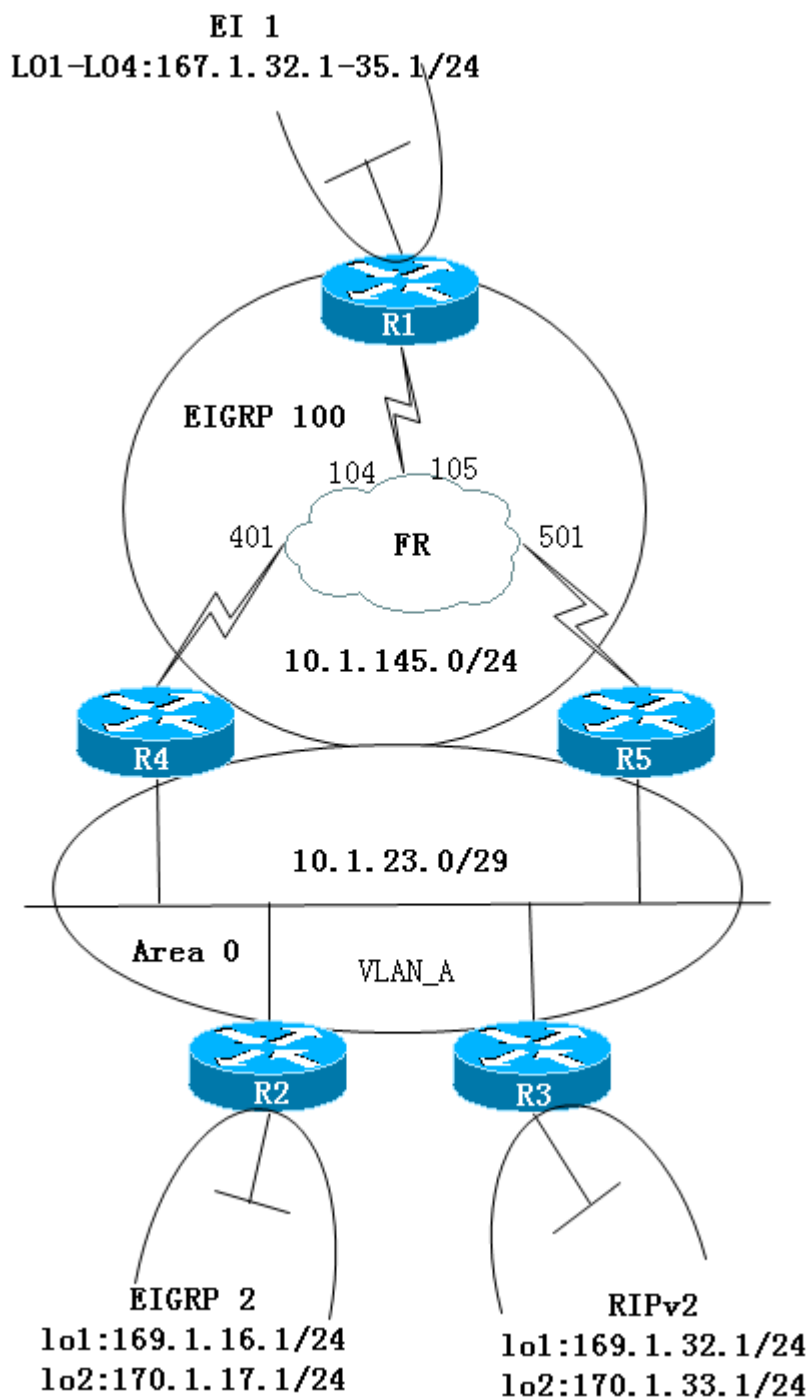
R2初始配置

```
key chain wolf
 key 1
 key-string wolf
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
interface Loopback1
 ip address 199.172.2.254 255.255.255.0
interface Loopback2
 ip address 199.172.3.254 255.255.255.0
interface Loopback3
 ip address 199.172.4.254 255.255.255.0
interface Loopback4
 ip address 199.172.1.254 255.255.255.0
interface Serial0
 ip address 150.100.12.2 255.255.255.0
 ip rip authentication mode md5
 ip rip authentication key-chain wolf
 clockrate 64000
 no shutdown
router rip
 version 2
 network 2.0.0.0
 network 150.100.0.0
 network 199.172.1.0
 network 199.172.2.0
 network 199.172.3.0
 network 199.172.4.0
```

笔记:

R1上关水平分割

## EIGRP



一. 桥接

帧中继中R1的s0接口只用于子接口 (仅使用图中所提供的DLCI)

二. RIP

R3的lo1, lo2运行RIP v2, RIP和OSPF做双向重分布.

三. eigrp

eigrp 1

R1上起4个环回口

lo1:167. 1. 32. 1/24

lo2:167. 1. 33. 1/24

lo3:167. 1. 34. 1/24

lo4:167. 1. 35. 1/24

宣告在eigrp1中

eigrp100

R1, R4, R5在eigrp 100中, R1, R4, R5的环回口以及VLAN\_A为Eigrp 100域内路由

R4上看R1/R5的环回口

D 10. 1. 1. 1/32 [90/256000] via 10. 1. 145. 1, serial0

D 10. 1. 5. 5/32 [90/256000] via 10. 1. 145. 5, serial0

R5上看到R1/R4的环回口

D 10. 1. 4. 4/32 [90/256000] via 10. 1. 145. 4, serial0

D 10. 1. 1. 1/32 [90/256000] via 10. 1. 145. 1, serial0

R1上看到R4/R5的环回口

D 10. 1. 5. 5/32 [80/1657856] via 10. 1. 145. 5, serial0. 1

D 10. 1. 1. 1/32 [80/1657856] via 10. 1. 145. 4, serial0. 1

R4, R5收到167. 1. 32. 0/22的一条汇总路由. eigrp 1 向lo1只发送一条10. 1. 0. 0/16的路由, 向其他接口公布明细的路由信息.

eigrp 2

R2的lo1, lo2口在eigrp2中, eigrp2与ospf双向重分布.

#### 四. OSPF

R2/R3/R4/R5运行OSPF, R3/R4/R5之间都只能形成two-way状态(含义:R2为DR), 当VLAN\_A中再加入一台新设备时, R2还是DR.

R2, R3只能看到167网段和10. 1. 145. 0网段负载均衡(R4和R5的重分布R1的环回口路由是Metric不一样, 也就上让R5重分布时设大Metric), R2访问R1的合胃口正常情况下走R4, 当R2/R4间链路断了后走R5. R3访问R1的环回口走R5, 当R3/R5之间断了后, 就丢包

#### 五. 过滤

eigrp1和eigrp2不接受RIP的路由, RIP也不接受他们的路由

注:要求全网全通, 所有loopback0接口的地址为10. 1. x. x/32(x为路由器号), 本试验主网段为10. 1. 0. 0/16. 不允许出现静态路由.

笔记:

为什么选择mu而不用p-t-p(不稳定)

主接口和多点子接口都需要同时关闭反向ARP

配帧中继映射一般配置运行广播

帧中继中: frame intf-type dce(二层)与clo r 64000(三层)并不是一个概念.

水平分割只需在子接口上关闭

EIGPR必须双向指neighbor, RIP和OSPF只需单向指neighbor

K值是什么东西?

修改eigrp的度量值: 进程下: distance eigrp 80 (D内部度量值) 170 (DEX外部的度量值)

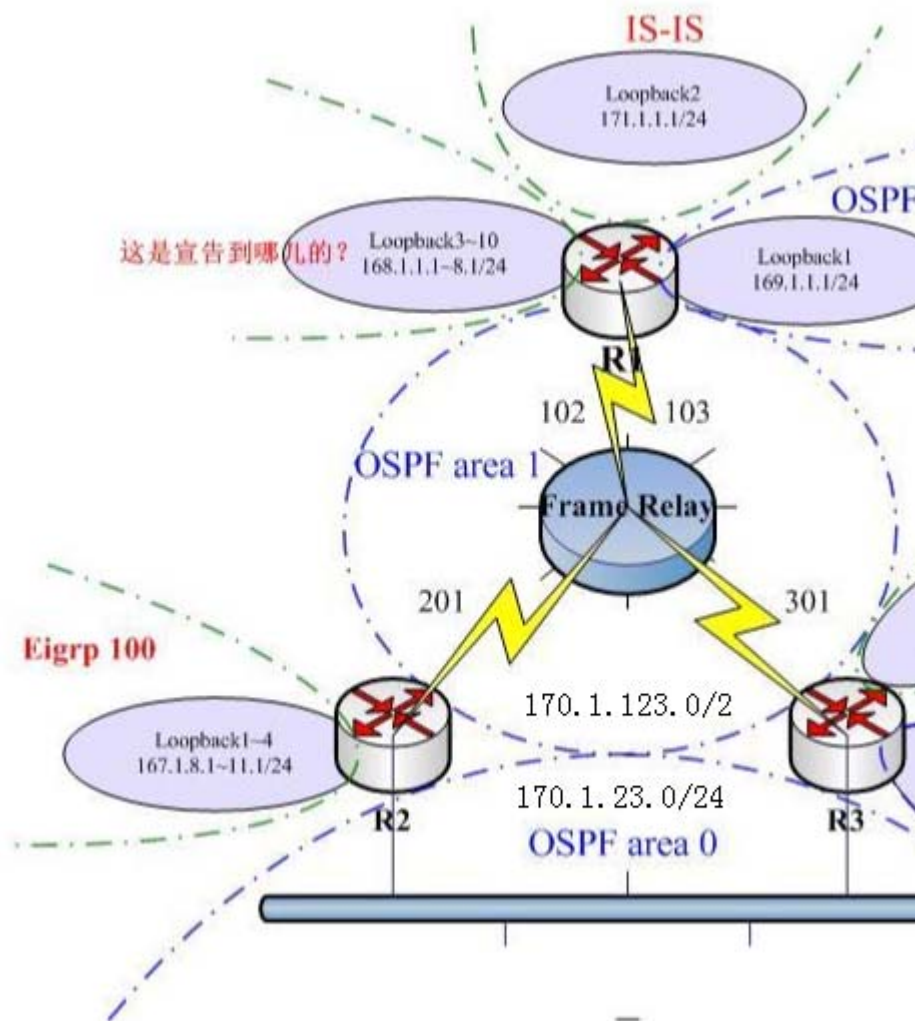
第五个问:: (route-south????????)

实验:

RIP中HUB&SPOKE的SPOKE能否学到对方spoke的路由

$(10^7 / \text{沿路最小带宽} + \text{延迟之和}) * 256$

## OSPF



### 一、 桥接

帧中继中R1的S0接口只用于子接口，要求PING通所有接口(还要ping通自己的接口)（仅使用图中所提供的DLCI）

### 二、 OSPF

- 1、 R1、 R2、 R3的S0接口在OSPF的AREA 1 中， R2、 R3的E0接口在OSPF的AREA 0 中， R1的LOOPBACK 1在ospf的area 2中， R3的loopback1 在OSPF的area3中。
- 2、 OSPF的帧中继中不允许使用NBMA和广播模式。
- 3、 Area2只接收OSPF的域内和域间路由 stub
- 4、 R3在日后会从area3中接收到一些LSA7类型的路由以及area3中会有一条默认路由(area 3 nssa default....)。 total nssa
- 5、 Area0 使用明文验证， area1使用更安全的认证方式， 验证密码为cisco。（做虚链路把没有和Area0直接相连的Area相连接到Area0）

- 6、所有loopback0接口均在ospf域内。
- 7、R1的loopback3到loopback10接口不允许直接宣告进OSPF域内。（只有直接宣告到ospf才需要打ip os net p-p）

## 二、EIGRP

- 1、R2的loopback1到loopback4在EIGRP 100中。
- 2、EIGRP和ospf在R2上做双向重分发，EIGRP只向OSPF只发送一条路由（不允许是167.1.0.0/16）

## 三、RIP

- 1、R3的loopback2到loopback5在RIP域中（R3上要大no-re....???)
- 2、RIP和ospf在R3上做双向重分发。
- 3、在R1和R2上只能看到RIP域过来的一条汇总路由（不能是166.1.0.0/16）。

## 四、

- 1、R1的loopback2接口不允许宣告在任何路由协议中。

## 五、过滤

- 1、在R2上可看见这样的一些路由168.1.x.0（X为奇数）
  - 2、在R3上可看见这样的一些路由168.1.Y.0（Y为偶数）
- 注：要求全网全通，不允许出现任何主机路由，除P2M外，所有loopback0接口的地址为170.1.x.x（x为路由器号），本试验主网段为170.1.0.0/16。

## BGP (25)

1. R1,R2在一个自制系统内,R1与R2是对等体, R1与R3是对等体,R1上的loopbank21到loopback24接口在BGP域内, IP地址分别为199.168.128.2/24,  
199.168.129.2/24, 199.168.130.2/24, 199.168.131.2/24.  
R2上BGP表为 (15)  
S>199.168.128.0  
S>199.168.129.0  
\*>199.168.130.0  
\*>199.168.131.0  
\*>199.168.128.0/22
2. R3的loopback21接口在BGP域内,地址为197.1.1.1/24,并且此地址只在本自制系统内出现,并咬去全部AS内部都能看到这个路由的属性;(5)
3. R1只向R3发送1条路由.(5)  
\*>199.168.128.0/22

~~~~~  
~~~~~

## 笔记:

帧中继中的DLCI号是本地本接口有效,一个接口不能出现相同的dlci  
使用p-to-M前提:支持广播  
为什么EIGRP可用在接口和进程上作汇总,而RIP不能  
ODR:按需路由(全是cisco设备,Hub&Spoke结果)

为什么EIGRP可用在接口和进程上作汇总, 而RIP不能

ODR:按需路由 (全是cisco设备, Hub&Spoke结果)

管理距离160, 基于CDP

#cdp run

if#cdp enable

router odr ()

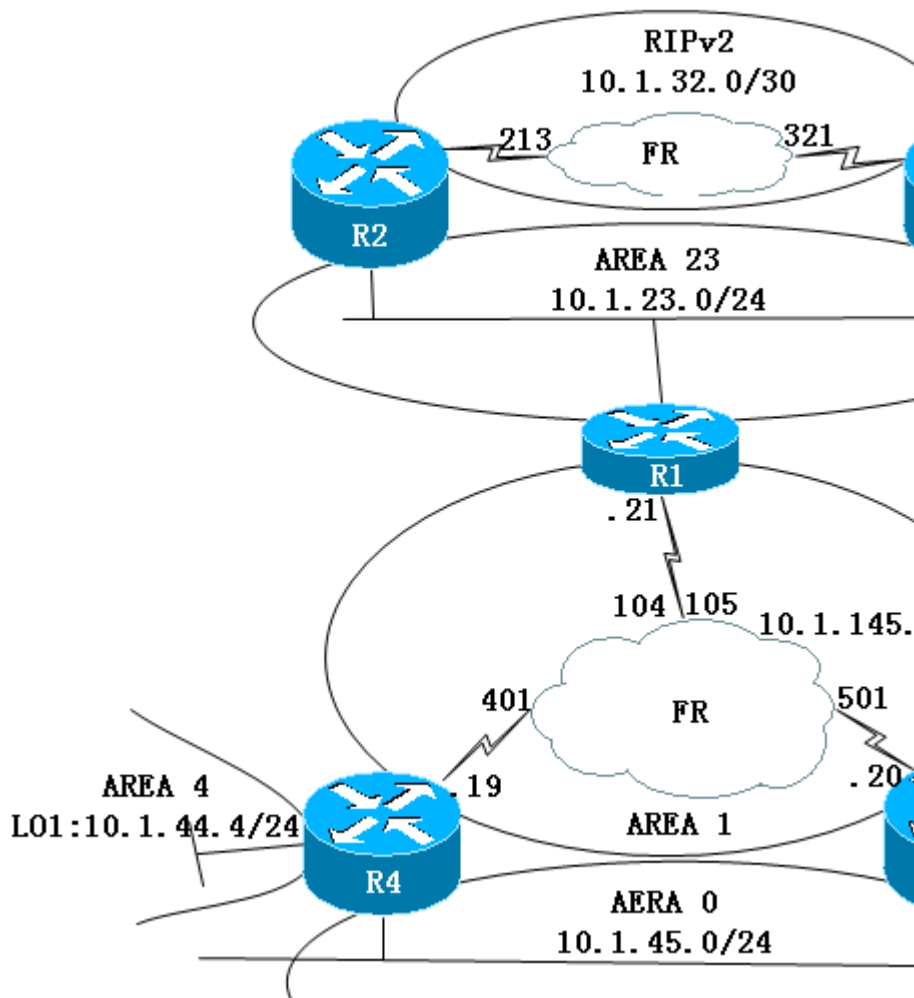
在Spoke端使用了任何协议, 那么ODR将完全实效

TEST:Spoke端使用静态路由, 会不会对ODR有影响

ODR重分布到OSPF时, 除了HUB与SPOKE直连的接口外其他接口都可用直

接重分布到OSPF中(要把这些直连接口做重分布)

## IGP小结实验4



### 一、桥接

帧中继中R1的S0接口和R2的S0接口用子接口（仅使用图中所提供的DLCI）

### 二、EIGRP

R5的lo2-lo7运行在eigrp 100中

eigrp->ospf 要求路由表如下：

```
OE2 169.1.16.0 [110/0]
OE2 169.1.17.0 [110/150]
OE2 169.1.18.0 [110/0]
OE2 170.1.32.0 [110/150]
OE2 170.1.33.1 [110/150]
OE2 171.1.12.0 [110/150]
```

```
red ei 100 swb 100 E 150
route E per 10
match ip add 1
set metrip 0
```

### 三、RIP

R2、R3运行RIP v2，R2、R3的lo0口以及R3的lo1口是RIP域内路由。R3的lo1口地址为192.168.1.3/24

R2、R3间用很安全的认证，密码为CISCO。R2、R3间在路由稳定时不发送路由更新(触发更新)。

正常情况下R2访问RIP域外的网络时走R3，当R3的E0口DOWN掉后，走R1。(把RIP的AD改小，重分布直连或用tag，重分布直连也需把metric type改掉)

R3访问RIP域外的网络时走R1，当自己的E0口当掉后，走R2。(打tag)要求R2、R3间链路全网可见。且每一跳，cost值会改变。(该为OE1)

#### 四、OSPF

OSPF的帧中继中不允许使用neighbor命令。R1、R4、R5的环回口运行在OSPF域内；R4、R5的E0口运行在area 0内；R1、R2、R3的E0口运行在area 23内；R1、R4、R5的串 口运行在area 1中；R4的lo1运行在area 4中。R5的路lo1运行在area 5中。由于我们不能配置交换机且在以太网上跑OSPF，所以请正确选择OSPF的接口网络类型。area 0明文认证，密码为cisco。R4和R1互相看对方的lo0口路由为24位()。R4不向area 4发送任何LSA(ip ospf dead-intervd.ini..4)。当R2在150秒内没收到HELLO包(if:ip ospf dead-intervd)，邻居关系也不会当。

R4的E0口一秒钟发送4个HELLO包(if:ip ospf dead-intervd.ini..4)。当R2在150秒内没收到HELLO包(if:ip ospf dead-intervd)，邻居关系也不会当。

#### 五、过滤???

RIP不接收171.1.12.0这条路由，不能用列表匹配。但RIP的路由器能通过R2来访问171.1.12.0网断内的主机(进程:default interfa orig..)。

#### 六、feature

在R3上配置一个lo1口将其宣告进RIP域内。掩码为24位。

假定这个环回口连一台主机，地址为192.168.1.133。当在这台主机上trace R4的lo1口时。(做tuntle)

trace信息为

1 192.168.1.134

2 10.1.4.4

注：要求全网全通，除lo0外，不允许出现任何主机路由，所有loopback0接口的地址为10.1.x.x/32 (x为路由器号)，本试验主网段为10.0.0.0/8。不允许出现任何的静态路由。

笔记：

两个帧中继相连接口类型为NNI

R2与R3相连的接口类型为p2p

p2p子接口只需在主接口关闭反向解析

p2p子接口用:frame-relay interface-dlci XXX (本地DLCI号)

p2p子接口只需在主接口关闭反向解析  
p2p子接口用: frame-relay interface-dlci XXX (本地DLCI号)  
改hello, dead会变; dead变, hello不变

RIP:

lo口宣告到RIP里面, 需要pass

或者: 1. pass default 2. no pass s0.1

if: ip rip triggered : 自动生成一条 `timesbase 30 180 0 240`

RIP如果是直连, 一定要打开水平分割 (只有帧中继中主接口默认关闭水平分割, 其他默认打开水平分割)

P2MP会出现32位路由

RIP的认证

P2P上配手工帧中继映射: ip int

LAB

EIGRP汇总出现一条打 的路由

OSPF汇总会出现一条打 的路由

双向重分布

## BGP

### BGP知识点

#### 1. BGP的一些特性以及BGP的三张表

什么时候需要BGP

- 1) 穿越ISP
- 2) 对等策略控制
- 3) 多出口

什么时候不需要BGP:

BGP与IGP的比较

| BGP         | IGP             |
|-------------|-----------------|
| 1. AS见      | 1. 一个AS内部       |
| 1. 丰富的策略选路  | 2. 单一的选路, 简单的策略 |
| 3. 稳定收敛慢    | 3. 收敛快          |
| 4. Internet | 4. 相当于BGP少      |

### BGP特性

1. 可靠, 基于TCP连接 179端口 :抓BGP包:access 101 tcp eq 179/BGP
2. keepalive (60秒)
3. 增量触发更新(还是有间隔的)
4. 丰富的度量值
5. 可用组建大型网络以至于Internet

### IGP的特性

1. 它执行拓扑发现
2. 它尽量完成快速收敛
3. 它需要周期性的更新来确保路由小子信息的
4. 它受同一个管理结构的控制
5. 它采取共同的路由选择策略
6. 它提供了优先策略路由能力

### IGP不合适当域间路由选择协议

一种域间路由选择协议应该能够提供广泛的策略控制

当前缀的数量处于Internet的水平时, IGP路由的周期性刷新特征是不具有扩展特性的.

### BGP的三张表

1. 路由表      show ip rou bgp
2. 转发表      show ip bgp
3. 邻居表      show ip summary

### 2. EBG和IBGP的比较, 各有哪些要求, 解决IBGP路由黑洞问题的几种方法. 解决IBGP同步的几种方法

IBGP 的AS值:200

EBGP 的AS值:20

IBGP :1个router BGP AS号(1~65535) 64512~65535是私有

EBGP: 分属于两个的路由器建立邻居 (OSPF是接口分区域的, 而BGP是路由器分区域的)

IBGP: 属于相同AS的

IBGP优点:减少IGP路由条目

EBGP优点:丰富的策略, 减少路由条目

EBGP的跳数建议用具体的数值

BGP负载均衡默认1条, 最大6条.

### IBGP需求:

IBGP表与IGP同步

只有从IGP学到路由AS内所有的路由器

### 什么时候可用关闭同步:

1. IBGP full mesh

1. IBGP full mesh
2. 开始同步, BGP重分布到IGP

EBGP运行在AS与AS之间的边界路由器上, 默认情况下, 需要直连, 如果不是直连, 必须指EBGP多跳, 定义了AS的边界, 即定义了管理的边界, 体现了管理控制  
IBGP运行在AS内部, 不需要直连. IBGP建议FULL MESH, 主要减少了区域内

3. nei和network语句的含义, 以及邻居的FLAPPING, 下一跳在多访问网络中的实现. 解决IBGP的同步问题.

**Neighbor命令含义:**

1. nei x. x. x. x remote-AS YY : 允许x. x. x. x这个地址来访问我的179端口.
2. 而我访问它时也是访问x. x. x. x这个地址的179端口.

**network含义:**

**IGP**

1. 匹配一个网段
2. 宣告这个协议  
可以走我

3. 这个接口所在的网段运行这个协议3. NO au时, nei必须严格匹配路由表的条目; 在Auto时, 可以对路由表中的主类起作用.

**BGP Auto**时情况下, 可以将主类路由引用BGP.

**BGP**

1. 引入BGP
2. 可用告诉对方, 到这个网络

**BGP防环的五机制:**

1. AS-path EBGp : 记录经过了哪些区域
2. 水平分割 IBGP : 从一个IBGP邻居学过来路由不会传会原来的IBGP
3. class-id : 非客户端, 收到的Class-id和自己的一样, 这个路由就不收
4. RR or.. ID : RR产生的, 在客户端收到一条非RR送来的与RR送来相同的OR--ID, 则不收
5. 从IBP学过来的路由, Next-hop为自己的接口, 丢弃.

**邻居的FLAPPING**

BGP建邻居, IGP表中一定要出现

**主类边界: ABC类网络**

4. BGP的各自包, 以及建立邻居的各种状态

**BGP的各种包:**

OPEN  
KEEPALIVE  
UPDATE

**连接建立过程**

IDLE 初始和终止  
Connect 尝试三次连接  
A 三次握手  
Open S 发OPEN

UPDATE  
NOTIFICATION

Open S 发OPEN  
Open C 确认  
establish 建立

**BGP包的公告延时:**发给IBGP 5s 发给EBGP 30s

**同步:**

BGP表中最优的放入路由表

关闭同步, 使其同步: 关闭BGP的同步, 使BGP的转发表和IP路由表同步

IBGP全互联或包的走向链路上全部运行BGP, 可以关闭同步

## 5. BGP聚合路由以及各参数

BGP 的Auto-Summary (影响本地) 对于Network; Redistribute 都有影响, 都汇总成主类

| 重分布    | EBGP    | IBGP          |
|--------|---------|---------------|
| BGP转发表 | 有(一定相信) | 有             |
| 有明细    | 有       | 有             |
| 有汇总    | IGP     | 有             |
| IP路由表  | 有       | 同步 NO 关闭同步 OK |
| 没汇总    |         |               |

**手工汇总方法:** sh irara

summary : 进程

net静态 : 先在路由表写条到NULL0汇总

aggregate: BGP转发表中, 有明细就能聚合 (自己参数的Weight: 36768)

聚合路由会继承每条AS的属性

**aggregate的参数:**

advertis-map: 承匹配的路由的属性.

as-set: 聚合路由会继承每条AS的属性, 继承的AS-path以乱续的方式并写在大括号内, 在传递出去的时候, 这个AS-path参与选路, 并且算一个AS

可以继承每条明细路由的AS-path; Local (只在本区传); commu

attribute-map: 可以清除属性; 可以设置属性. 可以选择继承哪个AS的属性.

没有suppress-map和summary-only: 会发明细

summary-only: 只发汇总, 不发明细

suppress-map: 抑制自己定义的明细

unsuppress-map:

如果继承了commu属性, 必须要在aggregate时要把commu属性删除, 才能传给EBGP邻居.

nlri:

attribute-map和route-map: 清除所有属性, 但不清除AS-path的属性.

route-map:

## 6. BGP的各种属性, BGP选路原则, 以及路由最优的必要条件.

## 6. BGP的各种属性, BGP选路原则, 以及路由最优的必要条件.

### BGP选路原则:

weight:选小的值, 影响出站量  
local-preference:大, 影响出  
优先本路由器上BGP的路由.  
as-path:选短的, 影响出  
起源代码  
Origin code (IGP<EGP<Incomplete):选小的  
medtric:默认为0, 越大越优先(BGP), 影响入;而(IGP)影响出.  
最近的EBGP邻居  
最近的IBGP邻居  
EBGP优于IBGP  
最老的  
最低的Router-ID

## 7. Dampening, 联邦, 发射器, 对等体组

### dampening(只对EBGP路由有效, 而IBGP无效):

路由惩罚, 当邻居down和UP一次时, 记作一个dampening. 惩罚值加1000(这个值不能修改).  
到达了一个高度, 达到抑制门限(默认值:2000, 最大抑制门限=重用门限\*(最大抑制时间(默认半衰期的4倍:60分钟)/半衰期)). 每隔一段时间(默认15分钟), 值降到原来的一半(称为半衰期), 在降到一段时间后到重用期(默认值:750), 此时, 这个路由重新可以.  
dampening值一直在降:10S降一次  
命令: bgp dampeing route-map DNM  
route-map DAN per 10  
match ip add pre 1  
set dampening 20(半衰期) 700(重启限制) 2500(最大抑制时间) 80( )

### 联邦:解决IBGP全互联的数目

命令: bgp confederation identifier {大AS号}  
bgp confederation peers {小AS号}  
选路原则:先选外部EBGP, 在选内部EBGP, 再选IBGP  
联邦既有IBGP的属性也有EBGP的属性:一个联邦的 B G P 路由, 在另一个联邦可以学到, 但下一跳不发送改变  
但同一区域联邦用lookback建立邻居的时候是要打多跳的。

### IBGP的水平分割原则:从一个邻居学过来的IBGP路由不会发给另一个BGP

### 反射:

EBGP->RR->EBGP, IBGP客户, IBGP非客户;  
IBGP客户->IBGP, IBGP客户, 非客户  
IBGP非客户->EBGP

R1:nei 2.2.2.2 route-reflector-client >R1是RR, 2.2.2.2是客户  
同一簇(class-id(Cluster list)相同):有两个RR相连, 如果上同一个簇,  
路由很有可能出问题.

class-id(即cluster list):是RR产生的

非客户不能通过客户与RR建立邻居关系.

客户也不能通过非客户跟RR建立邻居关系

在大型的BGP里面, cluster会记录所经过的class-id, 但传递给EBGP的时候, 就会把cluster的path删除

Originator:参数这条路由的Router-id.

对等体组的好处:

当客户端全互联时, 要达

### BGP重分布:

EBGP的路由可以直接重分布到IGP; 而IBGP的路由需要忽略同步问题进行重分布(bgp redistribute inter...). 原因: IBGP认为路由已经同步, IBGP的路由IGP路由表上已经存在, 所有, 不允许重分布IGP的路由.

### 对等体:

好处:

减少命令行

拥有完全相同的出站策略

可以设置不同入站策略

### BGP基本配置:

闭关自动汇总:no au

闭关同步:no au

Router-id

默认情况下不同AS传过来的MED是不比较的.

### 命令解析:

Maximum-paths --修改负载均衡的条数, 默认是EBGP的, 要ibgp负载:maximum-paths

Ebgp时, 有多少个下一个就可以就多少个负载; IBGP时必须IGP负载, IBGP才能负载.

show tcp brief (show TCP的连接)

nei x.x.x.x next-hop-unchanged --下一跳不改变

进程:bgp dampening --修改dampening值

show ip bgp da --察看dampening值

show ip bgp nei 1.1.1.1 advertised-routes --察看向1.1.1.1发了那些露有

bgp cluster list --修改class-id

进程:no bgp client-to-client --当客户全互联时(全环路反射), 不会向其他客户做反射(防止路由环路)

bgp bestpath as-path ignore --隐藏命令:不使用as-path选路

bgp always-compare-med --比较不同As传来路由的MED值  
bgp bestpath med confed (missing-as-worst)--可以比较联盟内的MED值(如果没有med值,则把这个值设为最大)  
bgp bestpath compare-routerid:忽略最老路由,直接比较Router-ID  
bgp maxas-limit --离我AS多远的路由会使用.  
nei x.x.x.x local-path 100 --欺骗x.x.x.x我的AS为100  
nei x.x.x.x local-path 100 no-prepend  
nei x.x.x.x advertise-map 111 exist-map 222 --存在exist-map时发汇总  
nei x.x.x.x advertise-map 111 non-exist-map 222 --当exist-map不存在,时,aggrage这条汇总不发出去.只发明细  
nei x.x.x.x next-hop-unchange (只能在EBGP多跳时用)最好  
nei x.x.x.x capability orf prefix-list --两边协商可以通过哪些路由  
set as-pth prepend last-as 3 --最后的AS(起源的AS)号再出现多次.只能影响穿过的路由  
allowas-in --让路由器可以接收有本AS号的路由  
bgp default local-preference --适合从EBGP学到的BGP路由和自给产生的BGP路由。?

#### 其他知识点:

RIPv1    自动  
          手工  
RIPv2    自动    可以基于 RIP date base  
          手工    可以(可以汇总学过来的)  
EIGRP    自动    只能汇总本路由产生的  
          手工    可以(可以汇总学到的)  
OSPF     手工

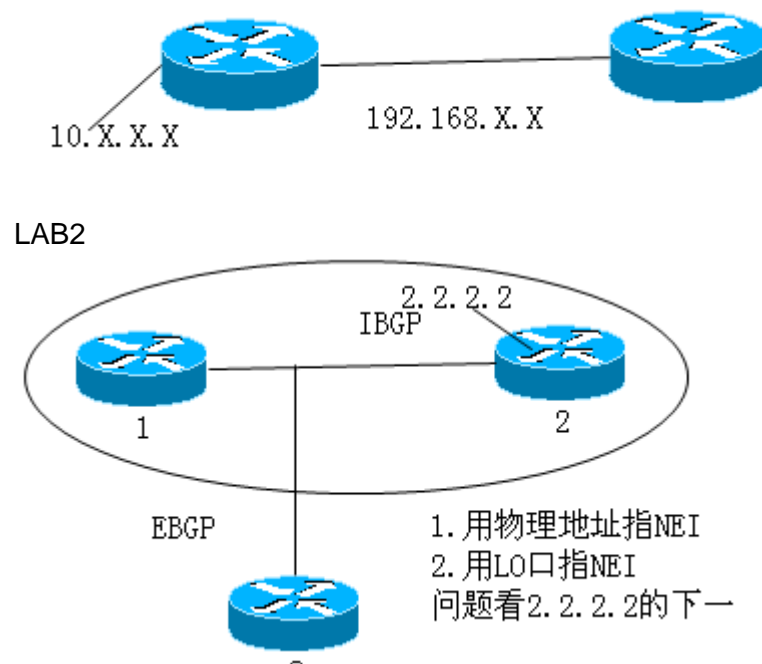
BGP的硬清是清除自己产生的,软清是清除学来的.

show ip pre  
set metric-type in... :会累加metric值  
ip route 使用的是前缀列表  
aggregate-address使用的而是正掩码

#### 作业:

1. unsuppress-map
2. addr...
3. route-map
4. unsuppress-map能不能于summary-only一起用

LAB1



### 试验现象:

R1与R2运行RIP协议;R2与R3建立静态路由.

用物理接口连接时:R3上看到1.1.1.1的NEXT-HOP为10.1.123.1;R2上

看到2.2.2.2的NEXT-HOP为10.1.123.2

用LO口建立连接时:R3上看到1.1.1.1与2.2.2.2的下一跳都是2.2.2.2

### 扩展

nei 2.2.2.2 remo 100

nei 10.1.123.1

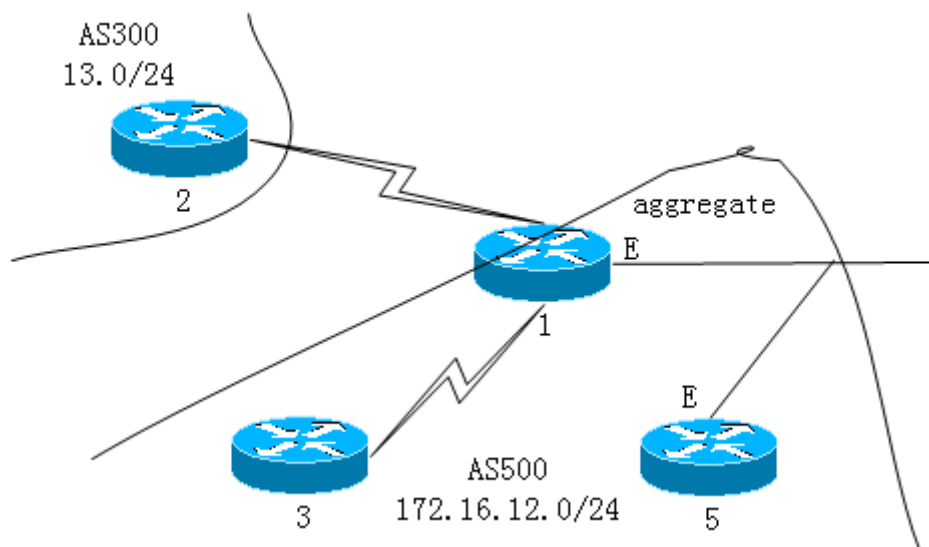
update-so 10.1.123.1

update-so 10.1.123.2

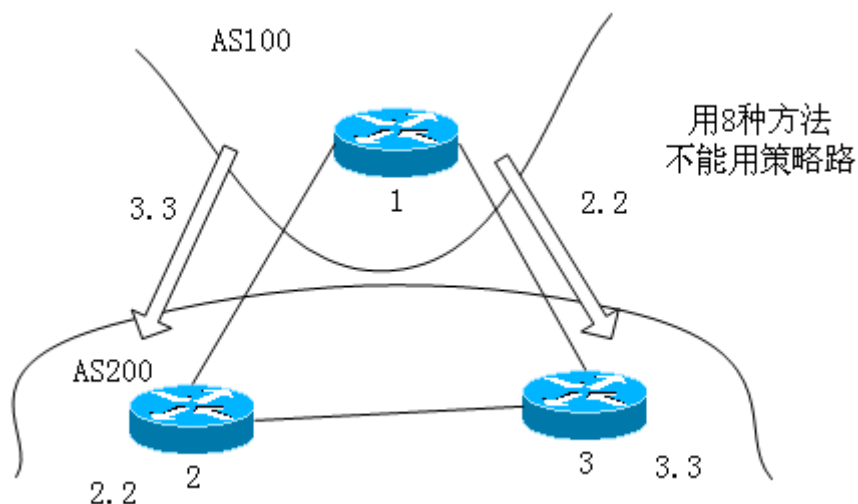
此时建立连接的是2.2.2.2的179端口. R1(show tpc b):

```
TCB Local
Address
Foreign Address
(state)
007D4DC4
10.1.123.1.11003
2.2.2.2.179
ESTAB
```

LAB3:



LAB4:

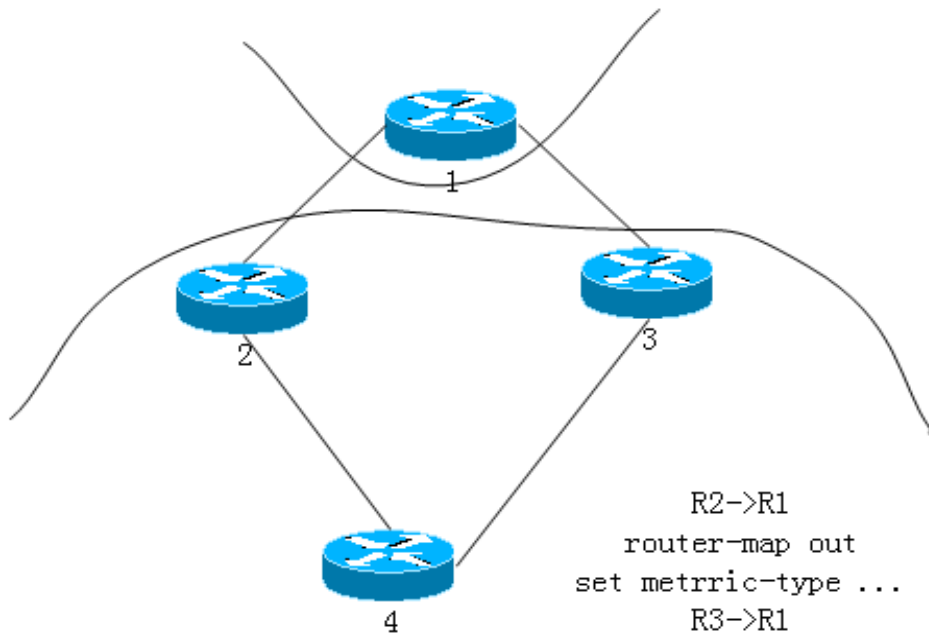


方法:

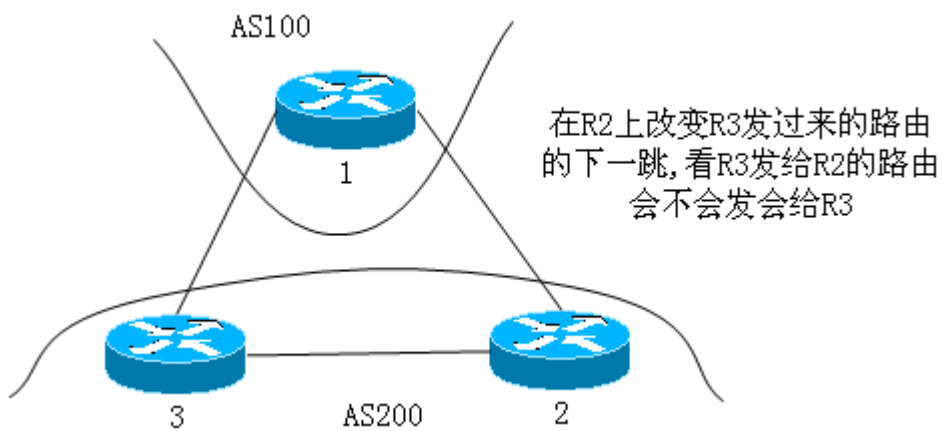
- 1.weight
- 2.MED (R1) 入 默认为0
- 2.1:MED (R2R3) 出
- 3.loc 入
- 4.Origin code :出? 入?
- 5.下一跳 R1 入? 出?
- 6.as-path :最短AS Path
- 7.network
- 8.prefix-list  
dist...list  
route...list
- 9.community
- 10.filter

11. 出站过来和出站过滤:nei 2.2.2.2 capability orf prefix-list  
both/receive/send 与邻居协商,只收/发prefix-list的路由.但一定要  
在network之前做才生效.

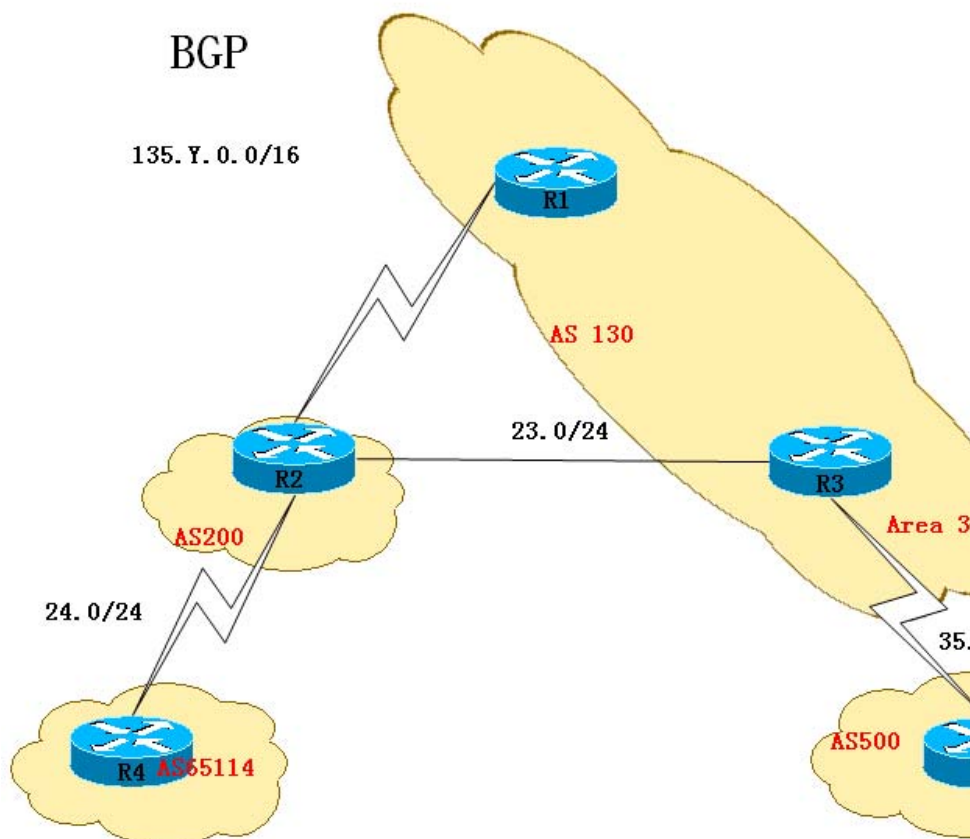
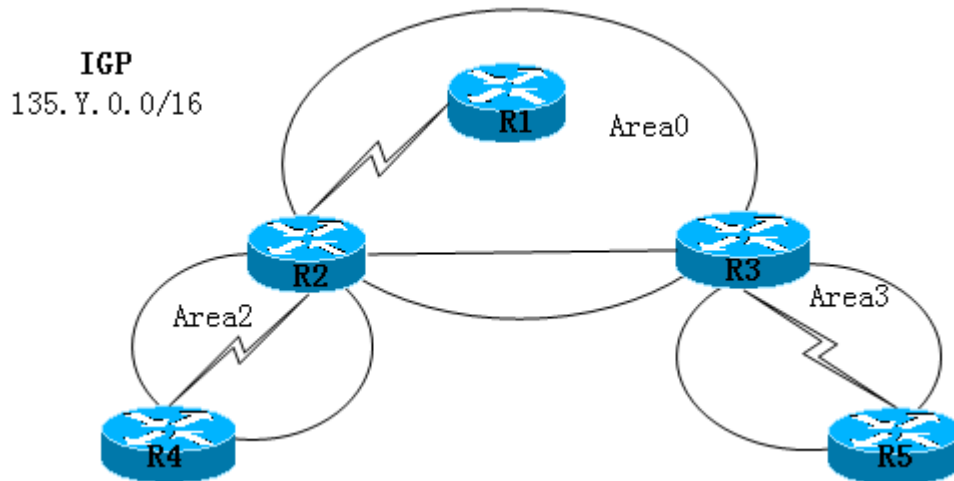
LAB5.



LAB6:



LAB7



总需求:

1. 不允许使用静态路由. 否则-20分.

2. 测试时间:2小时

IGP需求:30分

1. 为每个router建立loopback0接口(ip:135. y. x. x/24, y为RACK号, x为router编号. 例如, 对r1, x=1), 发布在适当的IGP路由进程中, 在IGP配置完成后, 在每台router上都一个看到所有router的loopback 0路由. (2)
2. R4上loop8--loop11, ip4. 4. 8. 4---4. 4. 11. 4/24, 重分布到ospf中, 并汇总. (5)
3. 汇总路由在ospf中传递时cost要发生改变. (3)
4. 如果以后运行ospf的路由器上存在Gibyte以太网口时, 其cost值要为5. 其他接口cost值也要相应的改变. (5)
5. area2的路由条目要尽量友好, 并且LSDB中要求存在7类默认路由. (5)
6. area3中不能存在外部路由, 但是应该存在域间路由. (5)
7. R3的E0口永远不能称为DR或者BDR. (5)

BGP需求: (70)

1. 所有BGP路径关系均使用loopback口作为update source. (5)
2. EBGP: R1--R2 R2--R4 R3--R5 (5)
3. IBGP: R1--R3 (5)
4. R4公布lo20, ip:200. 200. 4. 4/24, R2传递给R1时要去掉所有AS信息. (5)
5. R1上公布lo20, ip:200. 200. 1. 1/24, R2上公布lo20 ip:200. 200. 2. 2/24. (5)
6. R3上只有将200. 200. 4. 0/24和200. 200. 1. 0/24传给R5. (15) 指nei时  
=prefix-list
7. R5公布lo10--lo17, ip:199. y. 10. 5-----199. y. 17. 5/24. (5)
8. R3对其进行聚合, 只将聚合路由发给邻居, 不能使用summary-only参数, 不能使用路由过滤. (5) 用suppress-map和AS-PATH加本身 ;入方向comm no  
advitise(不会传出本路由器)
9. R1, R2, R3, R4都要可以PING 通R5公告的明细路由. (10)
10. R2监控R1的lo20, 震荡一次给1000惩罚值, 当惩罚值超过2500该路由被抑制, 每30分钟惩罚值降到原来的一半, 当惩罚值降到12000, 该路由被重新起用. (10)

LAB8:

R5配置:

R5

conf t

```
conf t
ho Backbone1-2
no ip do lo
li 0
no exec-t
logg s
end
```

```
conf t
int s0
no sh
ip add 110.100.1.254 255.255.255.0
```

```
int e0
no sh
ip add 110.100.2.254 255.255.255.0
```

```
int lo1
ip add 198.16.1.1 255.255.255.0
int lo2
ip add 198.16.2.1 255.255.255.0
int lo3
ip add 198.16.3.1 255.255.255.0
int lo4
ip add 198.16.5.1 255.255.255.0
int lo5
ip add 198.16.22.1 255.255.255.0
int lo6
ip add 198.16.23.1 255.255.255.0
```

```
router bgp 2
no au
no sy
bgp router-id 170.16.1.1
net 198.16.1.0
net 198.16.2.0
net 198.16.3.0
net 198.16.5.0
net 198.16.22.0
net 198.16.23.0
nei 110.100.1.1 remote 1
nei 110.100.1.1 route-map AS out
nei 110.100.2.1 remote 1
exit
```

```
route-map AS per 10
set as-path prepend 3

exit
```

## 试验

LAB1: 向BGP中注入IGP路由(宣告, 重分发)

LAB2: 向IGP中注入BGP路由(注入EBGP路由和注入IBGP路由)

注意: 默认IBGP是不能重分布给IBGP的, 但输入BGP:

LAB3: 没有IGP的BGP (静态路由? 默认路由?)

BGP产生默认路由:

network静态

```
nei 2.2.2.2 default-originate route-map 111
```

两边都上默认路由不可以

LAB4: IGP上的BGP(同步, 下一跳) (什么情况下可以关闭同步)

LAB5: EBGp多跳

LAB6: 聚合路由 (各种MAP)

LAB7: 使用LOCAL\_PREF (影响出)

LAB8: 使用MED

限制允许自己接收的最大前缀条目来保护他的边界路由器 :nei X.X.X.X

maximum-prefix (允许最多允许对少条路由) (可设置: 达到最大数量的

XX%开始警告; 重建邻居; 仅警告)

LAB9: BGP选路绕圈(几种方法)

修改holdtime (默认是多少, 最小是多少)和advertisement-interval

(IBGP, EBGp): holdtime为0s时, ???什么是HOLDTIME

neighbor命令的体会试验(一边指环回口, 一边指接口地址)

邻居的FLAPPING

auto-summary与network命令相互作用  
BGP选路时一些特定的BGP命令  
产生默认路由与route-map相结合  
BGP的负载均衡  
删除私有AS号的两种方法（联盟和remove-as）  
RR与关闭反射RR的路由  
soft-reconfiguration inbound  
keepalive和holdtime  
network和backdoor  
RR和联盟  
拆分路由与条件路由: `bgp inject-map 111 exist-map 222 copy-attributes`（拆分聚合路由, 继承聚合路由属性）

AS-PATH PREPEND  
BGP的下一跳  
允许接收经过多少AS的路由  
LOCAL\_AS

OSFP和IBGP; EIGRP和IBGP能不能做负载

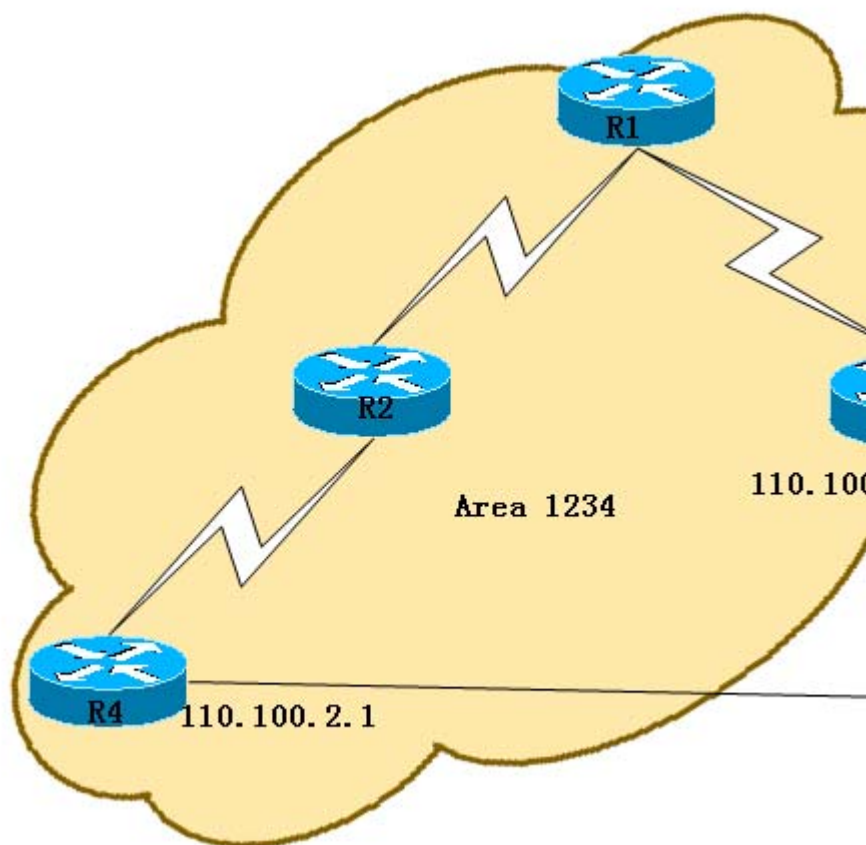
过滤列表与正则表达式

收多少前缀  
BGP的链路认证: `nei *.*.*.* pass .....`

删除私有AS号的两种方法(联盟和remove-as)  
RR与关闭反射RR的路由:  
`soft-reconfiguration inbound` :可以软清in方向的, 可以对这个逻辑软清, 对内存产生影响, 会多建一个buffer.  
keepalive和holdtime:  
`network`和`backdoor`  
`badkdoor`: 当我从IGP和EBGP同时学来一条路由, 优选IGP学来的路由. EBGp的管理距离改为200.  
RR与联盟

不懂: holdtime  
advertisement-interval

## 综合2



### BGP实验二

IGP全网互通，请用OSPF作为IGP协议，主网段为170.16.0.0/16。不能配置BACKBONE。

#### IBGP

R1、R2、R3、R4在AS1234里，R1/R2、R1/R3、R2/R4、R2/R3、R3/R4互为IBGP邻居关系，R2是RR，R3、R4是它的客户。IBGP邻居关系要尽可能的稳定。

#### EBGP

R3、R4分别和BACKBONE建立EBGP邻居关系，BACKBONE属于AS 2，BACKBONE只与AS 1建邻居。

R4与R5直连口的地址为110.100.2.1，R3与R5直连口的地址为110.100.1.1

#### 策略

R3上有一个环回口，10.20.200.3/24，宣告进BGP，只能在本AS内

内传递。

```
network 200.200.3.0 mark 255.255.255.0 route-map 111
route-map 111 per 10
set community local-AS (向全体邻居发这个属性)
```

R1, R2上分别有一个环回口: R1:10.20.20.1/27, R2:10.20.20.2/27。分别将其宣告进BGP。R3/R4

分别只发送200.200.1.0/27和200.200.2.0/27的路由给BACKBONE。让BACKBONE访问R1的环回口走R3, 访问R2的环回口走R4。

```
nei 10.1.12.2 prefix-list
```

聚合

在R4上面聚合, R4的BGP转发表如下

| Network         | Next Hop      | Metric | LocPrf | Weight |
|-----------------|---------------|--------|--------|--------|
| Path            |               |        |        |        |
| *>198.16.0.0/19 | 0.0.0.0       |        |        | 32768  |
| 2 i             |               |        |        |        |
| s>198.16.1.0    | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |
| *>198.16.2.0    | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |
| *>198.16.3.0    | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |
| s>198.16.5.0    | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |
| *>198.16.22.0   | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |
| *>198.16.23.0   | 110.100.2.254 |        |        | 1000   |
| 2 i             |               |        |        |        |

在R3上做策略, 当R3与BACKBONE之间链路正常的时候, R1访问BACKBONE走R3; 当R3与BACKBONE失去连接时, R1访问BACKBONE走R4。

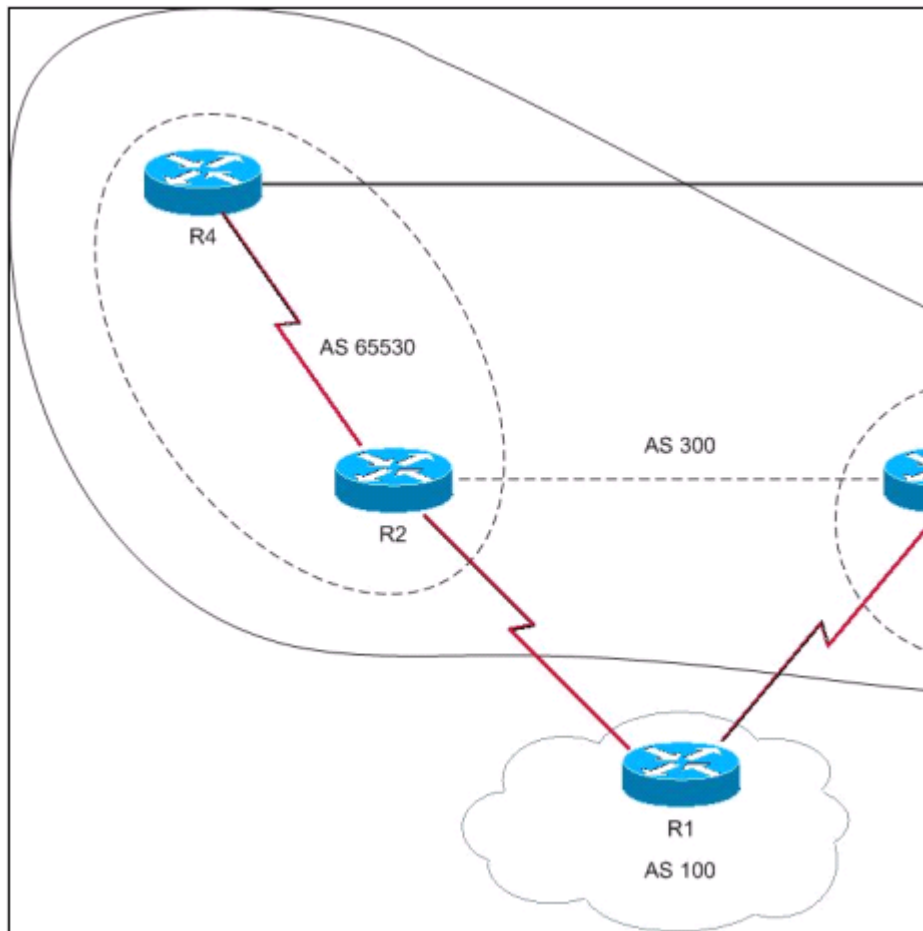
```
nei 110.100.1.254 prefix 1 out
prefix 1 per 200.200.1.0
```

R3只接受源自于AS 2或源自于AS 2直连AS的路由。

```
^2_[0-9]*$
```

R4上监控198.16.3.0这条路由, 如果其FLAPPING, 就对其进行惩罚, 当198.16.3.0, FLAPPING一次后, 经过20分钟, 惩罚值降到500, 当惩罚值超过2500时对其进行抑制, 当惩罚值降到700时, 将重新起用这条路由。

### 综合3



1、R1、R2、R3、R4运行OSPF area 0 将环回口放入OSPF中，不要出现32位路由。

主网段用10.1.0.0/16

2 . R1与R2、R3建立EBGP关系，R5与R3、R4建立EBGP邻居，R2与R4建立IBGP关系，与R3建立EBGP关系。

(loopback口建立邻居关系)

在建立BGP邻居关系的时候尽量采用稳定的邻居关系。

建立邻居后在R3、R4上不能改变R5的下一跳。

问题：如果在R1 BGP上network 1.1.1.0 这条路由，会出现不稳定情况(度量值所导致导致)，如何解决

3. R5公告172.1.12.0/24 172.1.13.0/24 172.1.14.0/24 172.1.15.0/24这些路由，

要求在R3上做聚合，并且抑制13.0和14.0，不能使用suppress-map，(route-map)

在R3上提高所学到的路由LOCALPREF值为300，不能使用router-map。  
(bgp default-localPre: 只对EBGP学来的路由和自己network的路由有效)

这些路由在R2上优选R4传来的路由。

在R3和R4上做配置使得这条聚合路由不能传回AS500，不能使用发布列表和社团属性。

不能用route-map做过滤(prepend)，R1收到的路由优选R3。

4. 在R4公告一条路由40.1.1.0/24，只能把这条路由公告给R5和AS65530中的路由器。要求在R4上配。

5. 在R2公告一条路由20.1.1.0/24，在AS300中看到这条路由的local-preference为202，

并且R2不能把这条路由传给R1，要求使用最简命令。配置后考虑R1如何ping通20的路由，

要求在R1上做配置。

6. 在R3公告一条路由30.1.1.0/24，要求R5访问30时走R3，访问20时走R4。不能在R5上做配置。

BGP：选路原则

下一跳是否可达；如果不可达则不会优化。

本地选路，优选最高Weight值

本路由器打上的Local-preference值不会传出本AS，传出去时会变回默认100。

本路由器产生的默认Weight值是32768。

## 基础

网络设计三层机构

接入层

分布层

核心层

冗余：用于防止单点故障

负责均衡

接入层用冗余链路连接到分布层，防止单点故障，分布层冗余链路连接到核心层，和接入层，核心层用全互联，为了快速的包交换

DHCP:动态主机控制协议

SAP(Service access point)描述上层的信息

ISO七层模型：下层是为上层服务的，上层是下层的负载

帧: SAP: tag 0x8000ipv4  
0x0806 ARP  
IP: SAP: protocol: 6 tcp  
17 UDP  
tcp:SAP: 端口号

~~~~~

### **DHCP:**

基于UDP的67和68端口，67是service端；68是client端  
建议不在串口做DHCP

### **DHCP命令:**

```
service dhcp --启动DHCP服务
no ip dhcp conflict logging --发生冲突时不产生提示信息
ip dhcp database ftp://user:xxxx --把DHCP提示信息存放在一个文件上
ip dhcp excluded-address 1.1.1.1 1.1.1.10 --1.1.1.1 到1.1.1.10
这些地址不允许分配
ip dhcp pool LAN
network 192.168.1.0 255.255.255.0 --可以分配的地址
default-router 192.168.1.1 --配网关
dns-server --配DNS Server
lease DAYS/infinite --地址被分配出去的时间
sh run | in dhcp 查看DHCP的配置
ip dhcp pool ??
show ip dhcp --查看dhcp的配置
ip add dhcp --启动DHCP客户端
```

DHCP工作原理:

- 1、客户端发一个DHCP discover
- 2、服务端回应一个: dhcp reply
- 3、客户: DHCP request
- 4、服务: DHCP ACK

服务端: debug ip dhcp server packet

客户端: debug dhcp da

DHCP的租期时间:

续约时间: 租期的一半

重新绑定时间: 租期的4/5

#### DHCP中继:

将DHCP的广播包, 中继到server和client端, 改为源地址为中继接口地址, 目的地址为helper-address地址, 根据单播包的源地址来选择用哪个pool地址分配。

#### 命令解析:

if#ip help-address 10.1.1.1 --DHCP的中继。把广播包转成  
10.1.1.1: 源地址: 本路由器收到这个广播地址的端口的地址, 源地址: help-address

#### 高级编址:

第一块: 可宽展的网络设计

好处:

- 1、可宽展性:
- 2、可预见性
- 3、灵活性

第二块: 层次的网络编址

好处:

- 1、减少路由表条目
- 2、更高效的个利用IP地址

#### 子网划分:

$X < 2^n$  (N为主机位, X为主机的数量)

网络以 $2^n$ 递增

DHCP. PDF

Adobe Acrobat  
7.0 Document

#### 正则表达式

. 匹配任何单个的字符, 包括空格

^ 匹配一个字符串的开始字符

\$ 匹配一个字符串的结束字符

\_ 下划线. 匹配一个逗号, 大括号, 一个输入字符串的开始, 一个输入字符串的结尾或一个空格

| 管道符. 它具有逻辑或(OR)的含义, 意思是可以匹配两个字符串中的一个  
\\ 转意字符, 用来将紧跟其后的扩展字符转变为常规字符

\* 匹配前面字符的任意序列 (0次或多次出现)  
+ 匹配前面字符的一个或多个序列 (1次或多次出现)  
? 匹配前面字符的0次或一次出现  
如何输入?: 先按ctrl+v 在按?

[] 表示一个范围

举例:

^a.\$ 可以匹配ab, a, ax等  
^100\_ 可以匹配100, 100 200, 100 300 400等

练习:

100\$ 匹配什么  
仅仅匹配(65000)如何写 : ^\\(65000\\)\$

abc\*d 匹配 abd, abcd, abccd

起源于AS 100, 中间可以隔一个AS的路由: ^.\*\_100\$

ip as-path access list 1

## 控制列表

### 访问控制列表

if#ip access-group {*NO.*/*WORD*} {in/out}  
后面一定要加:acc 101 per ip any any

### 时间访问控制列表:

#### 第一步:定义时间范围

time-range *WORD*  
absolute 定义绝对开始时间和结束时间  
必须跟start/end  
absolute start hh:mm date month year  
end hh:mm date month year

start和end至少要有有一个.

没有start, 则: 立即开始, 到end时间

没有end, 则: 重start时间开始, 没有停止时间

**periodic:**定义一个星期之内, 每天什么时间生效 例: periodic

weekdays 18:00 to 18:59(比要定义的时间少一分钟) ——下午六点  
到七点

## 第二步: 把时间范围和访问控制列表绑定起来

access 101 per ip a time-range *WORD*

```
access list 101
10 permit ip 170.16.0.0 0.0.255.255 any
20 permit tcp any host 10.1.123.254 eq www
30 deny tcp any any eq www time-range HTTP (active)
40 deny tcp any any eq ftp time-range http
50 deny tcp any any eq ftp -date time -range Http
60 permit ip any any
```

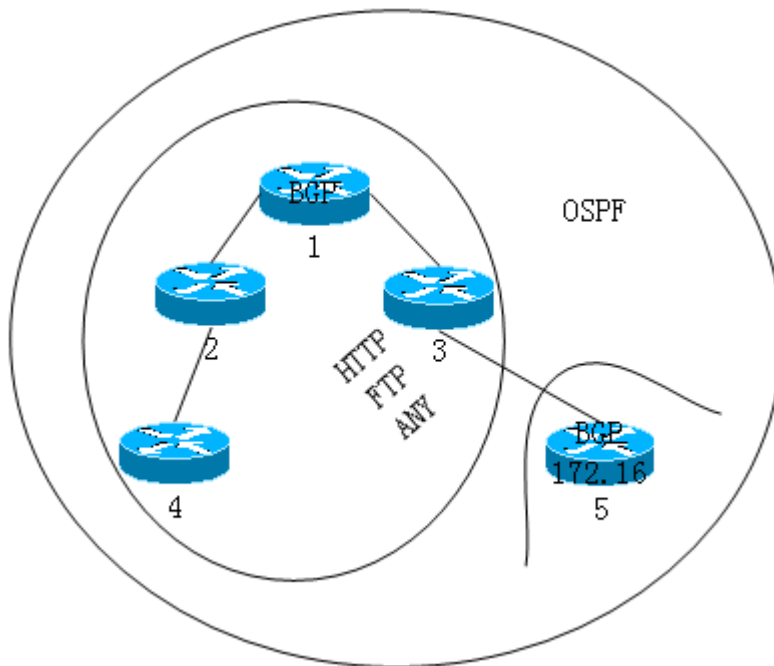
## 自反访问控制列表(reflect)

允许内部主动发包, 而外部相应

外部不可以主动发包

只对通过本路由器包起效, 而对本路由器的包不起效

只在立即的命名访问控制列表



R3:

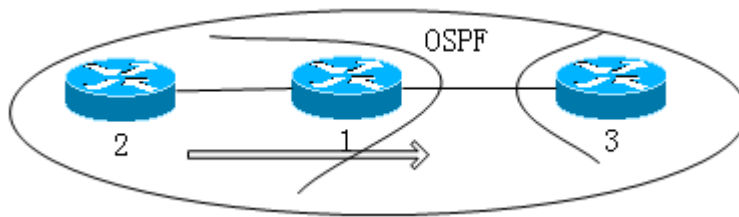
int s1

ip access-group ACCin in

```
ip access-group ACCout out
conf t:
 per tcp a eq bgp
 per tcp a eq bgp any
 per ospf any any
 per icmp any any
 per tcp any any eq www reflect(反射) REF
 per tcp any any eq ftp reflect REF
 per tcp any any eq ftp-data refltct REF
```

```
ip access-list extended ACCin
```

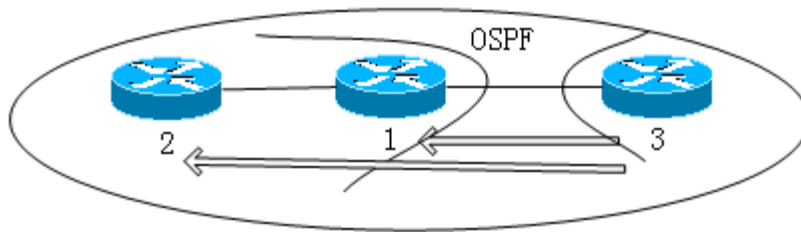
例:



```
R1
int s1:
ip access-group ACCin in
ip access-group ACCout out
conf t
ip access-list extended ACCin
per tcp host 3.3.3.3 eq telnet a established (允许有established标
记的进来)
per ospf any any
per icmp a a
evaluate REF (允许REF出去的包在进方向, 可以让对方反射一条进来)
exit
ip access-list extended ACCout
per os
per ospf any nay
per icmp any nay
per tcp any host 3.3.3.3 eq telnet established (允许本路由器访问本
路由器发起的)
per tcp a host 3.3.3.3 eq telnet eq telnet reflect REF
```

### 动态访问控制列表

那个先起, 那个生效



1. R2 telnet R1
  - 2.1 R1用本地密码验证
  - 2.2 动态生成access. 允许R2 telnet R3
  3. R3可以telnet R2
- access 101 dynamic WORD timeout 5 per tcp any host 2.2.2.2 eq 23

R1:

```
extended ip access list 101
permit ospf any any
permit icmp any any
permit tcp any host 10.1.14.1 eq telnet
Dynamic DYNN permit tcp any host 2.2.2.2 eq telnet
```

username ccnp autocmmand access-enable host timeout

## IPV6

### IPv4的限制:

1. 地址短缺, 2的32次方
2. 复杂的包头
3. 复杂的路由器和主机配置 (手工配(大网络), DHCP(配置服务器))
4. 重编地址比较困难
5. 很大的路由表(CIDR没有做好)
6. 部署安全、组播、mobile-ip比较麻烦

### IPv4地址短缺的解决方案

- 1、NAT
  - 静态NAT
  - 地址池
  - 地址轮循
  - 缺点: 1. 速度慢; 2. 耗费CPU资源(一个NAT映射占64K内存); 3. 破坏端到端模型; 4.
- 2、DHCP
- 3、CIDR

### IPV4与IPV6的比较

| 服务 | IPV4 | IPV6 |
|----|------|------|
|----|------|------|

| 服务   | IPV4          | IPV6                          |
|------|---------------|-------------------------------|
| 地址空间 | 32bit(8*4)    | 128bit (16*8)                 |
| 自动配置 | DHCP          | 无状态自动配置 /dhcp                 |
| 安全   | Ipssec        | 可以做到端到端的ipsec加密               |
| 移动性  | Mobile-ip     | Mobile-ip with direct routing |
| QOS  | 区分服务、集成服务     | 区分服务/集成服务                     |
| 组播   | Igmp/pim/MBGP | Mid/PIM/MBGP,scope identifier |

### IPV6特性

- 大的地址空间
  - 提高了autoconfigure的技术，即插即用
- 简单高效的包头
- 提高了安全和移动性
- 有丰富的IPV4到IPV6转换途径

### 大的地址空间

128bit

8段，每段16bit

2031:0000:130F:0000:0000:09C0:876A:130B =>

2031:0:130f:0:0:9c0:876a:130b(前导0省略)

=>

2031:0:130f::9c0:876a:130b(连续0省略--"::"只能出现一次)

IPV6有一个自动配置的技巧

路由器会向网络内，发布一些网络信息（包括前缀、缺省路由...）以组播的方法发送信息的。

新加

### IPV6自动配置编址技术

IPv6地址=前缀+接口标识

前缀：相当于v4地址的网络ID

接口标识：相当于v4地址中的主机ID

2001:A304:6101:1::E0:F726:4E58

IPv6地址：

MAC地址第7位为0的时候**全球唯一**，为1的时候**本地唯一**

EUI(ipv6)地址第7位为0的时候**本地唯一**，为1的时候**全球唯一**

MAC地址转换为EUI(IPV6)地址时，自动会把第七位转换**0=>1;1=>0**

注意：串口没有MAC地址，但串口的EUI会使用以太口的MAC地址。

MAC: 0010:0C2F:3F2B

IPv6: 0010:0C**FF:FE**2F:3F2B (用MAC地址填充，中间夹带了**FF:FE**)

例：网段：2031/64 的IPV6;MAC地址：003F:442B:A23E 构造一个全球唯一的IPV6地址：2031::023F:44FF:FE3B:A23E

### IPv6编址及寻址 --IP v6地址分类

- 单播地址 (Unicast Address)
- 组播地址 (Multicast Address)
- 泛播地址 (Anycast Address)

## 特殊地址

| 地址类型    | 二进制前缀               | IPv6标识    |
|---------|---------------------|-----------|
| 未指定     | 00...0 (128bits)    | ::/128    |
| 环回地址    | 00...1 (128bit)     | ::1/128   |
| 组播      | 11111111            | FF00::/8  |
| 本地链路地址  | 1111111010          | FE80::/10 |
| 本地站点地址  | 1111111011          | FEC0::/10 |
| 全局单播    | (其他)                |           |
| anycast | 可以和全局单播和本地站点一样 前缀:: |           |

## 单播地址

IPv6单播地址分类（根据地址范围）：

全局单播地址 例：2001:A304:6101:1::E0:F726:4E58

本地链路地址 例：FE80::E0:F726:4E58

本地站点地址 例：FEC0::E0F726:4E58

全局单播地址层次结构

| 001                        | 全局路由前景(45位) | 子网ID(16位) | 接口ID(64位) |
|----------------------------|-------------|-----------|-----------|
| 2000-3fff<br>::/3<br>(001) |             |           |           |

## 本地链路地址和本地站点地址

本地链路地址

设备链路地址

设备站点生成，在本地网络中使用

本地站点地址

相当于v4网络中的私网地址（需要配置）

## 组播地址

| 8-bit     | 4-bit    | 4-bit | 112-bit  |
|-----------|----------|-------|----------|
| 1111 1111 | Lifetime | Scope | Group-ID |

## Flags(lifetime)

用来表示永久地址(0000)或临时地址(0001)

## Scope

表示组播组的范围:0001—本地接口、0010—本地链路、0011—本地子网、0100—本地管理、0101—本地站点、1000—组织机构、1110—全球

## 组播指定地址

| 组播指定地址  | 范围   | 含义    | 描述           |
|---------|------|-------|--------------|
| FF01::1 | 节点   | 所有节点  | 在本接口范围的所有节点  |
| FF01::2 | 节点   | 所有路由器 | 在本接口范围的所有路由器 |
| FF02::1 | 本地链路 | 所有节点  | 在本接口范围的所有节点  |
| FF02::2 | 本地链路 | 所有路由器 | 在本接口范围的所有路由器 |
| FF05::2 | 站点   | 所有路由器 | 在本接口范围的所有路由器 |

## 被请求节点组播地址

此类地址由前缀FF02::1:FFXX:XXXX/104和单播或泛播地址的低24位组成，主要为两类特定类型的地址：替代IPv4中的ARP和重复地址检查

(DAD)

一个设备会对一个将要设置的本地链路，本地站点，都会产生一个被请求节点的组播地址

泛播地址

泛播是一种机制，它使到最近点的发现机制成为可能。泛播地址可以使用可聚合的全球单播地址，也可以使用本地站点或本地链路地址。因此要区分泛播地址和单播地址是不可能的。在IPv6中，只有少数几个应用使用泛播地址。移动IPv6是设计使用泛播的一个协议范围

回环地址

类型于IPv4协议，每个设备都有一个回环口

IPv4五元素：

原标签，目标

IPv6的QoS:流标签和

IPv4兼容的IPv6地址

此地址是有过滤机制使用的特殊单播IPv6地址，目的是在主机和路由器上，站点创造

**邻居发现协议**

**IPv6地址配置技术**

**自动配置**

无状态自动配置 (stateless autoconfiguration)

有状态自动配置 (stateful autoconfiguration)

**手工配置**

建议用于服务器和重要网络设备

无状态自动配置上iIPv6中最有吸引力和最有用的新特性之一，它运行本地链路上的节点根据路由器在本地链路上公告的信息自给配置单播IPv6地址。

无状态自动配置涉及都到以下机制：

前缀公告—在本地链路沙公告前缀和参数。IPv6节点理由前缀公告信息来配置IPv6地址。

重复地址检测（DAD）—确保杂接口上无状态自动配置的每个IPv6地址杂本地链路的范围内唯一的。

前缀重新编址——在本地链路上公告修改了的前缀或新的前缀和参数，重新编址一个已经公告过的前缀。

#### 前缀公告：

——前缀公告是无状态自动配置中的

#### 命令解析：

ipv6 up --启动IPv6

接口：ipv6 enable --自动用MAC地址转换成EUI地址建立了IPv6地址

在没有配置IPv6地址前可以用

IPV6 ADD AUTOCONF

或用

ipv6 enable

#### eui特性

如果没64位,就前面用0填充,然后后面用EUI补充

如果大于64位,后面的就用半段EUI填充.

如果没link-local的情况下使上面的情况

如果有link-local的话就用link-local的后24位来填充的.

#### 配置link-local

IPV6 ADD FE80:: link-local

link-local只定义了前10位,后面的随便.

icmp向源节点报告关于向目的地址传输IP数据包的错误和信息。

icmp v4和icmp v6共同使用的错误和信息性消息

| 消息    | 类型号 | 消息类型 |                     |
|-------|-----|------|---------------------|
| 目标不可达 | 1   | 错误   | 目的主机的ip地址或端口未处于活动状态 |
| 数据包超时 | 2   | 错误   | 数据包长度超过发送链路的最大MTU   |

|      |     |    |                     |
|------|-----|----|---------------------|
| 回应请求 | 128 | 错误 | 发送到目的地的消息, 请求以个回应消息 |
| 回应应答 | 129 | 错误 | 用来回答回应请求消息的消息       |

icmp v6的协议号是58, ipv6认为icmp v6数据包是一个上层协议, 像TCP和UDP一样, 意味着它被放在ipv6数据包中所有可能的扩展包头之后。

在IPv6中, 协议的几种机制和功能使用icmpv6消息:

- 代替地址解析协议 (ARP) 用新的ICMPv6消息
- 无状态自动配置 用新的ICMP v6消息
- 重复地址检测 (DAD) 用新的icmp v6消息
- 前缀重新编址 用新的icmp v6消息
- 路径发现协议 (PMTUD)

NDP协议用的几种ICMP v6消息

RS (router solicitation) (icmpv6 type 133) 路由请求

RA (router advertisement) (icmpv6 type 134) 路由应答、公告 (把前缀和一条缺省路由通告给PC)

NS (neighbor solicitation) (icmpv6 type 135) 邻居请求

NA (neighbor advertisement) (icmpv6 type 136) 邻居公告、应答

Redirect (icmpv6 type 137) 重定向

当接口配置地址, 就发送NS 以一个未定义的地址发, 目的是一个被请求的地址。没有监测到冲突~~~就发送DAD说明自给的link-local地址, 如何在以设备的地址来发送一个NS消息。

思科认为, 串口上是不连主机的, 所以不往串口发路由公告 (RA (每200秒发一次))

主机自动请求, 发RS

RA-interval 在以太链路上, 确认唯一的, 每个多少秒发Ra

如果是路由器就不会收到RA里的默认路由

默认发送RA有效时间是30天

DAD和代替ARP都是用NS/NA, DAD的NS/NA是没有源地址的, 是::, 而替代ARP的NS是源地址的, 源地址是自给接口的地址, 目的地址都是被请求节点地址。

手工指定neighbor

第二地址的测试

IPv6邻居发现协议 (NDP-RFC2461)

IPv6邻居发现协议 (NDP-RFC2461)

重复地址检测 (DAD) :

DAD监测--先监测link-local, 就是发FF

FF: 组播

0: 永久

2: 链路地址

DAD: 接口NO sh和自动配地址的时候会发生DAD

ARP协议的替代协议:

```
ipv nd ra-lifetime
```

```
ipv nd prefix 2100::/64 no-advertise
```

### IPv6的RIP配置

conf:

```
ipv6 unicast-routing
```

```
ipv6 route rip WORD
```

```
if#ipv6 WORD enable
```

RIRng

RIPng是以link-local当下一跳的。

以本接口的link-local地址为源地址, 默认更新地址: 以FF02::9为目的地址, UDP:521;可以修改

发送默认路由

```
int s0:ipv6 rip CISCO default-information
```

only: 只发默认不发明细

originate :即发明细, 又发默认

RIP是不用建邻居直接发路由表的

帧中继上耍的HUB端要关闭IPV6的水平分割

### IPv6的OSPF配置

```
if# ipv6 ospf process area area-id
```

### IPv6的路由选择

基于RFC2740

### 增加2个LSA

Intra-Area-Prefix-LSA及Linki-LSA

### 原来的2个LSA被改名

Summary-LSA被改名为Inter-Area-Prefix-LSA

ASBR-Summary-LSA被改名为INter-Router-Prefix-LSA

### 组播地址变化

ALLSPFRouters ff02::5

DR/BDR ff02:6

## IPv6的路由选择- ospfv3概念及基本配置

### Ipv6的ISIS配置:

ISIS for IPv6基于draft-ietf-isis-ipv6-05.txt

为了挟带IPv6路由信息, 引入2个新的TLV

IPv6 Reachability TLV (0xEC)

IPv6 interface TLV (0xE8)

一个新的网络层协议标识符 (NLPID) (0x8E)

### IPv6的路由选择

BGP-4基于RFC1771

BGP-4 plus 基于RFC2858, BGP-4的多协议扩展

BGP Update中只有以下字段与IPv4具有相关性

next-hop

aggregator

NLRL

**BGP-4 plus 增加了2个属性**

NJP\_UNREACH\_NLRI

### MBGP

它统一有IBGP和EBGP的概念

和eBGP一样, 它是用地址 来做的

router bgp 1

bgp router-id X.x.x.x (如果没有loopback就必须手工配置)

no au

no sy

no bgp default ipv4-unicast 关闭IPv4的单播路由

address-family ipv6

adress-family ipv6 active --激活建立, IPv6邻居必须在地址簇里激活

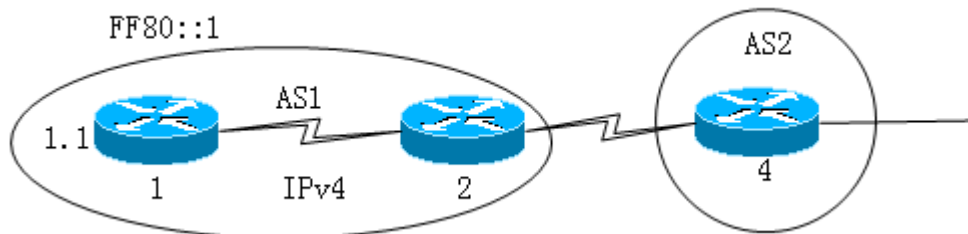
no bgp default ipv4-unicast --关闭IPv4的单播路由, 可打可不打如果打了ipv4的邻居也要在ipv4的地址簇里激活

exit

```
int s0
no sh
clo r 64000
```

```
show bgp sum = show ipv6 bgp sum
```

MBGP中 IPV6 add承载 IPv4的路由



### IPv4和IPv6网络的共存与整合

**双栈协议 (Dual stack)**—网络中的主机，服务器和路由器可以同时使用IPv4和IPv6协议栈

**隧道协议 (tunneling)**—隧道使用孤立的IPv6主机、服务器、路由器和域利用现有的IPv4基础设施与其他IPv6网络通信，（隧道头和未要是双栈）

**协议转换**—使用IPv6网络上的IPv6单协议网路节点与 IPv4网路上的IPv4当协议网路节点进行相互转换

### IPv4到IPv6的转换

#### 1、双栈 (DUAL-stack)

在一台设备上即运行IPv4又允许IPv6

#### 2、Tunneling

不能link-local地址来建立ipv6的tunnel

必须用一个ipv6的全局地址来建立tunnel

可以用GRE来做，GRE可以放在一些IPv4

以前有：

GRE、MPLS、IP（基于链路）

native ip over data link layers

ATM PVC

dWDM Lambda

dWDM Lambda  
Frame-relay PVC  
serial  
Sonet/SDH  
Ethernet

现在有了些新的方法

- 6to4（基于包）
- 自动配置（Automatic tunnels using IPv4）
- ISATAP（主机对主机）
- compatible ipv6 address

### GRE tunnel

```
int tunnel 0
ipv6 enable
ipv6 address 3ffe:b00::3/128
tunnel source 192.168.99.1
tunnel destination 192.168.30.1
tunnel mode gre ipv6
```

ipv6 to ip的tunnel

```
int tu 0
ipv6 enable
ipv6 address 3ffe:b00::3/128
tunnel source 192.168.99.1
tunnel destination 192.168.30.1
tunnel mode ipv6ip
```

### 6to4的原理

先定义前16bit，然后用将公网ipv4 32bit地址转换成ipv6的后32bit，接着的子网任意

tunnel source写1.1.1.1，destination不写，写了就是静态的了  
那怎么建tunnel？用包来出发。

0101；0101 0202；0202

巧妙的涉及，当源访问这个地址的时候，会把这个地址32位提取簇来，做它的目的。

配置：

```
int tun0
tun s 1.1.1.1
```

```
ipv6 add 2002:10101::1
tun mode ipv6 ip
```

### 要设计ipv6

#### 6to4隧道

1、要根据tunnel source来改造我的内部的ipv6的网路  
前16bit要一样（所有需要建立动态tunnel的ipv6网路前16bit都要一样）  
紧跟着16bit的32bit是tunnel source的ipv4地址转换成16进制后来填充。

```
int tunnel 0
tunnel source lo0 (1.1.1.1)
ipv6 unnumbed E0 (2002:101:101::1/64)
tunnel mode ipv6ip 6to4
```

#### 2、建立一条静态路由

```
ipv6 route 相同的16bit/16 走 tunnel 0
```

#### 3、发包

访问2002:202:202::202，首先我看路由表，看到静态路由，走TUNNE口，  
TUNNEL还没有建立起来，把目的地址的17到48位（即32位）提取出来，转换为ipv4地址，然后用以这个地址为tunnel destination，建立tunnel。

就可以发包了

```
show ipv6 tun --察看tunnel
```

### 交换

#### BCMSN知识点

## 交换

### BCMSN知识点

#### 1、网络设计的三层结构

核心层：高速的包交换

分布层：做路由，选录

访问层：只有提高端口密度，用户的接入，用access-list做简单的安全，做VLAN，做账户密码

#### 2、VLAN/TRUNK/VTP

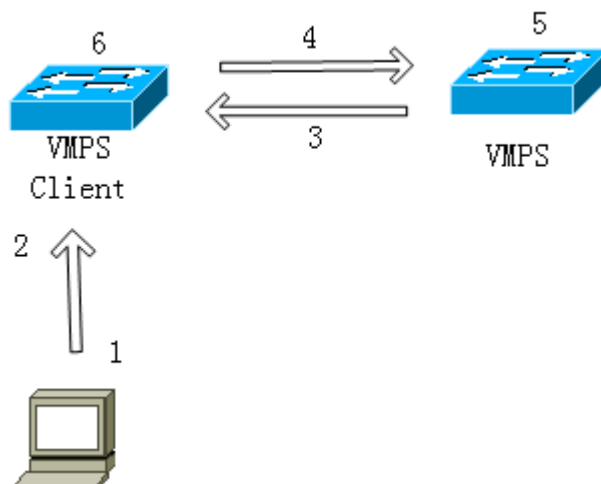
VLAN就是不受地理限制的广播域

把接口划入VLAN的方法：

##### 1、静态 ——基于接口

```
int f0/1
switchport mode access
switchport access vlan 2
```

##### 2、动态 ——基于MAC (VMPS vlan管理策略服务器)



步骤：

- 1 the pc send a frame to the switch
- 2 the VMPS client learns the PC MAC address ont the dynamic port
- 3 the VMPS client sends a VQP request to the VMPS.the request:contains the VMPS client: ip address, the PC MAC address, the PC port number, and the VTP Domain.
- 4 the VMPS parses its database file for PC VLAN assignment
- 5 the VMPS sends a VQP response to the VMPS client
- 6 if the VQP response contains a VLAN assignment, the VMPS client assigns it to the VLAN. Otherwise, it denies the PC access.

测试MAC地址表 :MAC地址老化时间5分钟

静态指定mac地址到某个vlan: #mac-address-table static H.H.H

vlan vlannumber int f0/X

查看: sh mac-address-table int f0/1

在路由器上将某个地址放入arp表中

#arp x.x.x.x H.H.H arpa

如果在VMTP server里找不到这个机器MAC映射, 则禁止它访问

配置:

1、配置VMPA服务器在哪

#vmps server x.x.x.x

if#sw acc vlan vlannumber

2、接口划入动态vlan

if#sw mo ac

if#sw ac vlan vlannumber

如果在VMTP server里找不到这个机器MAC映射, 则禁止它访问

Trunk

cisco建议Trunk为Point-to-Point, 不建议为共享链路

Trunk是2层的概念, 一条链路上可用承载多条Thunk, Trunk一般放在访问层和发布层, 核心层跑路由

二层接口的几种模式:(必须在Trunk上输入)

access:

trunk:

nonegotiate:

dynamic desirable: (默认)

dynamic auto:

查看: shwo int f0/1 trunk

show

DTP(动态trunk协议):

| on主动发DTP帧        | auto    | desirable主动发DTP帧 | nonegotiate |
|------------------|---------|------------------|-------------|
| on主动发DTP帧        | 可以trunk | 可以trunk          | 可以trunk     |
| auto被动发DTP帧      | 可以trunk | 不可以trunk         | 不可以trunk    |
| desirable主动发DTP帧 | 可以trunk | 可以trunk          | 不可以trunk    |
| nonegotiate      | 可以trunk | 不可以trunk         | 不可以trunk    |

|     |          |          |          |          |
|-----|----------|----------|----------|----------|
| off | 不可以trunk | 不可以trunk | 不可以trunk | 不可以trunk |
|-----|----------|----------|----------|----------|

不需要协商最终肯定是Trunk的是on和nonegotiate, auto和desirable是否能形成trunk就需要协商。

只有on和desirable主动发DTP帧。不主动发DTP的是auto, 不发DTP的是nonegotiate。

1. 修改为desirable模式: swl(config-if)#switchport mode denamic desirable

2. 修改为auto模式: swl(config-if)#switchport mode denamic auto

3. 修改为off模式: swl(config-if)#switchport mode access

4. 修改为on模式: a. swl(config-if)#switchport trunk encapsolution dot1q (封装为trunk之前先封装为802.1Q

b. swl(config-if)#switchport mode trunk

5. 修改为nonegotiate模式: a. swl(config-if)#switchport trunk encapsolution dot1q (封装为trunk之前先封装 为802.1Q

b. swl(config-if)#switchport mode trunk

c. swl(config-if)#switchport nonegotiate

前两个会自动协商, 包括封装。后两个不会自动协商, 因此要输入之前先要指明封装。

可以debug dtp 等观察dtp信息。 DBD包每30s发送一次300s超时协商的时候, 既能学到trunk又能学到封装模式

2900 默认isl封装, 不支持??/

ISL是思科私有的协议。在原始数据帧的前面、后面分别加头和尾, 可用由ASIC处理, 效率很高。

ISL头部是26字节, 后面有4字节的ISL的效验

trunk可用学习到对方MAC地址。

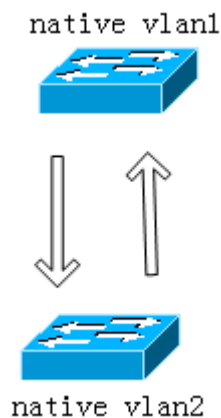
802.1Q是IEEE的标准。

ISL比802.1q要高效, 原因: ISL与802.1Q在封装时都要计算校验。ISL不需要破坏原始数据帧的结构, 则在ISL拆封时, 不需要计算校验。

802.1Q缺点, 破坏了以太帧, 需要重新验证

ISL支持1024个VLAN

802.1Q 两边的native vlan必须要一样。否则CDP会报错, 因为会产生桥接环路



当trunk做冗余时，会block端口，当不同vlan选的根桥不一样的时候，两个vlan block的端口也不一样。

假如上边的交换机native vlan为vlan1，下边的交换机native vlan是vlan2，上面的交换机左边的端口发出一个vlan1的数据帧，这边会误以为是vlan 2的，假如是一个广播帧，那么它会往vlan 2 的端口去返回。vlan2 上面那条链路会往左走，那么就原始以太网帧就发回去，这边收到原始以太网帧会认为是vlan 1的数据帧，然后vlan 1 的数据帧再进行泛洪。

802.1tunnel, 服务器提供商可能为客户提高二层服务。如何区分多个用户。（Q-in-Q）

## VTP

vlan.dat在flash里面

vtp协议的作用, vlan trunk protocol.

advertises VLAN configuration information

Maintains VLAN configuration consistency throughout a common administrative domain

Sends advertisement on trunk ports only

发的是vlan信息，相当于把show vlan发过去了。

这些信息保存在哪的，是flash下面的vlan.dat

运行VTP的条件。

- 1、domain 一致
- 2、trunk端口
- 3、C/s

默认VTP模式为V1

VTP模式: sever、client、Transparent

Transparent:其他VTP不对它产生任何效果，它只复杂传输VTP信息，即

即使两个不同的VTP也能传输。

修订版本好：大的覆盖小的(`configuration Revision`)

#### VTP有三种报文类型：

summary:由server发过来的

subset:子设置报文，主要client去同步。

request:负责请求vtp的信息。

VTP域名不一样，trunk协商会失败。

DTP中也包括vtp的信息

#### VTP修剪：

VTP修剪为了增大可用带宽，可用减少不必要的泛洪流量。

#show vtp

在VTP的prune信息里面，包括哪个接口包括哪个vlan。

VTP版本1版本2的区别：

### 网桥与交换机

为了打破冗余的桥接环路而出现的一种技术

#### 两个交换机有出现两个端口互相连接时

一、TTL是经过一台路由器才会减小

二、多帧拷贝

如果300S没有收到MAC地址的时候，就删除MAC地址。MAC地址有可能是空的。

三、MAC地址的学习，会出现MAC地址的翻动

交换机中，最小的MAC地址是???

### 3、Spanning-Tree Protocol

根桥上所有接口都是指定端口

CISCO有个feature，可用对广播风暴进行一个控制。叫storm-control

破坏环路的方法：

1、物理破坏

2、

发送报文：BPDU(桥接协议传输单元)

生成树只有一个树根：root bridge

One root bridge per network

One root prot per nonroot bridge

One designated prot per segment

Nondesignated prots are blocked

根桥的都是

RP是per交换机。DP是per segment

BPDU的参数：

1、在生成树里用来标识网桥：Bridge ID

8个字节：2个字节Bridge Priority最大值65535 默认32768 0x8000

6个字节 每个交换机的base mac地址 （是交换机端口）

STP的优先级一定要是4096的倍数( $2^{12}$ )

2、path cost

| Link Speed | Cost (Revised IEEE Spec) | Cost (Previous IEEE Spec) |
|------------|--------------------------|---------------------------|
| 10G        | 2                        | 1                         |
| 1G         | 4                        | 1                         |
| 100M       | 19                       | 10                        |
| 10M        | 100                      | 100                       |

整个BPDU的格式（越小越优先）

| Bytes | Field                                                         |
|-------|---------------------------------------------------------------|
| 2     | Protocol ID :0（代表上层协议是BPDU）1                                  |
| 1     | Version                                                       |
| 1     | Message Type:(2种:a.configuration bpdu b.TCN BPDU)             |
| 1     | Flags:(在802.1D里面，有8位，只有前后2位是使用的，前面一位叫TCA，一个上TC，用于拓扑变          |
| 8     | Root id:                                                      |
| 4     | Cost of path: 交换机收到BPDU的时候，保存到电端口的状态里面，累加path cost。发的时候不加cost |
| 8     | Bridge id: 谁发的BPDU的老化                                         |
| 2     | Prot id: 发送方的port id（0x80=128 0xXX）                           |
| 2     | Message age:                                                  |
| 2     | Forward Delay                                                 |

所有的交换机都要跟根桥的计时器保持一致

生成树是如何进行选举，如何去进行收敛的

Blocking

(loss of BPDU)

**生成树的状态:**

**block (20s):** 生成树如何打破网桥环路呢, 是将一个端口给阻塞。这个端口对用户数据是不敏感的, 很迟钝的 (不发也不收)

**Forward(15s):** 转发状态, 即发也收

**learning(15):** 对用户的数据帧不做处理, 不转发, 可用学习MAC地址。

**forwarding:** 可以对用户数据根据mac地址表, 进行转发。

交换机刚启动好, 需要经过多长时间?

端口在block状态下, 是做什么的

不处理用户数据, 但可以处理BPDU。接收BPDU, 不发送BPDU。

**生成树的选举原则:**

Lowest root BID

(该指定根桥)

Lowest path cost to root bridge (入的时候加) (改根端口的 spanning-tree cost)

(改指定端口)

SW(config-if)#spanning-tree cost

Lowest sender BID

mac+优先级

Lowest port ID

接口号+优先级128 (默认) /16递增

交换机的 端口都要保存能够发出或收到的最好的BPDU。

**生成树的选举:**

- 1、根桥
- 2、根端口

这个端口属于哪个VLAN, 就会在哪个VLAN里面选举根端口和指定端口

交换机并不知道根桥在哪里, 也不知道指定端口在哪里

## PVST

从流量的地方考虑, 来一个负载均衡。

最常用的方法, 调整根桥。

PVST提供了一定的灵活性。但并不能知道所有vlan的刘老师多少。并不能做到完全的负载均衡。

能做到完全的负载均衡。

一个PVST只能运行64个生成树。

$\text{ext-priority} = \text{vlan} + \text{ID}$

新的IOS的prot-id跟端口号一样，  
有没有可能一个交换机收到的prot-id一样。

self-loop,

不建议。

- 4、多层交换
- 5、高可用性、
- 6、组播
- 7、QoS

如果断了后，就让整个拓扑知道。  
发送TCN

发送TCN的情况

- 1、当一个交换机的端口被选为block端口时
- 2、当一个block被选为根端口或指定端口时
- 3、当一个block进入转发状态的时候发TCN

发TCN往根端口发。发到一个交换机时，肯定是指定网桥。确认TCA。发送下一个BPDU的时候TCA会置位。  
不断的传递到根桥。

为什么会有TCN的存在。

35S后能保证mac地址表要么是空的要么是正确。

命令解析：

vmps server 10.1.1.1 把这台交换机成为VMPS

mac - static XXXX(MAC) vlan 1 int f0/1 --这个MAC地址的接口放入VLAN 1

```
show mac-address-table
Q-in-Q(802.1Q-in-802.1Q)
 if#showchport mode dot1q tunnel
 showchport access vlan 30
delete flash:vlan.dat --删除vlan
show VTP status --查看VTP信息
show vtp counters --查看发送了多少条vtp信息
switchport trunk pruning vlan add/remote/none/except 2-10,12-1001
--修剪vlan(增加或减少)
if#storm-control broadcast level 30(百分比) --当标识广播到了带宽
的30%
if#storm-control action shutdown(出现广播风暴时shutdown端口)
show storm-control
spanning-tree vlan 1 priority 0/4096/8192
show int | in bia(所有接口的地址)
show spanning-tree vlan 1 --看vlan1的生成树
show spanning-tree vlan 1 int f01/1 detail --查看端口BPDU信息
spanning-tree cost bridge --改变端口的cost(改变端口的spanning-
tree cost)
spanning-tree mst configuration --定义mst的名字
instance 1(0~15个组) vlan 20-30 --把一组VLSN放入
spanning-tree mst 1-2,4 priority --改变这些组优先级

spanning-tree vlan 1 root
spanning-tree vlan 10 root primary --比root减少两个4096
spanning-tree
if#spanning-tree cost XX --改所有VLAN的cost
if#spanning-tree costvlan vlannumber cost XX --改单个VLAN的cost

spanning-tree vlan 10 ?
spanning-tree vlan 10 root spanning diameter --定义生成树的直径。
(从根到末节经过多少次交换机 默认7) 在根桥上改。
debug spanning-tree events
errdisable reconvery interval --(默认300S)
spanning-tree bpduguard enable --一收到BPDU包就本接口就会DOWN
debug spanning-tree uplinkfast
spanning-tree uplinkfast mas-update-rate
```

## 生成树的收敛

PVST

**MST:** 多个VLAN使用同一个生成树

1)

Port—Fast

BPDU guard

if#spanning-tree bpduguard enable/disable

此接口不接收BPDU。如果收到了BPDU，就将接口变成

uplink fast

访问层交换机的RP丢失后，另外一个端口会变成RP，经过

## 攻击课

**EIGRP的建立条件是什么？**

- 1、K值
- 2、AS号

**IGP的选路条件**

1. 下一跳可达
2. 最长匹配
3. AD值
4. METRIC值

**解决区域0分割的方法**

VL  
TUNNEL  
多进程重分布

**静态路由的AD**

默认：  
next hop是1  
接自己接口为0

接自己接口为0

#### 路由器修改MTU

串口:

mtu...

ip mtu...

以太口只能输入

ip mtu

#### OSPF默认路由的方法

1. 写静态, network
2. ip default network
3. 汇总
4. default info...

#### RIP和EIGRP的neighbor有什么不同

RIP可以单边指neighbor

EIGRP要两边互指neighbor

#### BGP的network有什么含义

1. 把路由条目引入BGP
2. 告诉对方到这条路由可以走我

#### OSPF过滤LSA

#### BGP keepalive时间间隔

60秒

#### EIGRP汇总方法?

- 1 auto
- 2 接口
- 3 进程

#### RIP汇总方法

1. VER1 没有
2. 接口: ip summary rip
3. network

#### EIGRP能不能汇总学来的路由?

自动汇总: 不可以

手工汇总: 可以

EIGRP支持什么认证: 只支持密文认证, 不支持明文认证

RIP的计时器： 30 180 180 240 使用timer basic调整

帧中继是怎么工作的？ 交换DLCI

BGP路由收到一跳路由隔多久发给对方？

I 5s E 30s

OSPF能不能重分布默认路由，如果可以，有哪些方法？ 不可以

OSPF的hello间隔时间

OSPF的exstand是干什么用的？

LSA的三个常用的标志位

RIP是单进程的

帧中继如何二层不支持广播？ 静态PVC时不打broadcast ， 关闭反向解析。

**RIP NETWORK静态的条件**

静态是的只能跟接口，不能跟下一跳

zhang

1:ospf邻居建立过程 错

2:虚链路的作用

3:ospf的几种包 zhu cui bi 提醒一次

4:BGP的防环

5:BGP三个表 错

6:BGP neighbour的含义 huan chun sheng 提醒 chenxu 提醒

7:BGP的几种状态 错

8:在什么时候可以关闭同步

9:BGP的antuosumary 错

10:图 错

11: 产生默认路由的方法

12: ospf组播

13: 没有默认路由

14: eigrp建邻居 错

15: IGP选路原则

- 15: IGP选路原则
- 16: 如何去掉neighbour
- 17: lp会不会发LSA
- 18: 解决区域0分割的几种方法 错
- 19: 非等价负载均衡 错, zhangyu来一个

cai

- 1: 交换机改mtu 错
- 2: 路由器mtu 错
- 3: eigrp的secondy地址 错
- 4: 建邻居的必要条件
- 5: RIP认证的密文, 明文有什么不同 错
- 8: ospf的几种包
- 6: PTOMP的两个优点
- 9: 产生默认路由
- 6: 虚链路的几种类型 错
- 7: 特殊区域
- 10: RIP的发送机制 错 huangchunsheng两次
- 11: RIP和EIGRP的passive的区别
- 12: RIP和EIGRP的neighbour
- 13: 什么是不连续子网
- 14: BGP的网络 错
- 15: 普通区域的两个ABR
- 16: MTU的最小值 错
- 17: ospf管理距离

黄

- 1: OSPF过滤LSA 错
- 2: ospf区域的双ABR默认路由 错
- 3: P2MP, NBMA: 的时间间隔
- 4: hello timer修改
- 5: 重分布的机制 错
- 6: BGP的四种包
- 7: BGP的keepalive多久一次 chengjungeng一次
- 8: EIGRP的汇总方法 错
- 9: RIP的汇总
- 10: EIGRP能不能汇总 错
- 11: 图
- 12: EIGRP认证方式 错
- 13: route-map
- 14: ospf起双进程
- 15: ospf邻接后就能学到各自的路由
- 16: 区域认证
- 17: RIP的计时器

18: 静态路由是否能负载均衡

19: 帧中继的包走的原理

罗:

1: BGP邻居

2: AGGAGRESS带community 错

3: IPV6分那几种地址 错

4: MAC地址第7位代表什么

5: ospf重分布默认路由

6: 图 错

7: 无类路由协议 chengjuangeng

8: TC发到交换机的MAC表的存活时间

9: ROUTE-ID

10: IGP类型

11: distri

12: 网络三层

13: 图

14: 图

15: ospf在NBMA的建邻居的过程错

16: 泛洪30分钟

17: RIP的环境下的no autosummary

18: ospf hello包间隔 错

19: 11 错

20: exstart 状态

21: 区域0 分割的TUNNEL

马:

1: 图 错

2: RIP端口号 错

3: ospf

4: ARP 错

5: 错

6: 帧中继

7: 图

8:

9: area range 错

10: IPV6的 错

11: NAT的三个缺点

12: EBGp用环回口建邻居，用环回口宣告进BGP 错

13: 静态路由

14: 交换机的trunk

15:

16:

17: 无类查询

- 18: 有类查询
- 19: EBGp多跳
- 20:

陈:

~~~~~

- 1: 图
- 2: 图
- 3: 图
- 4: DBD包通过什么确认 huang
- 5:
- 6: DR和BDR的选举过程
- 7: RIP是单进程或者多进程
- 8: TOTAL NSSA产生的默认路由是几型
- 9: 存在默认路由, NSSA区域, 一个路由器既是ABR又是ASBR在重分布的时候应该注意的地方
- 10: MA网络, 两个路由器一个DR, 另一个是什么什么状态
- 11: 如果在database表和路由表中都能看到5型但不能看到4型的路由原因
- 12: 图
- 13: 图
- 14: 图
- 15:
- 16: ospf和BGP的router-ID是否必须一致
- 17: 图
- 18:
- 18: BGP路由被优化后能不能放进路由表
- 19: IP包头
- 20: router-ID怎么选择
- 21: 4型的LSA是否都是指向ASBR上的
- 22: 图

陈:

- 1: ospf包在ospf中是以什么方式发送的
- 2: 在帧中继环境下
- 3: 图
- 4: 图
- 5: 图
- 6: 图
- 7: 全球组播地址
- 8: RIPnetwork默认路由
- 9: 图

~~~~~

#### 攻击课

- 1 BGP 的消息类型 以及包含的信息
- 2 ospf 七个过程
- 3 ospf 每个过程交换的包及每一个过程
- 4 igrp eigrp bgp tcp udp rip protocol id the function of protocol id
- 5 rip loop avoidance mechanism
- 6 rip timer
- 7 rip version v1 和version 2的区别
- 8 ospf lsa 类型和作用
- 9 bgp 建立邻居的过程
- 10 igp选路原则
- 11 ospf nssa 区域如何产生默认路由
- 12 ospf社么区域可以作为virtual-link传输区域 为社么? 标准区域具有全路由
- 13 lsa的传输过程
- 14 odr的意思 DEX ON1 ON2
- 15 ospf 建立邻居的时间 hello stub area flag authentication password area id
- 16 bgp 建立邻居的过程 show ip bgp summary
- 17 eigrp 建立邻居的过程
- 18 eigrp 建立邻居的必要条件 as号 k值
- 19 the difference of ppp() and hdlc(cisco proprietary)
- 20 ospf 组播更新的地址
- 21 vpn 是什么?
- 22 所有的管理距离
- 23 ppp lcp 能够实现那些功能
- 24 fram-relay 可以支持几条dlci 号? 1024 16-1007
- 25 isdn d信道标准
- 26 简述SIA及解决方法
- 27 isis nsap 地址格式
- 28 isis 和ospf 的异同点
- 29 bgp 每塌多长时间发送更新60s
- 30 bgp 属性
- 31 isis 支持的网络类型
- 32 ospf 社么情况下需要VIRTUAL-LINK 解决方法 tunnel 和多进程
- 33 VIRTUAL-LINK的网络类型
- 34 ospf 通过virtual-link建立对等体关系吗? (是的)
- 35 WAN的交换类型 (3个) isdn frame-relay
- 36 atm 同步和异步 区别
- 37 eigrp 如何保证他的可靠传输? (eigrp 如果没有在平均回程时间内没有受到ACK包就

- 重传 重传16次如果再收不到就放弃)
- 38 frame-relay 的参数 cir bc be de
- 39 isis向其余路由协议 重分发时要注意的问题（重分发直连）
- 40 239.112.23.166 写出对应的mac 地址01 00 5E17 16
- 41 eigrp 和ospf 所有时间间隔 ospf 中hello 包作用（发现 保持 建立 选举DR BDR）
- 42 实验 hub and spoke 中hub 配 p-to-m spoke 上配point-to-p 之后配hello-interval 结果是社么？
- 43 STP协议选择根网桥的过程
- 44 ospf 网络类型
- 45 stp 算法的状态 以及时间
- 46 二层如何隔离广播？（划VLAN）
- 47 dialer-watch 配置的过程 机制
- 48 ppp 验证过程中chap的验证过程
- 49 ospf 各个区域的lsa 类型
- 50 frame-relay 管理接口的类型
- 51 point-to-p frame-relay 模式 一面down 则另一面会down 吗？不会是up up 因为frame-relay switch没有down
- 52 rip 和ospf 触发更新有社么区别？ ospf是增量更新rip是全量更新
- 53 rip V1能不能传输子网信息？连续的情况下是可以的
- 54 3550上如何绑定mac地址？（switchport port-security mac-address xxx ）
- 56 ospf 选举DR BDR的过程
- 57 列举虚拟接口 （LOOPBACK 0 DIALER 0 TUNNEL 0 INYERFACE ASYNC 0 INTER VLAN 1 ）
- 58 动态NAT是不是一定要配在接口上？
- 59 cbwfq的配置步骤
- 60 eigrp as 号范围1-65535
- 61 lmi的操作过程
- 62 AAA的协议以及区别
- 63 odr 实现的条件
- 64 组播则样防止环路？（单播路径数决定发送接口）
- 65 rip bgp水平分割的区别
- 66 lsa四号类型是不是一定指向asbr的？不一定 如nssa 区域的abr
  
- 67 frame-relay的虚电路的三种状态？
- 68 一个自治系统写next-hop-self起作用吗？没有作用next-hop-self指的是不同自治系统之间的
- 69 pagp 和lagp 的区别
- 70 custom queueing 和priority queueing的区别
- 71 cisco产品支持几种队列？ llq pq wfq cbwfq cq
- 72 backbonefast and uplinkfast portfast
- 73 ethernet channel 配置
- 74 vtp三种模式
- 75 ospf 公告所有的路由给所有路由器命令 default-information

originate  
76 3550 配置路由协议则么配?  
77 hsrp 配置standby 1 ip  
                    standby priority  
                    standby 1 preempt  
                    standby 1 track s0  
78 ospf 中router-id 的选举 lo0 不在ospf内也选  
79 rstp的BPDU格式  
80 QoS服务类型 best-effort differetiated integrated  
81 cisco支持多少个vlan ,其中可以使用的多少?  
82 modem的信令标准  
83 BGP团队属性

## 组播

组播

**Why Multicast ?**

单播: 不能同时收到

组播: 可以同时收到

组播最大的优点, 解决数据的流量, 减少server负担

TCP只能是p2p单播

**组播路由协议分类:**

密集模式协议 (DVMRP和PIM-DM)

稀疏模式协议 (**PIM-SM**和CBT)

链路状态协议 (MOSPF)

1. 完全不支持组播的设备

## 2. 只发送而不接收组播

## 3. 即发送也接收组播

组成员：基于接口

**224.0.0.1**：所有支持组播的设备都会这个地址

一个接口可同时属于多个组

不但主机而且路由器的接口也是这个组的成员

在组播里面，把目标地址改为D类地址，源地址还是原来的地址

### 地址分类：

#### Reserved Link-Local Address：

-224.0.0.0 - 224.0.0.255

公网的组播地址

TTL=1

224.0.0.1 所有支持组播的设备都会监听这个地址

224.0.0.2 所有支持组播的网关设备(路由器)都要监听这个地址

224.0.0.5 OSPF的组播地址

224.0.0.13 PIMv2 Routers

224.0.0.22 IGMPv3

#### Other Reserved Addresses

224.0.1.0 - 224.0.1.255

Notlocal in scope (Transmitted with TTL>1)

examples:

224.0.1.1 NTP Network Time Protocol

225.0.2.32 Mtrace routers

224.0.1.78 Tibco Multicast1

#### Administratively Scoped Addresses(私有地址)

239.0.0.0 - 239.255.255.255

#### SSM(Source Specific Multicast) Range

232.0.0.0 232.255.255.255

Primarily targeted for Internet style Broadcast

FDDI不建议使用组播

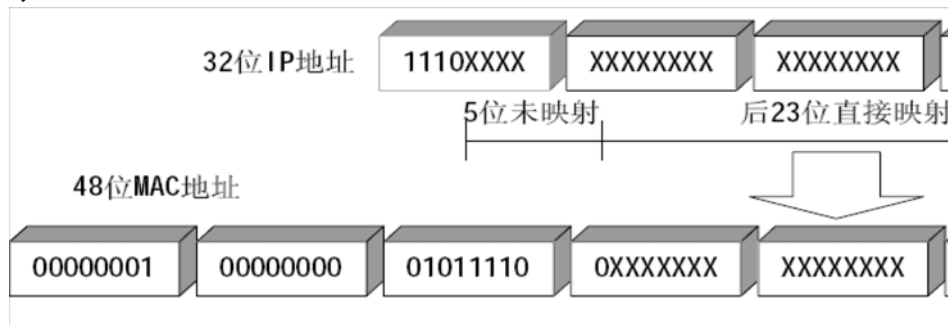
组播映射的是MAC的后23位

### IP 组播地址到 MAC 地址的映射

IANA 将 MAC 地址范围 01:00:5E:00:00:00 ~ 01:00:5E:7F:FF:FF 分配给组播使用

用这就要求将28位的 IP 组播地址空间映射到 23 位的 MAC 地址空间中具体

的映射方法是将组播地址中的低 23 位放入 MAC 地址的低 23 位如下图所示。



由于 IP 组播地址的后 28 位中只有 23 位被映射到 MAC 地址这样会有 32 个 IP 组播地址映射到同一 MAC 地址上

#### MAC映射计算:

239.255.0.1 => 1111 1111 0000 0000 0000 0001

从25位开始 => 0111 1111 0000 0000 0000 0001 => 7f 00 01  
01-00-5e-7f-00-01

### How are Multicast Addresses Assigned?

#### Host-Router Singaling:IGMP

RFC 1112 specifies version 1 of IGMP

RFC 2236 specifies version 2 of IGMP

RFC 3376 specifies version 3 of IGMP

IGMPv1:当没有报告收到的时候（3倍的报表时间大约3分钟）则认为已经离开了成员组

IGMPv2:当最好一个组成员离开时会向路由器发一个离开信息

经离开了成员组

IGMPv2:当最好一个组成员离开时会向路由器发一个离开信息

IGMPv3会定期发送Report报告

## **Multicast Distribution Trees**

### **Shortest Path or Source Distribution Tree**

Shared Distribution Tree

RP:PIM Rendezvous Point

Characteristics of Distribution Trees

**RPF:Reverse Path Forwarding (发向路径)**

RPF Check Fails

检测源，如果进来的地址是其他接口进来的地址，则不转发

RFP Checking:

Types of Multicast Protocols

任务:

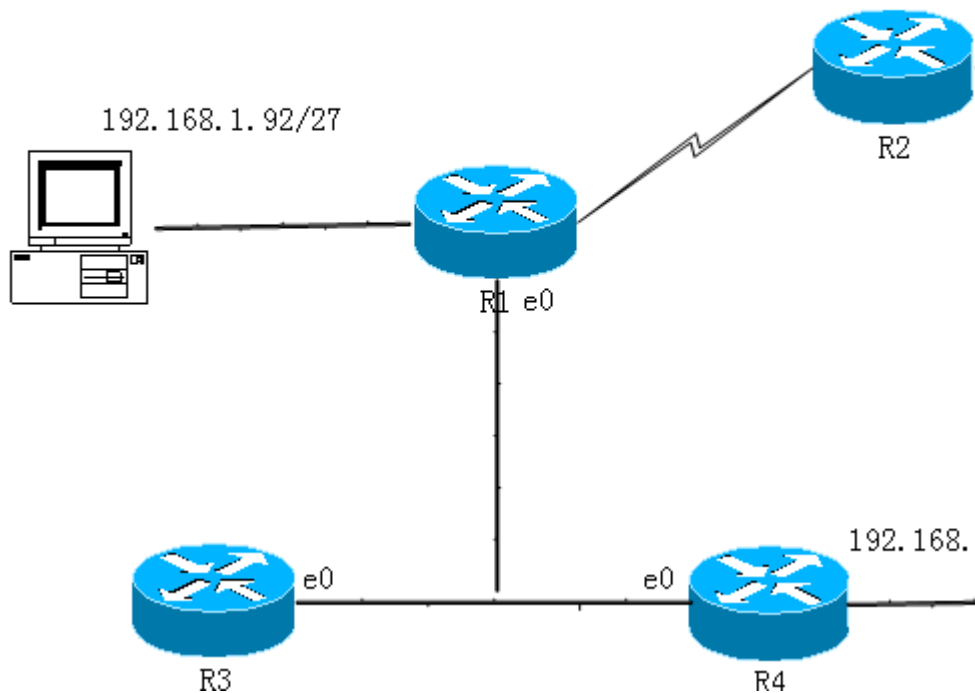
看IGMP

**NP+lab**

## NP+lab

## 4.3

## RIP



```
R1:interface e0 ip ad 10.1.1.1 255.255.255.0
```

```
R2:interface e0 ip ad 10.1.1.2 255.255.255.0
```

```
R3:interface e0 ip ad 10.1.1.3 255.255.255.0
```

1、R3不能向R4通告路由更新;

```
RACK16R1(config-router)#passive-interface ethernet 0 (e0成为被动接口, 只收不发路由条目)
```

```
RACK16R1(config-router)#neighbor 10.1.1.1 (指R1为邻居, 使用单播发路由条目)
```

2、R1是连接到Internet的路由器, 在R1上面通过RIP向内网注入一条缺省路由;

- RACK16R1(config-router)#default-information originate (向被宣告进RIP中的所有接口重分发一条缺省默认路由 0.0.0.0/0 [120/1] via 10.1.1.1, 00:00:04, Ethernet0)

- RACK16R1(config)#interface lo2

```
RACK16R1(config-if)# ip ad 12.0.0.1 255.255.255.0
```

```
RACK16R1(config)#ip default-network 12.0.0.0
```

3、修改R4的RIP计时器时间Update interval为5s; Invalid为10s; Holddown为20s; 但要求路由在没有收到Update后的15s就被删除;

- RACK16R4(config-router)#timers basic 5 10 20 15 (1. 发送路由更新的时间 2. 没有在接口上接收到路由更新后此路由条目在路由表中生存的时间 3. 本路由器接收到同路由表中存在相同的条目, 并且接收到的这个路由条目的跳数大于路由表中的条目, 此时这个时间产生, 20秒后, 这个路由条目被抑制 4. 在超过这个时间之前, invalid timer也超时了, 这个时候此条路由条目标记为不可达, 并且此条路由在路由表中被删除)
- 默认的时间阈值为: Sending updates every 30 , Invalid after 180 seconds, hold down 180, flushed after 240

4、在删除R1上发送出来的缺省路由后; R1与R4之间分别连接主机H1和H2 (用环回口代替); 发现不能相互访问对方, 请解决;

```
• router rip
 network 10.0.0.0
 network 192.168.1.0
 distribute-list 1 in Ethernet0
```

```
access-list 1 deny 0.0.0.0
access-list 1 permit any
```

是R4删除还是在R1删除呢?? 在R4上对R1发来的缺省默认路由进行过滤??

5、R1的S0口上级联一个RIP邻居, 要求仅在R1的S0接口上面发送和接收RIPv2的更新, 并且要求使用MD5加密验证; 密码为“WOLF”配置成功的话, 你应该从R2上收到一条路由 (199.172. x. 0) R1向R2发送 10网段的 明细路由; 要求R1R3R4能Ping通这些路由的IP地址 (例: 199.172. 1. 254)。

```
R1(config)#key chain wolf
 # key 1
 # key-string wolf
```

```
R1(config)#interface s0
R1 (config-if)#ip rip authentication mode md5
 ip rip authentication key-chain wolf
```

6、你不能用任何手段登陆到R2上查询或修改配置, 但要求在R1上配置, 能且只能允许R2成功直接TELNET到R1的“enable”模式 (即无需输入密码和输入enable命令), R2使用2.2.2.2 这个地址发出TELNET (10分); 其他路由器的访问将被拒绝 (10分)

```
no login
```

```
no login
privilege15
```

7、R3收到R1发送过来的199.172.x.0/24 网段路由， 要求奇数路由在路由表里面metric为10  
(5分)

R2初始配置

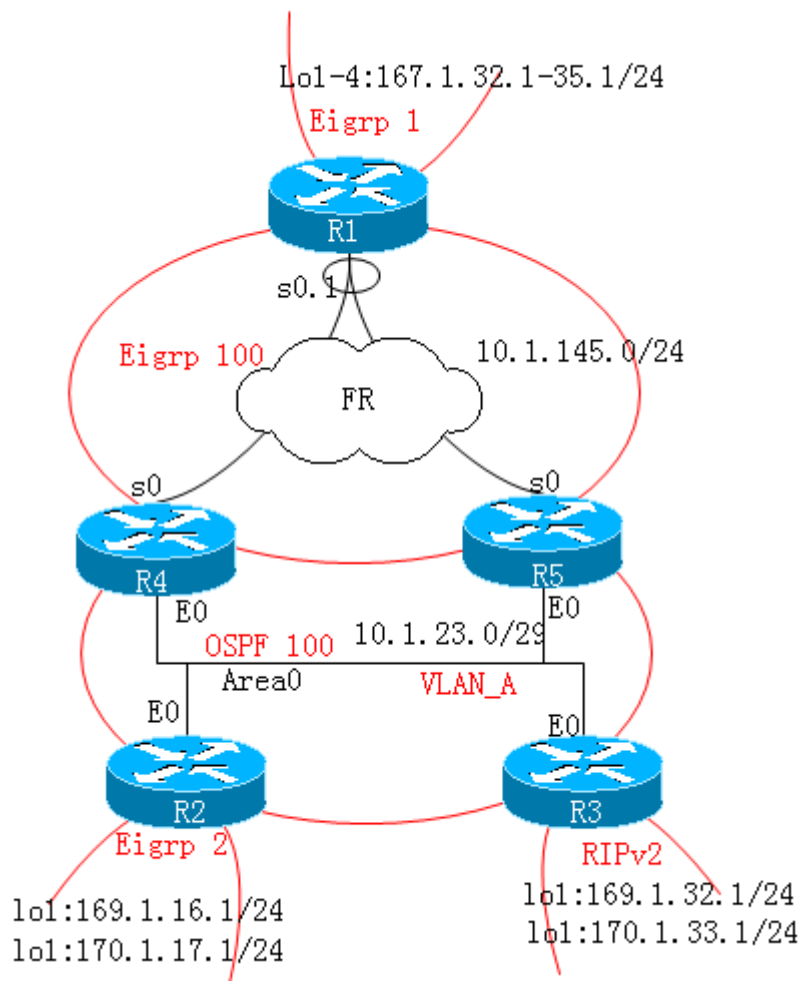
```
key chain wolf
key 1
key-string wolf
interface Loopback0
ip address 2.2.2.2 255.255.255.255
interface Loopback1
ip address 199.172.2.254 255.255.255.0
interface Loopback2
ip address 199.172.3.254 255.255.255.0
interface Loopback3
ip address 199.172.4.254 255.255.255.0
interface Loopback4
ip address 199.172.1.254 255.255.255.0
interface Serial0
ip address 150.100.12.2 255.255.255.0
ip rip authentication mode md5
ip rip authentication key-chain wolf
clockrate 64000
no shutdown
router rip
version 2
network 2.0.0.0
network 150.100.0.0
network 199.172.1.0
network 199.172.2.0
network 199.172.3.0
network 199.172.4.0
```

注：只能用默认的rip版本

注：也不能使用任何形式的静态路由和默认路由

#### 4.4

## EIGRP



一：桥接

帧中继中R1的S0接口只用子接口（仅使用图中所提供的DLCI）

（仅使用图中所提供的DLCI）由这个决定关闭frame-relay inverse-arp

1：主接口关闭

2：子接口关闭

运用多点子接口

```
RACK16R1(config-subif)#frame-relay map ip 10.1.105.4 104 broadcast
```

```
RACK16R1(config-subif)# frame-relay map ip 10.1.105.5 105
```

```
broadcast
```

配置帧中继交换机

1： fr sw

2： enc fram

3： fram lmi-type cisco

4： fram intf-ty dce

```
3: fram lmi-type cisco
4: fram intf-type dce
5: clo ra 2000000 （同步?? 按一定的时间发数据包，不确定包的大小）
（异步?? 随时发数据包，必须确定包的大小）为什么要打（物理层的DCE
和DTE连线，在物理层的DTE连接的路由器上要配置clo ra 2000000）
1: 一层的DCE?? clo ra 200000
2: 二层的DCE?? fram intf-type dce
二: rip
R3的lo1, lo2运行RIP v2, RIP和ospf做双向重分发。
打上no auto-summary
三: eigrp
eigrp 1
R1上起4个环回口
lo1: 167.1.32.1/24
lo2: 167.1.33.1/24
lo3: 167.1.34.1/24
lo3: 167.1.35.1/24
```

宣告在eigrp 1 中

```
eigrp 100
```

R1, R4, R5在eigrp 100中, R1, R4, R5的环回口以及vlan\_A为eigrp 100域内路由（有掩藏需求??

R4上看R1/R5的环回口

```
D 10.1.1.1/32 [90/256000] via 10.1.145.1 ,00:01:47, serial 0
D 10.1.5.5/32 [90/256000] via 10.1.145.5 ,00:01:47, serial 0
```

R5 上看R1/R4的环回口

```
D 10.1.4.4/32 [90/256000] via 10.1.145.4 ,00:01:47, serial 0
D 10.1.1.1/32 [90/256000] via 10.1.145.1 ,00:01:47, serial 0
```

R1上看R4/R5 的环回口

```
D 10.1.5.5/32 [90/256000] via 10.1.145.5 ,00:01:47, serial 0
D 10.1.1.1/32 [90/256000] via 10.1.145.4 ,00:01:47, serial 0
```

R4, R5收到167.1.32.0/22 的一条汇总路由, eigrp 1向lo1只发送一条10.1.0.0/16的路由, 向其他接口公告明细的路由信息。（eigrp 100 重分布的时候, 在lo1 下做汇总）

重分布进来后, 然后在R1的s0.1接口上做汇总。

```
eigrp 2
```

R2的lo1, lo2口在eigrp 2中, eigrp 2与ospf 双向重分发。

四: ospf

R2/R3/R4/R5运行ospf, R3/R4/R5之间都只能形成two-way状态, 当vlan\_A中再加入一台新设备时, R2还是DR。

重分布进来后，然后在R1的s0.1接口上做汇总。

eigrp 2

R2的lo1，lo2口在eigrp 2中，eigrp 2与ospf 双向重分发。

四：ospf

R2/R3/R4/R5运行ospf，R3/R4/R5之间都只能形成two-way状态，当vlan\_A中再加入一台新设备时，R2还是DR。

R2，R3只看到167网段和10.1.145.0网段负载均衡，R2访问R1的环回口正常情况下是走R4，当R2/R4间链路断掉了后走R5，R3访问R1的环回口走R5，当R3/R5之间断之后，就丢包。

五：过滤

eigrp 1 和 eigrp 2 不接受RIP的路由，RIP也不接受他们的路由（打tag或者route source）

注：要求全网互通，所有lp接口的地址为10.1.X.X/32（X为路由器号），本试验主网段为10.1.0.0/16。不允许出现静态路由。

注：子接口关闭no ip split-horizon ei 100

帧中继HUB-SPOKEN跑RIP的情况（指neighbor）

帧中继HUB-SPOKEN跑eigrp的情况

帧中继HUB-SPOKEN跑ospf的情况

distance xx（修改DX的）

distance eigrp（修改D）

## 协议层

```
R1-----
hostname R1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip address 167.1.32.1 255.255.255.0
```

```
!
interface Loopback2
 ip address 167.1.33.1 255.255.255.0
!
interface Loopback3
 ip address 167.1.34.1 255.255.255.0
!
interface Loopback4
 ip address 167.1.35.1 255.255.255.0
!
interface Ethernet0
 no ip address
 no ip mroute-cache
 shutdown
!
interface Serial0
 no ip address
 encapsulation frame-relay
 no ip mroute-cache
 no frame-relay inverse-arp
!
interface Serial0.1 multipoint
 ip address 100.1.145.1 255.255.255.0
 frame-relay map ip 100.1.145.4 104 broadcast
 frame-relay map ip 100.1.145.5 105 broadcast
 no frame-relay inverse-arp
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 1
 network 167.1.32.0 0.0.3.255
 no auto-summary
!
router eigrp 100
 network 10.1.1.1 0.0.0.0
 network 100.1.145.0 0.0.0.255
 neighbor 100.1.145.5 Serial0.1
 neighbor 100.1.145.4 Serial0.1
 metric weights 0 1 0 0 0 0
 distance 80 0.0.0.0 255.255.255.255 45
 no auto-summary
!
no ip http server
ip classless
!
!
```

```
!
!
access-list 45 permit 10.1.4.4
access-list 45 permit 10.1.5.5
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

```
R2-----
hostname R2
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
frame-relay switching
!
!
!
interface Loopback0
 ip address 10.1.2.2 255.255.255.255
!
interface Loopback1
 ip address 169.1.16.1 255.255.255.0
!
interface Loopback2
 ip address 170.1.17.1 255.255.255.0
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.2
 tunnel destination 10.1.23.3
!
interface Ethernet0
 ip address 10.1.23.2 255.255.255.248
 no ip route-cache
 ip ospf priority 2
 no ip mroute-cache
!
interface Serial0
 no ip address
```

```
no ip address
encapsulation frame-relay
no ip route-cache
no ip mroute-cache
clockrate 2000000
frame-relay intf-type dce
frame-relay route 104 interface Serial1 401
frame-relay route 105 interface Tunnel0 555
!
interface Serial1
no ip address
encapsulation frame-relay
no ip route-cache
no ip mroute-cache
clockrate 2000000
frame-relay intf-type dce
frame-relay route 401 interface Serial0 104
!
router eigrp 2
network 169.1.16.0 0.0.0.255
network 170.1.17.0 0.0.0.255
no auto-summary
!
router ospf 1
router-id 10.1.2.2
log-adjacency-changes
network 10.1.2.2 0.0.0.0 area 0
network 10.1.23.0 0.0.0.7 area 0
!
no ip http server
ip classless
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

```
R3-----
hostname R3
!
!
ip subnet-zero
no ip domain-lookup
!
frame-relay switching
```

```
frame-relay switching
!
!
interface Loopback0
 ip address 10.1.3.3 255.255.255.255
!
interface Loopback1
 ip address 169.1.32.1 255.255.255.0
!
interface Loopback2
 ip address 170.1.33.1 255.255.255.0
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.3
 tunnel destination 10.1.23.2
!
interface Ethernet0
 ip address 10.1.23.3 255.255.255.248
 no ip route-cache
 ip ospf priority 0
 no ip mroute-cache
!
interface Serial0
 no ip address
 no ip route-cache
 no ip mroute-cache
!
interface Serial1
 no ip address
 encapsulation frame-relay
 no ip route-cache
 no ip mroute-cache
 clockrate 2000000
 frame-relay intf-type dce
 frame-relay route 501 interface Tunnel0 555
!
router ospf 1
 router-id 10.1.3.3
 log-adjacency-changes
 network 10.1.3.3 0.0.0.0 area 0
 network 10.1.23.0 0.0.0.255 area 0
!
router rip
 version 2
 passive-interface default
 network 167.1.0.0
 network 170.1.0.0
```

```
network 170.1.0.0
no auto-summary
!
ip classless
no ip http server
ip pim bidir-enable
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

R4-----

```
hostname R4
!
!
ip subnet-zero
no ip domain-lookup
!
!
!
interface Loopback0
ip address 10.1.4.4 255.255.255.255
!
interface Ethernet0
ip address 10.1.23.4 255.255.255.248
ip ospf priority 0
no ip mroute-cache
!
interface Serial0
bandwidth 10000
ip address 100.1.145.4 255.255.255.0
encapsulation frame-relay
no ip mroute-cache
frame-relay map ip 100.1.145.1 401 broadcast
frame-relay map ip 100.1.145.5 401 broadcast
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
```

```
!
router eigrp 100
 passive-interface Ethernet0
 network 10.1.4.4 0.0.0.0
 network 10.1.23.4 0.0.0.0
 network 100.1.145.0 0.0.0.255
 neighbor 100.1.145.5 Serial0
 neighbor 100.1.145.1 Serial0
 metric weights 0 1 0 0 0 0
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 10.1.4.4
 log-adjacency-changes
 network 10.1.4.4 0.0.0.0 area 0
 network 10.1.23.0 0.0.0.7 area 0
!
ip classless
no ip http server
ip pim bidir-enable
!
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

R5-----

```
hostname R5
!
logging rate-limit console 10 except errors
!
ip subnet-zero
no ip finger
no ip domain-lookup
!
no ip dhcp-client network-discovery
!
!
!
interface Loopback0
```

```
interface Loopback0
 ip address 10.1.5.5 255.255.255.255
!
interface Ethernet0
 ip address 10.1.23.5 255.255.255.248
 ip ospf priority 0
 no ip mroute-cache
!
interface Serial0
 bandwidth 10000
 ip address 100.1.145.5 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 100.1.145.1 501 broadcast
 frame-relay map ip 100.1.145.4 501 broadcast
 no frame-relay inverse-arp
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 100
 passive-interface Ethernet0
 network 10.1.5.5 0.0.0.0
 network 10.1.23.5 0.0.0.0
 network 100.1.145.0 0.0.0.255
 neighbor 100.1.145.4 Serial0
 neighbor 100.1.145.1 Serial0
 metric weights 0 1 0 0 0 0
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 10.1.5.5
 log-adjacency-changes
 network 10.1.5.5 0.0.0.0 area 0
 network 10.1.23.0 0.0.0.7 area 0
!
ip kerberos source-interface any
ip classless
no ip http server
!
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 transport input none
line aux 0
```

```
line aux 0
line vty 0 4
!
end
```

### 策略层

```
R1-----
hostname R1
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip address 167.1.32.1 255.255.255.0
 ip summary-address eigrp 1 167.1.32.0 255.255.252.0 5
 ip summary-address eigrp 1 10.1.0.0 255.255.0.0 5
!
interface Loopback2
 ip address 167.1.33.1 255.255.255.0
!
interface Loopback3
 ip address 167.1.34.1 255.255.255.0
!
interface Loopback4
 ip address 167.1.35.1 255.255.255.0
!
interface Ethernet0
 no ip address
 no ip mroute-cache
 shutdown
!
interface Serial0
 no ip address
 encapsulation frame-relay
 no ip mroute-cache
 no frame-relay inverse-arp
```

```
no ip mroute-cache
no frame-relay inverse-arp
!
interface Serial0.1 multipoint
ip address 100.1.145.1 255.255.255.0
frame-relay map ip 100.1.145.4 104 broadcast
frame-relay map ip 100.1.145.5 105 broadcast
no frame-relay inverse-arp
!
interface Serial1
no ip address
shutdown
!
router eigrp 1
redistribute eigrp 100 route-map IN
network 167.1.32.0 0.0.3.255
no auto-summary
!
router eigrp 100
redistribute eigrp 1 route-map SUM
network 10.1.1.1 0.0.0.0
network 100.1.145.0 0.0.0.255
neighbor 100.1.145.5 Serial0.1
neighbor 100.1.145.4 Serial0.1
metric weights 0 1 0 0 0 0
distance 80 0.0.0.0 255.255.255.255 45
no auto-summary
!
no ip http server
ip classless
!
!
!
ip prefix-list 1 seq 5 permit 167.1.32.0/22
!
access-list 45 permit 10.1.4.4
access-list 45 permit 10.1.5.5
!
route-map IN deny 10
match tag 80
!
route-map IN permit 20
!
route-map SUM permit 10
match ip address prefix-list 1
set tag 81
!
!
```

```
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
!
end
```

```
R2-----
hostname R2
!
logging queue-limit 100
!
ip subnet-zero
no ip domain lookup
!
frame-relay switching
!
!
!
interface Loopback0
 ip address 10.1.2.2 255.255.255.255
!
interface Loopback1
 ip address 169.1.16.1 255.255.255.0
!
interface Loopback2
 ip address 170.1.17.1 255.255.255.0
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.2
 tunnel destination 10.1.23.3
!
interface Ethernet0
 ip address 10.1.23.2 255.255.255.248
 no ip route-cache
 ip ospf priority 2
 no ip mroute-cache
!
interface Serial0
 no ip address
 encapsulation frame-relay
 no ip route-cache
 no ip mroute-cache
 clockrate 2000000
 frame-relay intf-type dce
```

```
clockrate 2000000
frame-relay intf-type dce
frame-relay route 104 interface Serial1 401
frame-relay route 105 interface Tunnel0 555
!
interface Serial1
no ip address
encapsulation frame-relay
no ip route-cache
no ip mroute-cache
clockrate 2000000
frame-relay intf-type dce
frame-relay route 401 interface Serial0 104
!
router eigrp 2
redistribute ospf 1 metric 1000 100 255 1 1500 route-map IN
network 169.1.16.0 0.0.0.255
network 170.1.17.0 0.0.0.255
no auto-summary
!
router ospf 1
router-id 10.1.2.2
log-adjacency-changes
redistribute eigrp 2 subnets route-map E2
network 10.1.2.2 0.0.0.0 area 0
network 10.1.23.0 0.0.0.7 area 0
!
no ip http server
ip classless
!
!
!
!
route-map IN deny 10
match tag 80
!
route-map IN permit 20
!
route-map E2 permit 10
set tag 82
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
```

```
line vty 0 4
!
end

R3-----
hostname R3
!
!
ip subnet-zero
no ip domain-lookup
!
frame-relay switching
!
!
interface Loopback0
 ip address 10.1.3.3 255.255.255.255
!
interface Loopback1
 ip address 169.1.32.1 255.255.255.0
!
interface Loopback2
 ip address 170.1.33.1 255.255.255.0
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.3
 tunnel destination 10.1.23.2
!
interface Ethernet0
 ip address 10.1.23.3 255.255.255.248
 no ip route-cache
 ip ospf priority 0
 no ip mroute-cache
!
interface Serial0
 no ip address
 no ip route-cache
 no ip mroute-cache
!
interface Serial1
 no ip address
 encapsulation frame-relay
 no ip route-cache
 no ip mroute-cache
 clockrate 2000000
 frame-relay intf-type dce
 frame-relay route 501 interface Tunnel0 555
!
```

```
frame-relay route 501 interface Tunnel0 555
!
router ospf 1
router-id 10.1.3.3
log-adjacency-changes
redistribute rip subnets route-map OUT
network 10.1.3.3 0.0.0.0 area 0
network 10.1.23.0 0.0.0.255 area 0
!
router rip
version 2
redistribute ospf 1 metric 2 route-map IN
passive-interface default
network 167.1.0.0
network 169.1.0.0
network 170.1.0.0
no auto-summary
!
ip local policy route-map PB
ip classless
no ip http server
ip pim bidir-enable
!
access-list 1 permit any
access-list 100 permit ip any 167.1.32.0 0.0.3.255
route-map IN deny 10
match tag 81 82
!
route-map IN permit 20
!
route-map PB permit 10
match ip address 100
set ip next-hop 10.1.23.5
!
route-map OUT permit 10
set tag 80
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end

R4-----
```

```
R4-----
hostname R4
!
!
ip subnet-zero
no ip domain-lookup
!
!
!
interface Loopback0
 ip address 10.1.4.4 255.255.255.255
!
interface Ethernet0
 ip address 10.1.23.4 255.255.255.248
 ip ospf priority 0
 no ip mroute-cache
!
interface Serial0
 bandwidth 10000
 ip address 100.1.145.4 255.255.255.0
 encapsulation frame-relay
 no ip mroute-cache
 frame-relay map ip 100.1.145.1 401 broadcast
 frame-relay map ip 100.1.145.5 401 broadcast
 no frame-relay inverse-arp
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 100
 redistribute ospf 1 metric 1000 100 255 1 1500 route-map
O2E
 passive-interface Ethernet0
 network 10.1.4.4 0.0.0.0
 network 10.1.23.4 0.0.0.0
 network 100.1.145.0 0.0.0.255
 neighbor 100.1.145.5 Serial0
 neighbor 100.1.145.1 Serial0
 metric weights 0 1 0 0 0
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 10.1.4.4
 log-adjacency-changes
 redistribute eigrp 100 subnets route-map E2O
 network 10.1.4.4 0.0.0.0 area 0
```

```
redistribute eigrp 100 subnets route-map E2O
network 10.1.4.4 0.0.0.0 area 0
network 10.1.23.0 0.0.0.7 area 0
distance 180 10.1.5.5 0.0.0.0 1
!
ip classless
no ip http server
ip pim bidir-enable
!
access-list 1 permit 167.1.32.0 0.0.3.255
access-list 2 deny 167.0.0.0 0.255.255.255
access-list 2 deny 100.1.145.0 0.0.0.255
access-list 2 permit any
route-map O2E deny 10
match tag 81
!
route-map O2E permit 20
!
route-map E2O deny 10
match tag 82 80
!
route-map E2O permit 20
match ip address 2
set metric 19
!
route-map E2O permit 30
!
route-map O permit 40
match tag 10
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
!
end
```

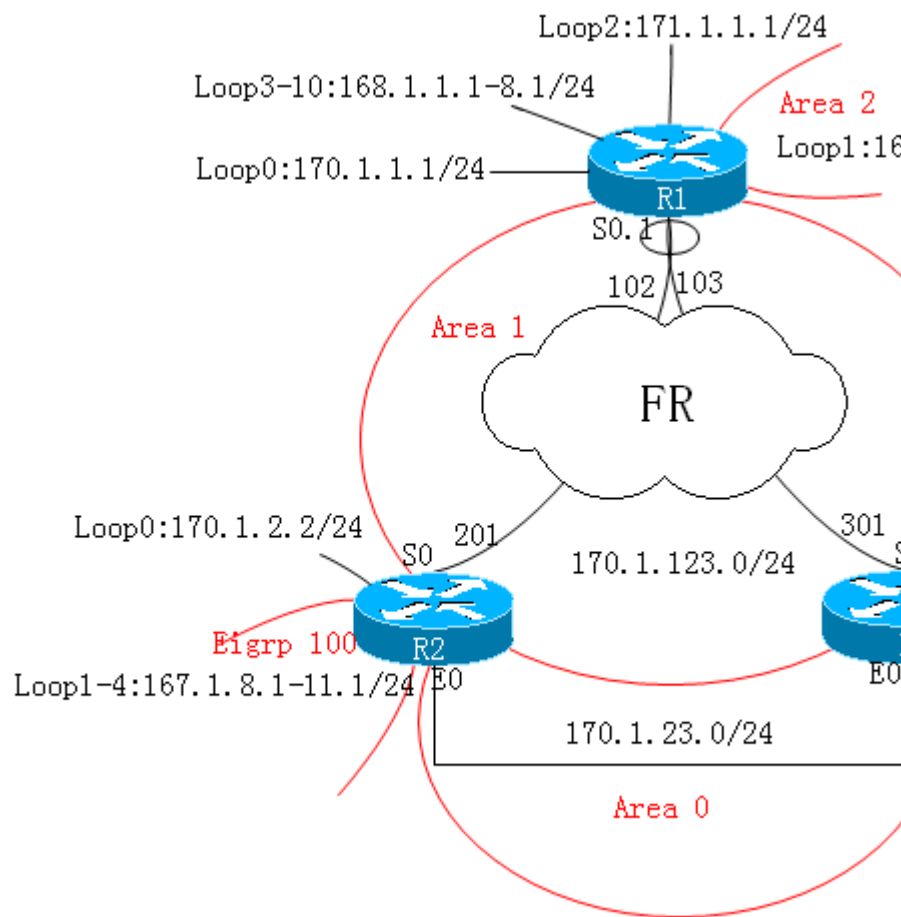
R5-----

```
hostname R5
!
logging rate-limit console 10 except errors
!
ip subnet-zero
no ip finger
```

```
ip subnet-zero
no ip finger
no ip domain-lookup
!
no ip dhcp-client network-discovery
!
!
!
!
interface Loopback0
 ip address 10.1.5.5 255.255.255.255
!
interface Ethernet0
 ip address 10.1.23.5 255.255.255.248
 ip ospf priority 0
 no ip mroute-cache
!
interface Serial0
 bandwidth 10000
 ip address 100.1.145.5 255.255.255.0
 encapsulation frame-relay
 frame-relay map ip 100.1.145.1 501 broadcast
 frame-relay map ip 100.1.145.4 501 broadcast
 no frame-relay inverse-arp
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 100
 redistribute ospf 1 metric 1000 100 255 1 1500 route-map
O2E
 passive-interface Ethernet0
 network 10.1.5.5 0.0.0.0
 network 10.1.23.5 0.0.0.0
 network 100.1.145.0 0.0.0.255
 neighbor 100.1.145.4 Serial0
 neighbor 100.1.145.1 Serial0
 metric weights 0 1 0 0 0 0
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 10.1.5.5
 log-adjacency-changes
 redistribute eigrp 100 subnets route-map E2O
 network 10.1.5.5 0.0.0.0 area 0
 network 10.1.23.0 0.0.0.7 area 0
```

```
network 10.1.23.0 0.0.0.7 area 0
distance 180 10.1.4.4 0.0.0.0 1
!
ip kerberos source-interface any
ip classless
no ip http server
!
access-list 1 permit 167.1.32.0 0.0.3.255
route-map E2O deny 10
 match tag 82 80
!
route-map E2O permit 20
!
route-map O2E deny 10
 match tag 81
!
route-map O2E permit 20
!
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 transport input none
line aux 0
line vty 0 4
!
end
```

## 4.5 OSPF



### 一、桥接：

帧中继中R1的S0接口只能用子接口，要求Ping通所有接口（仅使用图中提供的DLCI号）

### 二、OSPF

- 1、R1、R2、R3的S0接口在OSPF的Area1中，R2、R3的E0口在OSPF的Area0中，R1的Loopback1在OSPF的Area2中，R3的Loopback1在OSPF的Area3中。
- 2、OSPF的帧中继中不允许使用NBMA和广播模式
- 3、Area2只接收OSPF的inter和intra路由
- 4、R3在日后回从Area3中接收到一些LSA7类型的路由，以及Area3中会有一条默认路由。

5、Area0使用明文验证，Area1使用安全的认证方式，验证密码为cisco

6、所有Loopback0接口均在OSPF域内。

7、R1的Loop3到Loop10接口不允许直接宣告进OSPF域内。

### 三、EIGRP

1、R2的Loop1到Loop4在Eigrp中。

2、Eigrp和OSPF在R2上做双向重分布，Eigrp只向OSPF发送一条路由（不允许是167.1.0.0/16）。

### 四、RIP

1、R3的Loop2到Loop5在RIP域内。

2、RIP和OSPF在R3上做双向重分布。

3、在R1和R2上只能看到RIP域过来的一条汇总路由（不那是166.1.0.0/16）。要和Eigrp的汇总用不同的方法。

4、R1的Loop2接口不允许宣告在任何路由协议中。

### 五、过滤

1、在R2上只能看见这样的一条路由168.1.x.0(x为奇数)。

2、在R3上只能看见这样的一条路由168.1.x.0(x为偶数)。

**注：**要求全网互通，不允许出现任何主机路由。所有Loop0接口的地址为170.1.x.x(x为路由器号)。本实验主网段是 170.1.0.0/16

**注：**要求全网互通，不允许出现任何主机路由。所有Loop0接口的地址为170.1.x.x(x为路由器号)。本实验主网段是 170.1.0.0/16

### 配置

```
R1-----
hostname R1
!
logging rate-limit console 10 except errors
!
ip subnet-zero
no ip finger
no ip domain-lookup
!
no ip dhcp-client network-discovery
!
!
!
!
interface Loopback0
 ip address 170.1.1.1 255.255.255.0
!
interface Loopback1
 ip address 169.1.1.1 255.255.255.0
!
interface Loopback2
 ip address 171.1.1.1 255.255.255.0
!
interface Loopback3
 ip address 168.1.1.1 255.255.255.0
!
interface Loopback4
 ip address 168.1.2.1 255.255.255.0
!
interface Loopback5
 ip address 168.1.3.1 255.255.255.0
!
interface Loopback6
 ip address 168.1.4.1 255.255.255.0
!
interface Loopback7
```

```
interface Loopback7
 ip address 168.1.5.1 255.255.255.0
!
interface Loopback8
 ip address 168.1.6.1 255.255.255.0
!
interface Loopback9
 ip address 168.1.7.1 255.255.255.0
!
interface Loopback10
 ip address 168.1.8.1 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 no ip address
 encapsulation frame-relay
 no frame-relay inverse-arp
!
interface Serial0.1 multipoint
 ip address 170.1.123.1 255.255.255.0
 ip ospf message-digest-key 1 md5 cisco
 ip ospf network point-to-multipoint
 frame-relay map ip 170.1.123.1 103 broadcast
 frame-relay map ip 170.1.123.2 102 broadcast
 frame-relay map ip 170.1.123.3 103 broadcast
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 90
 network 171.1.1.0 0.0.0.255
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 170.1.1.1
 log-adjacency-changes
 area 0 authentication
 area 1 authentication message-digest
 area 1 virtual-link 170.1.2.2 authentication-key cisco
 area 1 virtual-link 170.1.3.3 authentication-key cisco
 area 2 stub
 redistribute connected subnets route-map CON
 network 169.1.1.0 0.0.0.255 area 2
```

```
network 169.1.1.0 0.0.0.255 area 2
network 170.1.1.0 0.0.0.255 area 1
network 170.1.123.0 0.0.0.255 area 1
!
ip kerberos source-interface any
ip classless
no ip http server
!
route-map CON permit 10
 match interface Loopback3 Loopback4 Loopback5 Loopback6
 Loopback7 Loopback8 Loopback9 Loopback10
!
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
 transport input none
line aux 0
line vty 0 4
!
end
```

R2-----

```
hostname R2
!
!
ip subnet-zero
no ip domain-lookup
!
frame-relay switching
!
!
interface Loopback0
 ip address 170.1.2.2 255.255.255.0
!
interface Loopback1
 ip address 167.1.8.1 255.255.255.0
 ip summary-address eigrp 100 167.1.8.0 255.255.252.0 5
!
interface Loopback2
 ip address 167.1.9.1 255.255.255.0
!
interface Loopback3
 ip address 167.1.10.1 255.255.255.0
!
interface Loopback4
```

```
!
interface Loopback4
ip address 167.1.11.1 255.255.255.0
!
interface Tunnel0
no ip address
tunnel source 10.1.23.2
tunnel destination 10.1.23.4
!
interface Tunnel1
no ip address
tunnel source 10.1.23.2
tunnel destination 10.1.23.5
!
interface Ethernet0
ip address 10.1.23.2 255.255.255.0
ip ospf authentication-key cisco
!
interface Serial0
no ip address
encapsulation frame-relay
clockrate 2000000
no frame-relay inverse-arp
frame-relay intf-type dce
frame-relay route 102 interface Tunnel0 204
frame-relay route 103 interface Tunnel1 205
!
interface Serial1
ip address 170.1.123.2 255.255.255.0
encapsulation frame-relay
ip ospf message-digest-key 1 md5 cisco
ip ospf network point-to-multipoint
clockrate 2000000
frame-relay map ip 170.1.123.1 201 broadcast
frame-relay map ip 170.1.123.2 201 broadcast
frame-relay map ip 170.1.123.3 201 broadcast
no frame-relay inverse-arp
!
router eigrp 100
redistribute ospf 1 metric 1000 100 255 1 1500
network 167.1.8.0 0.0.3.255
no auto-summary
no eigrp log-neighbor-changes
!
router ospf 1
router-id 170.1.2.2
log-adjacency-changes
area 0 authentication
```

```
area 0 authentication
area 1 authentication message-digest
area 1 virtual-link 170.1.3.3 authentication-key cisco
area 1 virtual-link 170.1.1.1 authentication-key cisco
redistribute eigrp 100 subnets route-map E2O
network 10.1.23.0 0.0.0.255 area 0
network 170.1.2.0 0.0.0.255 area 0
network 170.1.123.0 0.0.0.255 area 1
distribute-list 2 in
!
ip classless
no ip http server
ip pim bidir-enable
!
!
ip prefix-list 1 seq 5 permit 167.1.8.0/22
access-list 2 deny 168.1.0.0 0.0.254.0
access-list 2 permit any
route-map E2O permit 10
match ip address prefix-list 1
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
line aux 0
line vty 0 4
login
!
end
```

R3-----

```
hostname R3
!
logging rate-limit console 10 except errors
!
ip subnet-zero
no ip finger
no ip domain-lookup
!
no ip dhcp-client network-discovery
!
!
!
!
interface Loopback0
```

```
!
interface Loopback0
 ip address 170.1.3.3 255.255.255.0
!
interface Loopback1
 ip address 165.1.1.1 255.255.255.0
!
interface Loopback2
 ip address 166.1.16.1 255.255.255.0
!
interface Loopback3
 ip address 166.1.17.1 255.255.255.0
!
interface Loopback4
 ip address 166.1.18.1 255.255.255.0
!
interface Loopback5
 ip address 166.1.19.1 255.255.255.0
!
interface Ethernet0
 ip address 10.1.23.3 255.255.255.0
 ip ospf authentication-key cisco
 shutdown
!
interface Serial0
 no ip address
 shutdown
!
interface Serial1
 ip address 170.1.123.3 255.255.255.0
 encapsulation frame-relay
 ip ospf message-digest-key 1 md5 cisco
 ip ospf network point-to-multipoint
 clockrate 2000000
 frame-relay map ip 170.1.123.1 301 broadcast
 frame-relay map ip 170.1.123.2 301 broadcast
 frame-relay map ip 170.1.123.3 301 broadcast
 no frame-relay inverse-arp
!
router ospf 1
 router-id 170.1.3.3
 log-adjacency-changes
 area 0 authentication
 area 1 authentication message-digest
 area 1 virtual-link 170.1.2.2 authentication-key cisco
 area 1 virtual-link 170.1.1.1 authentication-key cisco
 area 3 nssa no-summary
 summary-address 166.1.16.0 255.255.252.0
```

```
summary-address 166.1.16.0 255.255.252.0
redistribute rip subnets
network 10.1.23.0 0.0.0.255 area 0
network 165.1.1.0 0.0.0.255 area 3
network 170.1.3.0 0.0.0.255 area 0
network 170.1.123.0 0.0.0.255 area 1
distribute-list 3 in
!
router rip
version 2
passive default
redistribute ospf 1 metric 2 route-map O2R
network 166.1.0.0
no auto-summary
!
ip kerberos source-interface any
ip classless
no ip http server
!
!
ip prefix-list 1 seq 5 permit 166.1.16.0/22
access-list 3 deny 168.1.1.0 0.0.254.0
access-list 3 permit any
route-map O2R deny 10
match ip address prefix-list 1
!
route-map O2R permit 20
!
!
!
line con 0
exec-timeout 0 0
logging synchronous
transport input none
line aux 0
line vty 0 4
!
end
```

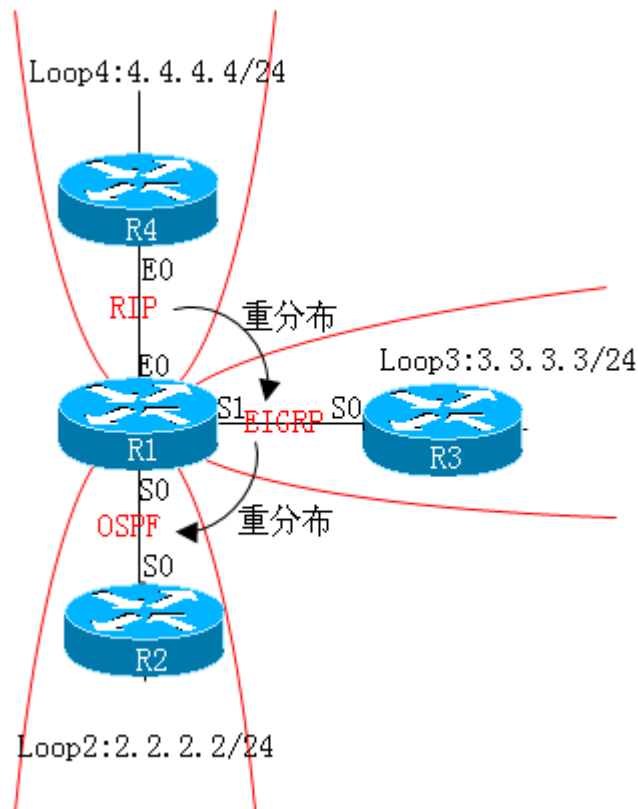
```
R4-----
hostname R4

frame-relay switching
!
!
!
!
interface Tunnel0
```

```
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.4
 tunnel destination 10.1.23.2
!
interface Ethernet0
 ip address 10.1.23.4 255.255.255.0
!
interface Serial0
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 201 interface Tunnel0 204
!
interface Serial1
 no ip address
!
R5-----
hostname R5

frame-relay switching
!
interface Tunnel0
 no ip address
 tunnel source 10.1.23.5
 tunnel destination 10.1.23.2
!
interface Ethernet0
 ip address 10.1.23.5 255.255.255.0
!
interface Serial0
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
 frame-relay route 301 interface Tunnel0 205
!
interface Serial1
 no ip address
 shutdown
```

#### 4.6 重分布



拓扑如上：当把运行RIP的R4中的Loop4重分布进EIGRP中后，再将EIGRP重分布进OSPF中，在R2上是否能看见Loop4的路由。

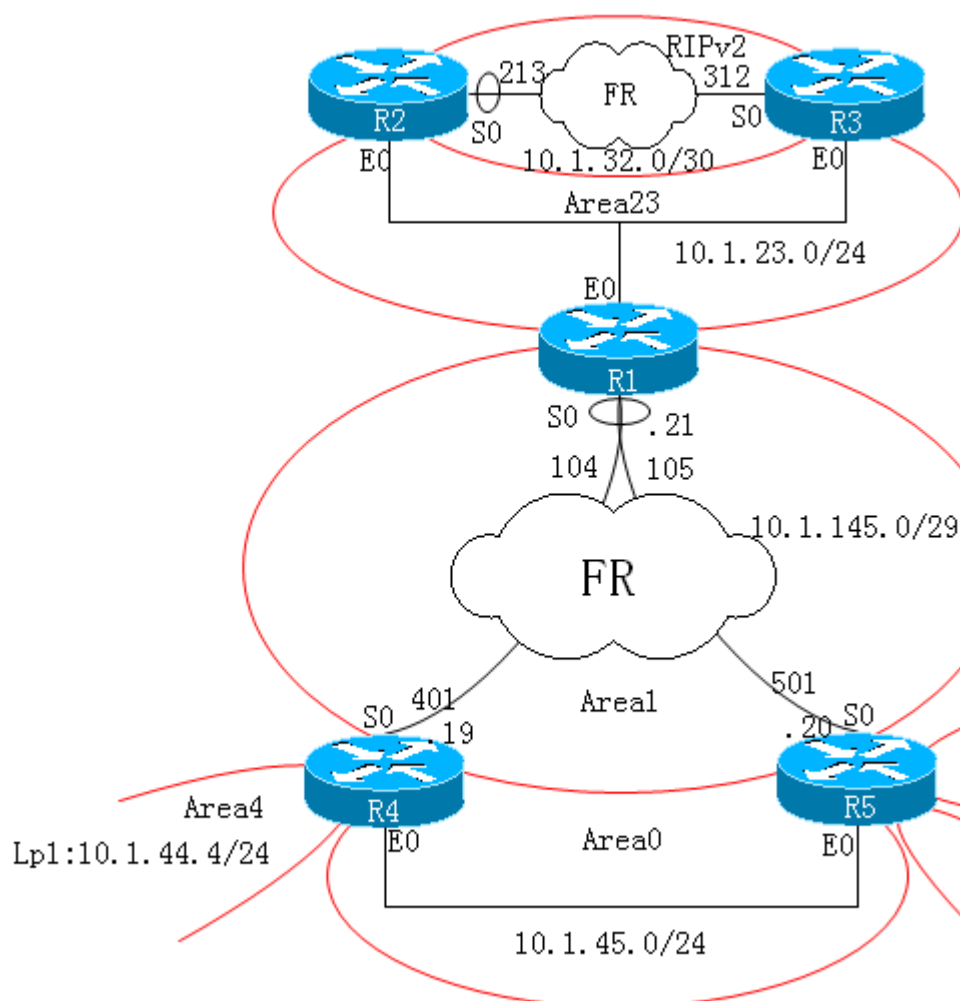
- **答案：**在R2上将不能看到Loop4的路由，因为在将RIP重分布进EIGRP中时，在R1的路由表中将不会看到打R的Loop4路由，由于重分布是基于路由表的，没有此路由，就不会重分布。

为什么在R1上没有打R的Loop4路由？

因为在Loop4重分布进Eigrp中后，在R3上的是看到的是打R的Loop4路由，R3会把收到的打R的Loop4路由，回传给R1，但是R1不会接收，因为，R1的S1口也是运行的Eigrp，因此回来的路由将回比较metric值，回来的的路由的metric值比R1传给R3的metric值更大，因此R1不会接收。

路由的AD的修改值只是本地有效，即本路由器修改后的AD不会传给其他的路由器。

## 4.6 综合



### 一、桥接

帧中继中R1的S0接口和R2的S0接口用子接口（仅使用图中所提供的DLCI）

### 二、EIGRP

R5的lo2-lo7运行在eigrp 100中

eigrp->ospf 要求路由表如下:

OE2 169.1.16.0 [110/0]

OE2 169.1.17.0 [110/150]

OE2 169.1.18.0 [110/0]

OE2 170.1.32.0 [110/150]

OE2 170.1.32.0 [110/150]

OE2 170.1.33.1 [110/150]

OE2 171.1.12.0 [110/150]

三、RIP

R2、R3运行RIP v2，R2、R3的lo0口以及R3的lo1口是RIP域内路由。R3的lo1口地址为192.168.1.3/24

R2、R3间用很安全的认证，密码为CISCO(大写)。

R2、R3间在路由稳定时不发送路由更新。

正常情况下R2访问RIP域外的网络时走R3，当R3的E0口DOWN掉后，走R1。

R3访问RIP域外的网络时走R1，当自己的E0口当掉后，走R2。

要求R2、R3间链路全网可见。且每一跳，cost值会改变

四、OSPF

OSPF的帧中继中不允许使用neighbor命令 R1、R4、R5的环回口运行在OSPF域内；R4、R5的E0口运行在area 0内；R1、R2、R3的E0口运行在area 23内；R1、R4、R5的串口运行在area 1中；R4的lo1运行在area 4中。R5的路lo1运行在area 5中。由于我们不能配置交换机且在以太网上跑OSPF，所以请正确选择OSPF的接口网络类型

area 0明文认证，密码为cisco。R4和R1互相看对方的lo0口路由为24

R4不向area 4发送任何LSA (ip os datafilter-list all)。R5会向area 5中的其他路由器发送一条LSA-7类的默认路由。

R4的E0口一秒钟发送4个HELLO包

当R2在150秒内没收到HELLO包，邻居关系也不会当

五、过滤

RIP不接收171.1.12.0这条路由，不能用列表匹配，但RIP的路由器能通过R2来访问171.1.12.0网段内的主机

六、feature

在R3上配置一个lo1口将其宣告进RIP域内。掩码为24位。

假定这个环回口连一台主机，地址为192.168.1.133。当在这台主机上trace R4的lo1口时。

trace信息为

1 192.168.1.134

2 10.1.4.4

注：要求全网全通，除100外，不允许出现任何主机路由，所有loopback0接口的地址为10.1.x.x/32（x为路由器号），本试验主网段为10.0.0.0/8。  
不允许出现任何的静态路由。

## OSPF

### 双ABR的NSSA

#### R1

```
!
interface Ethernet0
no ip address
shutdown
!
interface Serial0
ip address 12.0.0.1 255.255.255.0
!
interface Serial1
ip address 13.0.0.1 255.255.255.0
ip ospf cost 10
!
router ospf 1
router-id 102.0.0.1
log-adjacency-changes
area 1 nssa
network 12.0.0.0 0.0.0.255 area 0
network 13.0.0.0 0.0.0.255 area 1
!
```

#### R2

```
interface Ethernet0
no ip address
shutdown
!
interface Serial0
ip address 12.0.0.2 255.255.255.0
ip ospf cost 64
clockrate 2000000
!
interface Serial1
ip address 24.0.0.2 255.255.255.0
clockrate 2000000
!
router ospf 1
```

```
router-id 100.0.0.2
log-adjacency-changes
network 12.0.0.0 0.0.0.255 area 0
network 24.0.0.0 0.0.0.255 area 0
!
```

### R3

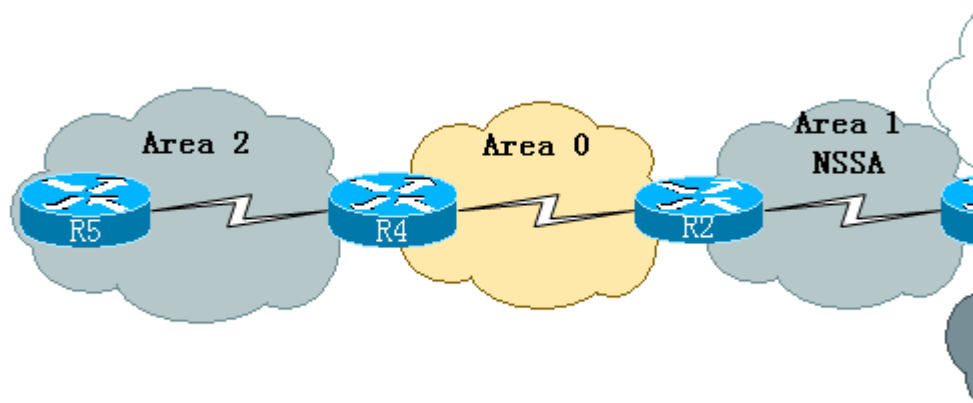
```
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Loopback2
 ip address 3.3.2.3 255.255.255.0
!
interface Ethernet0
 ip address 34.0.0.3 255.255.255.0
!
interface Serial0
 ip address 13.0.0.3 255.255.255.0
 clockrate 2000000
!
interface Serial1
 no ip address
 shutdown
!
router eigrp 9
 network 3.3.3.0 0.0.0.255
 auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 100.0.0.3
 log-adjacency-changes
 area 1 nssa
 redistribute eigrp 9 subnets
 network 3.3.2.0 0.0.0.255 area 1
 network 13.0.0.0 0.0.0.255 area 1
 network 34.0.0.0 0.0.0.255 area 1
!
```

### R4

```
interface Ethernet0
 ip address 34.0.0.4 255.255.255.0
!
interface Serial0
 ip address 24.0.0.4 255.255.255.0
!
interface Serial1
```

```
interface Serial1
no ip address
shutdown
!
router ospf 1
router-id 100.0.0.4
log-adjacency-changes
area 1 nssa
network 24.0.0.0 0.0.0.255 area 0
network 34.0.0.0 0.0.0.255 area 1
!
```

重分布+过滤+汇总



EIGRP产生的路由重分布到OSPF中要求如下

| IP ADD      | Metric | TAG |
|-------------|--------|-----|
| 172.16.12.0 | 50     | 20  |
| 172.16.13.0 | 50     | 10  |
| 172.16.14.0 | 50     | 20  |
| 172.16.15.0 | 50     | 10  |
| 172.17.32.0 | 50     | 10  |
| 172.17.33.0 | 50     | 10  |
| 172.17.34.0 | 50     | 10  |
| 100.1.1.0   | 100    | 20  |
| 192.168.1.0 | 100    | 10  |
| 192.168.5.0 | 100    | 10  |

## R1

```
interface Loopback0
ip address 1.1.1.1 255.255.255.0
!
```

```
interface Loopback1
 ip address 172.16.32.3 255.255.255.0
!
interface Loopback2
 ip address 172.16.33.3 255.255.255.0
!
interface Loopback3
 ip address 172.16.34.3 255.255.255.0
!
interface Loopback4
 ip address 172.16.35.3 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 12.0.0.1 255.255.255.0
 ip summary-address rip 172.16.0.0 255.255.128.0
!
interface Serial1
 ip address 13.0.0.1 255.255.255.0
!
router eigrp 9
 network 13.0.0.0
 auto-summary
 no eigrp log-neighbor-changes
!
router ospf 1
 router-id 100.0.0.1
 log-adjacency-changes
 area 1 nssa
 summary-address 172.16.16.0 255.255.252.0
 redistribute rip subnets
 redistribute eigrp 9 subnets route-map REDI
 network 12.0.0.0 0.0.0.255 area 1
!
router rip
 version 2
 network 172.16.0.0
 no auto-summary
!
ip kerberos source-interface any
ip classless
no ip http server
!
access-list 1 permit 172.16.12.0 0.0.2.255
access-list 2 permit 100.1.1.0
```

```
access-list 2 permit 100.1.1.0
access-list 3 permit 172.16.0.0 0.1.255.255
access-list 4 permit 192.168.1.0 0.0.4.255
route-map REDI permit 10
 match ip address 1
 set metric 50
 set tag 20
!
route-map REDI permit 20
 match ip address 2
 set metric 100
 set tag 20
!
route-map REDI permit 30
 match ip address 3
 set metric 50
 set tag 10
!
route-map REDI permit 40
 match ip address 4
 set metric 100
 set tag 10
```

## R2

```
interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Ethernet0
 no ip address
 shutdown
!
interface Serial0
 ip address 12.0.0.2 255.255.255.0
 clockrate 2000000
!
interface Serial1
 ip address 24.0.0.2 255.255.255.0
 clockrate 2000000
!
router ospf 1
 router-id 100.0.0.2
 log-adjacency-changes
 area 1 nssa
 summary-address 172.16.0.0 255.255.0.0
 network 2.2.2.2 0.0.0.0 area 0
 network 12.0.0.0 0.0.0.255 area 1
 network 24.0.0.0 0.0.0.255 area 0
```

```
network 24.0.0.0 0.0.0.255 area 0
distribute-list 2 in
!
ip kerberos source-interface any
ip classless
no ip http server
!
access-list 2 deny 192.168.0.0 0.0.1.255
access-list 2 permit any
!
```

### **R3**

```
interface Loopback0
 ip address 3.3.3.3 255.255.255.0
!
interface Loopback1
 ip address 172.16.12.5 255.255.255.0
!
interface Loopback2
 ip address 172.16.13.5 255.255.255.0
!
interface Loopback3
 ip address 172.16.14.5 255.255.255.0
!
interface Loopback4
 ip address 172.16.15.5 255.255.255.0
!
interface Loopback5
 ip address 172.17.32.5 255.255.255.0
!
interface Loopback6
 ip address 172.17.33.5 255.255.255.0
!
interface Loopback7
 ip address 172.17.34.5 255.255.255.0
!
interface Loopback8
 ip address 100.1.1.5 255.255.255.0
!
interface Loopback9
 ip address 192.168.1.5 255.255.255.0
!
interface Loopback10
 ip address 192.168.5.5 255.255.255.0
!
interface Loopback11
 ip address 172.16.0.5 255.255.255.0
```

```
ip address 172.16.0.5 255.255.255.0
!
interface Loopback12
ip address 172.16.1.5 255.255.255.0
!
interface Loopback13
ip address 172.16.2.5 255.255.255.0
!
interface Loopback14
ip address 172.16.3.5 255.255.255.0
!
interface Loopback15
ip address 10.4.5.0 255.255.0.0
!
interface Loopback16
ip address 10.5.5.0 255.255.0.0
!
interface Loopback17
ip address 10.6.5.0 255.255.0.0
!
interface Loopback18
ip address 10.7.5.0 255.255.0.0
!
interface Loopback19
ip address 192.168.16.5 255.255.255.0
!
interface Loopback20
ip address 192.168.17.5 255.255.255.0
!
interface Loopback21
ip address 192.168.18.5 255.255.255.0
!
interface Loopback22
ip address 192.168.19.5 255.255.255.0
!
interface Ethernet0
no ip address
shutdown
!
interface Serial0
ip address 13.0.0.3 255.255.255.0
clockrate 2000000
!
interface Serial1
ip address 35.0.0.3 255.255.255.0
shutdown
clockrate 2000000
!
```

```
!
router eigrp 9
 network 0.0.0.0
 no auto-summary
 no eigrp log-neighbor-changes
!
```

#### **R4**

```
interface Loopback0
 ip address 4.4.4.4 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback1
 ip address 4.0.0.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback2
 ip address 4.0.1.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback3
 ip address 4.0.2.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback4
 ip address 4.0.3.1 255.255.255.0
 ip ospf network point-to-point
!
interface Ethernet0
 ip address 45.0.0.4 255.255.255.0
!
interface Serial0
 ip address 24.0.0.4 255.255.255.0
!
interface Serial1
 no ip address
 shutdown
!
router ospf 1
 router-id 100.0.0.4
 log-adjacency-changes
 area 0 range 4.0.0.0 255.255.252.0
 area 2 filter-list prefix PR1 out
 network 4.0.0.0 0.0.3.255 area 0
 network 4.4.4.4 0.0.0.0 area 0
 network 24.0.0.0 0.0.0.255 area 0
 network 45.0.0.0 0.0.0.255 area 2
```

```
network 45.0.0.0 0.0.0.255 area 2
!
no ip http server
ip classless
!
!
!
ip prefix-list 1 seq 5 permit 192.168.0.0/24
ip prefix-list 1 seq 10 permit 192.168.1.0/24
!
```

## R5

```
interface Loopback0
ip address 5.5.5.5 255.255.255.0
!
interface Loopback1
ip address 192.168.0.5 255.255.255.0
ip ospf network point-to-point
!
interface Loopback2
ip address 192.168.1.5 255.255.255.0
ip ospf network point-to-point
!
interface Loopback3
ip address 192.168.2.5 255.255.255.0
ip ospf network point-to-point
!
interface Loopback4
ip address 192.168.3.5 255.255.255.0
ip ospf network point-to-point
!
interface Ethernet0
ip address 45.0.0.5 255.255.255.0
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
shutdown
!
router ospf 1
router-id 100.0.0.5
log-adjacency-changes
network 45.0.0.0 0.0.0.255 area 2
network 192.168.0.0 0.0.3.255 area 2
```

network 192.168.0.0 0.0.3.255 area 2  
!

## 生词本

|                    |                  |               |
|--------------------|------------------|---------------|
| prove              | [pru:v]          | 证明, 证实, 检验,   |
| 考验                 |                  |               |
| improve            | [im'pru:v]       | 改善, 改进        |
| achieve            | [E'tfi:v]        | 完成, 达到        |
| access             |                  | 通路, 访问, 入门    |
| arduous            | ['a:djuEs]       | 费劲的, 辛勤的, 险峻的 |
| approximate        |                  | 近似, 接近, 接近,   |
| 约计                 |                  |               |
| accommodate        |                  | 供应, 供给, 使适    |
| 应, 调节, 和解, 向... 提供 |                  |               |
| interactive        | [,intEr'ktiv]    | 交互式的          |
| boundary           |                  | 边界, 分界线       |
| burden             |                  | 担子, 负担        |
| binary             |                  | 二进位的, 二元的     |
| collaboration      | [kE,lAbE'reiFEn] | 协作, 通敌        |
| common             |                  | 共同的, 公共的, 公   |
| 有的, 普通的, 庸俗的, 伪劣的  |                  |               |
| comment            |                  | 注释, 评论, 意见    |
| command            |                  | 命令, 掌握, 司令部   |
| combine            | [kEm'bain]       | (使)联合, (使)结合  |
| compromise         |                  | 妥协, 折衷, 危     |
| 及...的安全            |                  |               |
| containment        |                  | 围堵政策, 牵制政策    |
| confine            |                  | 限制, 禁闭        |
| consume            |                  | 消耗, 消费, 消灭,   |
| 大吃大喝, 吸引           |                  |               |
| consuming          |                  | 强烈的           |
| consumption        | [kEn'sQmpFEn]    | 消费, 消费量, 肺病   |
| assume             |                  | 假定, 设想, 采取,   |
| 呈现                 |                  |               |
| resume             |                  | <美> 履历        |

|                  |                |             |
|------------------|----------------|-------------|
| consume          |                | 消耗, 消费, 消灭, |
| 大吃大喝, 吸引         |                |             |
| consuming        |                | 强烈的         |
| consumption      | [kEn' sQmpFEn] | 消费, 消费量, 肺病 |
| assume           |                | 假定, 设想, 采取, |
| 呈现               |                |             |
| resume           |                | <美> 履历      |
| contiguous       | [kEn' tigjuEs] | 邻近的, 接近的, 毗 |
| 边的               |                |             |
| consideration    |                | 体谅, 考虑, 需要考 |
| 虑的事项, 报酬         |                |             |
| correspond       |                | 符合, 协调, 通信, |
| 相当, 相应           |                |             |
| correspondent    |                | 通讯记者, 通信者   |
| correspondence   |                | 相应, 通信, 信件  |
| coordinate       | [kEu' C:dinit] | 同等者, 同等物, 坐 |
| 标(用复数)           |                |             |
| compose          | [kEm' pEuz]    | 组成, 写作, 排字, |
| (使)安定, 调解        |                |             |
| composite        |                | 合成的, 复合的    |
| chassis          | [' FAsi]       | 底盘          |
| distinct         |                | 清楚的, 明显的, 截 |
| 然不同的, 独特的        |                |             |
| delineate        |                | 描绘          |
| deprecate        |                | 轻视, 抗议, 不赞  |
| 成, 抨击, 反对, 藐视    |                |             |
| extent           |                | 广度, 宽度, 长度, |
| 范围, 程度, 区域       |                |             |
| extend           |                | 扩充, 延伸, 伸展, |
| 扩大               |                |             |
| attend           |                | 出席, 参加, 照顾, |
| 护理, 注意           |                |             |
| attend to        |                | 专心, 注意, 照顾  |
| attent/attention |                | [古]注意的, 留意的 |
| expansion        |                | 扩充, 开展, 膨胀, |

|                  |                |             |
|------------------|----------------|-------------|
| 顾, 护理, 注意        |                |             |
| attend to        |                | 专心, 注意, 照顾  |
| attent/attention |                | [古]注意的, 留意的 |
| expansion        |                | 扩充, 开展, 膨胀, |
| 扩张物, 辽阔, 浩瀚      |                |             |
| finite           |                | 有限的, [数]有穷  |
| 的, 限定的           |                |             |
| formally         |                | 正式地, 形式上    |
| intend           |                | 想要, 打算, 意指, |
| 意谓               |                |             |
| interconnection  |                | 互相连络        |
| immunity         |                | 免疫性         |
| incremention     |                | (月)盈; 渐增加   |
| eliminate        |                | 排除, 消除      |
| estimating       |                | 估计          |
| excessive        |                | 过多的, 过分的, 额 |
| 外                |                |             |
| leverage         | ['li:vEridV]   | 杠杆作用        |
| legacy           | ['legEsi]      | 遗赠(物), 遗产(祖 |
| 先传下来)            |                |             |
| resilient        | [ri'ziliEnt]   | 弹回的, 有回弹力的  |
| revision         |                | 修订, 修改, 修正, |
| 修订本              |                |             |
| procedure        | [prE'si:dVE]   | 程序, 手续      |
| subscription     | [sQb5skripFEn] | 捐献, 订金, 订阅, |
| 签署, 同意, [医]下标处方  |                |             |
| oversubscription |                | 超额认购        |
| manner           |                | 礼貌, 风格, 方式, |
| 样式, 习惯           |                |             |
| mitigate         | ['mitigeit]    | 减轻          |
| migration        |                | 移民, 移植, 移往, |
| 移动               |                |             |
| military         |                | 军事的, 军用的    |
| impact           | ['impAkt]      | 碰撞, 冲击, 冲突, |
| 影响, 效果           |                |             |

|                                            |                    |             |
|--------------------------------------------|--------------------|-------------|
| migration                                  |                    | 移民, 移植, 移往, |
| 移动                                         |                    |             |
| military                                   |                    | 军事的, 军用的    |
| impact                                     | [ˈ ɪmpʌkt]         | 碰撞, 冲击, 冲突, |
| 影响, 效果                                     |                    |             |
| invoke                                     |                    | 调用          |
| incorporate                                |                    | 合并的, 结社的, 一 |
| 体化的                                        |                    |             |
| insatiable                                 | [ɪnˈseɪəbəl]       | 不知足的, 贪求无厌  |
| 的                                          |                    |             |
| appetite                                   |                    | 食欲, 胃口, 欲望, |
| 爱好                                         |                    |             |
| intense                                    |                    | 强烈的, 剧烈的, 热 |
| 切的, 热情的, 激烈的                               |                    |             |
| compartmentalize                           | [kəmˈpɑːtməntlaɪz] | 划分          |
| cohesive                                   | [kəʊˈhiːsɪv]       | 粘着的         |
| adaptive                                   | [əˈdæptɪv]         | 适应的         |
| adequate                                   |                    | 适当的, 足够的    |
| approach                                   | [əˈprəʊtʃ]         | 接近, 逼近, 走进, |
| 方法, 步骤, 途径, 通路                             |                    |             |
| attractive                                 |                    | 吸引人的, 有魅力的  |
| streamline                                 |                    | 流线型的        |
| oriented                                   |                    | 导向的         |
| architecture                               |                    | 体系机构        |
| rendezvous                                 | [ˈrɒndɪvuː]        | 集合点         |
| PIM rendezvous points                      |                    | pim协议的集合点   |
| (路由器)                                      |                    |             |
| paradigm                                   |                    | 范例          |
| SONA service-oriented network architecture |                    | 基于服务的网络体系   |
| 结构                                         |                    |             |
| sort                                       |                    | 分类, 拣选      |
| sorting                                    |                    | 资料排架        |
| sparse                                     | [spɜːs]            | 稀少的, 稀疏的    |
| mechanism                                  |                    | 机械装置, 机构, 机 |
| 制                                          |                    |             |

|               |                    |
|---------------|--------------------|
| mechanism     | 机械装置, 机构, 机制       |
| evolution     | 进展, 发展, 演变, 进化     |
| revolution    | 革命                 |
| deploy        | 展开, 配置             |
| issue         | 问题, 结果             |
| impact        | 挤入, 撞击, 压紧,        |
| 对... 发生影响     |                    |
| integrity     | 正直, 诚实, 完整,        |
| 完全, 完整性       |                    |
| norm          | 标准, 规范             |
| optimal       | 最佳的, 最理想的          |
| overlay       | 覆盖, 覆盖图            |
| propagate     | 繁殖, 传播, 宣传         |
| plan          | 计划, 设计图, 平面图       |
| plane         | 平面, 飞机, 水平,        |
| 程度, 创         |                    |
| plant         | 设备, 植物, 庄稼,        |
| 工厂, 车间        |                    |
| robust        | 精力充沛的              |
| relate        | 使联系, 发生关系,         |
| 叙述, 讲         |                    |
| relation      | 关系, 联系, 叙述,        |
| 故事, 亲戚        |                    |
| ratify        | 批准, 认可             |
| ratio         | 比, 比率,             |
| determine     | 决定, 确定, 测定,        |
| 使下定决心, [律]使终止 |                    |
| stem          | [stem] 滋生, 阻止      |
| term          | [tE:m] 学期, 期限, 期间, |
| 条款, 条件, 术语    |                    |
| in terms of   | 根据, 按照, 用...       |
| 的话, 在... 方面   |                    |

间, 条款, 条件, 术语

in terms of

的话, 在... 方面

tenfold

utilization

utility

unbounded

ubiquity

普遍存在

vulnerability

accordance

in accordance with

span

根据, 按照, 用...

十倍的

利用

效用, 有用

极大的

到处存在, (同时的)

弱点, 攻击

一致, 和谐

与... 一致, 依照

跨度, 跨距, 范围

投标人资格

营业执照、税务登记证

项目授权书

法定代表人授权书、被授权人的身份证和复印件

收取资料费500元, 招标公司把招标文件、图纸、招标清单、投标邀请书给投标公司

不得自行修改图纸、设备清单

招标文件发放时间, 需要什么时间地点交投标文件。

投标须知:

投标人必须认真阅读一下内容, 以免造成竞标失败。

1: 工程名称: 一字不改

2: 工程地点: 一字不改

3: 招标内容: 综合布线、网络系统、机房系统、原厂商授权及售后服务

4: 合格的投标人

5: 投标的费用

6: 投标有效期: 遇见30天内价格有效 (货品价格可能浮动, 可能导致投标商不可预计损失)

致投标商不可预计损失)

关于废标:

投标人不足三家

出现违法违规

采购预算

甲方任务取消

标价总则:

写上去的东西不能改, 写错很麻烦

计算正确(单价与总价)

价格中途不能改

标价栏不要钱要写0, 如果空白则可能成为废标

报价含税

总报价

工程项目报价

投标总报价

不能在投标总价之外还有其他费用出现。

付款方式:

不设工程预付款

进度款

付到总价97%, 预留3%作为售后款, 一般一年后给。

工程要求: 45天。

用户要求和技术规格书

布线

网络系统(真正关心)

网络分层结构、引擎冗余、链路冗余、电源冗余、双机热备、VLAN实现、ACL实施、防火墙布局、

链路带宽、VPN

根据我司讨论要求整理如下:

机房

验收和售后服务  
进度计划（收钱）  
系统测试（重要）

投标文件组成  
商务文件、技术文件和唱标信封  
纸张为A4格式、电子板也要、正本一份、副本两份

