

TCP/IP 路由技术 学习手册

—— 似水年华

引言

从开始接触 **CISCO** 的认证体系到现在历时 **10** 个月时间，在考完 **CCNP** 认证之后，苦于没能找到工作，一直在潜心学习 **TCP/IP** 这本书，这份资料是看完 **TCP/IP** 书之后对自己的学习和笔记做的一个总结，本手册以大量的实验为基础验证理论！

因为自己在学习 **cisco** 技术的时候，一直苦于没人指点。无学习伴侣，故特将自己的一点学习心得发布出来，希望能起到抛砖引玉的作用，让更多的初学者少走一点点弯路。

同时也因为自己水平有限，在作品中不免会有笔误，理解错误等不实之处，也希望各位战友能给予理解和支持，另外，本作品不用做商业用途，只供学习之用。谢谢！

后续的交流部分和安全部分也正在陆续的学习和整理中，在不久的将来我会一起发出来，本作品以基础为主，而不是高深技术，目的指是针对初学者。

最后在这里希望所有人能够把基础学好，把理论学扎实。同时也很希望朋友们和我一起讨论技术问题，我的 **cisco** 群是：**68416476** 希望能有更多的战友们加入。

似水年华

CSCO11700217

2010 年 5 月 9 日

第一部分

RIP (routing information protocol)

RIP 作为一种距离矢量协议，从开发到现在一直沿用至今，虽然现在使用比较少。但是 RIP 还是小型企业使用比较多的，以其配置简单，易于管理等特点。

RIP 特点： 距离矢量路由协议；

以 HOP 来计算最佳路径；

最大为 15 跳，16 跳则不可达；

30 秒周期性更新；

最多六条，default 4 条 等价负载均衡

RIPv1: 不支持 VLSM (更新不包含 netmask)

更新使用广播

RIPv2 更新使用组播地址 224.0.0.9

RIPv1 和 RIPv2 的比较

	RIPv1	RIPv2
更新方式	广播 255.255.255.255	组播 224.0.0.9
类别	有类 更新不携带 netmask	无类
汇总	自动汇总	可手动汇总默认自动
认证	不支持	支持

RIP 协议的运行机制：

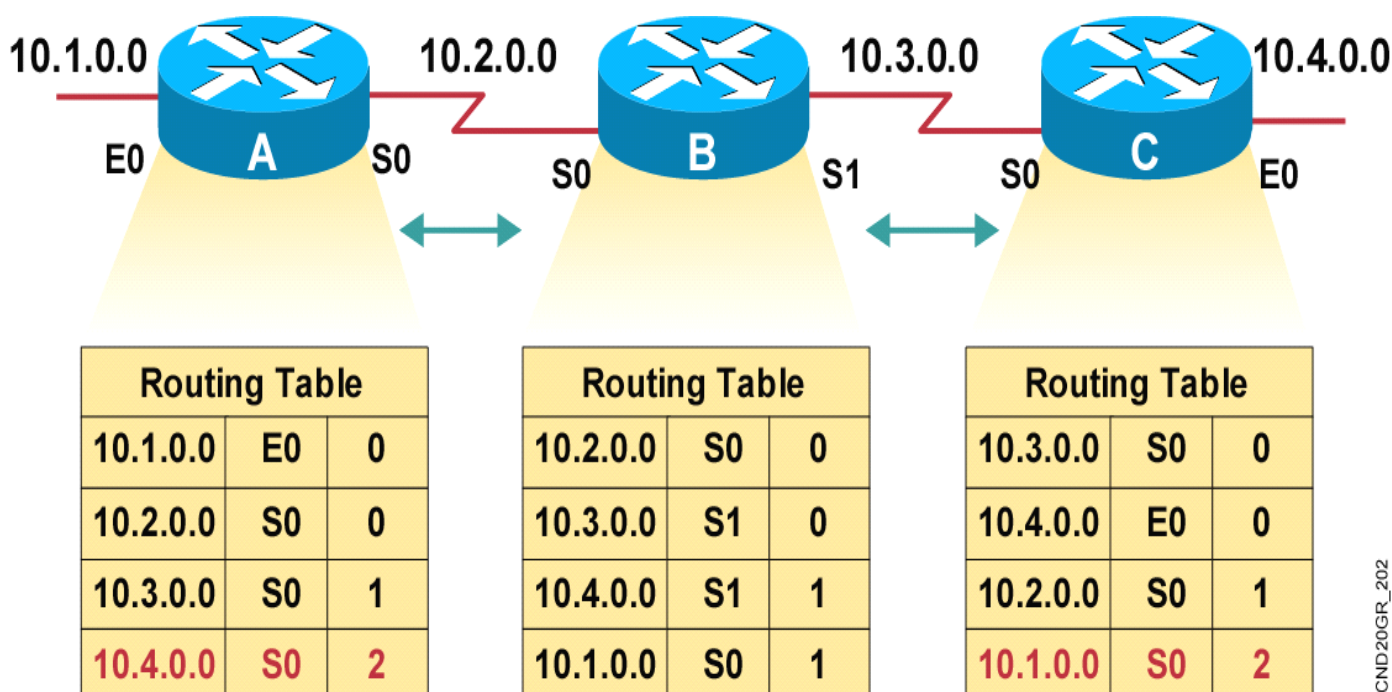
距离矢量路由选择协议是基于贝尔曼-福特算法的。

对于距离矢量路由协议来说，谁是邻居并不重要。

距离向量路由算法将完整的路由表传给相邻 router，然后这个 router 再把收到的表的选项加上

自己的表来完成整个路由表，这个叫做 routing by rumor(道听途说)；因为这个 router 是从相邻 router 被动地接受更新而非自己主动地去发现网络的变化。

更新机制



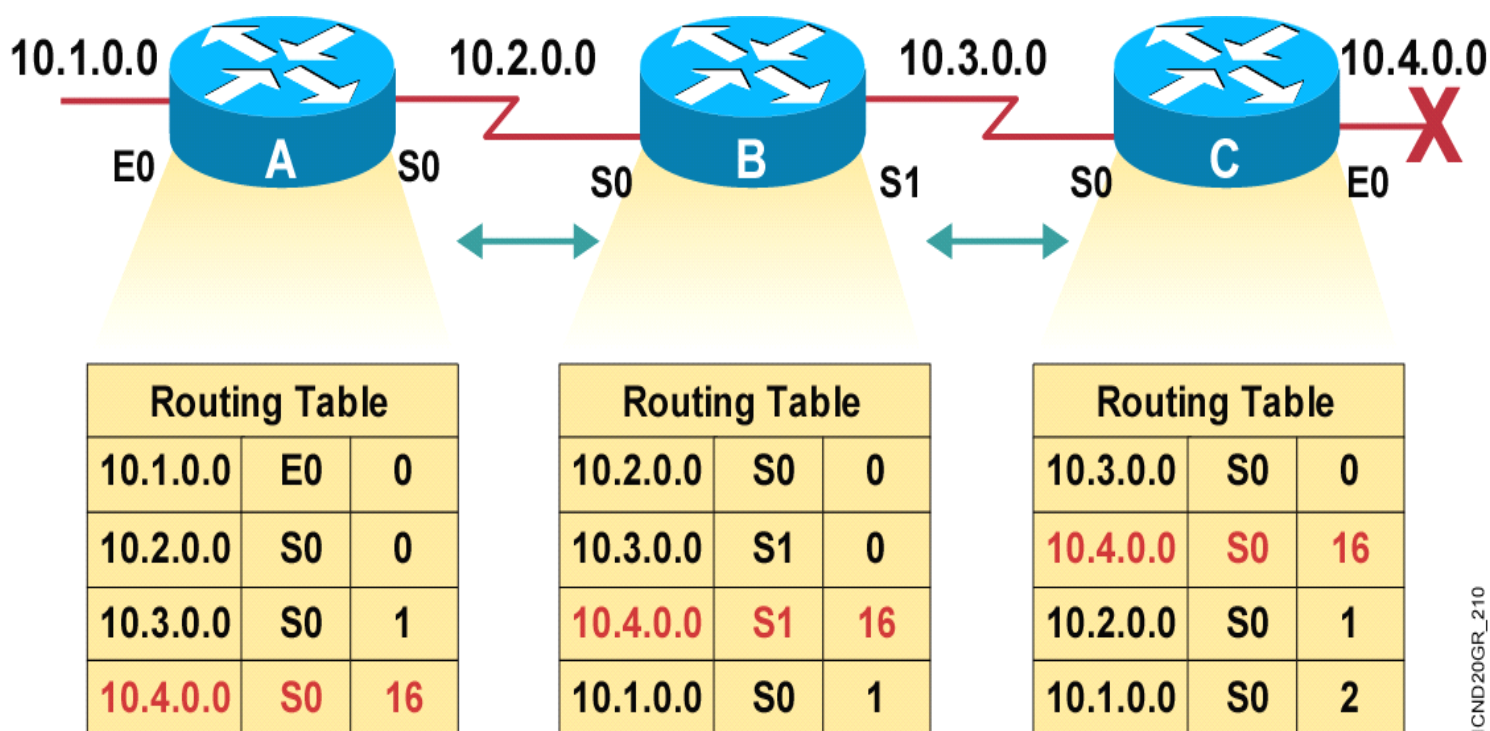
路由器接收到广播更新后就会查看更新，并将该信息与它当前的路由表进行比较；到新网络的路由或者到已知网络并有更好度量值的路由都将被安装到路由表中；然后继续向外广播它已更新过的路由表。距离矢量路由协议关心目的网络的距离和矢量（路由来源的方向）。在发送更新之前，每台路由器都将它自己的距离值添加到路由器的度量值中。当路由器接收到一个更新时，它将学到的网络与接收接口关

联起来。然后路由器将用这个接口来到达相应的目的地。

防环机制

方法一：设置为无穷大

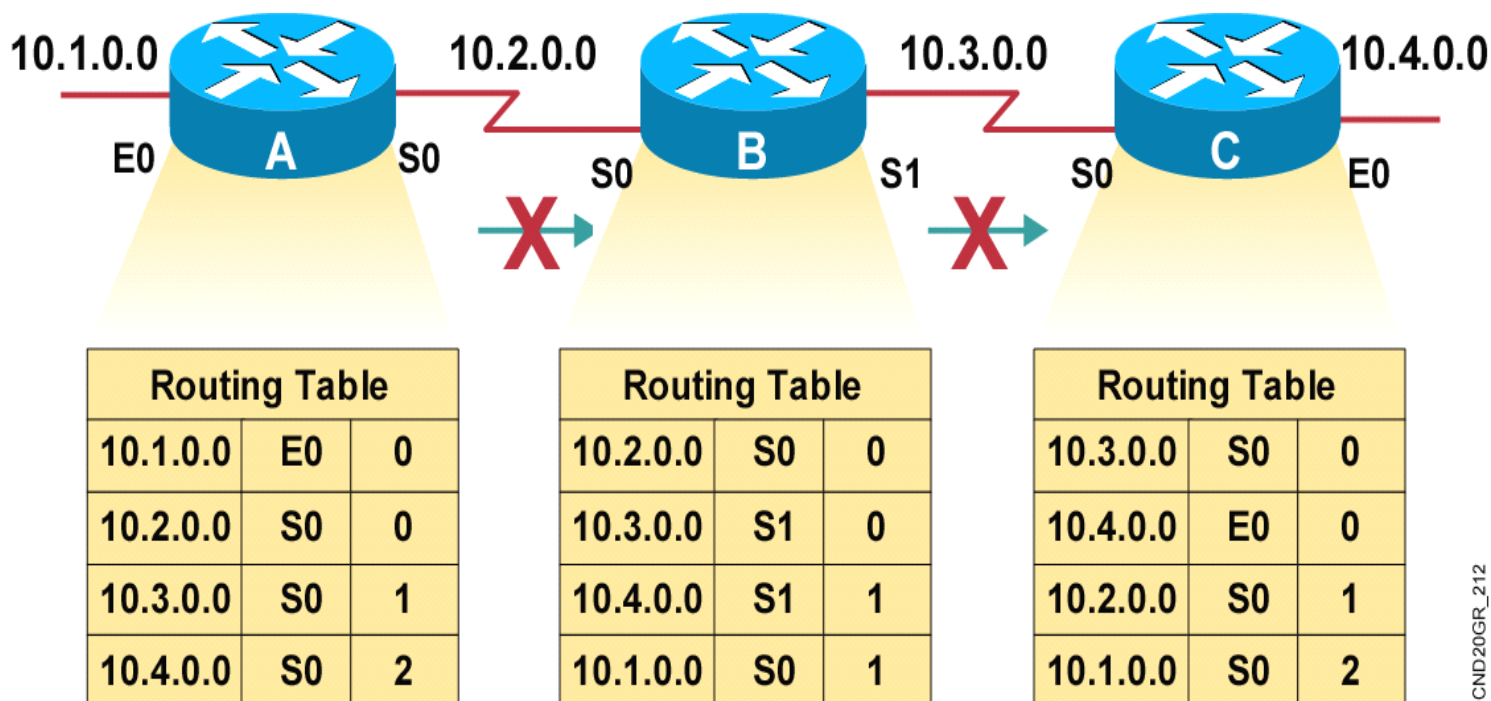
比如 RIP 约定 16 跳就是无穷大值.一旦距离达到这个值,这条路由将标记为 **possible down** 状态,但不会马上从路由表中清除,因为没有到达刷新时间,这个时候如果有数据流的话,仍旧被进行环回,直到刷新时间到时.所以,现在已经很难再见到距离矢量路由协议的身影,除非网络比较简。



红色部分已经将 C 的 10.4.0.0 的路由条目 **metric** 设置成 16

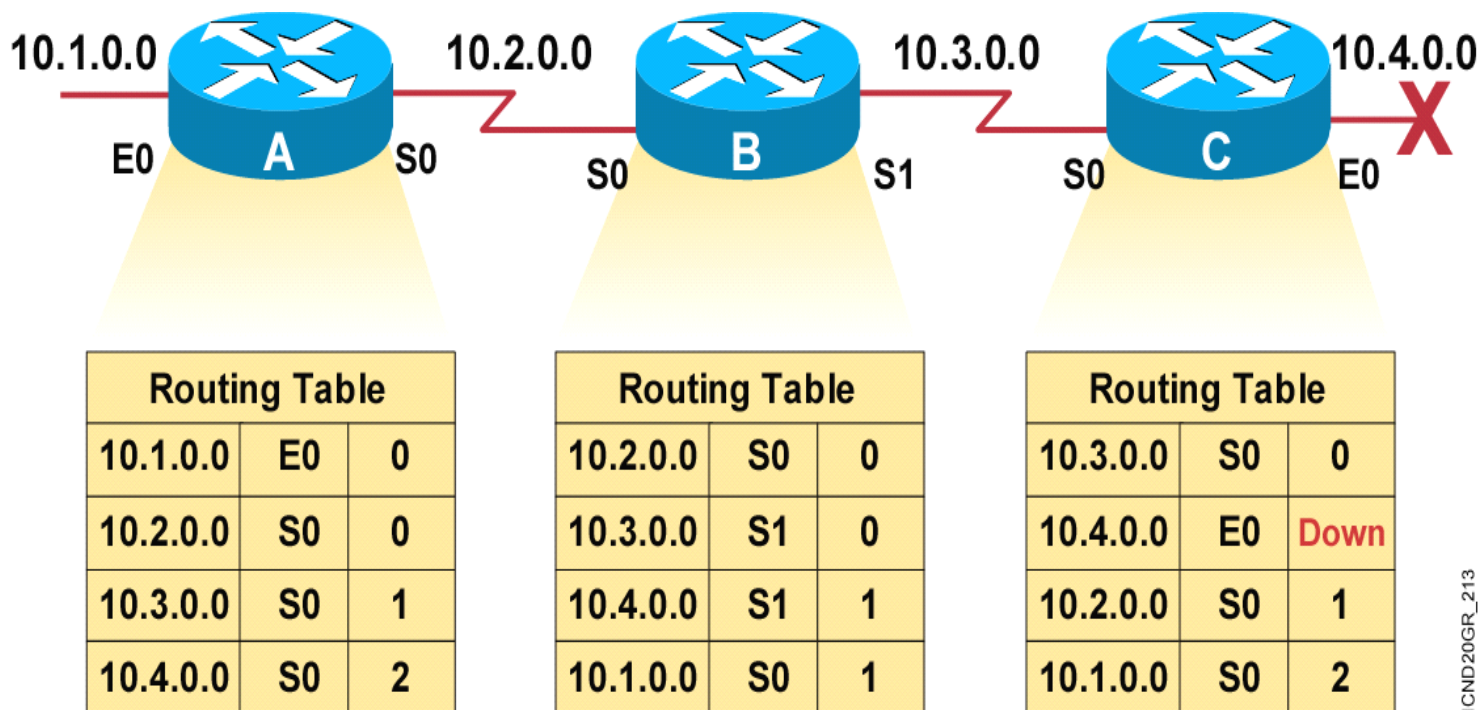
方法二：Split Horizon(水平分割)

路由器的某个接口接收到的路由信息不能再从这个接口反向再发送出去.这个方法减少了路由信息的不正确性并使设备负载大大降低.



A 的 10.4.0.0 路由是从 S0 接收到的,如果 A 的路由更新时间到期,就不能再从 S0 接口发送 10.4.0.0 路由了,其他的路由条目同理.

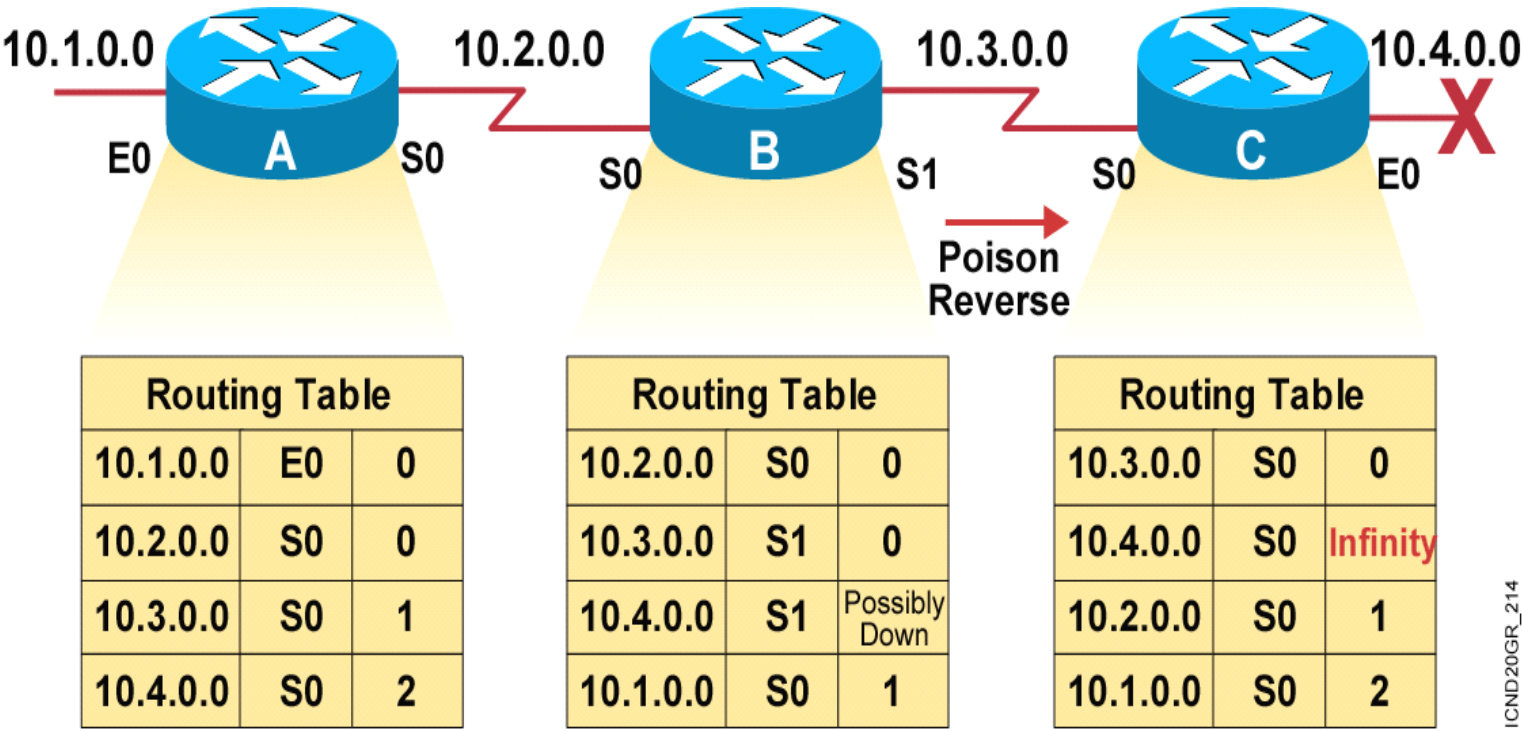
方法三：Route Poisoning(路由毒抑)



ICND20GR_213

计数到无穷不能从根本上避免路由环路.因为 C 并没有明确指明 10.4.0.0 路由不可用,因此,还需要经过一段时间才能到达 16 跳或刷新时间,明显不好.路由毒抑就是在 C 通告这条损坏的路由时,明确告诉邻居这条路由已经坏了,不需要计数到无穷,加快了收敛的时间.

方法四：毒抑反转



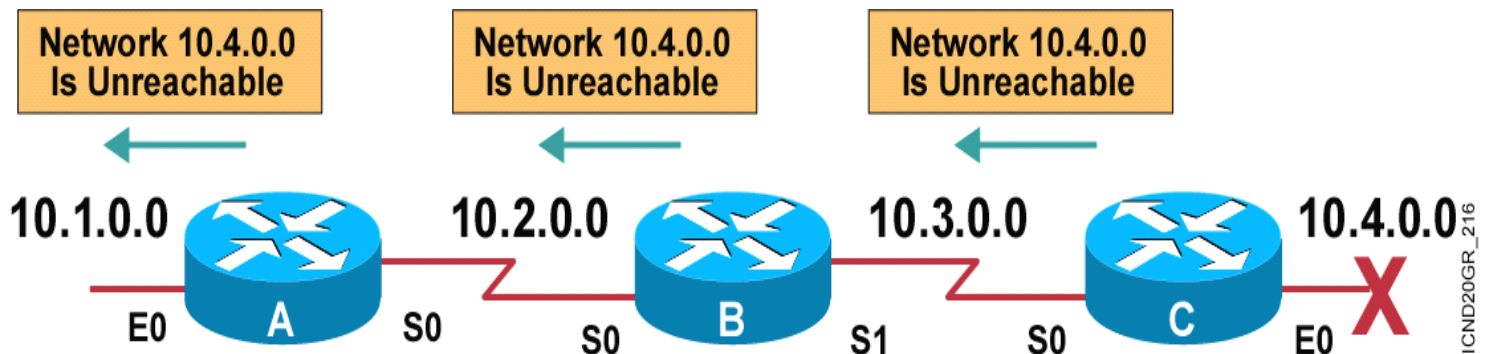
ICND20GR_214

一旦 B 收到毒抑路由之后,就把这条路由标记为 possibly down 状态.

当 B 的路由更新时间到时之后,把这条毒抑路由反向发给 C 告诉 C 我已经知道了,从而确保链路上相连的设备都知道毒抑路由的消息,确保正确性.

路由毒抑超越水平分割.也就是说水平分割对毒抑路由不起作用.

方法五: **Triggered Updates**(触发更新)



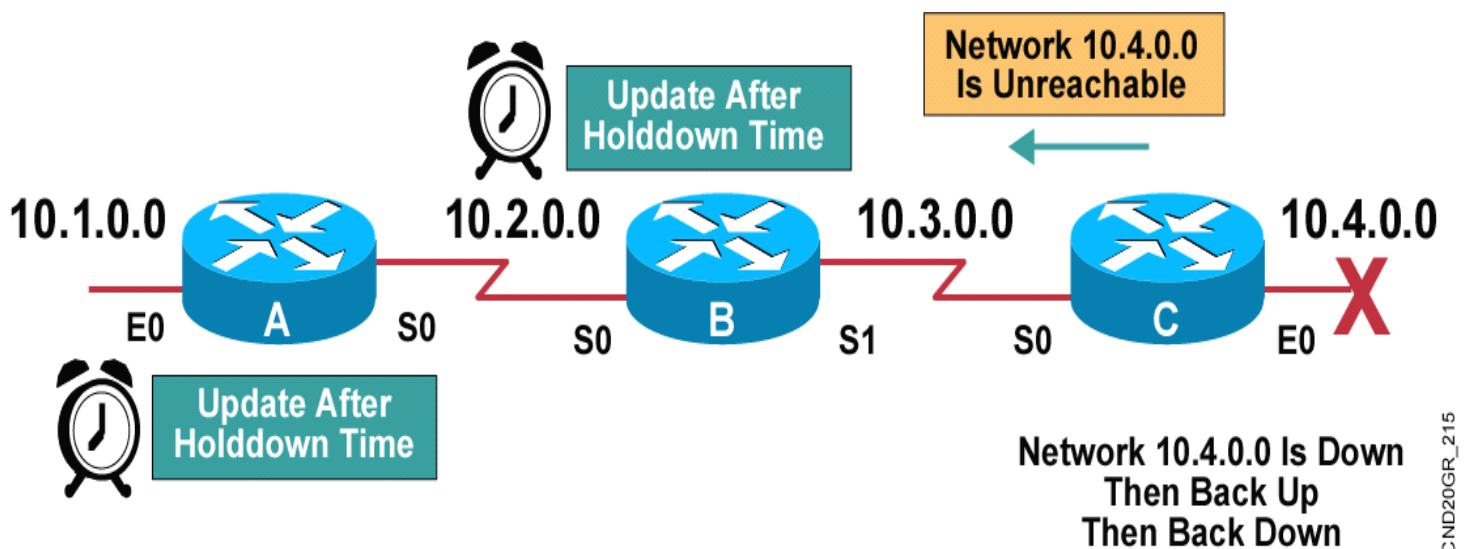
触发更新和正常的 RIP 更新不一样; 当路由表发生变化时, 更新信息立即广播给邻居路由器, 然后进行扩散, 而无需等待 30 秒的周期. 同样, 当一个路由器刚启动 RIP 时, 它广播(请求分组), 收到此消息的邻居路由器立即应答一个更新报文, 而不必等到下一个更新周期. 这样, 拓扑的改变会很快传播到全部 RIP 网络, 大大减少了收敛的时间.

方法六: **Holddown Timers**(抑制计时)

抑制计时是直接由触发更新设定的.如果一个路由条目无效,相连的路由器就会发出触发更新,而触发更新里包含的就是这条失效路由的抑制时间.

原理: 一个路由条目失效后,一段时间内这条路由处于抑制状态,即在一个特定时间段内不再接收关于这条路由的任何路由更新.然而在下列情况下,抑制时间解除:

- 1,抑制计时器超时.
- 2,收到一个更好度量值的路由更新(小于原来的度量值).
- 3,刷新时间到时.



RIP 的配置:

router rip //启动 **RIP** 进程 全局模式下

version 2 //选择版本 **2**

network 1.0.0.0

network 12.0.0.0 //宣告参与进程的接口

no auto-summary //关闭自动汇总

RIPV2 的认证

key chain cisco //定义 **key-chain** 的名字

key 1 //定义 **key** 可以是多个!

key-string cisco //定义 **key** 的密码

interface Serial0/0

ip address 12.1.1.1 255.255.255.0

ip rip authentication mode md5 //选择认证模式 **md5/text**

ip rip authentication key-chain cisco //调用定义好的 **keych**

serial restart-delay 0

clock rate 56000

DEBUG 信息

***Mar 1 00:10:43.779: RIP: received packet with MD5 authentication**

***Mar 1 00:10:43.783: RIP: ignored v2 packet from 12.1.1.1 (invalid authentication)**

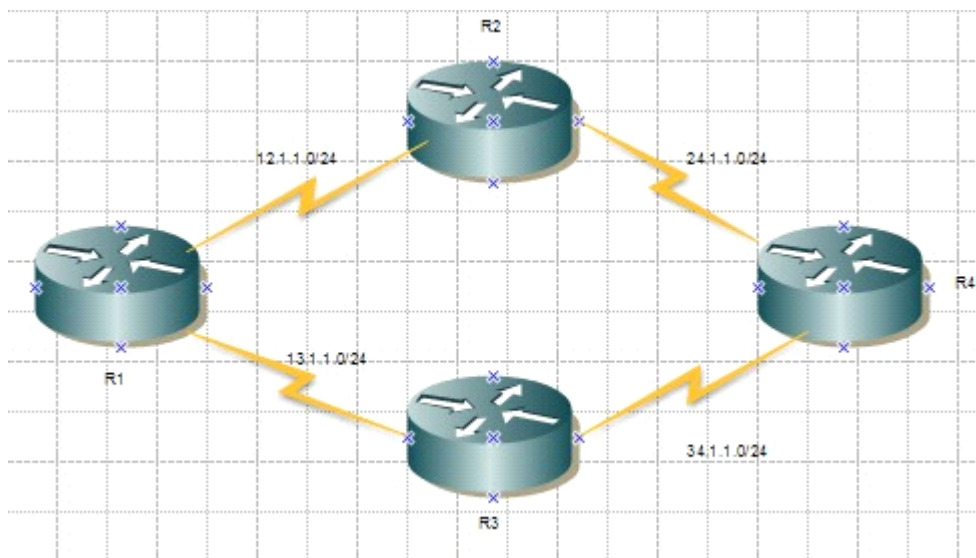
此处以显示收到 **MD5** 认证的包!!

RIP 的记数无穷大和毒化路由 **DEBUG** 信息

```
*Mar  1 00:23:46.099: RIP: received v2 update from 12.1.1.1 on Serial0/0
*Mar  1 00:23:46.103:      4.4.4.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
R2#
*Mar  1 00:23:54.867: RIP: sending v2 update to 224.0.0.9 via Serial0/0
(12.1.1.2)
*Mar  1 00:23:54.871: RIP: build update entries
*Mar  1 00:23:54.871:      2.2.2.0/24 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:23:54.875:      4.4.4.0/24 via 0.0.0.0, metric 16, tag 0
*Mar  1 00:23:54.879:      24.1.1.0/24 via 0.0.0.0, metric 1, tag 0
```

在上面我们不难看出。4.4.4.0 这条明细 down 后。马上就收到了一条跳数为 16 条的无穷大并且这条毒化路由从 s0/0 接口收到的，也从 s0/0 口发送出去了。这就证实了我们上面说的超越水平分割的特性。

RIP 负载均衡



R1#sh ip route

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, su - IS-IS summary, 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

```

      34.0.0.0/24 is subnetted, 1 subnets
R       34.1.1.0 [120/1] via 13.1.1.3, 00:00:08, Serial0/1
      1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
      2.0.0.0/24 is subnetted, 1 subnets
R       2.2.2.0 [120/1] via 12.1.1.2, 00:00:07, Serial0/0
      3.0.0.0/24 is subnetted, 1 subnets
R       3.3.3.0 [120/1] via 13.1.1.3, 00:00:08, Serial0/1
      4.0.0.0/24 is subnetted, 1 subnets
R       4.4.4.0 [120/2] via 13.1.1.3, 00:00:08, Serial0/1
      [120/2] via 12.1.1.2, 00:00:07, Serial0/0
      24.0.0.0/24 is subnetted, 1 subnets
R       24.1.1.0 [120/1] via 12.1.1.2, 00:00:07, Serial0/0
      12.0.0.0/24 is subnetted, 1 subnets
C       12.1.1.0 is directly connected, Serial0/0
      13.0.0.0/24 is subnetted, 1 subnets
C       13.1.1.0 is directly connected, Serial0/1
```

我们看到去往 **4.4.4.0** 出现了两个下一条。分别是 **13.1.1.3** 和

12.1.1.2 跳数都是 **2** 跳。这里出现了负载均衡。我们来看看效果

Reply to request 3 (144 ms). Received packet has options

Total option bytes= 40, padded length=40

Record route:

```

(12.1.1.1)
(24.1.1.2)
(4.4.4.4)
(34.1.1.4)
(13.1.1.3)
(1.1.1.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
```

End of list

Reply to request 4 (140 ms). Received packet has options

Total option bytes= 40, padded length=40

Record route:

(13.1.1.1)

(34.1.1.3)

(4.4.4.4)

(24.1.1.4)

(12.1.1.2)

(1.1.1.1) <*>

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

End of list

可以看出。第一个包是从 **12.1.1.2** 这个下一跳发出去的。回来是从 **13.1.1.3** 这个下一跳回来的。而第二个包正好相反，这就出现了两条链路负载均衡。

RIP 被动接口和单波更新。拓扑还是上图。我们将 **R1** 的所有接口改成被动接口。来看看实验效果

R1(config-router)#passive-interface s0/0 //设置 **s0/0** 为被动接口

```
*Mar  1 01:21:50.199: RIP: received v2 update from 13.1.1.3 on Serial0/1
*Mar  1 01:21:50.203:      3.3.3.0/24 via 0.0.0.0 in 1 hops
*Mar  1 01:21:50.207:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar  1 01:21:50.211:      24.1.1.0/24 via 0.0.0.0 in 2 *Mar  1 01:21:21.675: RIP:
received v2 update from 12.1.1.2 on Serial0/0
*Mar  1 01:21:21.679:      2.2.2.0/24 via 0.0.0.0 in 1 hops
*Mar  1 01:21:21.683:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar  1 01:21:21.687:      24.1.1.0/24 via 0.0.0.0 in 1 hops
*Mar  1 01:21:21.687:      34.1.1.0/24 via 0.0.0.0 in 2 hops
*Mar  1 01:21:22.459: RIP: received v2 update from 13.1.1.3 on Serial0/1
*Mar  1 01:21:22.463:      3.3.3.0/24 via 0.0.0.0 in 1 hops
*Mar  1 01:21:22.467:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar  1 01:21:22.467:      24.1.1.0/24 via 0.0.0.0 in 2 hops
*Mar  1 01:21:22.471:      34.1.1.0/24 via 0.0.0.0 in 1 hops
R1#
*Mar  1 01:21:50.199: RIP: received v2 update from 13.1.1.3 on Serial0/1
*Mar  1 01:21:50.203:      3.3.3.0/24 via 0.0.0.0 in 1 hops
```

```
*Mar 1 01:21:50.207:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar 1 01:21:50.211:      24.1.1.0/24 via 0.0.0.0 in 2 hops
*Mar 1 01:21:50.211:      34.1.1.0/24 via 0.0.0.0 in 1 hops
*Mar 1 01:21:51.039: RIP: received v2 update from 12.1.1.2 on Serial0/0
*Mar 1 01:21:51.043:      2.2.2.0/24 via 0.0.0.0 in 1 hops
*Mar 1 01:21:51.047:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar 1 01:21:51.051:      24.1.1.0/24 via 0.0.0.0 in 1 hops
*Mar 1 01:21:51.051:      34.1.1.0/24 via 0.0.0.0 in 2 hops
R1#
*Mar 1 01:21:58.859: RIP: received v2 update from 12.1.1.2 on Serial0/0
*Mar 1 01:21:58.859:      2.2.2.0/24 via 0.0.0.0 in 1 hops
*Mar 1 01:21:58.863:      3.3.3.0/24 via 0.0.0.0 in 3 hops
*Mar 1 01:21:58.867:      4.4.4.0/24 via 0.0.0.0 in 2 hops
*Mar 1 01:21:58.871:      13.1.1.0/24 via 0.0.0.0 in 3 hops
*Mar 1 01:21:58.871:      24.1.1.0/24 via 0.0.0.0 in 1 hops
*Mar 1 01:21:58.875:      34.1.1.0/24 via 0.0.0.0 in 2 hop
```

可以看出.经过了 **36** 秒的时间内本地接口只是收到了更新.却没有向外发送更新.而本地路由表的表象也发生了变化

Gateway of last resort is not set

```
      34.0.0.0/24 is subnetted, 1 subnets
R      34.1.1.0 [120/1] via 13.1.1.3, 00:00:00, Serial0/1
      1.0.0.0/24 is subnetted, 1 subnets
C      1.1.1.0 is directly connected, Loopback0
      2.0.0.0/24 is subnetted, 1 subnets
R      2.2.2.0 [120/1] via 12.1.1.2, 00:00:11, Serial0/0
      3.0.0.0/24 is subnetted, 1 subnets
R      3.3.3.0 [120/1] via 13.1.1.3, 00:00:00, Serial0/1
      4.0.0.0/24 is subnetted, 1 subnets
R      4.4.4.0 [120/2] via 13.1.1.3, 00:00:00, Serial0/1
           [120/2] via 12.1.1.2, 00:00:11, Serial0/0
      24.0.0.0/24 is subnetted, 1 subnets
R      24.1.1.0 [120/1] via 12.1.1.2, 00:00:11, Serial0/0
      12.0.0.0/24 is subnetted, 1 subnets
C      12.1.1.0 is directly connected, Serial0/0
      13.0.0.0/24 is subnetted, 1 subnets
C      13.1.1.0 is directly connected, Serial0/1
```

本地路由表的表象里面定期器的直已经到 **0** 了.要解决这个命令我们需要使用一条命令来手动指定邻居.让其向邻居发送路由更新.

R1(config-router)#neighbor 13.1.1.3


```
*Mar 1 01:27:33.931: RIP: sending v2 flash update to 12.1.1.2 via Serial0/0
(12.1.1.1)
*Mar 1 01:27:33.935: RIP: build flash update entries - suppressing null update
*Mar 1 01:27:33.939: RIP: sending v2 flash update to 13.1.1.3 via Serial0/1
(13.1.1.1)
*Mar 1 01:27:33.939: RIP: build flash update entries
*Mar 1 01:27:33.939: 2.2.2.0/24 via 0.0.0.0, metric 2, tag 0
*Mar 1 01:27:33.939: 24.1.1.0/24 via 0.0.0.0, metric 2, tag 0
R1#
*Mar 1 01:27:38.427: RIP: sending v2 update to 13.1.1.3 via Serial0/1 (13.1.1.1)
*Mar 1 01:27:38.431: RIP: build update entries
*Mar 1 01:27:38.431: 1.1.1.0/24 via 0.0.0.0, metric 1, tag 0
*Mar 1 01:27:38.435: 2.2.2.0/24 via 0.0.0.0, metric 2, tag 0
*Mar 1 01:27:38.439: 12.1.1.0/24 via 0.0.0.0, metric 1, tag 0
*Mar 1 01:27:38.439: 24.1.1.0/24 via 0.0.0.0, metric 2, tag 0
```

上面的 **debug** 信息清楚的显示了 **.Rip** 的更新报文并没有发送到组播地址 **224.0.0.9** 而是直接发向了我指定的邻居接口 **13.1.1.3/12.1.1.2** 被动接口的主要目的就是节约链路上的带宽.减少 **RIP** 更新向不必要的接口发送.

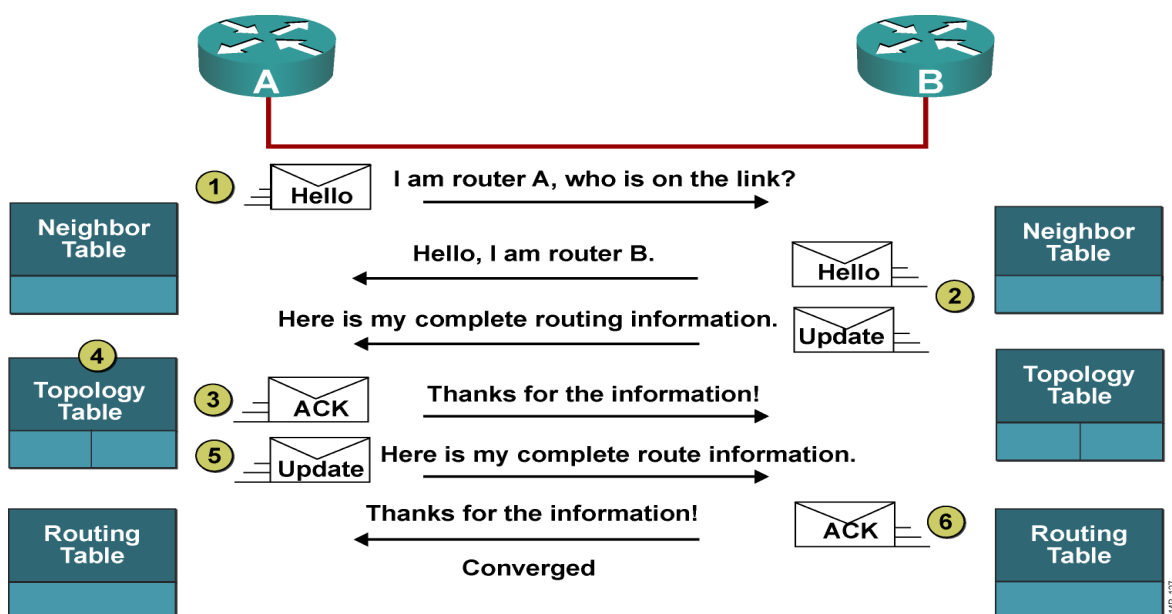
第二部分 **EIGRP** 增强型内部网关协议

Eigrp 协议是 **lgpr** 协议的一个高级扩展,增加了无类,自动汇总等功能.等价不等价负载均衡等。**Eigrp** 作为 **cisco** 的私有协议.以其快速的收敛、天生无环的保障、简单的配置、管理等特性而独步天下，下面我们将来讨论收敛最快速的路由选择协议。

Eigrp 的特性:

- 1>无类别路由选择协议支持 **vlsn** 更新携带 **netmask**
- 2>快速会聚，使用 **DUAL** 算法
- 3>部分更新，只有在拓扑发生变化时才会发送更新
- 4>多协议支持，可以和 **ip ipx appletalk** 等协议实现无缝连接
- 5>带毒性逆转的水平分割

Eigrp 使用的四种技术



2>可靠传输协议

3>DUAL 有限状态机

4>协议无关模块

Eigrp 的三张表

1>邻居表

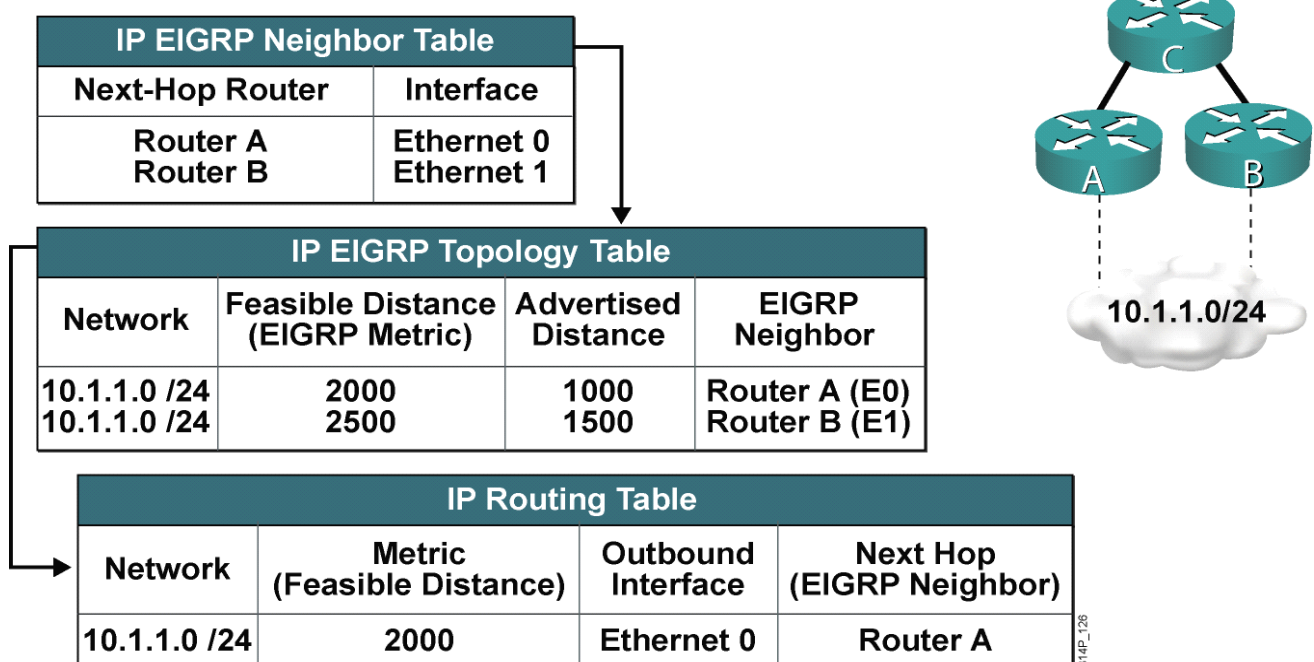
使用 **hello** 来发现和维护邻居关系,此表中包含:邻居的地址,可达邻居的端口

2>拓扑表

发现邻居后,将向邻居发送一个更新,告诉其有关自己的邻居信息,同时也将从邻居那收到有关的更新信息,将这些信息保存起来的表就叫做拓扑表.

3>路由表

通过拓扑表.使用 **DUAL** 算法计算出来得到前往特定网络的最佳



Eigrp 的报文:

Hello 报文: 用来维系邻居见的关系, 组播发送 **224.0.0.10** 不需要确认。

Update 报文: 提供路由器会聚时使用的路由, 只发送给受影响的路由器。

查询: 路由器计算路由未找到可行后继路由器时用来向邻居发送请求查询, 常用组播方式, 但也可以是单播。如果为单播则为可靠传输。

应答: 用来响应查询, 单播方式可靠传输

ACK: 用于确认更新, 查询和应答, 单播方式

Eigrp 的定时器:

Hello 时间 在大于 **T1** 线路上 (**1.544M**) **5S** 一次

在低与 **T1** 线路上 **60S** 一次

可以使用

Holddown time default 为 **3** 倍的 **hello time**

RTO 计时器: 如果 **RTO** 定时器到时后仍未收到 **ACK** 消息。则重传可靠分组消息, 直到 **16** 次后仍未收到 **ACK**, 则判断此邻居失效。

这个定时器主要作用是避免了网络的不稳定使得查询或者其他的一些消息在传输过程中丢失, 使用重传机制可以避免这类问题引起的网络不稳定性。

Eigrp 的度量:

Eigrp 的度量不象前面讲过的 **RIP** 那样简单。 **Eigrp** 的度量是充分结

合链路的信息。而不是简单的以条数来决定最佳路径的选择，从而避免了 **RIP** 中的有可能选择的可能不是最好的传输线路。

决定 **eigrp** 的度量的因素有 5 个分别是：

带宽：源去往目的之间的最小带宽

延迟：路径上所有延迟之和，也就是源去往目的之间的所有延迟之和

可靠性：源去往目的之间的最低可靠性

负载：源去往目的之间的最大负载

最大传输单元(MTU):源去往目的之间最小的 MTU

而且 **eigrp** 还使用了一系列的变量来加权求和

Default: K1=K3=1 K2=K4=K5=0

若 **K5=0**

则计算 **metric** 的公式为：

$M = K1 * \text{带宽} + [(K2 * \text{带宽}) / (256 - \text{负载})] + K3 * \text{延迟}$ ①

Default metric 的公式为：

$M = 256 * (\text{带宽} + \text{延迟})$

若 **K5!=0**

$M = m * (K5 / (\text{可靠性} + K4))$ **m** 为 1 式中求得的 **M**

Eigrp 的快速 DUAL

AD 通告距离：邻居路由器到目的的度量

FD 可行距离：本路由器到目的的度量，是 **AD** 和自己到达邻居的度量的和

Eigrp 的后继路由和可行后继路由

后继路由也就是加入路由表的路由条目。可行后继路由 **FS** 是备用路由条目。只有在 **successor** 发生故障 **down** 后才有可能启用 **FS**。当然可以使用不等价复杂均衡来使用 **FS**。正是因为 **FS** 的存在。使得 **eigrp** 有着快速收敛的特性。

存在 **FS** 的一个条件:

FD > successor 的 AD 那么此路由条目才会成为 **FS**

通过修改 **eigrp** 的 **varl** 来达到不等价负载均衡,最大 **eigrp** 支持 16 条等价/非等价负载均衡。

Eigrp 在特定链路下的设置

FR 链路

FR point2point 带宽设置为承诺速率 **CIR**

FR mutipoint 带宽设置为承诺速率之和

若永久虚链路 **PVC** 的 **CIR** 各不相同,则应该将带宽设置为最低的 **CIR** 和 **PVC** 条数的乘积。

Default,eigrp 最多占用据诶口带宽的 **50%** 来维持 **eigrp** 自己的运转,可以在接口下使用 **ip bandwidth-percent** 命令来修改 **EIGRP** 占用的带宽比例。

Eigrp 的认证

Eigrp 支持 **MD5** 认证,认证方法和 **RIP** 基本类似.将在后面以实验的方式给出具体的配置。

最后一个 **EIGRP** 的 **SIA** 状态,**state in active** 造成此状态的原因如下:

若 **successor** 不可用,而又无 **FS**,那 **router** 将向邻居发送查询报文以查找替代的条目,查询一般以组播的方式向除以前的 **successor** 以外的所有接口发送,若邻居有替代路由,则发送一个包含路由的应答消息来以确认,若邻居无替代路由,则向其邻居查询替代路由,**EIGRP** 收到每个查询消息的 **ACK** 后才会转为被动状态。

此时问题就出来了.如果去往邻居的链路出现了单向故障或者路由器处理不过来又或者中途丢包,或者其他的一些问题,不能够收到 **ACK** 的话.那么查询路由器将一直不能转为被动状态,**eigrp** 也就一直不能够收敛。

防范 **SIA**

Eigrp 包头中增加了 **SIA**-查询,**SIA**-应答来确保 **router** 的主动定时器达到一半时,若仍未收到邻居的应答,则发送 **SIA**-查询消息来确认邻居路由器是否正常。

主动定时器:**default 3min** 通过 **time active-time** 修改。

使用 **Eigrp** 末节路由器来防范 **SIA**

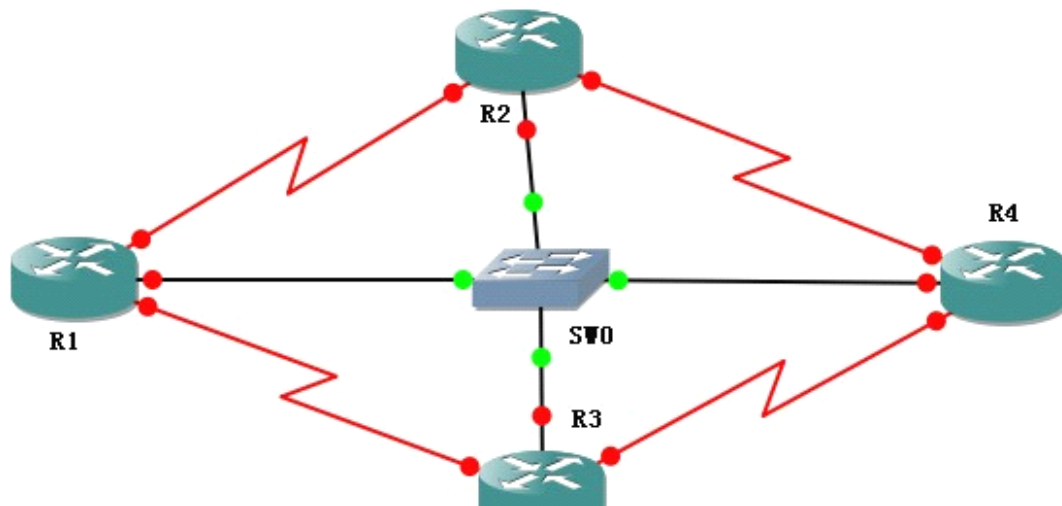
此技术是通过路由器的查询消息不进入 **EIGRP** 的末节路由器.从而间接的起到避免 **SIA** 状态的可能性,

使用汇总信息来防范 **SIA**

使用路由汇总,能够把明细路由汇总成一条汇总路由,也是通过减少查询消息的范围来防范 **SIA** 状态。

以上所有的问题都将通过下面的实验来做详细的解答:

实验拓扑:



1>eigrp 的基本配置

router eigrp 100 //选择 eigrp 进程 AS 号为 100 其中 AS 范围是 1-65536

network 1.1.1.0 0.0.0.255

network 12.1.1.0 0.0.0.255 //使用 network 命令选考参与进程的接口

no auto-summary //关闭自动汇总功能

Eigrp 的认证:

key chain cisco //定义 keychain

key 1 //定义 key 序列号

key-string cisco //定义 key 密钥

interface Serial0/0

ip address 12.1.1.1 255.255.255.0

ip authentication mode eigrp 100 md5 //选择认证模式 Eigrp 只支持 MD5

ip authentication key-chain eigrp 100 cisco //调用 keychain

Debug 信息

```
*Mar 1 00:27:56.767: EIGRP: Sending HELLO on Serial0/0
*Mar 1 00:27:56.771: AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iibQ un/rely 0/0
*Mar 1 00:27:57.043: EIGRP: Sending HELLO on Loopback0
*Mar 1 00:27:57.047: AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iibQ un/rely 0/0
*Mar 1 00:27:57.055: EIGRP: Received HELLO on Loopback0 nbr 1.1.1.1
*Mar 1 00:27:57.059: AS 100, Flags 0x0, Seq 0/0 idbQ 0/0
*Mar 1 00:27:57.099: EIGRP: received packet with MD5 authentication, key id = 1
*Mar 1 00:27:57.103: EIGRP: Received HELLO on Serial0/0 nbr 12.1.1.2
*Mar 1 00:27:57.103: AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iibQ un/rely 0/0
peerQ un/rely 0/0
```

此处标注地方为 **hello** 包已经被 **MD5** 认证.而且 **key-id** 就是我们定义的 **1**.

Eigrp 的 FS 与不等价负载均衡:

```
R1#sh ip eigrp topology
```

```
IP-EIGRP Topology Table for AS(100)/ID(1.1.1.1)
```

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
r - reply Status, s - sia Status

```
P 1.1.1.0/24, 1 successors, FD is 128256
  via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 156160
  via 120.1.1.2 (156160/128256), FastEthernet1/0
  via 12.1.1.2 (2297856/128256), Serial0/0
P 12.1.1.0/24, 1 successors, FD is 2169856
  via Connected, Serial0/0
P 120.1.1.0/24, 1 successors, FD is 28160
  via Connected, FastEthernet1/0
```

不难看出 **2.2.2.0/24** 这个我网段有两条路径可以到达,**eigrp** 已经选出了一个 **successor** **FD** 为 **15610** 的那条,也就是 **120.1.1.2** 那个下一跳.现在我们通过修改 **eigrp** 的 **variance** 让这两条都加入路由表.

命令:**R1(config-router)#variance 20**

Variance 的计算为 **FS** 的 **FD/successor** 的 **FD** 比出来的数取整.

```
1.0.0.0/24 is subnetted, 1 subnets
```

C 1.1.1.0 is directly connected, Loopback0

2.0.0.0/24 is subnetted, 1 subnets

D 2.2.2.0 [90/156160] via 120.1.1.2, 00:03:48, FastEthernet1/0

[90/2297856] via 12.1.1.2, 00:03:48, Serial0/0

12.0.0.0/24 is subnetted, 1 subnets

C 12.1.1.0 is directly connected, Serial0/0

120.0.0.0/24 is subnetted, 1 subnets

C 120.1.1.0 is directly connected, FastEthernet1/0

2.2.2.0 这条网络已经实现了负载均衡.而且度量是不一样的.这就是非等价负载均衡.

Eigrp 的汇总和 stub 区域:

汇 总 命 令 :R3(config-if)#ip summary-address eigrp 100

192.168.1.0 255.255.252.0

路由表信息:

Gateway of last resort is not set

1.0.0.0/24 is subnetted, 1 subnets

C 1.1.1.0 is directly connected, Loopback0

2.0.0.0/24 is subnetted, 1 subnets

D 2.2.2.0 [90/2297856] via 12.1.1.2, 00:02:15, Serial0/0

3.0.0.0/24 is subnetted, 1 subnets

D 3.3.3.0 [90/2297856] via 13.1.1.3, 00:02:15, Serial0/1

130.1.0.0/24 is subnetted, 1 subnets

D 130.1.1.0 [90/2172416] via 13.1.1.3, 00:02:15, Serial0/1

[90/2172416] via 12.1.1.2, 00:02:15, Serial0/0

12.0.0.0/24 is subnetted, 1 subnets

C 12.1.1.0 is directly connected, Serial0/0

13.0.0.0/24 is subnetted, 1 subnets

C 13.1.1.0 is directly connected, Serial0/1

D 192.168.0.0/22 [90/2297856] via 13.1.1.3, 00:00:03, Serial0/1

Eigrp stub 区域

R3(config-router)#eigrp stub ?

connected Do advertise connected routes

leak-map Allow dynamic prefixes based on the leak-map

receive-only Set IP-EIGRP as receive only neighbor

redistributed Do advertise redistributed routes

```
static          Do advertise static routes
summary        Do advertise summary routes
<cr>
```

Eigrp 的默认路由:

Eigrp 的默认路由产生有三种方法

1> 重分发默认静态路由进 eigrp 两个步骤

R1(config)#ip route 0.0.0.0 0.0.0.0 s0/1 //配置缺省路由

R1(config)#router eigrp 100

R1(config-router)#redistribute static //将其重分发进 eigrp

R2 的路由表

```
12.0.0.0/24 is subnetted, 1 subnets
C    12.1.1.0 is directly connected, Serial0/0
13.0.0.0/24 is subnetted, 1 subnets
D    13.1.1.0 [90/2172416] via 130.1.1.3, 00:41:20, FastEthernet1/0
D*EX 0.0.0.0/0 [170/2681856] via 12.1.1.1, 00:00:08, Serial0/0
D    192.168.0.0/22 [90/130816] via 130.1.1.3, 00:07:09, FastEthernet1/0
```

这时 **R2** 上就自动产生了一条默认路由标识为 **D*EX**.

2>使用 network 命令 两步

R1(config)#ip route 0.0.0.0 0.0.0.0 s0/1 //建立缺省路由

R1(config)#router eigrp 100

R1(config-router)#network 0.0.0.0 0.0.0.0 //network 进 eigrp

R2 的路由表:

```
12.0.0.0/24 is subnetted, 1 subnets
C    12.1.1.0 is directly connected, Serial0/0
13.0.0.0/24 is subnetted, 1 subnets
D    13.1.1.0 [90/2172416] via 130.1.1.3, 00:00:01, FastEthernet1/0
D* 0.0.0.0/0 [90/2681856] via 12.1.1.1, 00:00:01, Serial0/0
D    192.168.0.0/22 [90/130816] via 130.1.1.3, 00:00:02, FastEthernet1/0
```

R2 的路由表产生一条 D*路由.

3>ip default-network 命令宣告默认路由

要注意.本路由器必须要有去往 **network** 的路由

```
R1(config)#ip default-network 172.16.0.0 //宣告默认网络
```

```
R1(config)#ip route 172.16.0.0 255.255.0.0 null 0 //为了使本路由器产生  
去往 172.16.0.0 的路由
```

```
R1(config)#router eigrp 100
```

```
R1(config-router)#network 172.16.0.0 //将默认网络宣告进 eigrp
```

R2 的路由表:

```
D* 172.16.0.0/16 [90/2169856] via 12.1.1.1, 00:00:03, Serial0/0
```

```
130.1.0.0/24 is subnetted, 1 subnets
```

```
C 130.1.1.0 is directly connected, FastEthernet1/0
```

```
12.0.0.0/24 is subnetted, 1 subnets
```

```
C 12.1.1.0 is directly connected, Serial0/0
```

总结:

第一种产生是通过重分发技术,所以后面会产生 **D*EX** 的外部路由,管理距离为 **170**.

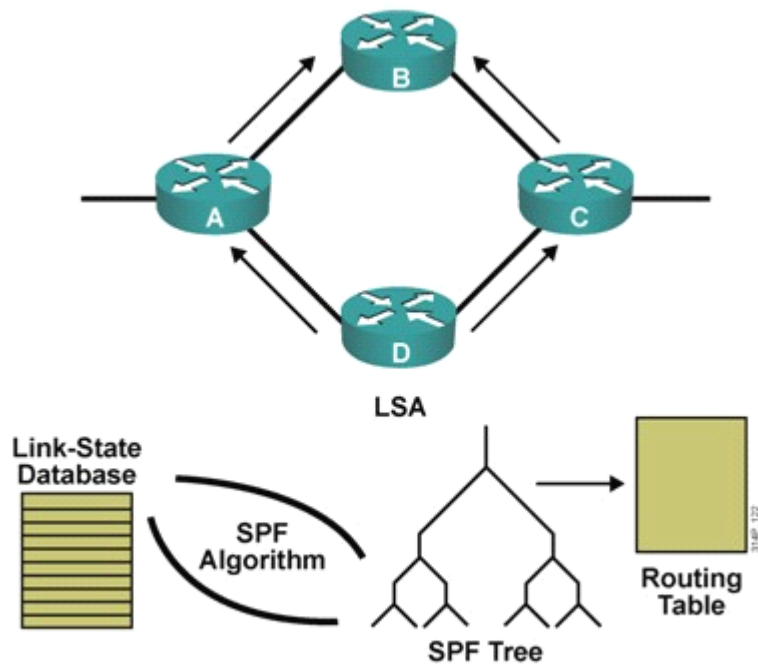
第二重是通过宣告技术,通过讲默认路由通过 **network** 宣告进 **eigrp** 的进程下面.所以产生的路由为 **D***,管理距离为**90**.

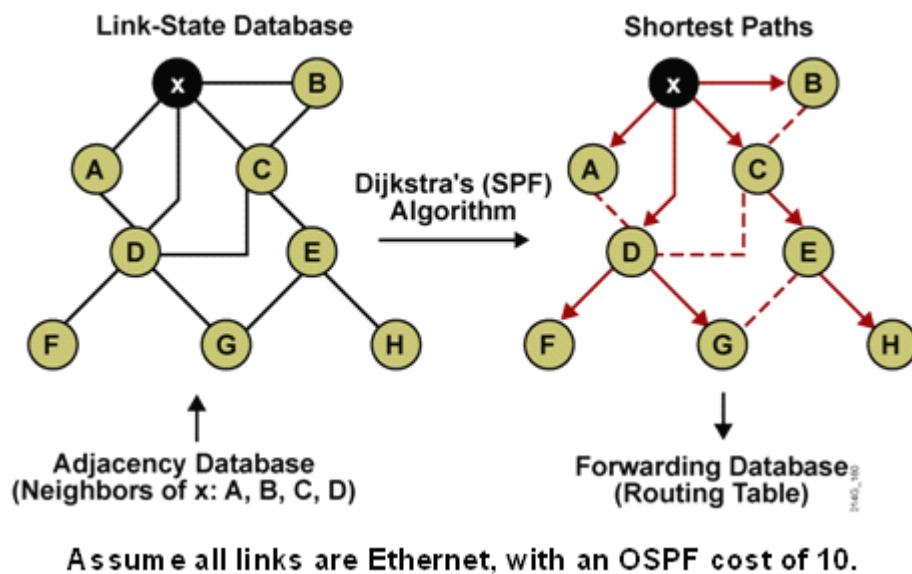
而第三种是通过默认网路的方式将流量指向一个地址,再将这个地址宣告进 **eigrp** 进程,从而会产生一个 **D*** 路由 管理距离为**90** 特别值得注意的是第三种方法产生的不是**0.0.0.0/0**的条目。

第三部分：ospf 开放式最短路径优先

Ospf 当今企业用处最广的一个路由选择协议，同时也是最难掌握的一种路由协议，通过它的多区域，多链路类型。多网络类型。多区域类型。多 **LSA** 类型等一系列的多，造就了这个协议的高扩展性，高稳定性的特点。下面就来揭开它的神秘面纱。

OSPF 是一种链路状态路由选择协议（**LINK-STATE ROUTING PROTOCOL**）采用 **SPF**（最短路径优先）算法来计算开销，一般以 10^8 /链路带宽。





Osfp 采用 **hello** 来维系邻居之间的关系，在 **T1** 带宽以上的链路上 **hello** 时间默认为 **10s**，在 **T1** 带宽以下的链路上 **hello** 时间一般为 **30s**

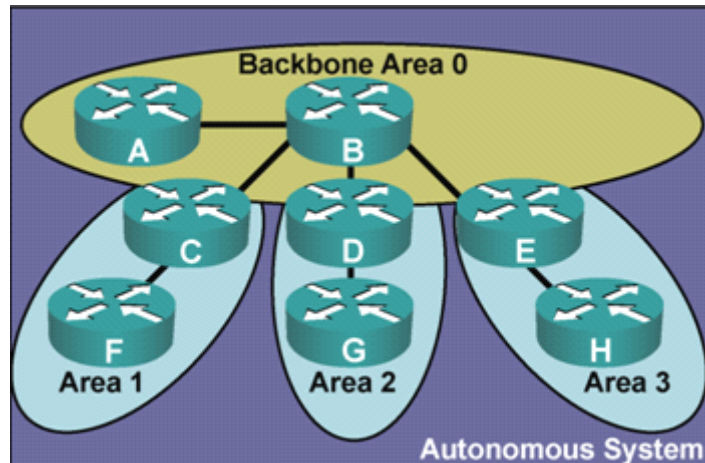
OSPF 的特性：

- (1) 快速适应网络变化
- (2) 触发式更新 当网络拓扑发生变化时发送更新，但是每 **30** 分钟需要发送一次链路状态通告，来确保区域内链路状态数据库的一致性。

链路状态通告 (**LSA**)：用来传递接口特征和邻居关系，区域内每台路由器的 **LSA** 会被其他路由器接受和保存，并且其他路由器会向自己的邻居洪范这个 **LSA**，区域内所有路由器的 **LSA** 必须相同。

LSA 洪泛到区域边界为止。

OSPF 的区域：



(1) 主干区域 (**backbone**): 也叫传输区域, 在多个非主干区域间传递数据和流量。

(2) 常规区域: 也叫非主干区域、普通区域, 每个普通区域必须和主干区域相连。

ABR (区域边界路由器): 连接多个区域, 并且其中至少包括 **area 0** 区域的路由器叫做 **ABR**

ASBR (自治系统边界路由器): 连接外部自治系统和 **OSPF** 区域的路由器

OSPF 的 packets

Hello: 用来维系邻居之间的关系, 可采用组播或者单播

DBD: 链路状态通告的一个摘要信息

LINK-STATE REQUEST: 链路状态请求, 用来请求链路状态更新

LINK-STATE UPDATE: 链路状态更新包,

LSAck 链路状态应答包

HELLO packet :

(1) router ID

(2) hello time 、 dead time *

(3) neighbors

(4) Area ID

(5) router priority *

(6) DR IP

(7) BDR IP

(8) authentication password *

(9) stub area flag *

OSPF 的 ROUTER-ID:

作用：标识一台路由器，如果只有物理接口，则为物理接口 **IP** 地址最高的，如果有 **LOOPBACK**

接口则 **LOOPback** 接口地址

可以手动使用 **router-id** 命令修改

但是要注意：

直连的两台路由器如果 **RID** 相同则无建立邻居关系

不直连的路由器如果 **RID** 相同则无法同步数据库

当 **RID** 一旦选中，则是非抢占的！

除非使用 **clear ip ospf pro** 命令清除 **ospf** 进程！

OSPF LSA TYPES

类型	谁产生	传播范围	信息/作用
1 router LSAs	各区域的路由器	本区域	包含自身的信息
2 network LSAs	DR/BDR	本区域	DR/BDR 的信息
3 summary network	ABR	整个 ospf 网络	ABR 直连区域的网络号
4 AS summary LSAs	离 ASBR 最近的 ABR	整个 ospf 网络	ASBR 的 RID
5 AS external LSAs	ASBR	整个 ospf 网络	用于到达外部网络
7 NSSA	NSSA 区域的 ASBR	整个 NSSA 区域	用于到达外部网络

Ospf 的 DR/BDR 选举过程:

在多路访问网路中 **OSPF** 要选定 **DR/BDR**,选举过程如下:

1>在路由器和邻居路由器之间简历了 **2way** 之后,将检查所有邻居路由器发送的 **hello** 包.检查 **RID**,优先级等,接着所有路由器都宣告自己是 **DR/BDR**,

2>从具有选举资格的路由器的列表中,创建一个还没有宣告为 **DR** 路由器的所有路由器列表.

3>若一个子集中有多个邻居,则优先级最高的邻居将被宣告为 **BDR**,在优先级相同的情况下,则宣告具有最高 **RID** 的路由器为 **BDR**

4>若子集中未有宣告自己为 **BDR** 的路由器,则具有最高优先级的邻居路由器将被宣告为 **BDR**.

5>如果一台或者多台路由器宣告自己为 **DR**,则具有最高优先级的路由器为 **DR**,如果优先级相同则具有最高 **RID** 的为 **DR**.

6>若没有路由器宣称自己是 **DR** 则新选举出来的 **BDR** 将成为 **DR**.

Ospf 的网络类型:

1>point2point 不选举 **DR/BDR** ->**FULL**

2>MA 多路访问网络 选举 **DR/BDR**

OSPF 的区域类型

1> stub area 到达其他区域只有 **1** 个出口,不接受 **5** 类 **LSA**

2>totally stub area 只接受 **1** 类 **2** 类 **LSA**

将区域设置成为 **STUB** 的规则:

a>所有路由器都必须为末节路由器

b>不能有 **ASBR**

c>不能为主干区域

d>不能有 **virtual-link** 穿越

若将某个区域配置成为 **stub** 则 **ABR** 会自动向内部产生一个 **3** 类的默认路由 **LSA**.

3>NSSA 区域 没有 5 类 LSA 产生一种新的 7 类 LSA

将区域配置成为 **NSSA** 区域的话,**ASBR** 产生 7 类的 **LSA**,此 **LSA** 经过 **NSSA** 区域到达普通区域时将转换成为 5 类的 **LSA** 在整个 **OSPF** 区域内传播.**NSSA** 区域的 **ABR** 不会向 **NSSA** 区域内宣告默认路由.

4>NSSA totally area:此为 **NSSA** 的一个扩展.将 3 类的 **LSA** 也拒绝掉,**ABR** 会自动产生一条默认路由引导区域内部的路由器访问外部网络.

Ospf 网络类型:

1>NBMA 非广播多路访问(default)

a>选举 **DR/BDR**,所有接口在同意子网,如果为 **hub-spoke** 模型,确保中心路由器为 **DR**

b>边缘路由器相互做 **DLCI** 映射

c>手动指定邻居

2>point-to-multipoint

a>所有接口在同意子网

b>多点 **FR** 子接口无需修改网络类型,邻接关系可正常建立但无法收到路由,仍需在多点子接口下修改网络类型。

C>边缘路由器无需做 **DLCI** 复用

3>broadcast

a>选举 **DR/BDR**

b>**DLCI** 复用

C>去保中心为 **DR**

4>point-to-point

a>不选 DR/BDR

b>hellotime 10sec

c>中心路由器划分子接口,边缘路由起不在同一子网

5>point-to-multipoint nonbroadcast 与点到多点一样.但需要手动指定邻居.

Ospf 的认证

1>邻居间 也就是接口上

2> 区域内

3>虚链路上

这一特别要声明一点,如果区域间做了认证那么虚链路必须得认证,因为虚链路是穿越区域的,而且虚链路的认证密码和模式必须得和区域认证一样!!

OSPF 的路由汇总:

(1) 区域间汇总: 在 **ABR** 上执行、将一个区域的明细路由汇总到别的 **OSPF** 区域内

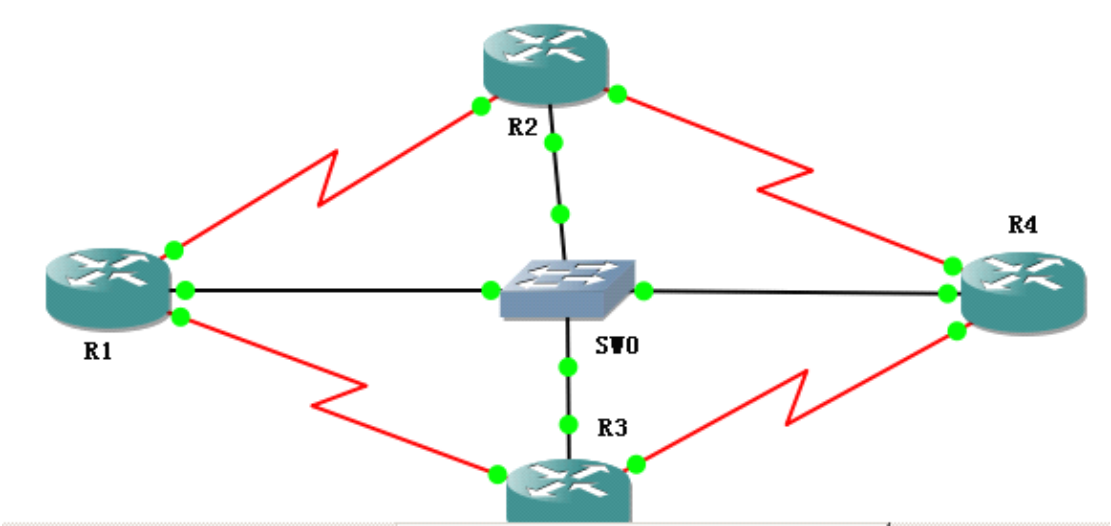
Router 模式 : Area area-id range address mask [advertise|not-advertise][cost cost]

(2) **AS** 间汇总: 在 **ASBR** 上执行、将来自 **OSPF** 外的路由汇总进 **OSPF**

Router 模式: Summary-address address mask tag tag

实验部分:

实验拓扑



Ospf 基本配置:

router ospf 1 //启动 ospf 进程 其中 **PID** 为本地有效范围 1-

42E,PID 只是本地有效

router-id 2.2.2.2 //手动制定 **RID**

network 12.1.1.0 0.0.0.255 area 0 //宣告参与 ospf 进程的接口

用反掩码来确认网络地址范围 并将这个地址划入哪个接口

network 24.1.1.0 0.0.0.255 area 1

Ospf 表项

邻居表:

Neighbor ID	Pri	State	Dead Time	Address	Interface
1.1.1.1	0	FULL/ -	00:00:34	12.1.1.1	Serial0/0
4.4.4.4	0	FULL/ -	00:00:30	24.1.1.4	Serial0/1

邻居 ID 优先级 状态 死亡时间 对方接口地址 本地接口

Ospf 的路由表

R2#sh ip route ospf

```
1.0.0.0/32 is subnetted, 1 subnets
O      1.1.1.1 [110/65] via 12.1.1.1, 00:09:50, Serial0/0
4.0.0.0/32 is subnetted, 1 subnets
O      4.4.4.4 [110/65] via 24.1.1.4, 00:09:11, Serial0/1
```

Ospf DR/BDR 选举过程

```
*Mar  1 00:16:13.767: OSPF: Nbr state is 2WAY
*Mar  1 00:16:13.839: OSPF: Rcv hello from 2.2.2.2 area 0 from
FastEthernet1/0 192.168.1.2
*Mar  1 00:16:13.839: OSPF: End of hello processing
R1#
*Mar  1 00:16:15.899: OSPF: end of Wait on interface FastEthernet1/0
*Mar  1 00:16:15.899: OSPF: DR/BDR election on FastEthernet1/0
*Mar  1 00:16:15.903: OSPF: Elect BDR 1.1.1.1
*Mar  1 00:16:15.903: OSPF: Elect DR 2.2.2.2
*Mar  1 00:16:15.903: OSPF: Elect BDR 1.1.1.1
*Mar  1 00:16:15.903: OSPF: Elect DR 2.2.2.2
*Mar  1 00:16:15.903:      DR: 2.2.2.2 (Id)   BDR: 1.1.1.1 (Id)
*Mar  1 00:16:15.903: OSPF: Send DBD to 2.2.2.2 on FastEthernet1/0 seq
0x262E opt 0x52 flag 0x7 len 32
*Mar  1 00:16:15.911: OSPF: Send hello to 224.0.0.5 area 0 on
FastEthernet1/0 from 192.168.1.1
```

从上面的 debug 信息我们可以清楚的看出 DR/BDR 的选举过程。也验证了我们前面所讲的 2WAY 状态下。先选举 BDR 然后选举 DR 的一个过程。

Ospf 的区域类型

```
router ospf 1
router-id 4.4.4.4
log-adjacency-changes
area 1 stub           //设置区域 1 为 stub 区域
network 4.4.4.0 0.0.0.255 area 1
network 24.1.1.0 0.0.0.255 area 1
```

这里我将 R2-R4 的区域 1 设置成了 STUB 区域.来 R4 上看看效果

R4 的路由表

Gateway of last resort is 24.1.1.2 to network 0.0.0.0

```
34.0.0.0/24 is subnetted, 1 subnets
C      34.1.1.0 is directly connected, Serial0/1
1.0.0.0/32 is subnetted, 1 subnets
O IA   1.1.1.1 [110/66] via 24.1.1.2, 00:01:53, Serial0/0
3.0.0.0/24 is subnetted, 1 subnets
D      3.3.3.0 [90/2297856] via 34.1.1.3, 00:02:59, Serial0/1
4.0.0.0/24 is subnetted, 1 subnets
C      4.4.4.0 is directly connected, Loopback0
24.0.0.0/24 is subnetted, 1 subnets
C      24.1.1.0 is directly connected, Serial0/0
12.0.0.0/24 is subnetted, 1 subnets
O IA   12.1.1.0 [110/128] via 24.1.1.2, 00:01:53, Serial0/0
O IA   192.168.1.0/24 [110/65] via 24.1.1.2, 00:01:53, Serial0/0
O*IA 0.0.0.0/0 [110/65] via 24.1.1.2, 00:01:54, Serial0/0
```

多了一条 **O*IA** 路由这缺省路由帮助本区域内的路由器找到去往外部的路径

完全末节区域

完全末节知识在末节区域的命令后面加个 **no-summary** 关键字.看看效果

Gateway of last resort is 24.1.1.2 to network 0.0.0.0

```
34.0.0.0/24 is subnetted, 1 subnets
C      34.1.1.0 is directly connected, Serial0/1
3.0.0.0/24 is subnetted, 1 subnets
D      3.3.3.0 [90/2297856] via 34.1.1.3, 00:05:16, Serial0/1
4.0.0.0/24 is subnetted, 1 subnets
C      4.4.4.0 is directly connected, Loopback0
24.0.0.0/24 is subnetted, 1 subnets
C      24.1.1.0 is directly connected, Serial0/0
O*IA 0.0.0.0/0 [110/65] via 24.1.1.2, 00:00:00, Serial0/0
```

和上面不同的是.**IA** 路由都被干掉了.也就是 **3 类 LSA** 给干掉了.

NSSA 区域

使用命令 **area X nssa | no-summary**

```
34.0.0.0/24 is subnetted, 1 subnets
O N2   34.1.1.0 [110/20] via 24.1.1.4, 00:00:02, Serial0/1
1.0.0.0/32 is subnetted, 1 subnets
O      1.1.1.1 [110/2] via 192.168.1.1, 00:01:10, FastEthernet1/0
2.0.0.0/24 is subnetted, 1 subnets
C      2.2.2.0 is directly connected, Loopback0
3.0.0.0/24 is subnetted, 1 subnets
O N2   3.3.3.0 [110/20] via 24.1.1.4, 00:00:03, Serial0/1
4.0.0.0/32 is subnetted, 1 subnets
O      4.4.4.4 [110/65] via 24.1.1.4, 00:00:03, Serial0/1
24.0.0.0/24 is subnetted, 1 subnets
C      24.1.1.0 is directly connected, Serial0/1
12.0.0.0/24 is subnetted, 1 subnets
C      12.1.1.0 is directly connected, Serial0/0
C      192.168.1.0/24 is directly connected, FastEthernet1/0
R2#
```

出现了 **N2** 路由.这就是类型 **7** 产生的路由条目.**N2 N1** 再后面的重分发中会中点讲述.顺便看下 **OSPF** 的数据库.

Type-7 AS External Link States (Area 1)

Link ID Tag	ADV Router	Age	Seq#	Checksum
3.3.3.0	4.4.4.4	149	0x80000001 0x003736 0	
34.1.1.0	4.4.4.4	149	0x80000001 0x00D081 0	

Type-5 AS External Link States

Link ID Tag	ADV Router	Age	Seq#	Checksum
3.3.3.0	2.2.2.2	143	0x80000001 0x000877 0	
34.1.1.0	2.2.2.2	143	0x80000001 0x00A1C2 0	

R2#

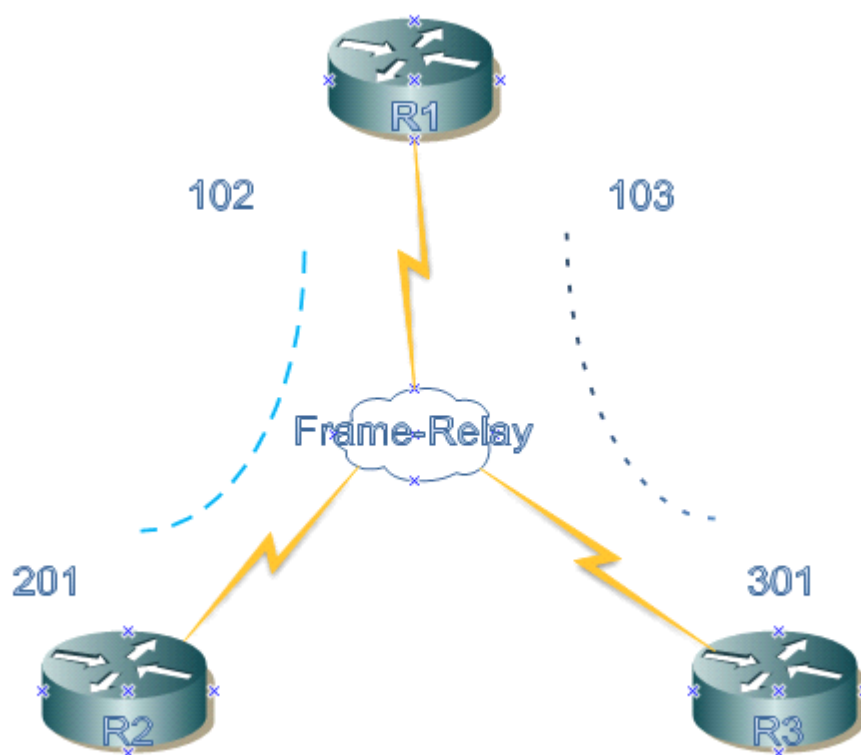
有类型 **7** 的信息.也有类型 **5** 的信息.因为我是站在一个 **ABR** 上的.**ABR** 会将类型 **7** 的信息转换成类型 **5** 的 **LSA** 这里得到了验证.

OSPF 的认证 **OSPF** 认证有很多中.有接口认证,区域认证虚链路认证等.这里我主要做一下区域认证.

```
ip ospf message-digest-key 1 md5 cisco //接口下定义认证密钥和认证模式
area 0 authentication message-digest //协议模式下调用密钥
*Mar  1 00:42:02.351: OSPF: rcv. v:2 t:1 l:48 rid:2.2.2.2
      aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x3C7ECE27 from Serial0/0
```

Ospf 在 **FR** 网络上的应用：

拓扑：



FR 作为一种非广播网络，我在这里就拿来模拟几种网络类型，看一些简单的配置，以及需要注意的事项。

1> point-to-point

```
interface Serial0/0.2 multipoint //定义子接口
ip address 12.1.1.1 255.255.255.0
ip ospf network point-to-point //设定网络类型
frame-relay map ip 12.1.1.1 102 broadcast
frame-relay map ip 12.1.1.2 102 broadcast //映射
interface Serial0/0.3 multipoint
```

```
ip address 13.1.1.1 255.255.255.0
ip ospf network point-to-point
frame-relay map ip 13.1.1.1 103 broadcast
frame-relay map ip 13.1.1.3 103 broadcast
```

```
Serial0/0.2 is up, line protocol is up
Internet Address 12.1.1.1/24, Area 0
Process ID 1, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost:
64
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

邻居表

Neighbor ID	Pri	State	Dead Time	Address	Interface
3.3.3.3	0	FULL/ -	00:00:33	13.1.1.3	Serial0/0.3
2.2.2.2	0	FULL/ -	00:00:37	12.1.1.2	Serial0/0.2

路由表

Gateway of last resort is not set

```
1.0.0.0/24 is subnetted, 1 subnets
C    1.1.1.0 is directly connected, Loopback0
2.0.0.0/24 is subnetted, 1 subnets
O    2.2.2.0 [110/65] via 12.1.1.2, 00:00:07, Serial0/0.2
3.0.0.0/24 is subnetted, 1 subnets
O    3.3.3.0 [110/65] via 13.1.1.3, 00:00:07, Serial0/0.3
12.0.0.0/24 is subnetted, 1 subnets
C    12.1.1.0 is directly connected, Serial0/0.2
13.0.0.0/24 is subnetted, 1 subnets
C    13.1.1.0 is directly connected, Serial0/0.3
```

通过上面几个表项我们可以看出 **point-to-point** 的网络类型 **hello** 时间 **10** 秒.直接 **FULL** 不选举 **DR/BDR**.不产生 **32** 位的主机路由证实了我们上面的理论.

2>point-to-multipoint

R1#sh ip ospf int s0/0

```
Serial0/0 is up, line protocol is up
Internet Address 192.168.1.1/24, Area 0
Process ID 1, Router ID 1.1.1.1, Network Type POINT_TO_MULTIPOINT, Cost: 64
Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT,
Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
```

接口状态为点多多点.**hello** 时间变成 **30S** 了.**dead 120 wait120**.可见

和点对点的区别.

interface Serial0/0

ip address 192.168.1.1 255.255.255.0

encapsulation frame-relay

ip ospf network point-to-multipoint //修改网络类型为点对多点

no arp frame-relay

frame-relay map ip 192.168.1.1 102 broadcast

frame-relay map ip 192.168.1.2 102 broadcast

frame-relay map ip 192.168.1.3 103 broadcast

no frame-relay inverse-arp

3.3.3.3 0 FULL/ - 00:01:59 192.168.1.3 Serial0/0

2.2.2.2 0 FULL/ - 00:01:36 192.168.1.2 Serial0/0

也是直接 FULL 了.并未选举 DR/BDR

1.0.0.0/32 is subnetted, 1 subnets

O 1.1.1.1 [110/65] via 192.168.1.1, 00:00:05, Serial0/0

3.0.0.0/24 is subnetted, 1 subnets

C 3.3.3.0 is directly connected, Loopback0

192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks

O 192.168.1.1/32 [110/64] via 192.168.1.1, 00:00:05, Serial0/0

C 192.168.1.0/24 is directly connected, Serial0/0

产生 32 位路由

3> NBMA 网络

interface Serial0/0

ip address 192.168.1.2 255.255.255.0

encapsulation frame-relay

ip ospf network non-broadcast //修改网络类型

ip ospf priority 0 //修改接口优先级<确保中心路由器为

DR>

no arp frame-relay

frame-relay map ip 192.168.1.1 201 broadcast

frame-relay map ip 192.168.1.2 201 broadcast

frame-relay map ip 192.168.1.3 201 broadcast //DLCI 复用

no frame-relay inverse-arp

这里注意一点 修改端口优先级来确保中心路由器被选举为 DR, 还有一个 DLCI 复用.

R2#sh ip ospf int s0/0

Serial0/0 is up, line protocol is up

Internet Address 192.168.1.2/24, Area 0

Process ID 1, Router ID 2.2.2.2, Network Type NON_BROADCAST, Cost:

64

Transmit Delay is 1 sec, State DROTHER, Priority 0

Designated Router (ID) 1.1.1.1, Interface address 192.168.1.1

No backup designated router on this network

Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5

可以看出 hello 时间和点对多点一样。

R1(config-router)#neighbor 192.168.1.3

R1(config-router)#neighbor 192.168.1.2 //手动指定邻居.因为 NBMA 网络不能传播广播和组播,ospf 的建立邻居的时候会以组播寻找邻居,需要手动指定邻居,让单播发送消息

R2#sh ip ospf nei

Neighbor ID	Pri	State	Dead Time	Address	Interface
1.1.1.1	1	FULL/DR	00:01:43	192.168.1.1	Serial0/0

NBMA 网络选举 DR/BDR.

R1#sh ip route

Gateway of last resort is not set

1.0.0.0/24 is subnetted, 1 subnets

C 1.1.1.0 is directly connected, Loopback0

2.0.0.0/32 is subnetted, 1 subnets

O 2.2.2.2 [110/65] via 192.168.1.2, 00:10:02, Serial0/0

3.0.0.0/32 is subnetted, 1 subnets

O 3.3.3.3 [110/65] via 192.168.1.3, 00:10:02, Serial0/0

C 192.168.1.0/24 is directly connected, Serial0/0

产生 32 位的路由这点与 multipoint 类似

总结:

	POINT-TO-POINT	NBMA	Point-to-multipoint
hellotime	10	30	30
指定邻居	no	yes	no
DLCI 复用	no	yes	no
32 位路由	no	yes	yes
DR/BDR	no	yes	no
子接口	yes	no	no
同一网段	no	yes	yes

第四部分 **IS-IS** 中间系统-中间系统 路由协议

ISIS 是一种链路状态基于 **OSI** 模型的一种无连接网络路由 **<CLNS>** 协议.和 **OSPF** 比较类似.

ISIS 的原理,首先解释一下 **ISIS** 用到的名词

IS--intermediate system 中间系统 可以理解为一个 **IS** 就是一个 **router**

ES-end system 终端系统 也就是 **PC**

SNPA -subnetwork point of attachment 子网连接点也就是与一个子网相连的那个接口

PDU-protocol data unit 协议数据单元比如我们说的帧、数据包等都是 **PDU**

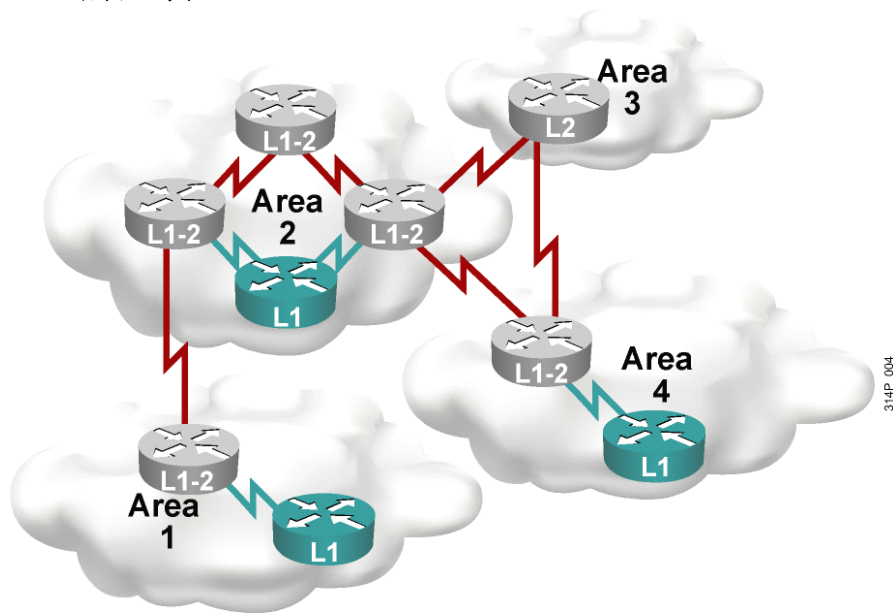
LSP-link state packet 与 **ospf** 中的 **LSA** 类似。但是一个 **LSP** 就是一个数据包。

ISIS 的 **LSDB** 类型:

L1 LSDB:装载 **L1** 的 **LSP**, 负责存储本地区域的拓扑信息;

L2 LSDB: 装载 **L2** 的 **LSP** 负责存储区域间的拓扑信息

IS-IS 的区域:



IS-IS 和 **ospf** 类似.但是也不同.**OSPF** 是以路由器为区分区域的节点.
而 **IS-IS** 是根据路由器的类型来区分区域的.

IS-IS 的路由器类型

L1-类似于 **ospf** 的非骨干路由器也就是区域内部路由器

L2-类似于 **ospf** 中的骨干路由器

L1/2-类似于 **ospf** 中的 **ABR**

Cisco default 时所有路由器都是 **L1/2** 的.可使用 **is-type** 来修改路由器的类型

和 **ospf** 相比,**IS-IS** 的区域边界显得不是那么明显.但是区域间通信必须经过 **L2** 区域.

L1 区域:区域内每台 **L1** 路由器维护相同的 **LSDB**,**default L1/2** 不会通告 **L2** 的路由给 **L1** 的 **router**,这样.**L1** 区域相当于 **ospf** 的完全末梢区域.

L2 区域:相当于 **ospf** 中的传输区域.

IS-IS 网络实体标题

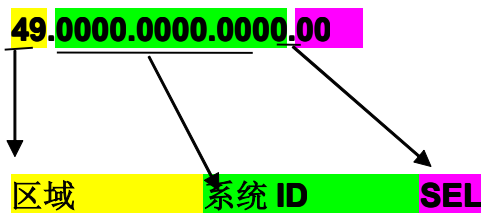
由于 **ISIS** 路由器完全处于一个单一的区域内,因此区域 **ID** 将整个路由器

关联起来,**default**,一个路由器可以最多存有 **3** 个区域地址,**IOS** 中使用 **max-area-addresses** 可将这个数字最大到 **254** 个.

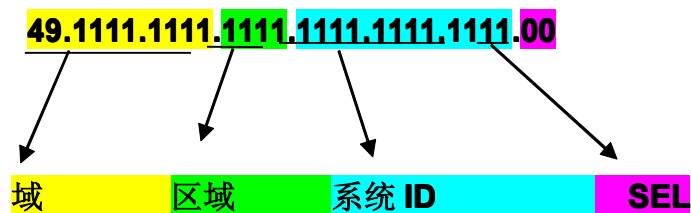
区域 ID	系统 ID	SEL
用于区域间的路由选择	用于区域内的路由选择	

这个实体标题总共有 **3** 种格式:

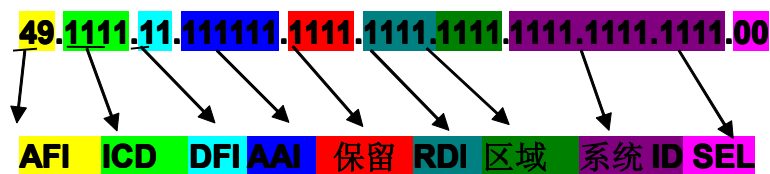
区域 ID/系统 ID 格式:



OSI NSAP 格式:



GOSIP NSAP 格式:



AFI:权限/格式标识符

ICD:国际代码标志符

DFI:域特定部分<DSP>格式标识符

AAI:管理权限标识符

RDI:路由选择域标识<AS>符

SEL:网络服务接入点<NSAP>选择符

不论什么格式,都要满足一下 **3** 中规则:

1>NET 地址必须以一个单个八位组字节的域开始(**47.xxxx.xxxx...**)

2>NET 地址必须以一个单个八位组字节的域结束并设置成 **0x00(...xxx.00)**

3>在 **cisco** 上.NET 地址的系统 **ID** 必须是 **6** 个八位组字节.

如果 **systemID** 不同,根据区域 **ID** 来定位,然后根据 **SYSID** 来定位;
若将某个路由器制定为 **L1router**,离其最近的 **L2router** 将向其宣告一条默认路由,用于引导其去往外部网络。

IS-IS 的功能结构:

ISIS 的网络类型

1>广播型子网

2>点到点

ISIS 邻居路由器和邻接关系

ISIS 通过交换 **hello PDU** 来发现和形成邻居关系, **hellotime 10S**

但是 **ISIS** 的邻居之间的 **hellotime** 可以不同

ISIS 形成邻居关系的几个规则:

a>两台 **L1-only** 路由器只有在它们的 **AID** 匹配才能邻接

b>两台 **L2-only** 路由器即使他们的 **AID** 不同也可以形成邻居

c>一台 **L1-only** 和一台 **L1/L2** 路由器只有在他们的 **AID** 匹配时才能形成一个 **L1** 邻接。

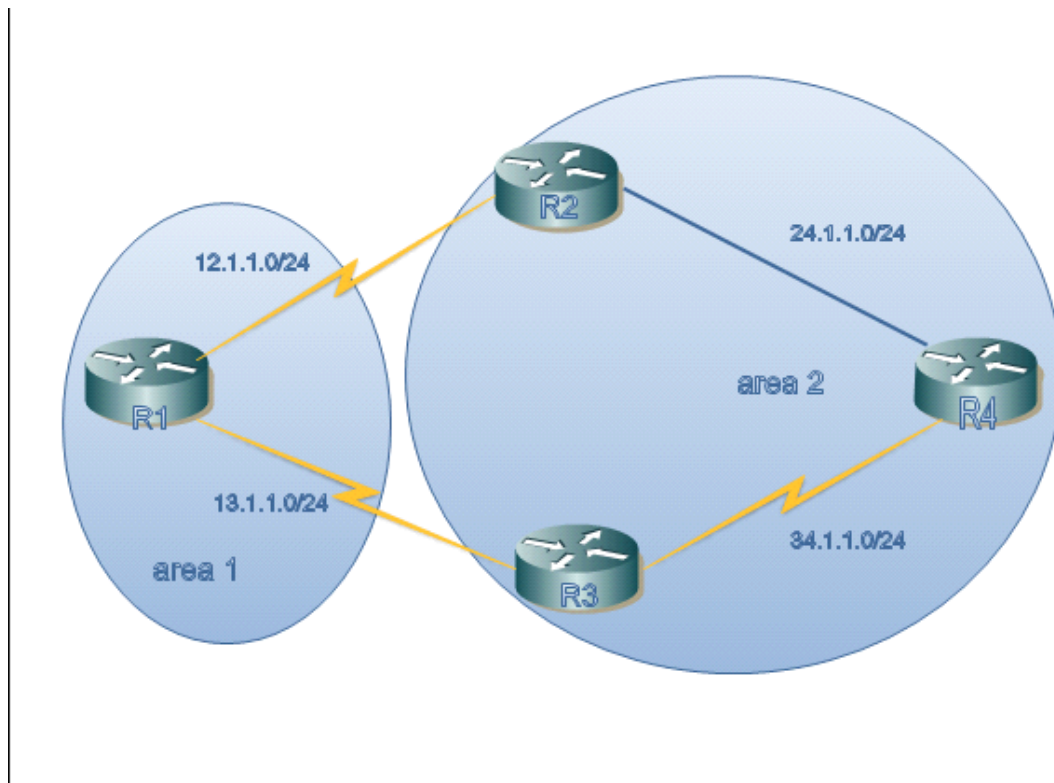
d>一台 **L2-only** 和一台 **L1/2** 路由器即使 **AID** 不同也能形成一个 **L2**

邻接。

e>若两台 L1/2 的 AID 匹配,可以同时形成 L1 和 L2 的邻接;若 AID 不匹配则只能形成 L2 邻接

实验配置:

拓扑:



基本配置:

router isis

net 49.0001.1111.1111.1111.00 //定义实体标识

interface Serial1/0

ip address 12.1.1.1 255.255.255.0

ip router isis //将接口加入 ISIS

Gateway of last resort is not set

```
34.0.0.0/24 is subnetted, 1 subnets
i L2    34.1.1.0 [115/20] via 13.1.1.3, Serial1/1
2.0.0.0/24 is subnetted, 1 subnets
i L2    2.2.2.0 [115/20] via 12.1.1.2, Serial1/0
3.0.0.0/24 is subnetted, 1 subnets
i L2    3.3.3.0 [115/20] via 13.1.1.3, Serial1/1
4.0.0.0/24 is subnetted, 1 subnets
i L2    4.4.4.0 [115/30] via 13.1.1.3, Serial1/1
        [115/30] via 12.1.1.2, Serial1/0
24.0.0.0/24 is subnetted, 1 subnets
i L2    24.1.1.0 [115/20] via 12.1.1.2, Serial1/0
12.0.0.0/24 is subnetted, 1 subnets
```

ISIS 的路由表.因为这些网段都是从区域 2 学习来的,所有被标识为 **L2** 的路由.

ISIS 的 **DR/BDR** 选举

```
*May  8 15:54:32.215: ISIS-Adj: L2 adj count 1
*May  8 15:54:32.215: ISIS-Adj: L2 adjacency state goes to Up
R2#
*May    8  15:54:32.215:  ISIS-Adj:  Run level-2 DR election for
FastEthernet2/0
*May    8  15:54:32.219:  ISIS-Adj:  New level-2 DR 4444.4444.4444 on
FastEthernet2/0
*May    8  15:54:32.443: ISIS-Adj: Sending L1 LAN IIH on FastEthernet2/0,
length 1497
*May    8  15:54:32.455: ISIS-Adj: Sending L2 LAN IIH on FastEthernet2/0,
length 1497
```

可以看出 **ISIS** 的 **DR** 的选举和 **ospf** 的区别.**isis** 的 **DR** 的选举是依靠优先级和 **MAC** 地址来决定的.选举顺序和 **OSPF** 类似.但是如果 **isis** 的优先级设置成为 **0** 的话.只是最低优先级而非 **ospf** 那样不参与选举

R2:address is ca02.0fdc.0038

R4:address is ca04.05f4.0038

ISIS 的认证:

key chain cisco

key 1

key-string cisco //定义 key

```
interface Serial1/0
ip address 12.1.1.1 255.255.255.0
ip router isis
serial restart-delay 0
isis authentication mode md5
isis authentication key-chain cisco  调用认证
```

因为 **ISIS** 使用比较少.本人水平也有限.只能写到这么多.望海涵,一般的基本内容基本已经提到,如果需要深入了解.看看 **TCP/IP** 卷一.

第五部分 路由重分发和策略路由

路由重分发:

此项技术是用来解决一个路由协议向另一种路由协议发布路由条目的.可以和分发列别结合选择需要发布的条目.这个技术还是比较简单.需要注意以下几点:

A>向 **EIGRP/RIIP** 重分发路由必须指定 **metric**,默认 **eigrp/rip** 的重分发 **metric** 为无穷大,向 **eigrp** 重分发路由时的度量需要注意的一点是延迟的单位为 **10** 毫秒.

B>在链路状态路由协议中使用分发列表需要注意,方向只能为 **in** 方向.

C>向 **ospf/isis** 路由协议重分发时.须指定条目类型,**ospf** 为 **E2/E1 /N2/N1**;isis 为 **L2/L1**

这两者的区别在于 **2** 在进程中传播 **metric** 不变,**1** 为变.

D>路由环路的避免,如果有多个分发点,则需要考虑路由环路,回馈路由等一个配置事例:

分发列表

```
access-list 1 permit 192.168.1.0 0.0.0.255 //定义感兴趣前缀
```

```
router ospf 1
```

```
    distribute-list 1 in //调用
```

重分发:

```
router eigrp 100
```

```
redistribute ospf 1 metric 10000 100 255 1 1500 //将 ospf 重分发进 eigrp
```

```
network 24.1.1.0 0.0.0.255
```

```
no auto-summary
```

```
router ospf 1
```

```
redistribute eigrp 100 metric-type 1 subnet //将 eigrp 重分发进 ospf 并制定类型为 1
```

第六部分 **BGP** 边界网关协议

BGP 作为一种 **AS** 间的路由协议，也是现在作为唯一一种 **AS** 间的路由协议，以其强大的属性，策略路由等特性，一直让人难以掌握。

BGP 在现实中除了 **ISP** 级别的骨干网络使用比较多外。其他的一些企业内部一般都不部署 **BGP**。

AS 的定义：在同一个技术掌控之下的一组路由器。

BGP 的路径选择：**AS** 号做决策类似与跳数，一个 **AS** 就是一条。

依赖于属性来完成决策

基于策略路由

BGP 的选路基本都是人为可以控制的，也证实因为灵活所以难度较大。

BGP 一般用在多个 **ISP** 出口，大型 **internet** 网络。

BGP 的更新：可靠更新，依赖于底层的 **TCP** 连接，**TCP 179** 端口，同样是因为 **TCP**。建立 **BGP** 并不需要本地直连。只要双方路由器能建立起 **TCP** 会话那就可以建立起 **BGP** 连接。这点和 **IGP** 协议很大的区别。依靠 **TCP** 的底层传输技术，可以保证大量数据的传输。

BGP 的消息：

Open 消息：在 **TCP** 建立完成之后，**BGP** 路由器会发送 **open** 消息来

协商建立 **BGP** 会话。

Keepalive：当 **BGP** 建立起 **PEER** 后，依靠 **keepalive** 消息来保持

TCP 会话的正常。时间为 **holdtime** 的 **1/3**，默认 **60S**，

如果 **holdtime** 设置为 **0**，则不发送 **keepalive**。

Update: 更新报文，链路有变化时才发送，触发更新。

Notification: 当 **BGP** 检测到错误时才发送。

BGP 的建立过程

建立 **TCP>open** 消息 **>keepalive 60S/ 次 >update>** 若有错误

notification 消息

若 **BGP** 发送 **open** 消息的 **source** 地址和远端指定的 **IP** 地址

(neighbor 命令)不一致则不能建立 **BGP** 会话!!!!!!

BGP 的防环机制:

BGP 是一种天生无环路的路由协议,原理是这样的:

如果一个 **BGP** 路由器从邻居那收到一个更新,发现更新包中的 **AS** 号

包含自己的 **AS** 号则认为有环路,而丢弃此更新包!

BGP 的水平分割:一个 **IBGP** 路由器从邻居那收到的更新包不会转发给自己的 **IBGP** 邻居.

BGP 的同步:

使用同步可以保证学习到的路由的准确性,机制是如果学习到一条路由,则此路由必须存在与自己的 **IGP** 表项中.

建立 **BGP** 时候的几种状态:

1> IDLE:查找本地路由表寻找到达邻居的可用路径,若不可达则停留在 **IDLE** 状态.

2>Connect:**BGP** 进程等待 **TCP** 连接完成,若 **TCP** 建立成功,则向邻居发送 **open** 消息并进入 **opensent** 状态.若 **TCP** 建立失败,则继续侦

听邻居初始化连接,重置连接定时器,并转到 **ACTIVE** 状态.

3>Active:BGP 进程尝试与其邻居初始化 **TCP** 连接.停留在 **active** 状态有以下几种原因:

A>邻居无到达源的路径

B>使用错误地址建立邻居

C>指向错误的 **AS** 号

4>opensent:已发送了 **open** 消息,**BGP** 一直等待知道收到邻居的 **open** 消息,会检查 **open** 消息的每个字段,如果发现差错,则发送 **notification** 消息

5>openconfirm 状态:**BGP** 将等待 **keepalive** 或 **notification** 消息,若收到 **keepalive** 消息则转到 **established** 状态,若收到 **notification** 消息,则断开 **TCP** 连接,转到 **idle** 状态.

6>established:**BGP peer** 已经建立完成,**peer** 间可相互交换 **update\keepalive** 和 **notification** 消息.若收到 **update/keepalive** 则重新启动 **holdtime** 定时器,若收到 **notification** 则转到 **idle**.

BGP 的属性:

BGP 属性分为 4 类

A> 众所周知强制属性

B>众所周知自选属性

C>可选传递属性

D>可选非传递属性

常用属性:

属性	类别	属性	类别
Origin/起源	周知强制	Aggregator/聚和	可选传递
AS_path	周知强制	Community/团体	自选传递
Next-hop	周知强制	MED	可选非传递
Local-pref	周知自选	Originator-id	可选非传递
Atomic-aggregate	周知自选	Cluster-list	可选非传递

1>origin:

IGP>EGP>Imcomplete

IGP:从源 **AS** 的协议中学到的

EGP: **EGP** 协议学到的

Imcomplete: 通过重分发学习到的

2>AS-path:

利用一串 **AS** 号来描述去往目的的 **AS** 见路径或者路由,有点类似与 **IGP** 中的条数,一个 **AS** 就是一跳的意思.可以这么理解.最小的 **AS-PATH**.

3>next-hop:

Next-hop 的规则:

如果宣告路由器与接收路由器不在一个 **AS**,则 **next-hop** 是宣告路由器的接口地址.

若宣告路由器与接收路由器在同一个 **AS** 且 **update** 的目的地址是同一 **AS** 内的地址,则 **next-hop** 是宣告该路由的邻居的 **IP** 地址.

若宣告和接受是 **ibgp**,且 **update** 的目的地址是不同 **AS** 内的地址,则 **next-hop** 是 **EBGP** 的 **IP** 地址.

4>Local-pref:

本地优先级仅用于 **IBGP**,不会传给 **EBGP** 邻居,最高优先.影响自己的流量出去.

5>MED(metric):

类似于 **IGP** 中的 **metric**,最小最优先,影响进入 **AS** 的流量.

6>atomic-aggregate&aggregate:

任何接收到带有 **atomic-aggregate** 属性的路由的下游 **BGP** 发言者都无法获得该路由更精确的信息,且将该条目宣告给其他 **Peer** 时必须加上 **atomic-aggregate** 属性 .

7>community:

主要用于简化策略的执行,将一组路由条目编排成同一个类型的团体.其类型包括:

a>internet 此团体无任何值,默认时都属于此团体.若属于此团体,则可以自由宣告该路由.

B>no-export(4294967041 或 0xfffff01)若收到该数值的路由.则不能将其传递给 **EBGP**,若配了 **bgp** 联盟,则不能将其传递到联盟之外.

C>no-advertise(4294967042 或 0xfffff02)若收到该数值的路由,则根本就不能宣告,包括 **EBGP** 和 **IBGP**.

D>Local-as(4294967042 或 0xffffffff03)若收到该数值的路由,讲不能宣告给 **EBGP peer** 包括联盟内其他自治系统中的 **peer**.

8>weight:

权重是 **cisco** 私有的 **BGP** 参数,仅用与单个路由器内的路由,该数值不与其他路由器交换.越大越优先.

BGP 选路规则:

1>权重最大最优先

2>local-pref 最大最优先

3>本地始发的优先

4>AS-path 最短最优先

5>优选来源编码最低的<编码 IGP<EGP<Incomplete>

6>MED 最小最优先

7>ebgp>联盟 EBGP>ibgp

8>优选 next-hop 最近的

9>如果设置了 maximum-paths 也就是负载均衡,则在 **loc-rib** 中安装所有等价路由

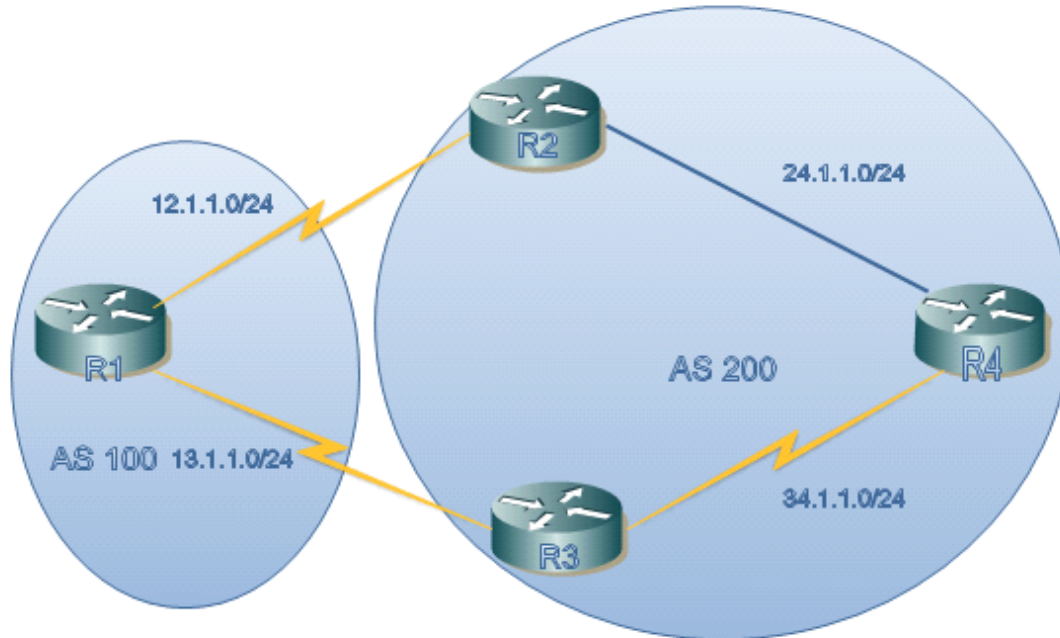
10>BGP router-id 最小最优先

往下一次比较,直到比出最优的



BGP 的配置:

拓扑:



基本配置

R1:

```
router bgp 100                                //启动 BGP 进程
no synchronization                          //关闭同步
bgp log-neighbor-changes
network 1.1.1.0 mask 255.255.255.0
network 12.1.1.0 mask 255.255.255.0
network 13.1.1.0 mask 255.255.255.0 //宣告网络,
neighbor 12.1.1.2 remote-as 200           //设置 EBGP 邻居
neighbor 12.1.1.2 ebgp-multihop 255       //设置 EBGP 多条
neighbor 13.1.1.3 remote-as 200
neighbor 13.1.1.3 ebgp-multihop 255
no auto-summary
```

R2:

```
router bgp 200
no synchronization
bgp log-neighbor-changes
network 2.2.2.0 mask 255.255.255.0
network 24.1.1.0 mask 255.255.255.0
```

```
neighbor 3.3.3.3 remote-as 200           //设置 IBGP 邻居
neighbor 3.3.3.3 update-source Loopback0 //设置更新接口为环回 0
neighbor 3.3.3.3 next-hop-self           //设置下一条为自己
neighbor 4.4.4.4 remote-as 200
neighbor 4.4.4.4 update-source Loopback0
neighbor 4.4.4.4 next-hop-self
neighbor 12.1.1.1 remote-as 100
neighbor 12.1.1.1 ebgp-multihop 255
no auto-summary
```

需要注意的是:

1>BGP 在建立邻居的时候可以使用环回口来指定邻居,这样可以避免链路故障造成不必要的路由翻滚.如果 **EBGP** 用 **lookback** 口来建立 **BGP** 会话,则需要指定静态路由,确保对方的环回口能够建立 **TCP** 会话,并且需要制定 **EBGP-MULTIHOP**,来修改 **EBGP** 包的 **TTL=1** 的限制。

2>建立 IBGP 会话时,AS 内的 IBGP 邻居必须全互联,因为 IBGP 路由器收到了路由不会转发给自己的 IBGP 邻居.并且要设置 next-hop-self 确保收到来自非邻居始发的路由能够最优.

```

Network      Next Hop      Metric LocPrf Weight Path
* > 1.1.1.0/24 0.0.0.0          0         32768 i
* > 2.2.2.0/24 12.1.1.2         0         0 200 i
* > 3.3.3.0/24 13.1.1.3         0         0 200 i
* > 4.4.4.0/24 12.1.1.2         0         0 200 i
* > 12.1.1.0/24 0.0.0.0          0         32768 i
* > 13.1.1.0/24 0.0.0.0          0         32768 i
* > 24.1.1.0/24 12.1.1.2         0         0 200 i
* > 34.1.1.0/24 12.1.1.2         0         0 200 i
R1#sh ip bap
```

1>local-pref

```
access-list 1 permit 3.3.3.0 0.0.0.255 //定义感兴趣流量
route-map local permit 10 //定义 route-map
  match ip address 1
  set local-preference 200 //设置策略
route-map local permit 20
  neighbor 12.1.1.2 route-map local in //调用 route-map
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 1.1.1.0/24	0.0.0.0	0		32768	i
*> 2.2.2.0/24	12.1.1.2	0		0	200 i
*> 3.3.3.0/24	13.1.1.3			0	200 i
*> 4.4.4.0/24	12.1.1.2		200	0	200 i
*> 12.1.1.0/24	0.0.0.0	0		32768	i
*> 13.1.1.0/24	0.0.0.0	0		32768	i
*> 24.1.1.0/24	12.1.1.2	0		0	200 i
*> 34.1.1.0/24	13.1.1.3			0	200 i

通过我们的设置，去往 **3.3.3.0** 网段选择了 **local-pref** 为 **200** 的那条路径，**local-pref** 是控制自己的流量出去的。

2> MED

原来的 **BGP** 表项

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 1.1.1.0/24	0.0.0.0	0		32768	i
*> 2.2.2.0/24	13.1.1.3			0	200 i
*> 3.3.3.0/24	12.1.1.2	0		0	200 i
*> 4.4.4.0/24	13.1.1.3	0		0	200 i
*> 12.1.1.0/24	0.0.0.0	0		32768	i
*> 13.1.1.0/24	0.0.0.0	0		32768	i
*> 24.1.1.0/24	13.1.1.3			0	200 i
*> 34.1.1.0/24	12.1.1.2	0		0	200 i

修改后的表项

```

Network      Next Hop      Metric LocPrf Weight Path
*> 1.1.1.0/24  0.0.0.0        0       32768 i
*> 2.2.2.0/24  13.1.1.3       0         0 200 i
*> 3.3.3.0/24  12.1.1.2       0         0 200 i
*> 3.3.3.0/24  13.1.1.3      40         0 200 i
*> 4.4.4.0/24  12.1.1.2       0         0 200 i
*> 4.4.4.0/24  13.1.1.3      40         0 200 i
*> 12.1.1.0/24 0.0.0.0        0       32768 i
*> 13.1.1.0/24 0.0.0.0        0       32768 i
*> 24.1.1.0/24 13.1.1.3       0         0 200 i
*> 34.1.1.0/24 12.1.1.2       0         0 200 i
*> 34.1.1.0/24 13.1.1.3       0         0 200 i
R1#
```

配置:

```

no synchronization
bgp log-neighbor-changes
network 1.1.1.0 mask 255.255.255.0
network 12.1.1.0 mask 255.255.255.0
network 13.1.1.0 mask 255.255.255.0
neighbor 12.1.1.2 remote-as 200
neighbor 12.1.1.2 ebgp-multihop 255
neighbor 13.1.1.3 remote-as 200
neighbor 13.1.1.3 ebgp-multihop 255
neighbor 13.1.1.3 route-map med in
no auto-summary
```

```

access-list 1 permit 3.3.3.0 0.0.0.255
access-list 1 permit 4.4.4.0 0.0.0.255
```

```

route-map med permit 10
 match ip address 1
 set metric 40
!
route-map med permit 20
!
```

其他的配置基本都和这个类似.都是这个模式.我就不再做例子了

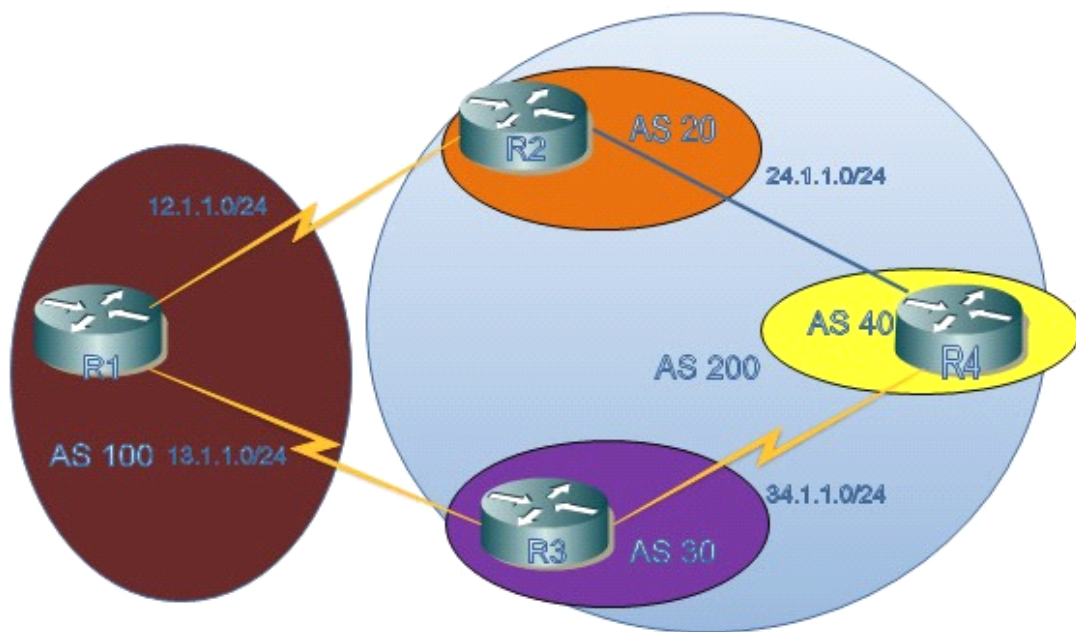
下面来看看两个比较重要的技术:

大家知道 **IBGP** 路由器收到更新后默认是不向自己的 **IBGP** 邻居发送的,这也就意味着 **IBGP peer** 之间必须全互联,现在我这里做实验是 **3** 台路由器全互联问题不大,如果是 **100** 台,**1000** 台,**10000** 台呢?那这种全互联带来的管理开销可想而知,下面的这两中技术都是解决这个问题的.

1>BGP 联盟:

BGP 联盟是将一个大的 **AS** 中间分割出若干过小的 **AS**,在外部看来依然是一个大的 **AS**,而内部确实各个小的 **AS** 间的 **EBGP** 会话,这就减少了 **IBGP** 全互联的约束.

实验拓扑



基本配置:

```
router bgp 40
no synchronization
bgp log-neighbor-changes
bgp confederation identifier 200
bgp confederation peers 20 30
network 4.4.4.0 mask 255.255.255.0
network 24.1.1.0 mask 255.255.255.0
network 34.1.1.0 mask 255.255.255.0
neighbor 2.2.2.2 remote-as 20
neighbor 2.2.2.2 ebgp-multihop 255
multihop 因为在内部会认为是一个 EBGP 连接
neighbor 2.2.2.2 update-source Loopback0
neighbor 2.2.2.2 next-hop-self
neighbor 3.3.3.3 remote-as 30
```

//启动 **BGP** 进程

//定义大 **AS**

//定义联盟内的 **AS** 邻居号

//这里一定要指定 **ebgp-**

```
neighbor 3.3.3.3 ebgp-multihop 255
neighbor 3.3.3.3 update-source Loopback0
neighbor 3.3.3.3 next-hop-self
no auto-summary
```

R4 的 BGP 表

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

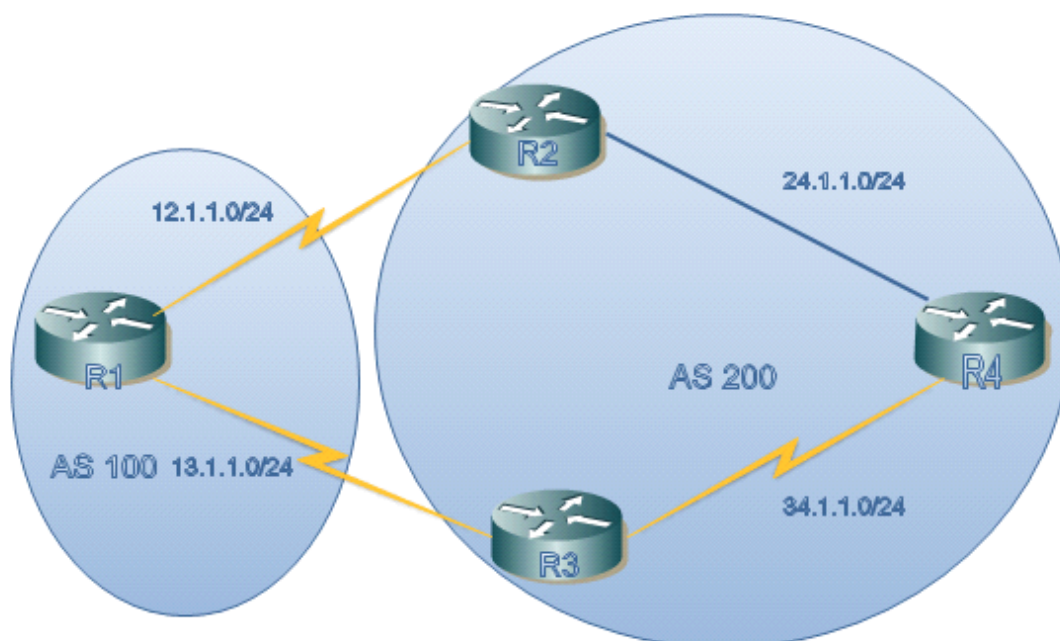
Network	Next Hop	Metric	LocPrf	Weight	Path
* 1.1.1.0/24	3.3.3.3	0	100	0	(30) 100 i
*> 2.2.2.0/24	2.2.2.2	0	100	0	(20) 100 i
*> 2.2.2.0/24	2.2.2.2	0	100	0	(20) i
*> 3.3.3.0/24	3.3.3.3	0	100	0	(30) i
*> 4.4.4.0/24	0.0.0.0	0		32768	i
r 12.1.1.0/24	3.3.3.3	0	100	0	(30) 100 i
r> 12.1.1.0/24	2.2.2.2	0	100	0	(20) 100 i
r 13.1.1.0/24	3.3.3.3	0	100	0	(30) 100 i
r> 13.1.1.0/24	2.2.2.2	0	100	0	(20) 100 i
*> 24.1.1.0/24	0.0.0.0	0		32768	i
*> 34.1.1.0/24	0.0.0.0	0		32768	i

没问题的.**BGP** 表项全部学到了

2>BGP 路由反射器

BGP 的路由反射器,也是减少 **IBGP** 会话连接的,配置和管理比 **BGP** 联盟相对简单一点.可以这么理解.定义为反射器的那台路由器就象一个镜子,将学习到的 **BGP** 条目反射给内部的客户端.

实验拓扑:



我们将 **R4** 设置成为反射器.把 **R2** 的条目反射给 **R3**

未设置反射器的 **BGP** 表

```
Origin codes: i - IGP, e - EGP, ? - Incomplete
  Network          Next Hop          Metric LocPrf Weight Path
*> 1.1.1.0/24      13.1.1.1              0         0 100 i
*> 3.3.3.0/24      0.0.0.0              0        32768 i
*>i4.4.4.0/24      4.4.4.4              0        100      0 i
*> 12.1.1.0/24     13.1.1.1              0         0 100 i
*> 13.1.1.0/24     0.0.0.0              0        32768 i
*                  13.1.1.1              0         0 100 i
*> 34.1.1.0/24     0.0.0.0              0        32768 i
R3#
```

设置反射器后的 **BGP** 表

```
Origin codes: i - IGP, e - EGP, ? - incomplete
  Network          Next Hop          Metric LocPrf Weight Path
* i1.1.1.0/24      2.2.2.2              0        100      0 100 i
*>                13.1.1.1              0         0 100 i
*>i2.2.2.0/24      2.2.2.2              0        100      0 i
*> 3.3.3.0/24      0.0.0.0              0        32768 i
*>i4.4.4.0/24      4.4.4.4              0        100      0 i
r>i12.1.1.0/24     2.2.2.2              0        100      0 i
r                  13.1.1.1              0         0 100 i
*> 13.1.1.0/24     0.0.0.0              0        32768 i
*                  13.1.1.1              0         0 100 i
r>i24.1.1.0/24     2.2.2.2              0        100      0 i
*> 34.1.1.0/24     0.0.0.0              0        32768 i
R3#
```

配置:

router bgp 200

no synchronization

bgp log-neighbor-changes

network 4.4.4.0 mask 255.255.255.0

neighbor 2.2.2.2 remote-as 200

neighbor 2.2.2.2 update-source Loopback0

neighbor 2.2.2.2 next-hop-self

neighbor 3.3.3.3 remote-as 200

neighbor 3.3.3.3 update-source Loopback0

neighbor 3.3.3.3 route-reflector-client //设置要反射给的邻居

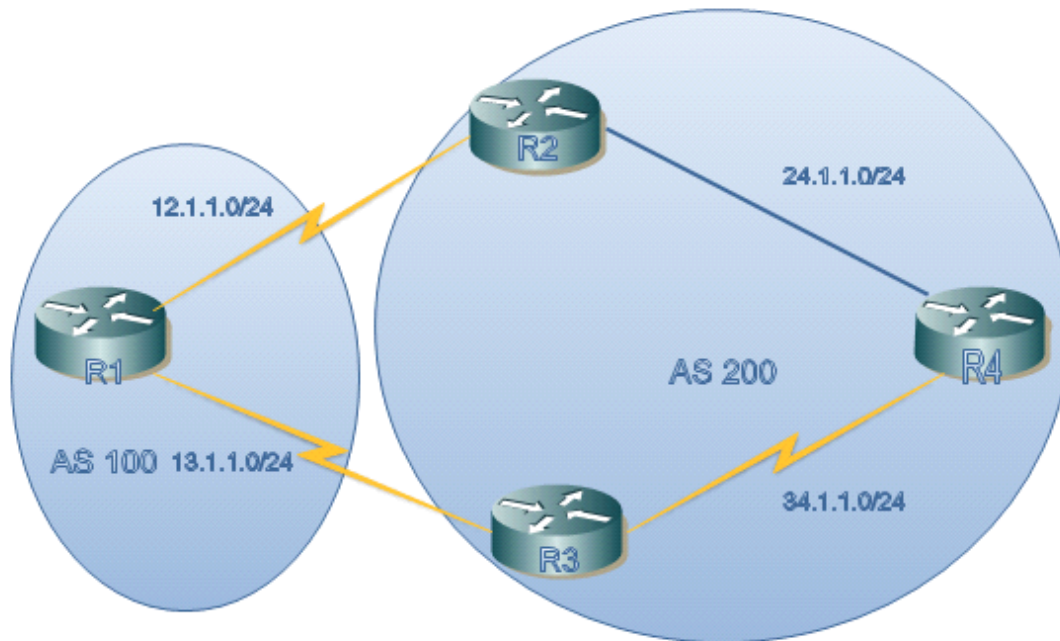
neighbor 3.3.3.3 next-hop-self

no auto-summary

反射器的配置还是比较简单的.一条命令就搞定了!!

最后讲讲 **BGP** 的聚合:

拓扑:



在 **R4** 上模拟几个环回口.并宣告进 **BGP** 进程.

未使用聚合的 **BGP** 表

```
Network      Next Hop      Metric LocPrf Weight Path
*>i2.2.2.0/24 2.2.2.2       0      100      0 i
*> 3.3.3.0/24 0.0.0.0       0      32768 i
*>i4.4.4.0/24 4.4.4.4       0      100      0 i
r>i12.1.1.0/24 2.2.2.2       0      100      0 i
*> 13.1.1.0/24 0.0.0.0       0      32768 i
r>i24.1.1.0/24 2.2.2.2       0      100      0 i
*> 34.1.1.0/24 0.0.0.0       0      32768 i
*>i172.16.1.0/24 4.4.4.4       0      100      0 i
*>i172.16.2.0/24 4.4.4.4       0      100      0 i
*>i172.16.3.0/24 4.4.4.4       0      100      0 i
R3#
```

聚合之后的 **BGP** 表

```
Network      Next Hop      Metric LocPrf Weight Path
* i1.1.1.0/24 2.2.2.2       0      100      0 100 i
*> 13.1.1.1 13.1.1.1      0      0 100 i
*>i2.2.2.0/24 2.2.2.2       0      100      0 i
*> 3.3.3.0/24 0.0.0.0       0      32768 i
*>i4.4.4.0/24 4.4.4.4       0      100      0 i
r 12.1.1.0/24 13.1.1.1      0      0 100 i
r>i 2.2.2.2 2.2.2.2       0      100      0 i
* 13.1.1.0/24 13.1.1.1      0      0 100 i
*> 0.0.0.0 0.0.0.0       0      32768 i
r>i24.1.1.0/24 2.2.2.2       0      100      0 i
*> 34.1.1.0/24 0.0.0.0       0      32768 i
*>i172.16.0.0/22 4.4.4.4       0      100      0 i
```

配置也比较简单:

R4(config-router)#aggregate-address 172.16.1.0 255.255.252.0

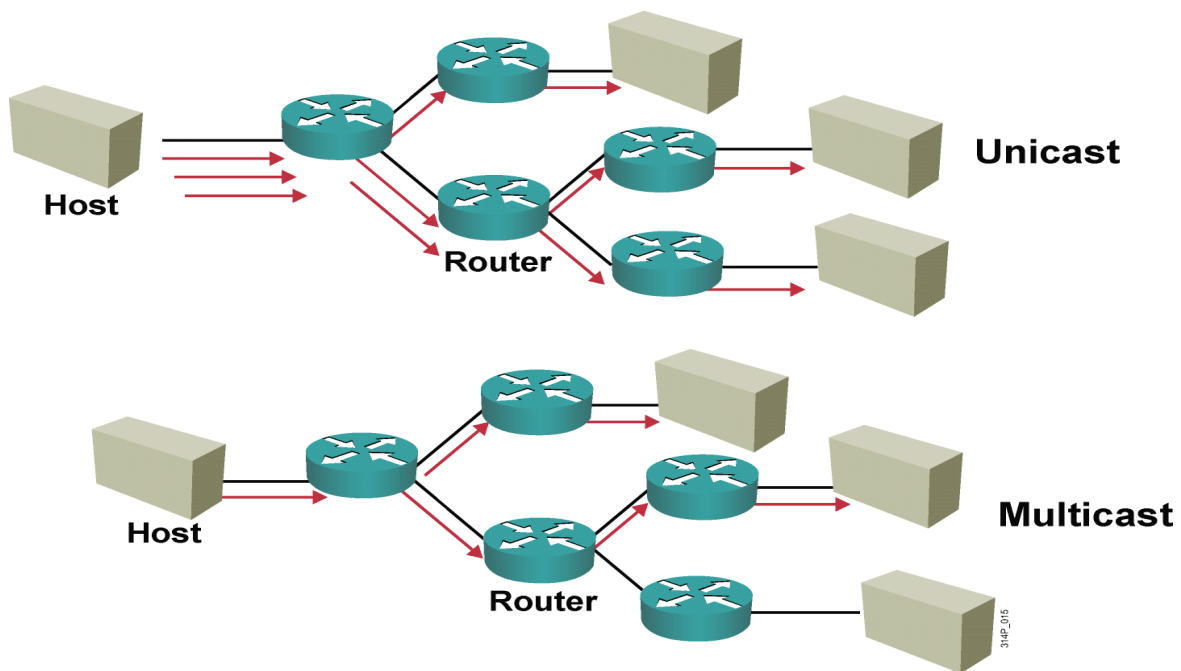
summary-only //这里值得注意的是, 如果不加 **summary-only** 的话
那么明细条目和聚合条目将同时宣告给 **bgp peer**。

当然 **BGP** 还有很多内容这里没有提到, 有兴趣的可以参考 **TCP/IP**
路由技术卷二

Ip multicast 多播

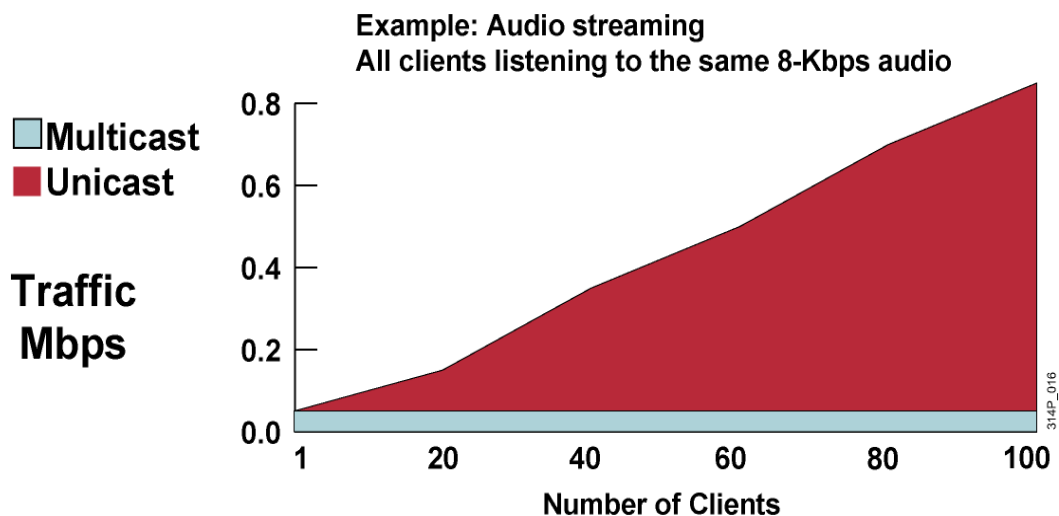
多播是一个相对于单播和广播的概念，使用多播可以提高网络效率，降低网络资源。先来比较三者的区别。

Multicast VS Unicast



从上图不难发现，如果右边三台主机想观看 **Host** 提供的视频服务，如果使用单播，那么 **host** 将需要发送三份拷贝出来，相对于下图使用组播，**host** 只需要发送一份拷贝出来，这就起到了节省网络资源，提高网络效率的特点。

如果使用广播的话。那么将产生一个问题，也就是如果右边只有 **2** 台主机需要这份数据的话。那么不管另外一台需要与否。都将发送一份拷贝过去。而路由器默认是隔离广播的。虽然可以采取定向广播的技术，但是这种带来的开销显然是不能够承受的。



从上图我们不难看出 **multicast** 和 **unicast** 的特性刚好相反，用户越少 **unicast** 的优点越明显，而用户越多 **multicast** 的优点越明显！单播关心流量往哪走，组播关心流量从哪来!!!!

既然 **multicast** 有这么多优点。那么它有没有缺点呢？肯定有的它的缺点就是 **UDP** 的缺点。因为 **multicast** 是基于 **UDP** 传输的。

1>与单播协议相比没有纠错机制，发生丢包错包后难以弥补，但可以通过一定的容错机制和 **QOS** 加以弥补。

2>现行网络虽然都支持组播的传输，但在客户认证、**QOS** 等方面还需要完善，这些缺点在理论上都有成熟的解决方案，只是需要逐步推广应用到现存网络当中。

组播的编址：

组播使用 **D** 类地址**224.0.0.0-239.255.255.255**

224.0.0.0~224.0.0.255为预留的组播地址（永久组地址），地址**224.0.0.0**保留不做分配，其它地址供路由协议使用；

224.0.1.0~238.255.255.255为用户可用的组播地址（临时组地址），全网范围内有效；**239.0.0.0~239.255.255.255**为本地管理组播地址，仅在特定的本地范围内有效。常用的预留组播地址列表如下：

224.0.0.0 基准地址（保留）

224.0.0.1 所有主机的地址

224.0.0.2 所有组播路由器的地址

224.0.0.3 不分配

224.0.0.4 dvmrp 路由器

224.0.0.5 ospf 路由器

224.0.0.6 ospf dr

224.0.0.7 st 路由器 **224.0.0.8 st** 主机

224.0.0.9 rip-2 路由器

224.0.0.10 Eigrp 路由器

224.0.0.11 活动代理

224.0.0.12 dhcp 服务器/中继代理

224.0.0.13 所有 **pim** 路由器

224.0.0.14 **rsvp** 封装

224.0.0.15 所有 **cbt** 路由器

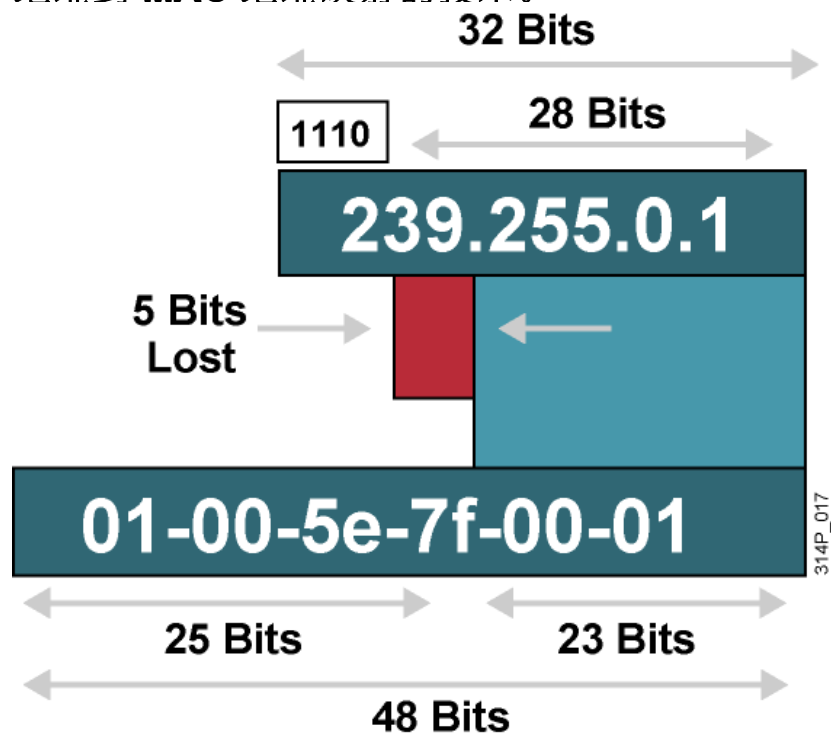
224.0.0.16 指定 **sbm**

224.0.0.17 所有 **sbms**

224.0.0.18 **vrrp**

Ethernet 上的多播：

以太网使用 **MAC** 地址来寻址，所以 **multicast** 如果需要在以太网上使用的话，就必须能够有对应的 **MAC** 地址。**IETF** 定义了一种 **IP** 地址到 **MAC** 地址映射的技术。



IETF 规定组播 **MAC** 地址的前 **25bit** 为 **01-00-5e** 而 **D** 类组播地址是以 **1110** 开头的, 所以这三位不需要做映射, **48** 位的 **MAC** 地址去掉前面规定的 **25** 位 **MAC** 地址, 还剩下 **23** 位。**IETF** 规定将 **IP** 地址的后 **23** 位一一映射到 **MAC** 地址的后 **23** 位, 这样就产生了一个组播 **MAC** 地址, 但是因为有 **5bit** 未能够映射下来, 所以就产生了一个问题, 一个组播 **MAC** 地址对应 **2** 的 **5** 次方个 **IP** 地址也就是说一个组播 **MAC** 地址对应 **32** 个组播 **IP** 地址。

这种 **MAC** 地址方案有两个好处:

1>本网络的组播源或者路由器只需向多播 **MAC** 地址发送一份数据帧,即可保证 **LAN** 中的组成员都能接收到。

2>由于知道了 **multicast** 地址之后,即可以知道 **MAC** 地址,所以不需要使用 **ARP**。

组成员关系协议 (**IGMP**)

IGMP 协议运行于主机和与主机直接相连的组播路由器之间, 主机通过此协议告诉本地路由器希望加入并接受某个特定组播组的信息, 同时路由器通过此协议周期性地查询局域网内某个已知组的成员是否处于活动状态 (即该网段是否仍有属于某个组播组的成员), 实现所连网络组成员关系的收集与维护。

IGMP 有三个版本, **IGMPv1** 由 **RFC1112** 定义, 目前通用的是 **IGMPv2**, 由 **RFC2236** 定义。**IGMPv3** 目前仍然是一个草案。

IGMPv1中定义了基本的组成员查询和报告过程，**IGMPv2**在此基础上添加了组成员快速离开的机制，**IGMPv3**中增加的主要功能是成员可以指定接收或指定不接收某些组播源的报文。这里着重介绍 **IGMPv2**协议的功能。

IGMPv2通过查询器选举机制为所连网段选举唯一的查询器。查询器周期性的发送普遍组查询消息进行成员关系查询；主机发送报告消息来应答查询。当要加入组播组时，主机不必等待查询消息，主动发送报告消息。当要离开组播组时，主机发送离开组消息；收到离开组消息后，查询器发送特定组查询消息来确定是否所有组成员都已离开。

通过上述 **IGMP** 机制，在组播路由器里建立起一张表，其中包含路由器的各个端口以及在端口所对应的子网上都有哪些组的成员。当路由器接收到某个组 **G** 的数据报文后，只向那些有 **G** 的成员的端口上转发数据报文。至于数据报文在路由器之间如何转发则由路由协议决定，**IGMP** 协议并不负责。这里我们主要讨论 **IGMPV2**

IGMPV2主机消息类型：

1>Membership Report 成员关系报告：用于指示某主机希望加入某多播组，如果本地多播 **router** 并不知道主机所加入的多播会话，那么将向上游多播源发送一个请求消息，接受到数据后，就会向请求了组成员关系的主机所在的子网转发数据。成员关系报告中

的目的地址是组地址,**RFC2236**建立发送间隔为**10S** 以确保本地路由器能够收到此消息。

2>Leave Group 离组消息:此消息用在主机需要离开多播组时。发送给**224.0.0.2**地址,

IGMPV2路由器消息类型:

General Query 常规查询:路由器利用此消息来查询其所连接的子网,以确定是否有组成员.**default,60S** 发送一次查询消息。使用 **ip igmp query-interval** 修改 范围 **0-65535**. 发送地址 **224.0.0.1**; 主机以成员关系报告响应此消息,**default** 最大响应时间为**10S 1/10**为单位,

Group-Specific Query 特定组查询消息:如果有主机离组,那么将发送特定组查询消息,特定组查询消息中包含了组地址,且用该地址作为查询的目的地址,路由器会以**1S** 为间隔发送两条特定组查询消息,防止特定组查询消息的丢失。

IGMP Snooping 的实现机理是:交换机通过侦听主机发向路由器的 **IGMP** 成员报告消息的方式,形成组成员和交换机接口的对应关系;交换机根据该对应关系将收到组播数据包只转给具有组成员的接口。

IGMP Snooping 工作原理:

1>多播 router 周期性发送 IGMP 查询到 VLAN

2>主机发送成员关系报告响应查询

3>switch 监听每个 IGMP 组播包,并保持跟踪.

4>switch 将创建每个 VLAN 的条目,将其保存在自己的目录表中

5>如果主机要离开组,router 将查询;若 switch 收不到主机的响应,则将条目删除.

CGMP(Cisco Group Management Protocol)是 **Cisco** 基于客户机/服务器模型开发的私有协议,在 **CGMP** 的支持下,组播路由器能够根据接收到的 **IGMP** 数据包通知交换机哪些主机何时加入和脱离组播组,交换机利用由这些信息所构建的转发表来确定将组播数据包向哪些接口转发。**GMRP** 是主机到以太网交换机的标准协议,它使组播用户可以在第二层交换机上对组播成员进行注册。

CGMP 包:

虽然 **CISCO** 的路由器和交换机都能配置 **CGMP**,但是只有路由器能够产生 **CGMP** 包,交换机只是监听和读取 **CGMP** 包。

1>join 包:路由器发出,告诉交换机有成员要加入多播组

2>leave 包:路由器发出,告诉交换机有成员从多播组中删除.

CGMP 的包的目的地址始终是保留的 **MAC** 地址 **0100.0cdd.dddd**,启用了 **CGMP** 的交换机始终监听此地址.

当 **router** 启用 **CGMP** 时,发送一条 **GDA** 被设置成为**0,USA** 设置成自身 **MAC** 的 **CGMP Join** 包,告知交换机,当启用 **CGMP** 交换机收到此包后,就知道连接在收到包的端口上连接了一台多播路由器,路由器每隔**60S** 发送一次此包.

GDA(Group Destination Address)组目的地址

USA(Unicast Source Address)单播源地址

多播的加组过程:

1>主机发成员关系报告到**224.0.0.1**,在未得到答复前每**10S** 发送一次;

2>路由器收到报告,将其加入到 **IGMP** 组中,路由器每**60S** 发送一次普通查询给所有主机以确认这些主机是否存在于组中(至少有一台活动主机);

3>主机收到普通查询开启随机递减计数器,当计数器清零会发报告来相应路由器,若有一台主机响应了,其他的降不再响应.

离组过程;

1>主机发送离组消息到224.0.0.2

2>router 收到此消息,发送特定组查询消息,来检测子网中是否还有其他活动组成员,如果发送**3**次未得到响应,则认为此组中已无活动组成员,停止组流量的发送.

如若子网中有多台路由器,将进行查询者选举.最高 **IP** 地址的将成为查询者,其他路由器则监听查询者,如若**120S** 发现查询者无响应,则认为自己为查询者.

分发树:

源分发书(**S,G**)也叫最短路径树,从名字就能看出就是源和主机之间建立的分发书.路径最短.在源和接收方直接建树,占用资源较多

共享分发树(***,G**):要选 **RP**, 流量从 **RP** 向下分发, **source** 向 **RP** 请求。接收方都指向 **RP**,**RP** 负担较重.

两种树的相同点:

1>建立无环拓扑

2>动态加组或离组

3>如果子网子网一个主机向最近的 router 加组,都会向上请求

流量,如果都不请求,会剪切向上的流量.

Upstream(上行链路)&downstream(下行链路):组播一般总是沿着源到目的的下行方向传递,去往特定组的数据只能到达上行链路,并从下行链路转发出去.上行链路更靠近组播源,下行链路更靠近接收者.



反向路径转发(Rreverse Path Forwarding):组播路由协议的功能是用来确定上行接口;组播路由有些是用来确定去往源最近的路径和 **IGP** 等刚好相反,所以组播包的转发成为反向路径转发.**RPF** 要求源来目的的地址和路由表中的一致.

组播模式:

1>Dense mode 密集模式:网络中的接受者占网络中的成员数量较多.特点:

A>常用在 LAN

B>PUSH 模型,向外推流量,如果不需要则进行剪切.

C>常用(S,G)

2>Sparse mode 稀疏模式:组成员占网络中成员的小部分.其特点是:

A>常用 WAN

B>PULL 模型,路由器等待分包再创建分发书

C>常用(*,G)

PIM(Protocol Independent multicast) 协议无关组播,使用单播路由表确认数据流的源地址,不关注底层哪种协议创建的路由表.使用协议号**13** 组播地址**224.0.0.13**.

下面两种 **PIM** 的模式:

PIM-DM 特点:

1>将数量 flooding

2>如果不需要流量,则路由器发送修剪包,每个3分钟洪泛和修剪一次.使用(S,G)

PIM-SM 特点:

1>使用两种分发树

2>共享分发树 RP-接受方

3>源分发树 源-RP

流程

RP 发现 (自动 RP,静态指定)>建立共享分发树>源到 RP 注册

>switchover

上面有很多新名词.下面一个一个解释.

自动 **RP,cisco** 开发的,我们来看看它的特点

CISCO auto-rp protocol:

1>允许远程路由器动态发现 RP

2>在任意组任意时间只能有一个 RP

3>每个组播组都要映射到 RP

4>为防止单独 RP 失效,有备份 RP 叫 CRP

5>CRP 会通告自己有可能成为 RP 到224.0.0.39 每60S

一个 **CRP** 可以通告为多个组的 **CRP**; 一个路由器监听

224.0.0.39做 **mapping agents** <映射代理>选出 **RP** 以及 **RP** 和组的关系,并保存在 **Group-to-mapping** 缓存表中,每**60S** 发送一次 **auto-rp discovery** 消息.

共享分发树的构建:

1>如果主机希望接收组流量,向最近的 **router** 发送组成员报告 (**IGMP**);

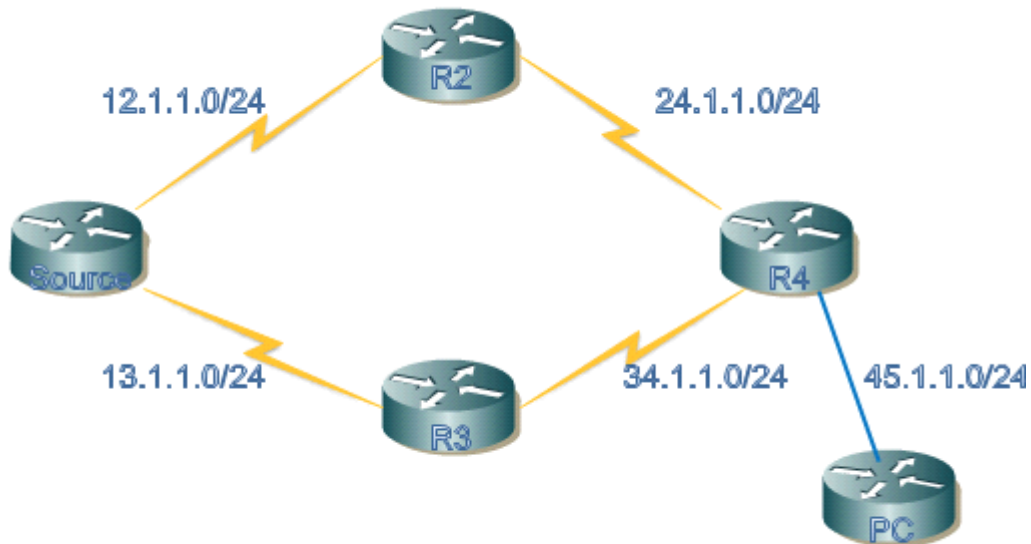
2>Router 先查看自己是否知道 **Group-to-rp** 映射,若知道则转发组播流量,若不知道,则向上发送 **Join** 消息;

3>RP 检查是否有此组的消息,若没有则创建 **(*,G)** 树;并创建 **(S,G)** 到源的树;从源获得流量并将其转发给底下所有用户.

源注册:

源路由器发送 **register** 消息给 **RP**,**RP** 发送 **JOIN** 消息给源路由器,如果 **RP** 收到组播流后,则发送 **Register Stop** 消息给源.

PIM-DM 的配置:



基本配置:

```
ip multicast-routing
```

//打开多播路由

```
interface FastEthernet1/0
```

```
ip address 45.1.1.4 255.255.255.0
```

```
ip pim dense-mode
```

//将接口运行 **pim dense-mode**

```
interface FastEthernet0/0
```

```
ip address 45.1.1.5 255.255.255.0
```

ip igmp join-group 224.1.1.1 //在 **PC** 上加如多播组.这里因为是模拟.
在实际中可能是运行某个软件什么的

R4#

***Mar 1 00:12:25.851: IGMP(0): Received v2 Report on FastEthernet1/0 from 45.1.1.5 for 224.1.1.1**

***Mar 1 00:12:25.855: IGMP(0): Received Group record for group 224.1.1.1, mode 2 from 45.1.1.5 for 0 sources**

***Mar 1 00:12:25.859: IGMP(0): WAVL Insert group: 224.1.1.1 interface: FastEthernet1/0Successful**

***Mar 1 00:12:25.863: IGMP(0): Switching to EXCLUDE mode for 224.1.1.1 on FastEthernet1/0**

***Mar 1 00:12:25.867: IGMP(0): Updating EXCLUDE group timer for 224.1.1.1**

***Mar 1 00:12:25.867: IGMP(0): MRT Add/Update FastEthernet1/0 for (*,224.1.1.1) by 0**

***Mar 1 00:13:03.767: IGMP(0): Send v2 general Query on FastEthernet1/0**

***Mar 1 00:13:08.971: IGMP(0): Received v2 Report on FastEthernet1/0 from 45.1.1.5 for 224.1.1.1**

***Mar 1 00:13:08.975: IGMP(0): Received Group record for group 224.1.1.1, mode 2 from 45.1.1.5 for 0 sources**

在这上面我们可以看到加入组的一瞬间收到了成员关系报告,并且也能看到发送常规查询

R1

***Mar 1 00:12:54.427: PIM(0): Received v2 Join/Prune on Serial0/0 from 12.1.1.2, to us**

***Mar 1 00:12:54.431: PIM(0): Prune-list: (1.1.1.1/32, 224.1.1.1)**

***Mar 1 00:12:54.435: PIM(0): Prune Serial0/0/224.1.1.1 from (1.1.1.1/32, 224.1.1.1)**

上面是我发送流量的时候,收到的 **join** 消息

R4上面的多播路由表:

(*, 224.1.1.1), 00:08:42/stopped, RP 0.0.0.0, flags: DC

Incoming interface: Null, RPF nbr 0.0.0.0

Outgoing interface list:

FastEthernet1/0, Forward/Dense, 00:08:42/00:00:00

Serial0/1, Forward/Dense, 00:08:42/00:00:00

Serial0/0, Forward/Dense, 00:08:42/00:00:00

(1.1.1.1, 224.1.1.1), 00:01:27/00:01:36, flags: T

Incoming interface: Serial0/1, RPF nbr 34.1.1.3

Outgoing interface list:

Serial0/0, Prune/Dense, 00:01:26/00:01:33

该接口已经被修剪掉了

FastEthernet1/0, Forward/Dense, 00:01:27/00:00:00

(12.1.1.1, 224.1.1.1), 00:01:27/00:01:35, flags: T

Incoming interface: Serial0/0, RPF nbr 24.1.1.2

Outgoing interface list:

Serial0/1, Forward/Dense, 00:01:27/00:00:00

FastEthernet1/0, Forward/Dense, 00:01:27/00:00:00

(13.1.1.1, 224.1.1.1), 00:01:28/00:01:35, flags: T

Incoming interface: Serial0/1, RPF nbr 34.1.1.3

Outgoing interface list:

Serial0/0, Forward/Dense, 00:01:28/00:00:00

FastEthernet1/0, Forward/Dense, 00:01:28/00:00:00

(*, 224.0.1.40), 00:09:36/00:01:56, RP 0.0.0.0, flags: DCL

Incoming interface: Null, RPF nbr 0.0.0.0

Outgoing interface list:

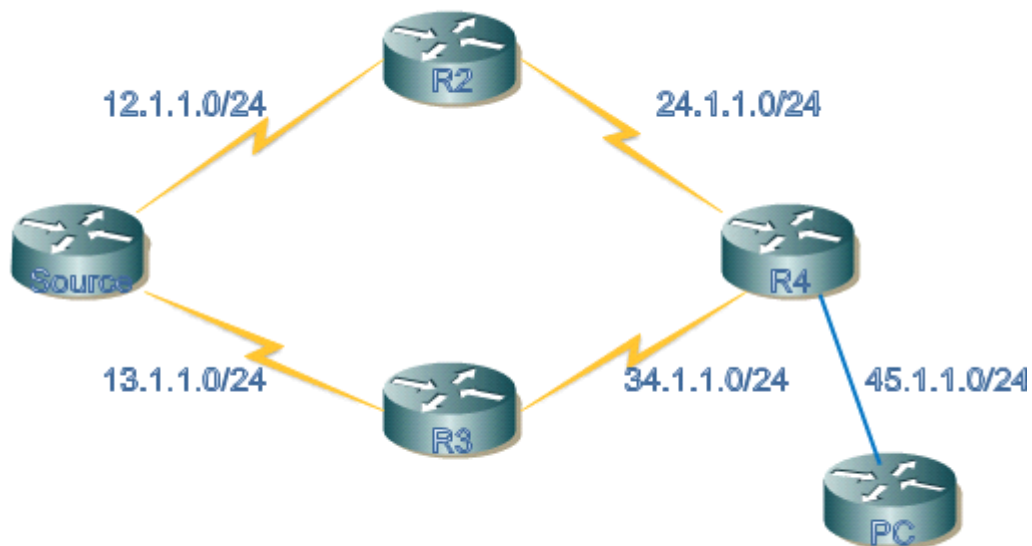
Serial0/1, Forward/Dense, 00:09:25/00:00:00

Serial0/0, Forward/Dense, 00:09:28/00:00:00

Loopback0, Forward/Dense, 00:09:36/00:00:00

不难看出**(S,G** 的表象)第一个和最后一个**(*,G)**是 **CISCO** 创建的,不起转发作用.

PIM-SM:



静态制定 **RP** 的我这里就不配了.只需要使用 **ip pim rp-address x.x.x.x** 来手动制定 **RP** 就行了再每一台运行组播的路由器上打上.我这里就配一个动态 **RP** 的.

```
Interface Serial0/1
ip address 34.1.1.4 255.255.255.0
ip pim sparse-mode //启用 pim-SM
serial restart-delay 0
```

```
ip pim autorp listener //设置此路由器为 AUTO-RP 的监听者
```

```
ip pim send-rp-announce 24.1.1.2 scope 5
```

```
ip pim send-rp-announce 34.1.1.3 scope 5 //设置参与选举的 CRP 的地址
```

```
ip pim send-rp-discovery scope 5 //设置发送 RP 发现消息
```

```
*Mar 1 01:05:01.659: Auto-RP(0): Build RP-Announce for 34.1.1.3, PIMv2/v1, ttl 5, ht 181
*Mar 1 01:05:01.659: Auto-RP(0): Build announce entry for (224.0.0.0/4)
*Mar 1 01:05:01.663: Auto-RP(0): Send RP-Announce packet on Serial0/0
*Mar 1 01:05:01.667: Auto-RP(0): Send RP-Announce packet on FastEthernet1/0
*Mar 1 01:05:01.671: Auto-RP: Send RP-Announce packet on Serial0/1
*Mar 1 01:05:01.671: Auto-RP(0): Received RP-announce, from 34.1.1.4, RP_cnt 1, ht 181
*Mar 1 01:05:01.675: Auto-RP(0): Update (224.0.0.0/4, RP:34.1.1.3), PIMv2 v1
R4#
```

上面列举了 **RP** 选择的过程.不难看出这个过程很精彩.

```
(* , 224.1.1.1), 00:05:47/stopped, RP 34.1.1.3, flags: SJC
Incoming interface: Serial0/1, RPF nbr 34.1.1.3
Outgoing interface list:
FastEthernet1/0, Forward/Sparse, 00:05:47/00:02:08
```

//共享树,**RP** 到接受者的共享树

```
(13.1.1.1, 224.1.1.1), 00:02:21/00:00:53, flags: JT
Incoming interface: Serial0/1, RPF nbr 34.1.1.3
Outgoing interface list:
FastEthernet1/0, Forward/Sparse, 00:02:21/00:02:08
```

//源到 **RP** 的源树

```
(* , 224.0.1.39), 00:04:07/00:02:07, RP 0.0.0.0, flags: DCL
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet1/0, Forward/Sparse, 00:04:07/00:00:00
Serial0/1, Forward/Sparse, 00:04:07/00:00:00
Serial0/0, Forward/Sparse, 00:04:07/00:00:00
Loopback0, Forward/Sparse, 00:04:07/00:00:00
```

```
(* , 224.0.1.40), 00:06:48/00:02:09, RP 0.0.0.0, flags: DCL
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
Serial0/1, Forward/Sparse, 00:06:48/00:00:00
Serial0/0, Forward/Sparse, 00:06:48/00:00:00
Loopback0, Forward/Sparse, 00:06:48/00:00:00
```

当然 **PIM** 的模式还有一种 **DM** 和 **SM** 结合使用的我这里就不列举了.通过上面两个实验.我们可以很清楚的知道多播的整个过程.如有兴趣可以去看看 **DVMRP Mospf** 等多播协议.

IPV6

随着互联网的飞速发展，现有的地址空间 **IPV4** 已经快要枯竭，据最新统计，**IPV4** 地址空间可能在 **2011** 年就要耗尽，所以我们急需一种更新的地址空间，更好的规划，所以引入了 **IPV6**。

与 **IPV4**相比，**IPV6**具有以下几个优势：

1>IPv6具有更大的地址空间。**IPv4**中规定 **IP** 地址长度为**32**，即有 **$2^{32}-1$** （符号[^]表示升幂，下同）个地址；而 **IPv6**中 **IP** 地址的长度为**128**，即有 **$2^{128}-1$** 个地址。

2>IPv6使用更小的路由表。**IPv6**的地址分配一开始就遵循聚类（**Aggregation**）的原则，这使得路由器能在路由表中用一条记录（**Entry**）表示一片子网，大大减小了路由器中路由表的长度，提高了路由器转发数据包的速度。

3>IPv6增加了增强的组播（**Multicast**）支持以及对流的支持（**Flow Control**），这使得网络上的多媒体应用有了长足发展的机会，为服务质量（**QoS, Quality of Service**）控制提供了良好的网络平台。

4>IPv6加入了对自动配置（**Auto Configuration**）的支持。这是对 **DHCP** 协议的改进和扩展，使得网络（尤其是局域网）的管理更加方便和快捷。

5>IPv6具有更高的安全性。在使用 **IPv6**网络中用户可以对网络层的数据进行加密并对 **IP** 报文进行校验，极大的增强了网络的安全性。

IPv6 编址规则:

从 IPv4 到 IPv6 最显著的变化就是网络地址的长度。**RFC 2373** 和 **RFC 2374** 定义的 IPv6 地址，就像下面章节所描述的，有 **128** 位长；IPv6 地址的表达形式一般采用 **32** 个十六进制数。

IPv6 中可能的地址有 3.4×10^{38} 个。也可以想象为 **16** 个因为 **32** 位地址每位可以取 **16** 个不同的值。

在很多场合，IPv6 地址由两个逻辑部分组成：一个 **64** 位的网络前缀和一个 **64** 位的主机地址，主机地址通常根据物理地址自动生成，叫做 **EUI-64**（或者 **64**-位扩展唯一标识）。

IPv6 地址表示

IPv6 地址为 **128** 位长，但通常写作 **8** 组，每组为四个十六进制数的形式。例如：

2001:0db8:85a3:08d3:1319:8a2e:0370:7344 是一个合法的 IPv6 地址。

如果四个数字都是零，可以被省略。例如：

2001:0db8:85a3:0000:1319:8a2e:0370:7344 等价于

2001:0db8:85a3::1319:8a2e:0370:7344 遵从这些规则，如果因为省略而出现了两个以上的冒号的话，可以压缩为一个，但这种零压缩在地址中只能出现一次。因此：

2001:0DB8:0000:0000:0000:0000:1428:57ab

2001:0DB8:0000:0000:0000::1428:57ab

2001:0DB8:0:0:0:0:1428:57ab

2001:0DB8:0::0:1428:57ab

2001:0DB8::1428:57ab 都是合法的地址，并且他们是等价的。但

2001::25de::cade 是非法的。(因为这样会使得搞不清楚每个压缩中有几个全零的分组)

同时前导的零可以省略，因此：

2001:0DB8:02de::0e13 等价于 **2001:DB8:2de::e13** 如果这个地址实际上是 **IPv4** 的地址，后 **32** 位可以用 **10** 进制数表示；因此：

ffff:192.168.89.9 等价于 **::ffff:c0a8:5909**，但不等价于

::192.168.89.9 和 **::c0a8:5909**。

ffff:1.2.3.4 格式叫做 **IPv4** 映像地址，是不建议使用的。

而 **::1.2.3.4** 格式叫做 **IPv4** 一致地址。

IPv4 地址可以很容易的转化为 **IPv6** 格式。举例来说，如果 **IPv4** 的一个地址为 **135.75.43.52**(十六进制为 **0x874B2B34**)，它可以被转化为 **0000:0000:0000:0000:0000:0000:874B:2B34** 或者 **::874B:2B34**。同时，还可以使用混合符号 (**IPv4-compatible address**)，则地址可以为 **::135.75.43.52**。

IPV6 的地址空间:

Unicast:

1>AUGA(可聚合全球单播地址): 范围: **2000::/3** ; **internet** 使用 **2001::/16** 而 **2002::/16** 为 **6to4** 转换专用。

3	13	8	24	16	64 位
FP	TLA ID	RES	NLA ID	SLA ID	Interface ID

其中 **FP = Format prefix** (格式前缀), 对于可聚集全局单播地址, 其值为“001”**TLA ID = Top-level Aggregation Identifier** (顶级聚集标识符) **RES = Reserved for future use** (保留以备将来使用) **NLA ID = Next-Level Aggregation Identifier** (下一级聚集标识符) **SLA ID = Site-Level Aggregation Identifier** (站点级聚集标识符)

2>本地链路地址(Link-local):FE80::/10

10 位	54 位	64 位
1111111010	0	接口 ID

链路本地地址用于在单个链路上对节点进行寻址。来自或发往链路本地地址的数据包不会被路由器转发。

3>Site-local 本地单播地址, 公网不路由:FEC0::/10

10 位	38 位	16 位	64 位
1111111011	0	子网 ID	接口 ID

站点本地地址应在同一站点内使用。路由器不会转发任何站点本地源地址或目标地址是站点外部地址的数据包。

4>未制定地址 ::

5>Lookback 地址 ::1 类似 IPV4 中的 127.0.0.1

Multicast: FF00::/8

8 位	4 位	4 位	112 位
11111111	标志	范围	组 ID

地址开头的“**FF**”标识该地址是一个多播地址。

“标志”字段是一组四个标志“**000T**”。高位顺序的三位是保留位，必须为零。最后一位“**T**”说明它是否被永久分配。如果该值为零，说明它被永久分配，否则为暂时分配。

“范围”字段是一个四位字段，用于限制多播组的范围。例如，值“**1**”说明该多播组是一个节点本地多播组。值“**2**”说明其范围是链路本地。

“组 ID”字段标识多播组。以下是一些常用的多播组：所有节点地址 = **FF02:0:0:0:0:0:0:1**（链路本地）

所有路由器地址 = **FF02:0:0:0:0:0:0:2**（链路本地）所有路由器地址 = **FF05:0:0:0:0:0:0:2**（站点本地）

Anycast:

一到最近的（从单播地址空间分配）。

多个设备共享相同的地址。

所有的选播节点应该提供统一的服务。

源设备发送数据包任播地址。

路由器决定最接近设备达到这一目标。

适合用于负载均衡和内容交付服务。

嵌入 **IPV4** 地址:

1>兼容 IPV4的 IPV6地址:地址通过其低位顺序的 **32** 位携带 **IPv4** 地址。

80 位	16 位	32 位
零	0000	IPv4 地址

2>映射 IPv4 的 IPv6 地址:有一种特殊类型的 **IPv6** 地址，其中包含嵌入的 **IPv4** 地址。可以采用这种地址将只支持 **IPv4** 的节点的地址表示为 **IPv6** 地址。该地址特别适用于既支持 **IPv6** 又支持 **IPv4** 的应用程序。

80 位	16 位	32 位
零	FFFF	IPv4 地址

NDP:邻居发现协议:

NDP 的功能:

1>路由器发现(router discovery) :当一个节点连接到 **IPV6** 链路时,能够自动发现本地路由器.不需要借助 **DHCP**.

2>前缀发现(prefix discovery):当一个节点连接到 **IPV6** 时,它能自动发现分配给该链路的前缀.

3>参数发现(parameter discovery):节点能够发现它所连接的链路的参数.

4>地址自动配置(address autoconfiguration):节点能够确定它的完整地址.

5>地址解析(address resolution): 节点不需要使用 **ARP** 就能够发现链路上其他节点的 **MAC**

6>下一跳确定(next-hop determination): 链路上的节点能够确定到达目的的下一条 **MAC** 或本地目的节点或到达节点的路由器。

7>邻居不可达检测(neighbor unreachability detection): 节点能够检测到链路上的邻居何时不可达.

8>地址冲突检测(duplicate address detection):节点能够检测到地址是否已经被链路上其他节点使用.

9>重定向(redirect):对于非连接的目的节点,路由器能够通过重定向消息通知主机存在比它更好的下一跳路由.

NDP 消息通常在链路本地范围内收发,封装 **NDP** 消息的数据包也始终使用 **IPV6** 链路本地地址或者链路本地范围内的 **multicast**.

NDP 消息:

路由器通告(**router advertisement,RA**)消息:由路由器发起,通告路由器的存在和链路细节参数.

路由器请求(**router solicitation,RS**)消息:由主机发起,用来请求路由器发送一个 **RA**.

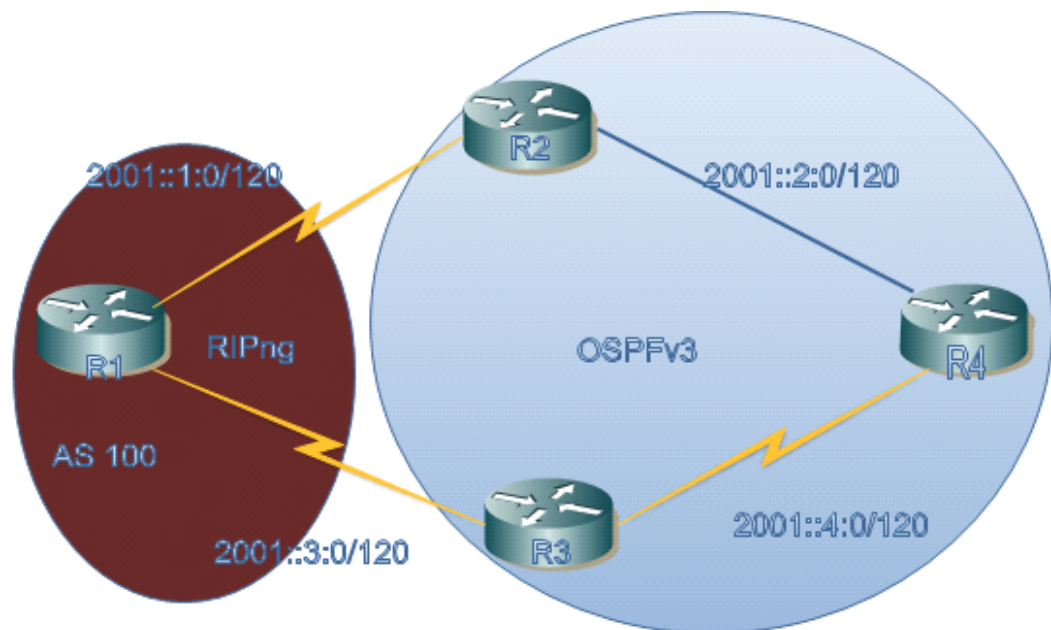
邻居请求(**neighbor solicitation,NS**)消息:有节点主机发起,用来请求另一台主机的链路层地址,也可用来实现冲突检测,邻居不可达等功能.

邻居通告(**neighbor advertisement,NA**)消息:用来响应 **NS** 消息
重定向(**redirect**)消息:和 **IPV4** 的 **ICMP** 中的相同.

其实 **IPV6** 很多原理和机制和 **IPV4** 还是很相同的,毕竟 **IPV6** 是 **IPV4** 的一个升级版本.有兴趣的可以看看 **IPV6** 专门介绍的知识.这里只是粗略的写了一点 **IPV6**,让大家能有个概念和了解.下面我着重来谈一谈 **RIPNG OSPFV3** 这两个 **IPV6** 的协议,其实原理和 **IPV4** 中一样,我用实验来配置一下,看看 **ipv6** 的基本配置情况.后面我们再谈一谈 **IPV4** 到 **IPV6** 的过渡.

IPV6 实验基本配置

实验拓扑:



RIPNG 基本配置:

ipv6 unicast-routing //开启 **IPV6** 路由功能

ipv6 router rip cisco //启动 **ripng**

interface Loopback0

no ip address

ipv6 address 1111::1/128

ipv6 rip cisco enable //将接口加如 **RIPNG**

OSPFV3 基本配置:

ipv6 unicast-routing //开启 **IPV6** 路由功能

ipv6 router ospf 1 //启动 **OSPFV3** 进程

router-id 4.4.4.4 //设置路由 **ID**

interface FastEthernet2/0

no ip address

duplex auto

speed auto

ipv6 address 2001::2:4/120

ipv6 ospf 1 area 0 //将接口加入 **OSPFV3** 进程.

其实 **RIPNG** 和 **OSPFV3** 的总体机理和 **IPV4** 的 **RIP,OSPF** 差异不是很大,基本上一致.配置起来也不复杂.

OSPFV3 的 DR/BDR 选举:

```
*Mar 1 00:14:24.131: OSPFv3: Nbr state is 2WAY
*Mar 1 00:14:24.291: OSPFv3: end of Wait on interface FastEthernet2/0
*Mar 1 00:14:24.291: OSPFv3: DR/BDR election on FastEthernet2/0
*Mar 1 00:14:24.295: OSPFv3: Elect BDR 4.4.4.4
*Mar 1 00:14:24.295: OSPFv3: Elect DR 4.4.4.4
*Mar 1 00:14:24.299: OSPFv3: Elect BDR 2.2.2.2
*Mar 1 00:14:24.299: OSPFv3: Elect DR 4.4.4.4
*Mar 1 00:14:24.299:      DR: 4.4.4.4 (Id)    BDR: 2.2.2.2 (Id)
```

RIPNG 和 OSPF 的重分发:

其实这个重分发和 **IPV4** 中的区别不是很大.命令如下:

ipv6 router ospf 1

router-id 2.2.2.2

log-adjacency-changes

redistribute rip cisco metric 20 metric-type 1

ipv6 router rip cisco

redistribute ospf 1 metric 3

R1 的路由表:

```
          ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
LC  1111::1/128 [0/0]
    via ::, Loopback0
C   2001::1:0/120 [0/0]
    via ::, Serial0/0
L   2001::1:1/128 [0/0]
    via ::, Serial0/0
C   2001::3:0/120 [0/0]
    via ::, Serial0/1
L   2001::3:1/128 [0/0]
    via ::, Serial0/1
R   2001::4:0/120 [120/4]
    via FE80::CE02:DFF:FE10:10, Serial0/0
R   3333::3:3/128 [120/4]
    via FE80::CE02:DFF:FE10:10, Serial0/0
R   4444::4:4/128 [120/4]
    via FE80::CE02:DFF:FE10:10, Serial0/0
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
```

R4 的路由表:

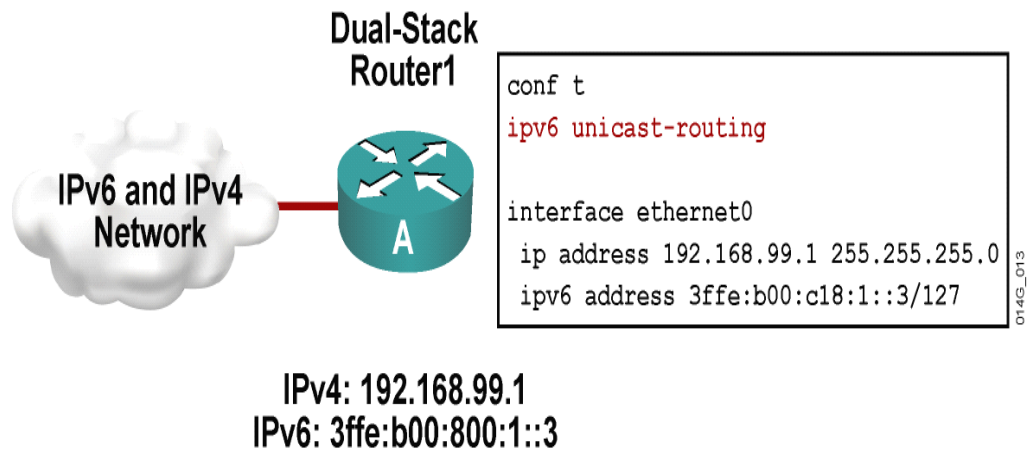
```
          ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OE1  1111::1/128 [110/21]
    via FE80::CE02:DFF:FE10:20, FastEthernet2/0
C   2001::2:0/120 [0/0]
    via ::, FastEthernet2/0
L   2001::2:4/128 [0/0]
    via ::, FastEthernet2/0
OE1  2001::3:0/120 [110/21]
    via FE80::CE02:DFF:FE10:20, FastEthernet2/0
C   2001::4:0/120 [0/0]
    via ::, Serial0/1
L   2001::4:4/128 [0/0]
    via ::, Serial0/1
O   2222::2:2/128 [110/1]
    via FE80::CE02:DFF:FE10:20, FastEthernet2/0
O   3333::3:3/128 [110/64]
    via FE80::CE03:DFF:FE10:10, Serial0/1
LC  4444::4:4/128 [0/0]
    via ::, Loopback0
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
R4#
```

可以看出和 **IPV4** 区别不大.

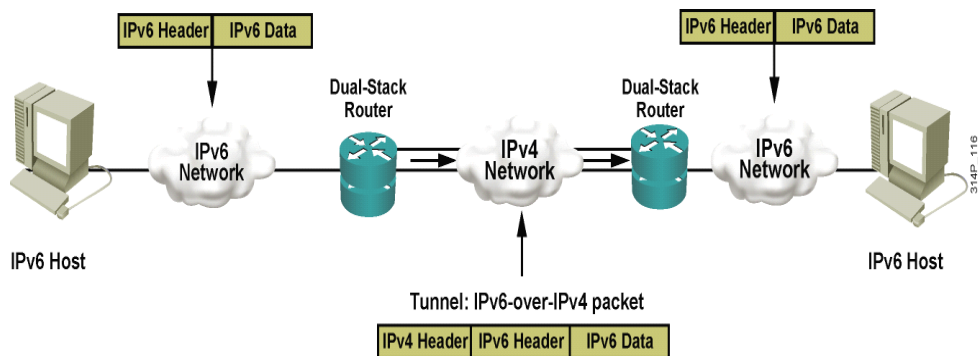
IPv4 到 IPv6 过渡:

4 到 6 过渡可以使用 3 种机制,常用的两种:

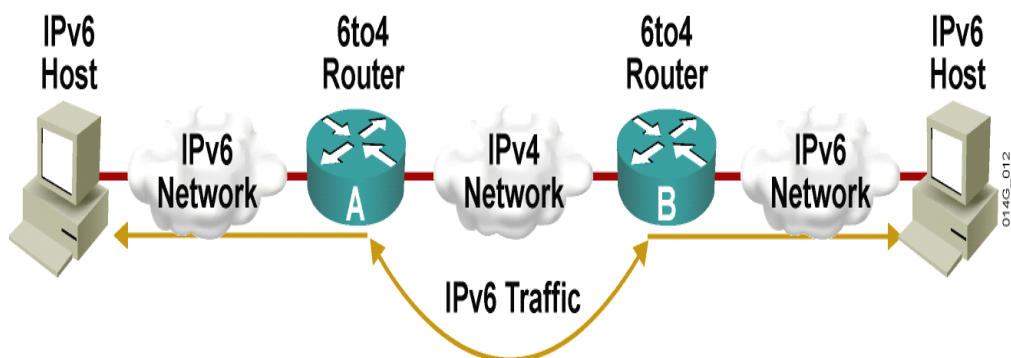
1>双栈



2>6to4 tunnel



3>NAT-PT



1>双协议栈 (Dual Stack)

采用该技术的节点上同时运行 **IPv4**和 **IPv6**两套协议栈。这是使 **IPv6**节点保持与纯 **IPv4**节点兼容最直接的方式，针对的对象是通信端节点（包括主机、路由器）。这种方式对 **IPv4**和 **IPv6**提供了完全的兼容，但是对于 **IP** 地址耗尽的问题却没有任何帮助。由于需要双路由基础设施，这种方式反而增加了网络的复杂度。

2>隧道技术 (Tunnel)

隧道技术提供了一种以现有 **IPv4**路由体系来传递 **IPv6**数据的方法：将 **IPv6**的分组作为无结构意义的数​​据，封装在 **IPv4**数据报中，被 **IPv4**网络传输。根据建立方式的不同，隧道可以分成两类：(手工)配置的隧道和自动配置的隧道。隧道技术巧妙地利用了现有的 **IPv4**网络，它的意义在于提供了一种使 **IPv6**的节点之间能够在过渡期间通信的方法，但它并不能解决 **IPv6**节点与 **IPv4**节点之间相互通信的问题。

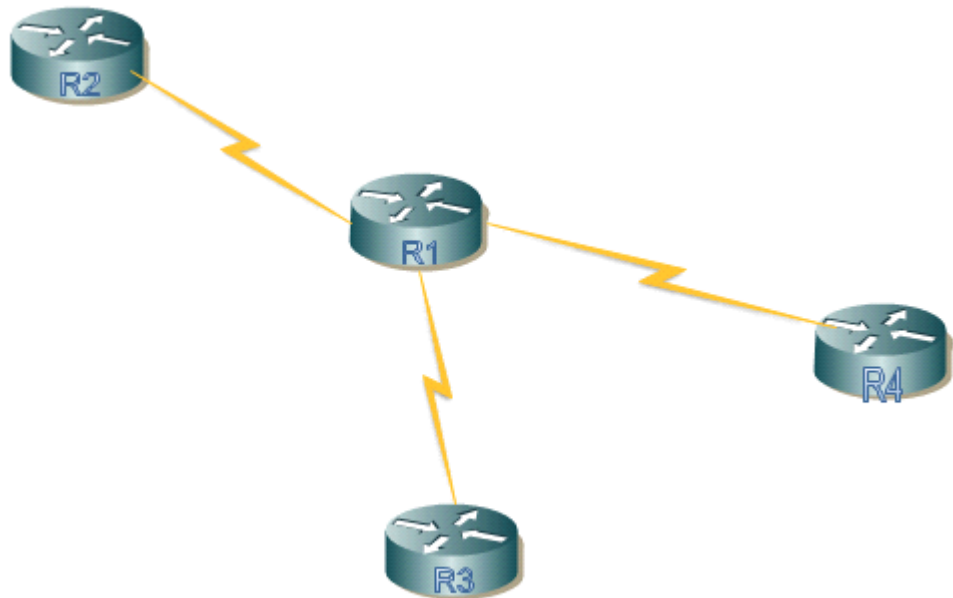
3> NAT-PT

转换网关除了要进行 **IPv4**地址和 **IPv6**地址转换，还要包括协议并翻译。转换网关作为通信的中间设备，可在 **IPv4**和 **IPv6**网络之间转换 **IP** 报头的地址，同时根据协议不同对分组做相应的语义翻译，从而使纯 **IPv4**和纯 **IPv6**站点之间能够透明通信。

下面我来延时两个实验分别是现在国内使用的双栈和**6to4tunnel**:

1>双栈:

拓扑



R1为双栈路由器，**R3**代表 **IPV6**网络，**R4**代表 **IPV4**网络，**R2**代表客户

配置其实很简单，**R1**上所有接口运行 **IPV4**和 **IPV6**地址。然后运行两个协议，**R2**上配置默认路由给 **R1**，模拟主机，接口也是两个地址。这样就形成了双栈。

R1配置:

```
interface Serial0/0
```

```
ip address 12.1.1.1 255.255.255.0
```

```
ipv6 address 2001::12:1/120
```

```
ipv6 rip cisco enable
```

serial restart-delay 0

interface Serial0/1

ip address 14.1.1.1 255.255.255.0

ipv6 address 2001::14:1/128

ipv6 rip cisco enable

serial restart-delay 0

interface Serial0/2

ip address 13.1.1.1 255.255.255.0

ipv6 address 2001::13:1/128

ipv6 rip cisco enable

serial restart-delay 0

R2配置:

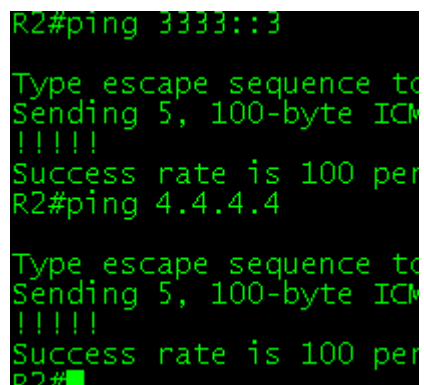
R2(config)#ipv6 route ::/0 2001::12:1 //缺省 ipv6路由

interface Serial0/0

ip address 12.1.1.2 255.255.255.0

ipv6 address 2001::12:2/120

PingR3 R4回还效果:



```
R2#ping 3333::3
Type escape sequence to abort
Sending 5, 100-byte ICMP Echos to 3333::3: 5 received, 0 timeouts, 5 successes
!!!!
Success rate is 100 percent (5/5)
R2#ping 4.4.4.4
Type escape sequence to abort
Sending 5, 100-byte ICMP Echos to 4.4.4.4: 5 received, 0 timeouts, 5 successes
!!!!
Success rate is 100 percent (5/5)
R2#
```

6to4tunnel:

实验拓扑:



蓝色区域为 **IPV6** 区域,绿色区域为 **IPV4** 区域,要求两个 **IPV6** 区域能互访!

具体配置:

```
interface Tunnel0
  no ip address
  ipv6 address 2001::14:1/120
  ipv6 rip cisco enable
  tunnel source 14.1.1.1
  tunnel destination 14.1.1.4
  tunnel mode ipv6ip
interface Serial0/0
  no ip address
  ipv6 address 2001::12:1/120
  ipv6 rip cisco enable
  serial restart-delay 0
interface Serial0/1
  ip address 14.1.1.1 255.255.255.0
  serial restart-delay 0
```

实验结果:

```
R2#ping 3::3
Type escape sequence
Sending 5, 100-byte I
!!!!
Success rate is 100 p
R2#
```

其实配置还是比较简单的.原理就是 **R1-R4** 中间打个隧道配上 **IPV6** 的地址来穿越 **IPV4** 的网络.

后记:

经过一个多星期的写作,正个 **TCP/IP** 路由技术学习手册就此告一段落了,非常感谢朋友的关注和支持,也有很多热心的读者一直给我提建议,发现我的错误.在此深表感谢.

本人水平有限,可能有很多地方交代的不是很清楚,还望大家多多见谅,也希望这本手册能给您学习的过程中带来一点点小小的帮助.我会继续我的整理之路.

本着互相学习的原则,后续会陆续推出一系列的实验作品,会是以视频的方式展出来,我的原则是最少的机器完成尽可能多的实验!

至此,整个 **tcp/ip** 学习手册就写完了.如有错误之处在所难免,还望大家多多包涵!!谢谢

似水年华

2010 年 5 月 10 日深夜