IPv6 技术笔记

红茶三杯 CCIE 学习文档

文档版本： 3.0 更新时间： 2013-07-02

文档作者： 红茶三杯 文档地址： http://ccietea.com

文档备注： 访问 ccietea.com 以获得最新的文档版本。文档的编撰倾注了作者大量的心血、时间和精力，供广大技术爱好者交流分享，请勿用于商业用途，转载请保留作者信息。

1 IPv6 概述

1. IPv6 特点

- 128bits 的地址方案，为未来数十年提供了巨大的 IP 地址空间
- 多等级层次有助于路由聚合，提高了因特网网络路由的效率及可扩展性
- 自动配置过程允许 IPv6 网络中的节点更加便捷的接入 IPv6 网络
- 重新编址机制使得 IPv6 提供商之间的转换对最终用户是透明的
- 无需 NAT
- 不再有广播、不再有 ARP
- IPv6 的包头比 IPv6 更有效率，数据字段更少，去掉了包头校验和。更简单的报头提高了路由器的处理效率。新的扩展包头替代了 IPv4 的选项字段，并且提供了更多的灵活性
- 更有效的支持移动性和安全性
- v4v6 过渡方式丰富多彩

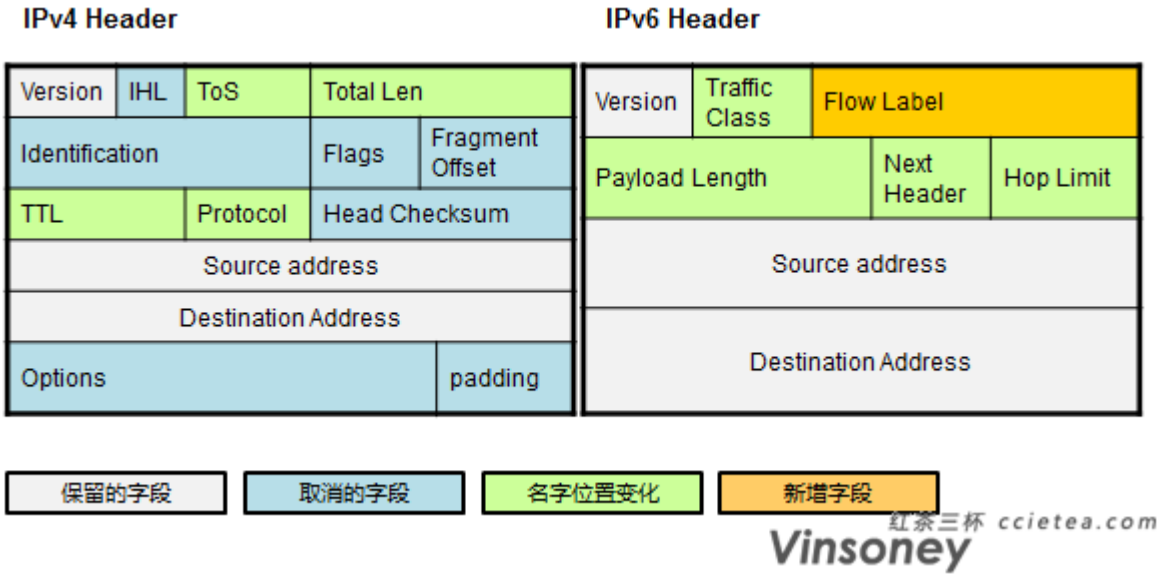
2. 6bone

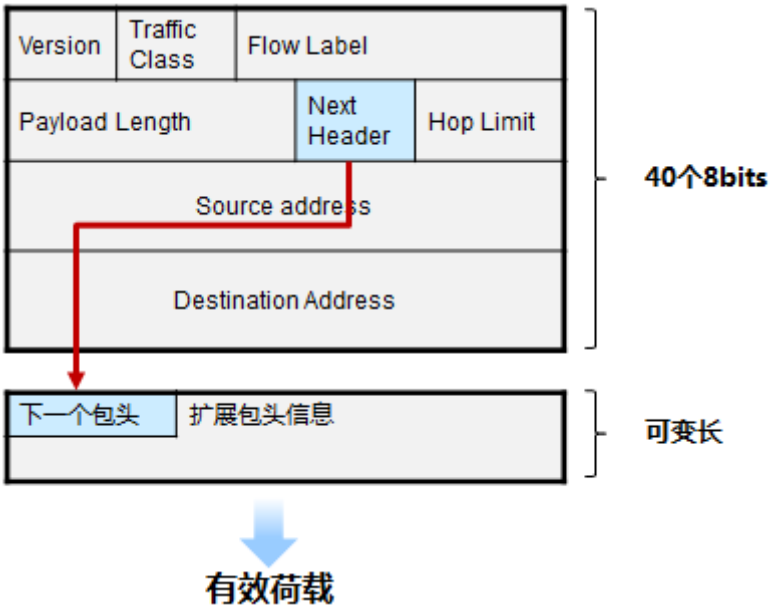
6bone 是一个世界范围的非正式合作的 IPv6 试验床。也是 IETF IPng 项目的成果。始于 1996 年，是运行在因特网上的由 IPv6 的 IPv4 隧道组成的虚拟网络，网络在缓慢的向纯 IPv6 链路迁移。更多的消息，请关注

2 IPv6 Header

2.1 IPv6 Header

1. **包头格式**





- **基本的 IPv4 报头包含 12 的字段，20 个字节长。options 和 padding 字段在需要时添加。**

  IHL 包头长度；

  Total Length 总长度；

- **基本的 IPv6 报头 40 个 8 位 bit，即 40 个字节长，共 8 个字段。**

  IPv6 数据包由一个基本报头加上 0 个或多个扩展报头再加上上层协议单元构成。

  几个字段的含义：

  下一个报头(Next Header)： 该字段定义了紧跟在 IPv6 报头后面的第一个扩展报头(如果存在)的类型，或者上层协议数据单元中的协议类型。

  跳限制(Hop Limit)： 类似于 IPv4 中的 TTL 字段。它定义了 IP 数据包所能经过的最大跳数。每经过一个路由器，该数值减去 1，当该字段的值为 0 时，数据包将被丢弃

  FlowLabel： **流标签，用于 QoS**

2. IPV6 包头的改进

- 取消了 IP 的校验

  第二层和第四层的校验已经足够健壮了，因此 IPv6 直接取消了 IP 的三层校验。

- 取消中间节点的分片功能

  分片重组功能由源目端自己进行，通过 PMTU 机制来发现路径 MTU

- 定义最长的 IPv6 报头

  有利于硬件的快速处理，如此一来中间节点可以避免处理而节省大量的资源

- 安全选项的支持

  IPv6 提供了对 Ipsec 的完美支持，如此上层协议可以省去许多安全选项，例如 OSPFv3 就取消了认证

- 增加流标签

  提高 QoS 效率

3. 扩展报头

在 IPv6 中，v4 中的选项被移到了扩展报头中。一个 IPv6 数据包可能包括 0 个或多个扩展包头，当使用多个扩展包头时，通过前面的包头的 Nexthead 字段指明该扩展包头后的扩展包头。有了扩展包头，中间路由器就不需要处理每一个可能出现的选项，提高了路由器处理数据包的速度，提高了其转发性能。在扩展报头链的最后就是有效负载。

扩展报头可选，只有需要该扩展报头对应的功能，发送主机才会添加相应扩展报头

- **逐跳选项报头 Hop-by—Hop Options Header（协议 0）**

  传送路径上每个路由器都要处理，用于巨型数据包和路由器警报。如 RSVP 资源预留协议

- **目标选项报头 Destination Options Header（协议 60）**

  该包头承载特别针对数据包目的地的可选信息，例如用在移动节点和家乡代理之间交换注册信息，移动

IP 是这样的一个协议，即使移动节点改变了连接点，仍允许它保持永久的 IP 地址

- **路由包头 Routing Header （协议 63）**

  在数据包发往目的地的途中，该包头能够被 IPv6 源节点用来强制数据包经过特定的路由器，当路由类型字段为 0 时，在路由包头中可以指定中间路由器列表，这个功能类似 IPv4 中的松散源路由选项。

- **分段报头 Fragment Header （协议 44）**

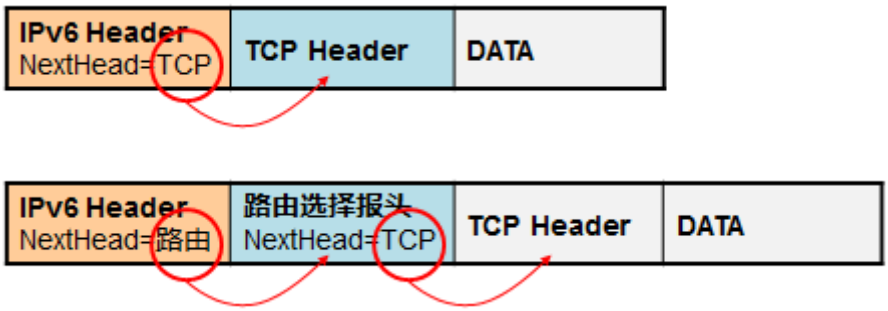
  在 IPv6 中，建议所有的节点使用 PMTUD 机制来发现链路的 MTU，这样就不需要对数据进行分段了。但是如果 IPv6 节点不支持 PMTUD，但又必须发送比路径 MTU 还大的数据包时，就不得不分段了，这时候需使用分段包头，节点将数据分段，使用分段包头发送每个分段。

- 认证报头 Authentication Header （协议 51）

  AH 头，这个很熟悉了

- 封装安全有效载荷报头 Encapsulating Security Payload Header（协议 51）

  ESP 头



3 IPv6 编址

3.1 基础

1. **IPv6 地址简写**

   IPv6 地址在简写的时候多个前导 0 可以省略成一个 0；

   如：2001：**00**a8：**0**207：0000：0000：0000：0000：8207

   可缩写成：2001：a8：207：0：0：0：0：8207

   一个或多个连续的 16 比特字段为 0 时，可用：：表示，但整个缩写中只允许有一个：：

   如上面的例子，可以进一步缩写成：2001：a8：207：：8207

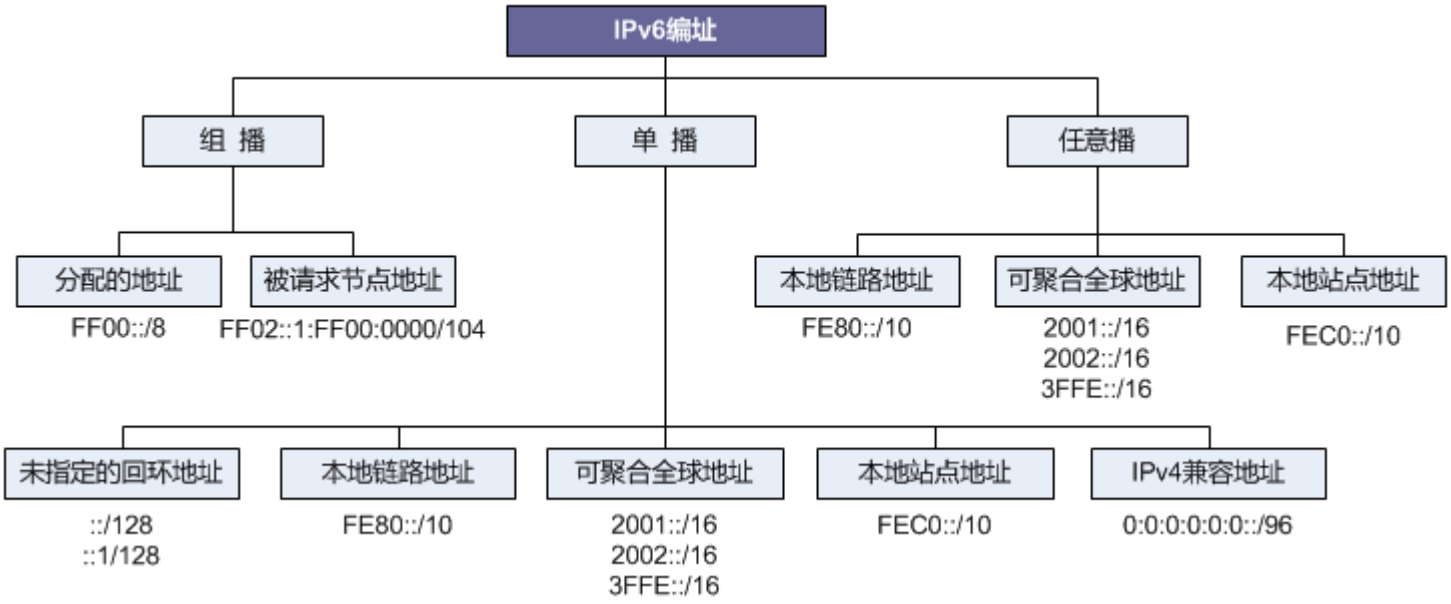
2. IPv6 地址的 URL 表示

http://[ipv6 地址]：8080　　!! 必须用 [ ] 括起来

3. IPV6 和子网划分

IPv6 与 v4 的不同之处是网络前缀范围内没有保留广播号，在 v4 中，第一个和最后一个地址是被保留给网络号和广播号的。而在 IPv6 中，不再有广播了。而且事实上 IPv6 的地址空间如此之大，以至于 VLSM 都不是特别有必要了。

3.2 地址分类



整个 IPv6 空间的已分配部分：

二进制的前缀（高 bits）	Hex	Size	Allocation
0000 0000 xxxx xxxx	0000 – 00FF	1/256	Reserved 未指定、回环、IPv4 兼容地址
0000 0001 xxxx xxxx	0100 – 01FF	1/256	Unassigned
0000 001x xxxx xxxx	0200 – 03FF	1/128	NSAP
0000 010x xxxx xxxx	0400 – 05FF	1/128	IPX -> moving to Unassigned
0000 011x xxxx xxxx	0600 – 07FF	1/128	Unassigned
0000 1xxx xxxx xxxx	0800 – 0FFF	1/32	Unassigned
0001 xxxx xxxx xxxx	1000 – 1FFF	1/16	Unassigned
001x xxxx xxxx xxxx	2000 – 3FFF	1/8	可聚合全球单播地址（IANA to registers）
010x xxxx xxxx xxxx	4000 – 5FFF	1/8	Unassigned

011x xxxx xxxx xxxx	6000 – 7FFF	1/8	Unassigned
100x xxxx xxxx xxxx	8000 – 9FFF	1/8	Unassigned
101x xxxx xxxx xxxx	A000 – BFFF	1/8	Unassigned
110x xxxx xxxx xxxx	C000 – DFFF	1/8	Unassigned
1110 xxxx xxxx xxxx	E000 – EFFF	1/16	Unassigned
1111 0xxx xxxx xxxx	F000 – F7FF	1/32	Unassigned
1111 10xx xxxx xxxx	F800 – FBFF	1/64	Unassigned
1111 110x xxxx xxxx	FC00 – FDFF	1/128	Unassigned
1111 1110 0xxx xxxx	FE00 – FE7F	1/512	Unassigned
1111 1110 10xx xxxx	FE80 – FEBF	1/1024	本地链路
1111 1110 11xx xxxx	FEC0 – FEFF	1/1024	本地站点
1111 1111 xxxx xxxx	FF00 – FFFF	1/256	组播

几点说明：

- 200::/7 保留用于网络业务接入点 NSAP ,当前没有使用 NSAP 保留空间 ,NSAP 地址主要用于 ATM 技术中。
- 2000::/3 (2000::到 3FFF::)是可聚合全球单播地址空间，

3.2.1 单播地址

1. **可聚合全局单播地址（Aggregatable global unicast address）**

   目前已经分配的全球单播地址前缀都是以 001 开头的，(2000：： 到 3FFF:FFFF:….FFFF)

   以下是这个区间地址的分配情况：

Unicast Global [001]		
2001::/16	0010 0000 0000 0001	IPv6 InternetARIN, APNIC, RIPE NCC, LACNIC
2002::/16	0010 0000 0000 0010	6 to 4 transition mechanisms
2003::/16	0010 0000 0000 0011	IPv6 InternetRIPE NCC
2400:0000::/19 2400:2000::/19 2400:4000::/21	0010 0100 0000 0000	IPv6 InternetAPNIC
2600:0000::/22 2604:0000::/22 2608:0000::/22 260C:0000::/22	0010 0110 0000 0000 0010 0110 0000 0100 0010 0110 0000 1000 0010 0110 0000 1100	IPv6 InternetARIN
2A00:0000::/21 2A01:0000::/23	0010 1010 0000 0000 0010 1010 0000 0001	IPv6 InternetRIPE NCC
3FFE::/16		6bone

一般从运营商处申请到的 IPv6 地址空间为/48，再由自己根据需要进一步规划：

如下图：

2. 本地站点地址 (Site-local address)

类似 IPv4 中的私有地址。

以 FEC0 : : /10 为前缀。其中前 10 bits 固定为 1111111011，紧跟在后面的是连续 38 bits 的 0。因此，对于站点本地地址来说，前 48bits 总是固定的。在接口 ID 和高位 48bits 特定前缀之间有 16bits 子网 ID 字段，供机构在内部构建子网。站点本地地址不是自动生成的，是手工配置的。

本地站点地址永远不会用于与全球 ipv6 因特网通信。一般用于内网通信。当然，其实 IPv6 地址空间如此之庞大，以至于根本不需要用到 Site-local 地址，直接用全局单播地址即可。

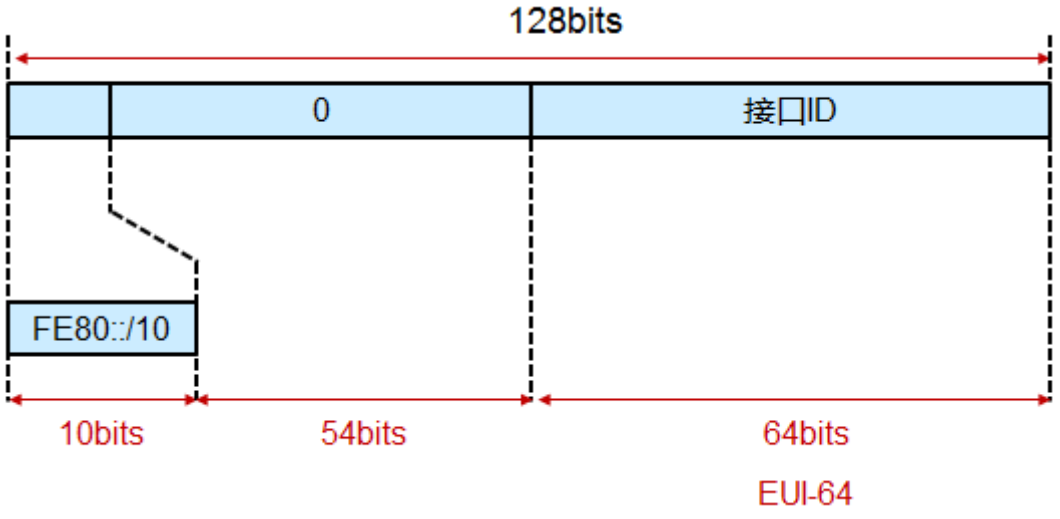
3. 链路本地地址 (Link-local address)

以 FE80::/10 为前缀，11-64 位为 0，外加一个 64bits 的接口标识（一般是 EUI-64）。

只能在连接到同一本地链路的节点之间使用，也就是链路范围内有效。

用于自动地址配置、邻居发现、路由器发现等。

当一个节点启动 IPv6 协议栈时，节点的每个接口会自动配置一个链路本地地址。这种机制使得两个连接到同一链路的 IPv6 节点不需要做任何配置就可以通信 。缺省网关建议使用链路本地地址，因为这个地址是最稳定的。



4. **特殊的地址**

- **未指定地址：全 0 地址 ::　　0:0:0:0:0:0/128　或者 :/128**

  该地址可以表示某个接口或者节点还没有 IP 地址，可以作为某些报文的源 IP 地址（比如作为 DAD 报文的 DHCP 初始化过程中客户端的源 IP）。用于 IPv6 节点在没有获取 IPv6 地址时的场景

- **回环地址：::1　　0:0:0:0:0:1/128　或者::1/128**

  用于 IPv6 节点发送报文给自己

- **IPv4 兼容地址**

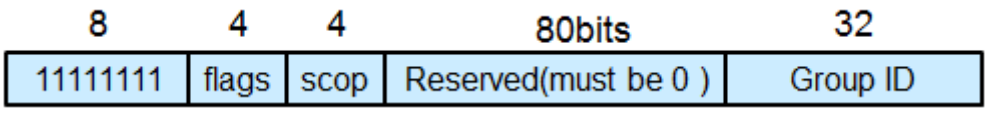
  在过渡技术中会使用到一些包含 IPv4 地址的 IPv6 地址，为了让 IPv4 地址显得更加突出一些，定义了内嵌 IPv4 地址的 IPv6 地址格式。内嵌 IPv4 地址格式是过渡机制中使用的一种特殊表示方法。在这种表示方法中，IPv6 地址的部分使用十六进制表示，IPv4 地址部分可用十进制格式。

  0:0:0:0:0:0:192.168.1.2 或者::192.168.1.2（96 个 0）

  IPv4 兼容地址用于过渡机制，如自动 ipv4 兼容隧道 及 NAT-PT（详细请见本文档 IPv6 过渡技术章节）。

3.2.2 Multicast

1. **地址结构**



组播地址最高位前 8 位固定为全 1　　FFXX：：/8

Flags　4bits，0000：永久分配或众所周知的；0001：临时的

Scop　用来限制组播数据流在网络中发送的范围。

- 0：预留；
- 1：节点本地范围；　单个接口有效，仅用于 Loopback 通讯
- 2：链路本地范围；　如 FF02::1 表示链路上的所有节点；FF02::9：表示链路上的所有 RIP 路由器
- 3：本地子网范围；
- 4：本地管理范围；
- 5：本地站点范围；
- 8：组织机构范围；
- E：全球范围；

- F：预留。

Group-ID 该字段长度可以为 112 位，用来标识组播组，而 112 位最多可以生成 2112 个组 ID ，RFC2373 并没有将所有的 112 位都定义成组标识 ,而是建议仅使用该 112 位的最低 32 位组 ID，将剩余的 80 位都置 0。

2. IPv6 有一些特殊的组播地址，这些地址有特别的含义

FF01::1 节点本地范围所有节点组播地址

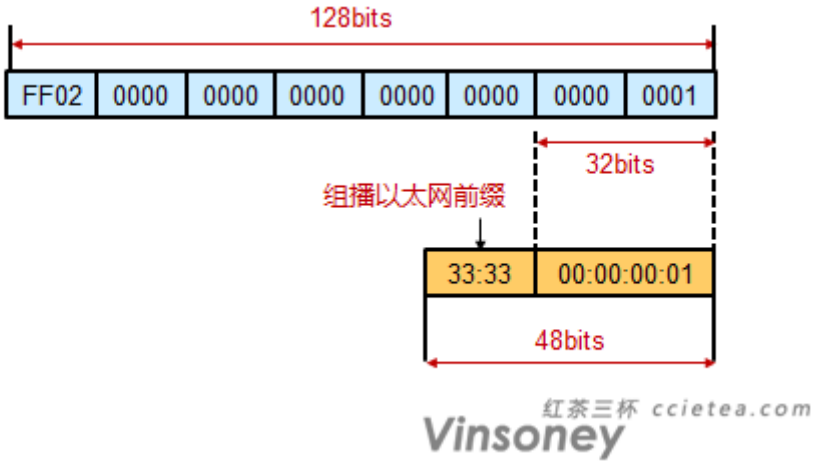
FF01 ::2 节点本地范围所有路由器组播地址

FF02::1 链路本地范围所有节点组播地址

FF02::2 链路本地范围所有路由器组播地址

.....

3. IPv6 组播地址的 MAC 地址映射



4. 被请求节点组播地址 Solicited-node

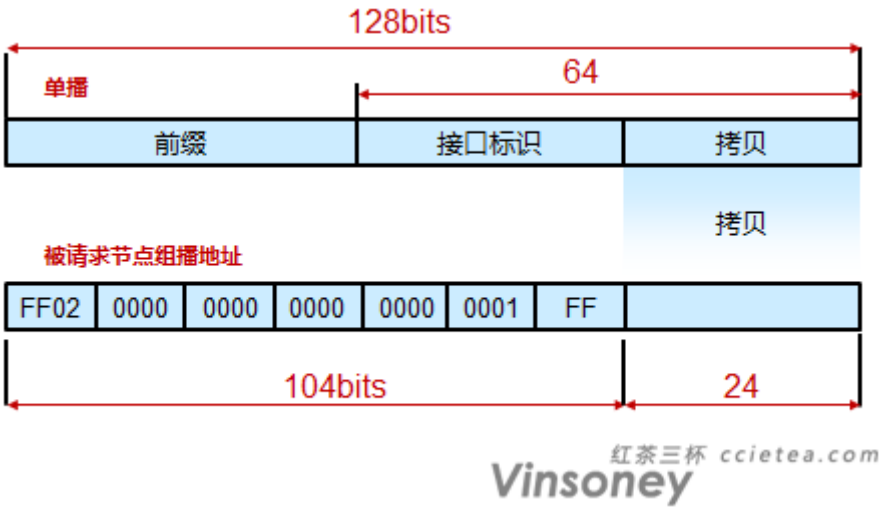
在 IPv6 组播地址中，有一种特别的组播地址，叫做 Solicited-node 地址(被请求节点组播地址)。

Solicited-node 地址是一种特殊用途的地址．主要用于重复地址检测（DAD）和替代 IPv4 中的 ARP。

Solicited-node 地址由前缀 FF02::1:FF00:0 / 104 和 ipv6 单播地址的最后 24 位组成。

一个 IPv6 单播地址对应一个 Solicited-node 地址。Solicited-node 地址有效范围为本地链路范围。

地址格式： FF02:0:0:0:0:1:FFXX:XXXX，具体的对应关系如下：



3.2.3 Anycast

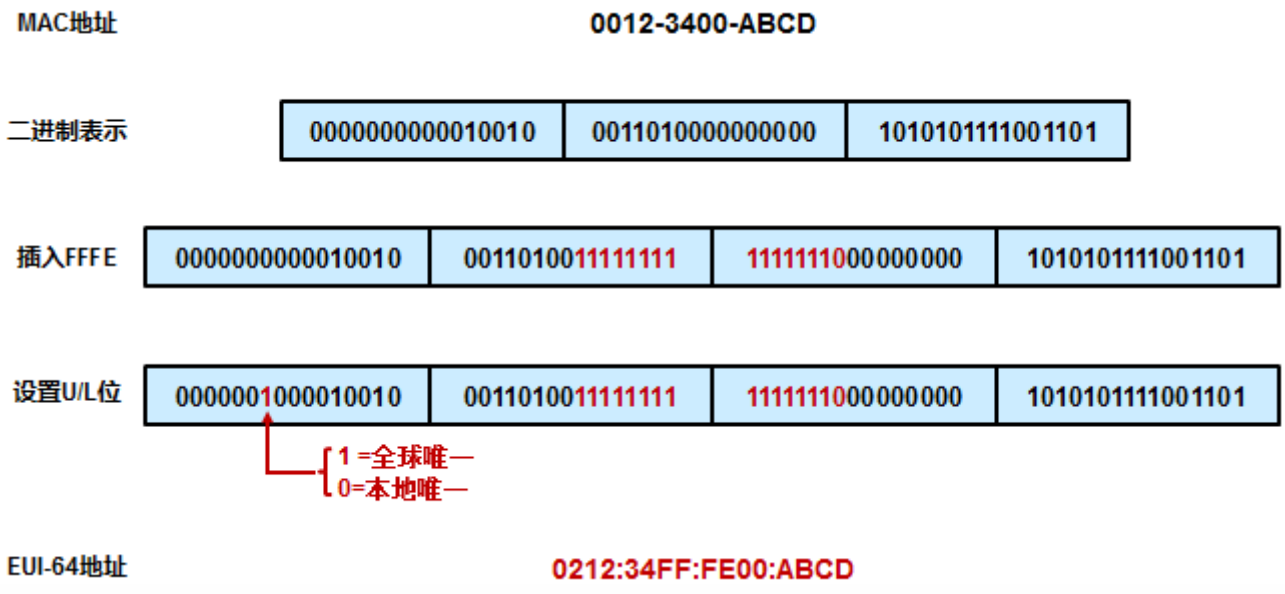
任播地址没有专门的地址空间，使用的是单播的地址空间。一个 IPv6 地址被分配给多个接口，仅用于路由器，发往任播地址的数据包被路由给分配了任播地址中的最近的一个（由路由协议判断远近）。适合于 One-to-One-of-Many(一对一组中的一个)的通信场合。接收方只需要是一组接口中的一个即可，如移动用户上网就需要因地理位置的不同而接入离用户最近的一个接收站，这样才可以使移动用户在地理位置上不受太多的限制。任播地址只能作为目的 IP，不能作为源。

3.3 接口 ID

接口 ID 为 64bits，可以根据 IEEE 的 EUI-64 规范将 48 比特的 MAC 地址转化为 64 比特的接口 ID。

MAC 地址的唯一性保证了接口 ID 的唯一性。

设备自动生成，不需人为干预，由接口的 MAC 地址转换得到 64bits 的 EUI-64 格式接口 ID 的过程如下：



将 48bits 的 MAC 对半劈开，插入 FFFE，然后设置第 7 位，也就是 U/L 位，该比特位确定 48bits 的 MAC 地址的唯一性与否。一个以太网地址可以有两种含义，地址可被全球管理或本地管理。全球管理指使用如 0800-2bxx-xxxx 之类的厂商 MAC，本地管理指使用自己的值重写 MAC 地址，在这种情况下，这个特殊的位=1 表示本地管理；为 0 表示全球管理。但是在 EUI-64 格式中，U/L 的含义正好相反，0 表示本地管理，1 表示全球管理，所以使用 EUI-64 格式的地址 IPv6 地址，U/L 位为 1，则地址是全球唯一的，如果为 0，则为本地唯一。

3.4 必须具备的 IPv6 地址

- **节点必须具备的 IPv6 地址：**

链路本地地址	FE80::/10
回环地址	::1
所有节点组播地址	FF01::1；FF02::1
分配的可聚合全球单播地址	2000::/3
所使用的每个单播和任意播地址对应的被请求节点组播地址	FF02::1:FFxx:xxxx 其中 xx:xxxx 是每个单播或任意播地址的低 24 比特
主机所属的所有组的组播地址	FF00::/8

以上是主机节点必须具备的地址，路由器的有所差异，多了以下地址：

所有路由器组播地址：FF01::2 ； FF02::2 ； FF05::2

子网路由器任意播地址：UNICAST_PREFIX :0:0:0:0

其他任意播配置地址：2000::/3

3.5 IPv6 地址配置方法

1. 手工配置

```
R1(config)# interface fast0/0
R1(config-if)# ipv6 enable
R1(config-if)# ipv6 address 2001:0001::/64 eui-64
```

上面这条命令的意思是使用 2001:0001::/64 作为前缀，并且追加 64bits 的 EUI-64 格式接口 ID，构成接口的全局唯一 IPv6 地址。

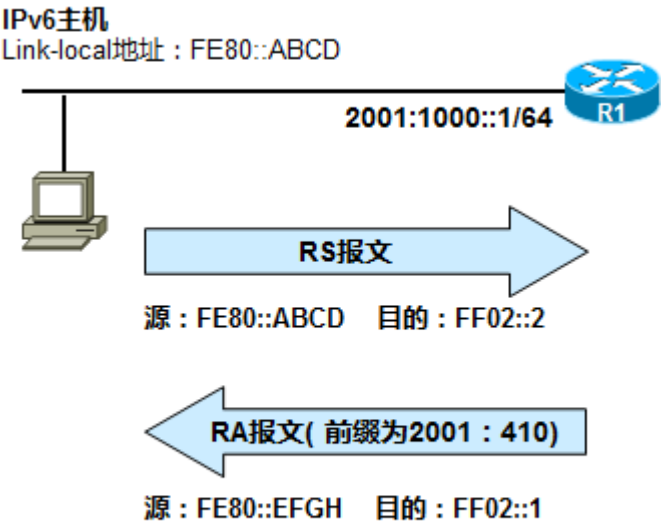
2. 自动配置

IPv6 地址的自动配置一般有两种方式：

- **有状态地址自动配置（DHCPv6）**
- **无状态地址自动配置**

  为了自动获得这个前缀，只要在路由器和主机之间运行一个协议即可。使用 NDP 协议的 Router Solicitation 恳求和 Router Advertisement 通告消息。前者用于发现路由器，并促使路由器发送 Router Advertisement 消息通报前缀信息。RA 消息中包含前缀、生存期、缺省网关等信息。大致过程如下图，更详细的内容见本文档相关章节。

1. 主机发送router Solicitation报文
2. 路由器回应Router Advertisement报文
3. 主机获得前缀及其它参数
4. 路由器周期性地向外发送RA报文



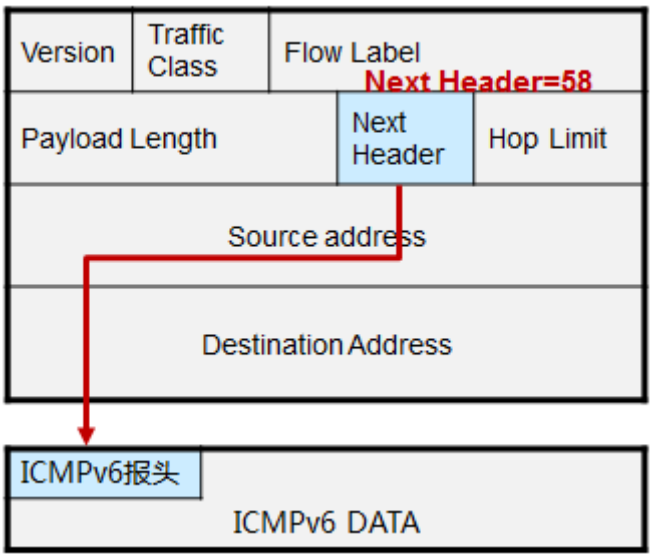
获取到的地址：2001:1000::ABCD

4 ICMPv6（RFC2463）

4.1 Foundation

ICMPv6 是 IPv6 的基础协议之一。协议号 58，该协议号在 IPv6 报头的“下一个包头”字段中。

ICMP 报文有两种：差错消息及信息消息



4.2 消息类型

消息类型	TYPE	名称	CODE
差错消息	1	目的不可达	0 无路由
			1 因管理原因禁止访问
			2 未指定
			3 地址不可达
			4 端口不可达
	2	数据包过长	0
	3	超时	0 跳数到 0
			1 分片重组超时
	4	参数错误	0 错误的包头字段
			1 无法识别的下一包头类型
			2 无法识别的 ipv6 选项
信息消息	128	Echo request	0
	129	Echo reply	0

还有一些其他报文，在下面介绍

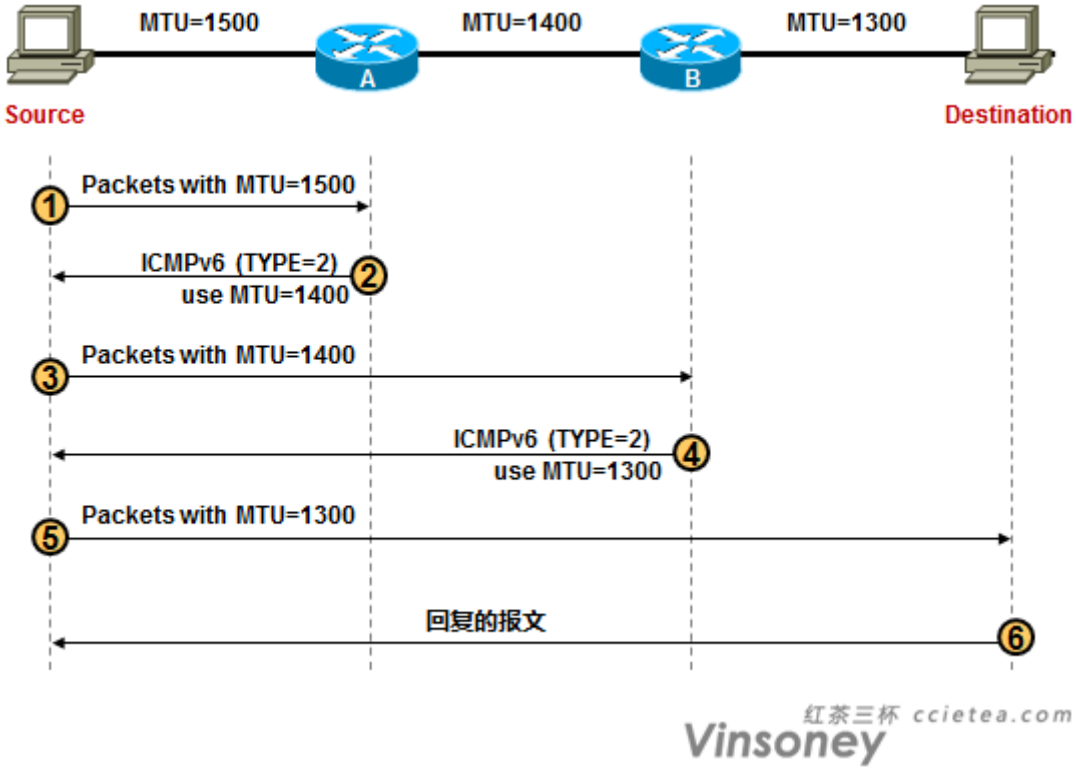
4.3 PMTUD

PMTU 就是路径上的最小接口 MTU。

PMTUD 的主要目的是发现路径上的 MTU，当数据包被从源转发到目的地的过程中避免分段。依赖 PMTUD 源节点可以使用所发现到的最大 PMTU 与目的地节点进行通信，这样可以避免数据包在从源传输到目的的过程之中，被中途的路由器分片，而导致性能的下降。

因此 IPv6 的分片不是在中间路由器上进行的，仅当路径 MTU 比欲传送的数据包小的时候，由源节点自己对数据包进行分片。

在 RFC1981 中定义了 PMTU 发现协议。IPv6 PMUTD 使用 ICMPv6 Type 2 消息。典型过程如下：

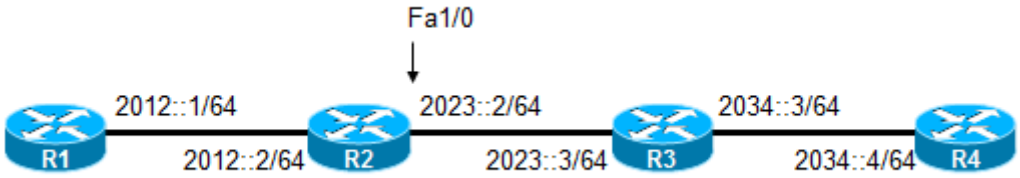


1. 首先 Source 用 1500 字节作为 MTU 向目标节点发送一个 IPv6 数据包
2. 中间路由器 A 意识到数据包过大，MTU 为 1400，于是回复一个 ICMPv6 type=2 消息向 Source 应答，在该 ICMPv6 消息中指定较小的 MTU=1400
3. Source 开始使用 MTU=1400 发送 IPv6 数据包，该数据包到了 B
4. 然而 B 意识到本地接口 MTU 为 1300，于是回复一个 ICMPv6 type=2 消息向 Source 应答
5. Source 开始使用 MTU=1300 发送 IPv6 数据包，该数据包顺利到达了目的地。
6. Source 和 Destination 之间的会话被建立起来。

注意：

- 这里的 PMTU，指的是单向的，沿路上的数据包出接口 MTU 中最小的。
- IPv6 要求的链路层所支持的 MTU 最小为 1280bytes
- 用 IPv6 PMTUD 发现的 MTU 值被源节点缓存，使用 CISCO IOS，可以用 show ipv6 mtu 来显示每个缓存的目的地 PMTUD 值

测试：



1. 准备工作

完成环境的搭建及设备的基本配置，保证全网互通。

R1 去 ping R4，能够 ping 通。

2. 实验测试

现在，将 R2 的 Fa1/0 口的接口 ipv6 MTU 修改为 1400 字节（上图中默认接口 IPv6 MTU 都是 1500 字节）：

```
Interface fast1/0
  Ipv6 mtu 1400
```

完成后，在 R2 上简单的 show 一下：

```
R2#show ipv6 interface f1/0
FastEthernet1/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE01:CFF:FEF0:10
  Global unicast address(es):
    2023::2, subnet is 2023::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::5
    FF02::6
    FF02::1:FF00:2
    FF02::1:FFF0:10
  MTU is 1400 bytes
.....
```

现在，我们再到 R1 上，

```
R1#ping 2034::4 repeat 1 size 1500          !! 让 R1 产生一个报文大小为 1500 的包，发给 R4
Type escape sequence to abort.
Sending 1, 1500-byte ICMP Echos to 2034::4, timeout is 2 seconds:
B
```

结果不通，因为报文在传递到 R2 时，超出了 R2 Fa1/0 口的 MTU，因此 R2 回送一个 TYPE=2（packet too big）的 ICMPv6 消息给 R1。在这个回送给 R1 的 ICMPv6 消息中，包含允许的 MTU=1400，以便告知 R1，后续发送的报文不要超过 1400 字节。这个过程抓包如下：

Time	Source	Destination	Protocol	Length	Info
62.942000	2012::1	2034::4	ICMPv6	1514	Echo (ping) request id=0x02
62.963000	2012::2	2012::1	ICMPv6	1294	Packet Too Big

其中，R2 回送给 R1 的 ICMPv6 差错消息如下：

```
Ethernet II, Src: cc:01:0c:f0:00:00 (cc:01:0c:f0:00:00), Dst: cc:00:0c:f0:00:
Internet Protocol Version 6, Src: 2012::2 (2012::2), Dst: 2012::1 (2012::1)
Internet Control Message Protocol v6
  Type: Packet Too Big (2)
  Code: 0
  Checksum: 0xbb6f [correct]
  MTU: 1400
⊞ Internet Protocol Version 6, Src: 2012::1 (2012::1), Dst: 2034::4 (2034::4)
⊞ Internet Control Message Protocol v6
```

此时，在 R1 上进一步查看一下：

```
R1#sh ipv6 mtu
MTU        Since       Source Address            Destination Address
1400       00:00:27    2012::1                   2034::4
```

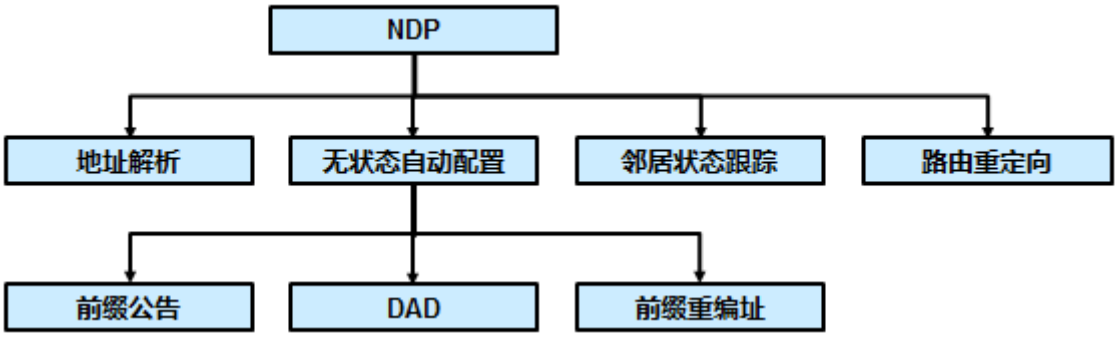
我们发现，R1 创建了一条缓存，标识从 2012::1 到 2034::4，PMTU=1400，因此 R1 在后续发送给 2034::4 的报文中，如果超出 1400 字节，R1 自己就会进行分片，例如我们再做测试，在 R1 上继续 ping 2034::4，报文大小为 1500 字节，则：

```
2012::1   2034::4   IPv6      1414 IPv6 fragment (nxt=ICMPv6 (0x3a) off=0 id=0x1)
2012::1   2034::4   ICMPv6     170 Echo (ping) request id=0x06c8, seq=0
2034::4   2012::1   ICMPv6    1514 Echo (ping) reply id=0x06c8, seq=0
```

上图中，前面两个包就是 R1 发送给 R4 的 ICMPv6 报文的两个分片，第三个报文是 R4 的 reply 包

所以这样 R1 发送大包给 R4 的时候，就不会有问题了。直接在 R1 源头这就分片。当然，我们可以继续试验，在 R3 的 Fa1/0 接口上将 IPv6 MTU 设置为 1300 字节，然后继续 R1 去 ping 大包给 R4，最终 R1 上的缓存条目，PMTU 会变成 1300 字节。

5 NDP 邻居发现协议（RFC2461）



NDP 可以帮助我们实现以下功能：

1. **地址解析(代替 ARP 协议，使用 ICMP 完成地址解析)**

   IPV6 取消了 arp 使用 NS 和 NA 来做，利用的是 solicited address

   通过邻居请求（NS）和邻居通告（NA）报文来解析三层地址对应的链路层地址。

2. **邻居的状态跟踪**

   Show ipv6 neighbor

3. **无状态自动配置**
4. **重复地址检测（DAD）**
5. **前缀重编址**
6. **路由器重定向**

   路由器向一个 Ipv6 节点发送 icmpv6 消息，通知它在相同的本地链路上才能在一个更好的到达目的地望楼的路由器地址

5.1 为 NDP 定义的 icmpv6 消息

ICMPv6 TYPE	消息名称
133	路由器请求（RS）
134	路由器通告（RA）
135	邻居请求（NS）
136	邻居通告（NA）
137	重定向消息

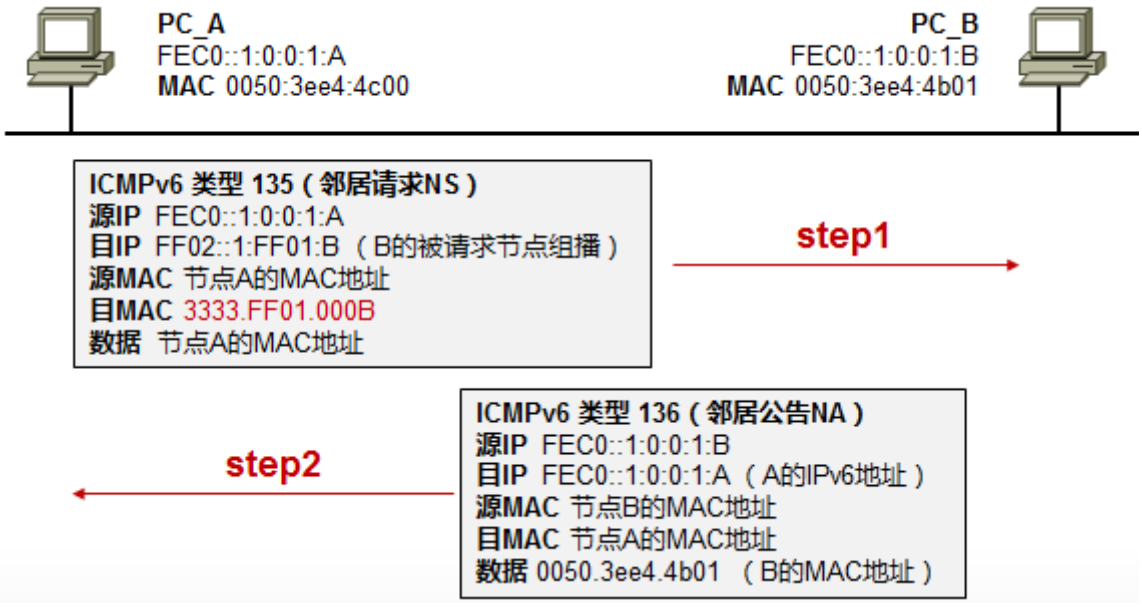
NDP 机制使用的 ICMPv6 消息：

机制	RS 133	RA 134	NS 135	NA 136	重定向消息 137
报文介绍	主机可以发送 RS 要求路由器立即产生 RA	包含 MTU、前缀信息等	用来判断邻居的链路层地址，也用于 DAD 等		
替代 ARP			X	X	
前缀公告	X	X			
前缀重新编制	X	X			
DAD			X		
路由重定向					X

RS 133 指的是使用 ICMPv6 TYPE133 Code0 的 RS 报文，其他报文类似。

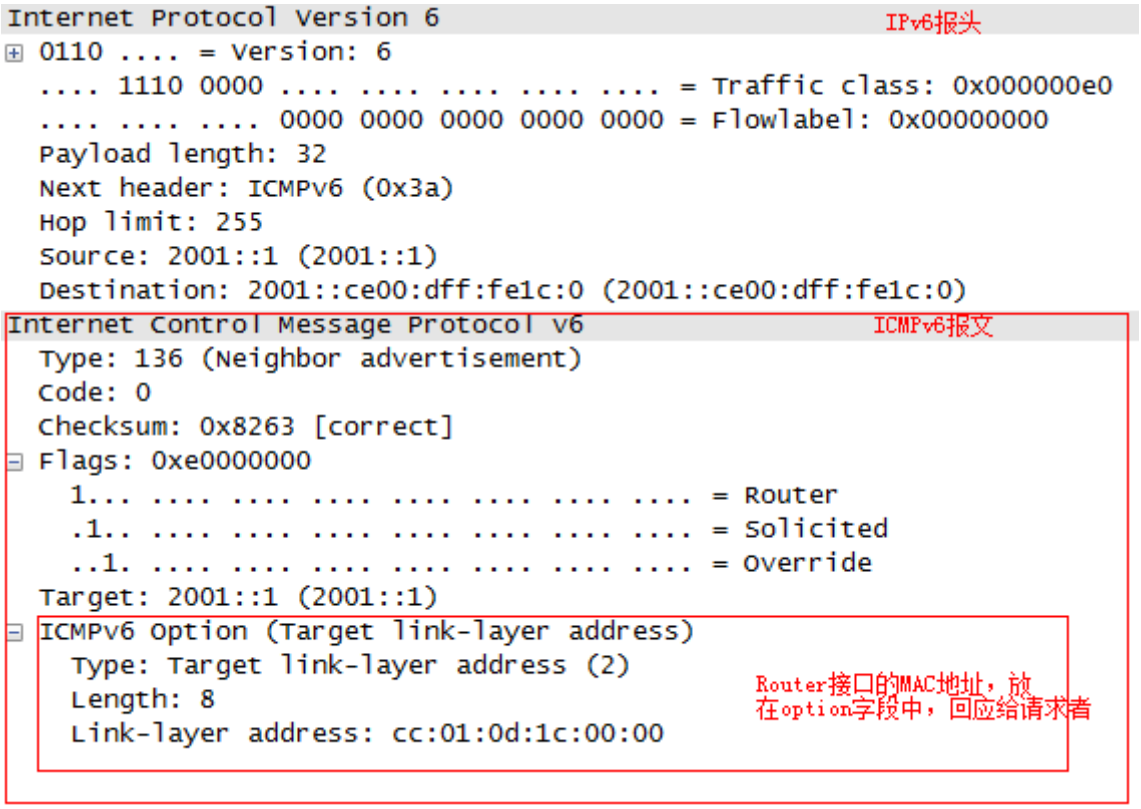
5.2 地址解析

在 IPv6 中，对节点链路层地址的确定，使用 NS、NA 和被请求节点组播地址的组合来完成。这比 IPv4 的 ARP 要高效得多。我们来看一下典型的地址解析过程：



其中 33:33:FF:01:00:0B 是 ipv6 目的地址 FF02::1:FF01:B 的多播映射。

以下是一个 NA 报文的示例：



注意到其中的 Flag 字段，

R 位置 1 表示这是台路由器，

S 位置 1 表示这是响应某个邻居请求的通告，置 0 则为主动发送的，

O 位置 1 表示 NA 消息中的条目是否覆盖已有，收到该消息的邻居会覆盖已有条目

- **使用 show ipv6 neighbors 可以查看邻居表项**

R1#show ipv6 neighbors

IPv6 Address	Age	Link-layer Addr	State	Interface
2012::2	11	cc01.0cf0.0000	STALE	Fa0/0
FE80::CE01:CFF:FEF0:0	11	cc01.0cf0.0000	STALE	Fa0/0

其中 State 如果为 REACH 表示邻居可达；为 STALE 意味着这些邻居在最后的 30s 内是不可达的

- **使用“ipv6 neighbor ipv6 地址 接口编号 mac 地址”，可添加一个静态表项到邻居发现表，例如：**

Router(config)# ipv6 neighbor 2012::2 fastEthernet 0/0 cc01.0cf0.0000

R1# show ipv6 neighbors

IPv6 Address	Age	Link-layer Addr	State	Interface
2012::2	-	cc01.0cf0.0000	REACH	Fa0/0

- **使用“clear ipv6 neighbors”，会清楚邻居发现表中的动态表项**

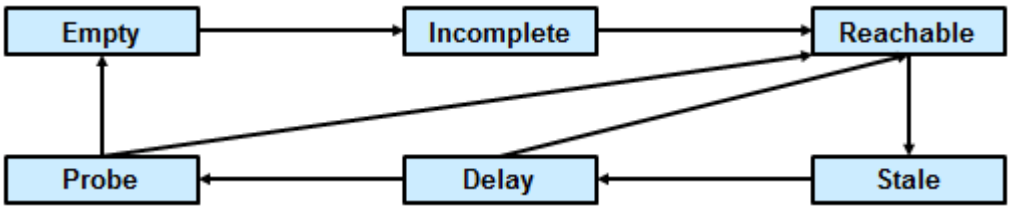
- **调整 NDP 消息的参数**

  Ipv6 nd ns-interval !! 用于设置邻居请求消息的时间间隔，默认是 1000ms 也就是 1S

  Ipv6 nd reachable-time !! 一个邻居在由某个事件证实它的可达性后，在这段时间内这个邻居被认为是可达的，默认 30S（在 CISCO IOS 上 show 出来的，书上说的是 30min，估计是写错了）

5.3 邻居状态

Incomplete	邻居请求已经发送到目标节点的请求组播地址，但没有收到邻居的通告
Reachable	可达，收到确认，不续再发包确认
Stale	从收到上一次可达性确认后过了超过 30s。
Delay	在 stale 状态后发送过一个报文，并且 5s 内没有可达性确认
Probe	每隔 1s 重传邻居请求来主动请求可达性确认，直到收到确认



1. A 发送 NS，并生成缓存条目，A 上条目的状态为 Incomplete
2. 若 B 回复 NA，则 A 上关于 B 的邻居状态就由 Incomplete ->Reachable。但若 A 发出 NS 消息后一定时间内仍没有收到任何的回复，则由 Incomplete->Empty，即删除条目
3. 如果在 reachable 状态上经过 ReachableTime（默认 30S），A 路由器上关于 B 的条目状态 Reachable->stale 或如果在 reachable 状态上，收到 B 的非请求 NA，且链路层地址不同，则马上->stale
4. 在 Stale 状态若 A 要向 B 发送数据，可直接发送，并从 A 上关于 B 的条目由 Stale->Delay，同时会等待应用层的提示信息，提示邻居是否可达
5. 如果在 Delay_First_Probe_Time（默认 5S）内，有 NA 应答或者应用层的提示信息（例如我发了 ICMP 包给对端，对端回复我 ICMP 了，那就是上层可达），则 Delay->Reachable，如果无应用层提示信息，Delay->Probe
6. 在 Probe 状态，每隔 RetransTimer（默认 1S）发送单播 NS，发送 MAX_Unicast_Solicit 个后再等 RetransTimer，有应答则 Reachable，无则进入 Empty，即删除条目

测试的时候可开 debug ipv6 nd

5.4 无状态自动配置

涉及机制

- 前缀公告
- DAD
- 前缀重新编址

5.4.1 路由器公告 RA

```
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x4a68 [correct]
  Cur hop limit: 64
⊟ Flags: 0x00
    0... .... = Not managed
    .0.. .... = Not other
    ..0. .... = Not Home Agent
    ...0 0... = Router preference: Medium
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊟ ICMPv6 Option (Source link-layer address)
    Type: Source link-layer address (1)
    Length: 8
    Link-layer address: cc:01:0d:1c:00:00
⊟ ICMPv6 Option (MTU)
    Type: MTU (5)
    Length: 8
    MTU: 1500
⊟ ICMPv6 Option (Prefix information)
    Type: Prefix information (3)
    Length: 32
    Prefix length: 64
  ⊟ Flags: 0xc0
      1... .... = Onlink
      .1.. .... = Auto
      ..0. .... = Not router address
      ...0 .... = Not site prefix
    Valid lifetime: 2592000
    Preferred lifetime: 604800
    Prefix: 2001::
```

- Cur Hop Limit： 表示当 PC 使用该 RA 通告的前缀构建 IPv6 地址，那么该 PC 发送的 IPv6 报文 hoplimit 都是该值（64）
- Flags 如果 not managed 位=1 表示主机使用有状态地址自动配置（如 DHCPv6）；not other 位为 1 则表示主机使用有状态地址自动配置来获取除了地址以外的参数
- RouterLifetime 该 RA 关联的前缀发给 PC 后，PC 将 Router 视为网关，这个 lifetime 就是这个意思
- Reachable time 及 retrans timer 如果为 0 则表示为指定

PC 收到一个 RA 报文，会做一个检测：

1. RA 数据包的源地址，是否为一个 linklocal 地址
2. RA 数据包的 IPv6 报头的 hop limit 是否为 255。因为只有 255 才表示是本地网络中路由器产生的
3. 如果 RA 数据包包含认证信息，那么认证是否正确
4. ICMPv6 的 checksum 是否正确
5. ICMPv6 code 是否为 0
6. ICMPv6 长度大于等于 16
7. 所有的 options 长度大于等于 0

当然，如果一台 PC 初始化，也可以主动发送一个 RS 请求关于前缀信息

5.4.2 前缀公告

1. **前缀公告概述**

   前缀公告是无状态自动配置的初始机制。利用 RA（ICMPv6 Type134）和所有节点的组播地址（FF02::1），路由器 RA 消息在本地链路上周期性的发送到 FF02::1 上。

   在无状态自动配置中，前缀长度为 64 比特

   Show ipv6 interface xxx perfix !!显示接口上公告的前缀参数

2. **在 CISCO IOS 路由器上公告 IPv6 前缀**

   只要在网络接口上配置了一个本地站点或者全局可聚合单播 IPv6 地址及其前缀长度，就启用了 CISCO 路由器上的 IPv6 前缀公告。

   以下是自动配置过程期间和之后使用的参数：

   - **IPv6 前缀**

     默认情况下，无状态自动配置公告的前缀长度为 64 位，节点收到 IPv6 前缀，将自己的 EUI-64 地址附加在这 64 位的前缀后面，构成 128 位的 IPv6 地址。
   - **生存期**
   - **有效生存期（Valid lifetime）**
   - **首选生存期（preferred lifetime）**

   【前缀的两个时间】

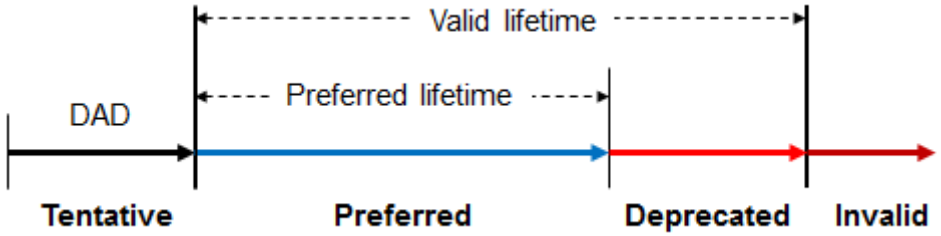
   IPv6 地址可以通过 state configuration 和 stateless configuration 获得到，简单的说

   状态化配置是通过手工添加或是通过 dhcpv6 获得，无状态化是通过 prefix advertisement 获得。不管哪种方式获得的地址，都会有几个时间状态。

其实 IPv6 地址的 lifetime 主要有两个，一个是 **Valid lifetime** ，另一个是 **preferred lifetime**。首先在 DAD 检查阶段。在 DAD success 前，地址一直处于 tentative (试验状态)状态，这时的地址是不可用状态，直到 DAD success。然后进入 Preferred 状态，也就是首选状态，这个状态过程中的 IPv6 address 的有效性是由 preferred lifetime 决定的。

Valid lifetime 是一个类似于 dhcp 中的 lease time。如果地址超过了 valid lifetime，节点就会把这个地址置为 inactive 状态，然后 delete。在 preferred time 和 valid lifetime 之间叫做 deprecated 状态，这种状态算是一种 buffer，大概意思叫做不赞成使用的状态，当地址达到这个时间段的时候，地址不能主动的发起连接只能是被动的接受连接，这也是为了保证上层应用而设计的，但是过了 valid lifetime 时间地址就变为 invalid，这时任何连接就会 down 掉。

在 CISCO IOS 上，默认 valid time 为 30 天（2592 000s），prefer time 为 7 天（604 800s）。



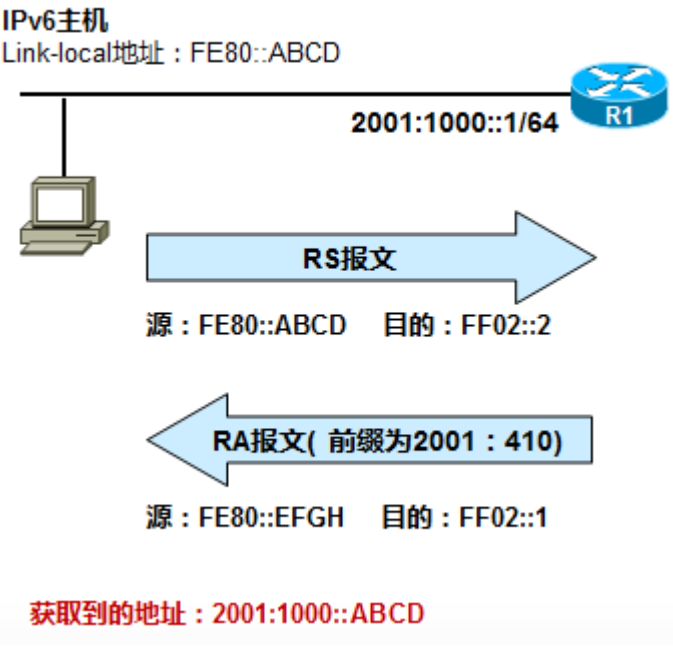
- **默认路由器信息**

  提供关于默认路由器 IPv6 地址的存在和生存期信息。在 IPv6 中，节点使用的默认路由器地址是路由器的本地链路地址（linklocal address），这样一来，即使前缀被重新编址，默认网关也是可达的（linklocal address 不会变）

- **标志/选项**

3. 前缀公告是如何工作的

1. 主机发送router Solicitation报文
2. 路由器回应Router Advertisement报文
3. 主机获得前缀及其它参数
4. 路由器周期性地向外发送RA报文



4. 调整前缀公告的参数

```
R1(config-if)#ipv6 nd prefix 2001::/64 ?
  <0-4294967295>   Valid Lifetime (secs)
  at               Expire prefix at a specific time/date
  infinite         Infinite Valid Lifetime
  no-advertise     Do not advertise prefix
  no-autoconfig    Do not use prefix for autoconfiguration
                   // 当在特定前缀后开启该参数后，该前缀不能用于无状态自动配置
  no-rtr-address   Do not send full router address in prefix advert
  off-link         Do not use prefix for onlink determination
```

举例：

```
Ipv6 nd prefix 2012::/64 30 15
!! 配置 validtime 及 prefered time，这是一个相对时间
```

配置该命令后，邻居 R2 收到 prefix，显示如下：

```
R2#sh ipv int f 0/0
FastEthernet0/0 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::CE00:DFF:FE48:0
Global unicast address(es):
```

2001::CE00:DFF:FE48:0, subnet is 2001::/64 [PRE]　　!! 状态为 prefer

valid lifetime 25 preferred lifetime 10　　!! 两个时间在不断递减

当 preferred lifetime 先到 0 时，状态变为[DEP]，当 valid time 变 0 时，地址抹去

Ipv6 nd prefix 2012::/64 at xxx yyyy

!! xxx 为 validtime，yyy 为 prefered time，这是绝对时间

Ipv6 nd prefix 2012::/64 890000 720000 off-link

!! off-link 和 L 比特有关，这个比特位在 RFC2461 中定义。当可选的 off-link 关键字在 CISCO IOS 中被配置时，L 比特被关闭。而如果 L 比特被打开（默认打开），他表示在 RA 消息中的前缀是分配给本地链路的。因此，向包含这个指定前缀的地址发送数据的节点认为目的地是本地链路可达的。

Ipv6 nd prefix 2012::/64 890000 720000 no-autoconfig

!! no-autoconfig 和 A 比特有关，RFC2461 中定义。A 比特也成为自治地址配置标志。当这个可选关键字 no-autoconfig 在 CISCO IOS 中被配置（就是配了上面的命令），A 比特被置 0。如果 A 比特被置 1（默认就是 1），它指示本地链路的主机可以使用该前缀进行无状态自动配置。

ipv nd prefix 2012::/64 no-advertise

该前缀将不包含在 RA 消息中。默认情况下，CISCO IOS 关闭这个 no-advertise，也就是前缀都会被包含在 RA 消息中。

ipv nd prefix default ?

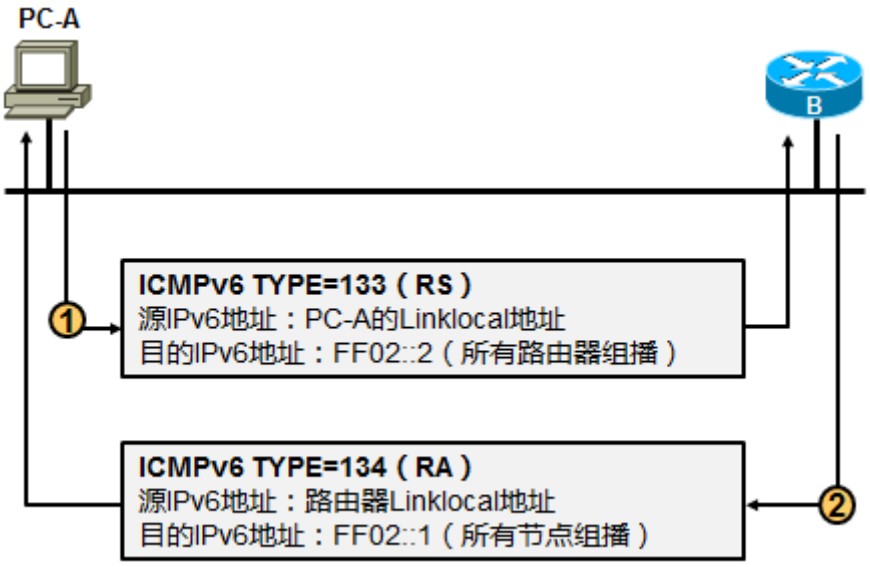
可以修改接口下，nd 前缀通告的默认参数

Ipv6 nd suppress-ra

在接口上关闭路由器公告。默认情况下，当全局命令 ipv6 unicast-routing 被启用，CISCO 设备的以太网口、FDDI 和令牌环接口上具有 RA 功能

5. **使用 RS 来请求 RA**

PC 可以主动发送 RS 来触发路由器发送 RA，而不用等 RA 的周期性发送时间到了才收 RA。



为了避免路由器请求消息在本地链路上泛滥，在启动时每个节点只能发送 3 个 RS 消息。

5.4.3 调整 IPv6 nd 参数

- **Ipv6 nd ra-lifetime**

  RA 消息的生存时间，默认 1800s

- **Ipv6 nd ra-interval**

  RA 消息的通告间隔，这个时间小于或等于 ra-lifetime。默认 200S。

- **ipv6 nd managed-config-flag**

  配置这条命令后，该接口上的前缀信息将不能被链路上的主机用于无状态自动配置，主机必须使用有状态自动配置例如 DHCPv6 来获取地址。默认情况下这命令没配，也就是说，managed-config-flag 为 0，也就是说，主机可以通过无状态配置获取地址。

```
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x4a68 [correct]
  Cur hop limit: 64
⊟ Flags: 0x00
    0... .... = Not managed
    .0.. .... = Not other
    ..0. .... = Not Home Agent
    ...0 0... = Router preference: Medium
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊞ ICMPv6 Option (Source link-layer address)
⊞ ICMPv6 Option (MTU)
⊞ ICMPv6 Option (Prefix information)
```

- **ipv nd other-config-flag**

  Hosts should use DHCP for non-address config

  如果配置了该命令，则主机需使用 dhcp 配置除了 ipv6 地址外的其他信息，如 DNS，域名什么的

  这个标志也与有状态自动配置有关，当它没有置位（默认情况下），节点不应该使用有状态自动配置机制来配置除了 IPv6 地址以外的其他参数。

5.4.4 DAD（Duplicate Address Detection）

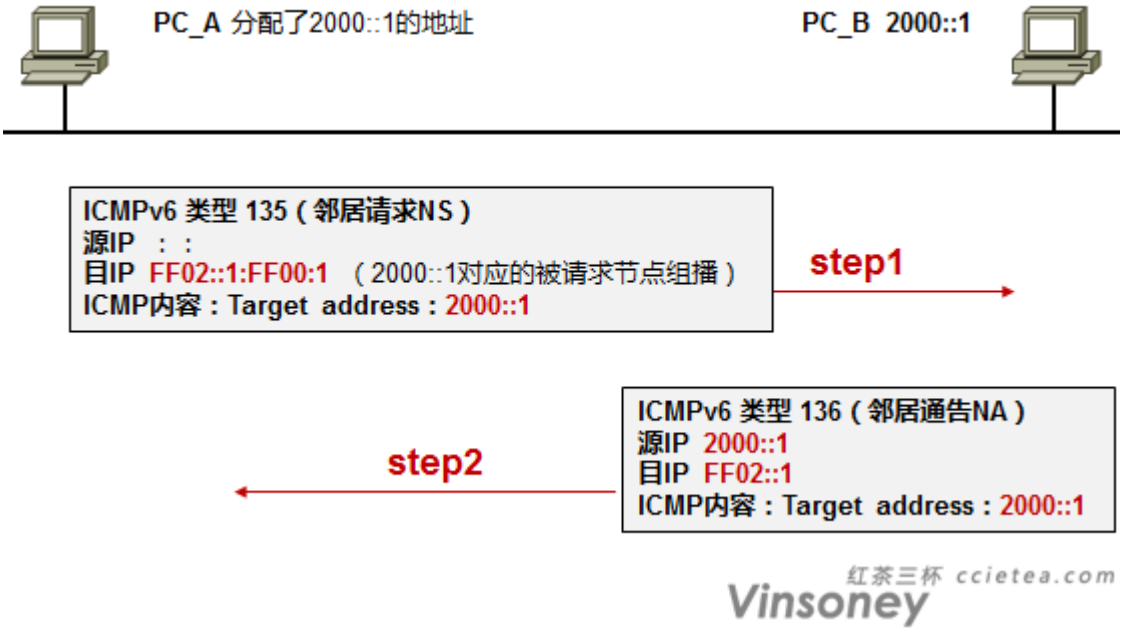
1. **机制概述**

   无状态配置，和节点启动时的一个 NDP 机制。用于保证节点准备启用的 IPv6 单播地址在链路上的唯一性。这个机制使用 NS 消息（ICMP 135），使用源地址（ ::）、目的地址为获取到的 v6 地址对应的被请求节点组播地址的 NS 报文来完成这个任务。

2. **机制原理**

   一个地址在通过重复地址检测（DAD）之前称为“tentative 地址”，试验地址。接口还暂时不能使用这个试验地址进行正常单播通讯，但是会加入和 tentative 地址所对应的 Solicited-Node 组播组。

   重复地址检测 :节点向一个自己将使用的 tentative 地址所对应的 Solicited-Node 组播地址发送一个 NS，如果收到某个其他站点回应的 NA，就证明该地址已被网络上使用，节点将不能使用该 tentative 地址通讯。



如果 1S 后没有检测到冲突，A 就会发送 non-solicited advertisement（一个 NA 消息），宣告大家我将正式使用这个 IPv6 地址。

3. 调整 DAD

- **Ipv6 nd dad attempts x**

  默认情况下，CISCO 路由器启用 DAD，在确定一个地址的唯一性之前，在本地链路上发送 NS 消息的个数为 1。使用上述命令，能修改 NS 消息的个数。为 0 则关闭 DAD。

5.4.5 前缀重新编址

前缀重新编址允许从以前的网络平稳过渡到新的前缀，站点内节点使用无状态自动配置（或者其他重编址方法，但是不如无状态自动配置的方法透明），这样可以使得在前缀重新编址过程对站点内的节点完成透明，也就是说，站内节点完全“不知情“或”无感知 “，部署的动作在路由器上完成，切换过程平滑。

路由器接口配置新、老两个前缀，并且都进行公告，老前缀的生存期较短，这样站内的节点可以同时使用两个地址，当老前缀失效后，仅剩下新的前缀被使用，即可实现切换。

首选 ,站点中所有路由器继续公告当前的前缀(老前缀),但是有效和首选生存期被减小到接近于 0 的一个值，然后，路由器开始在本地链路公告新的前缀，因此每个本地链路上至少有两个前缀存在。当旧的前缀完全被废止时（生存期已过），路由器公告消息仅包括新前缀。

配置前缀重新编制

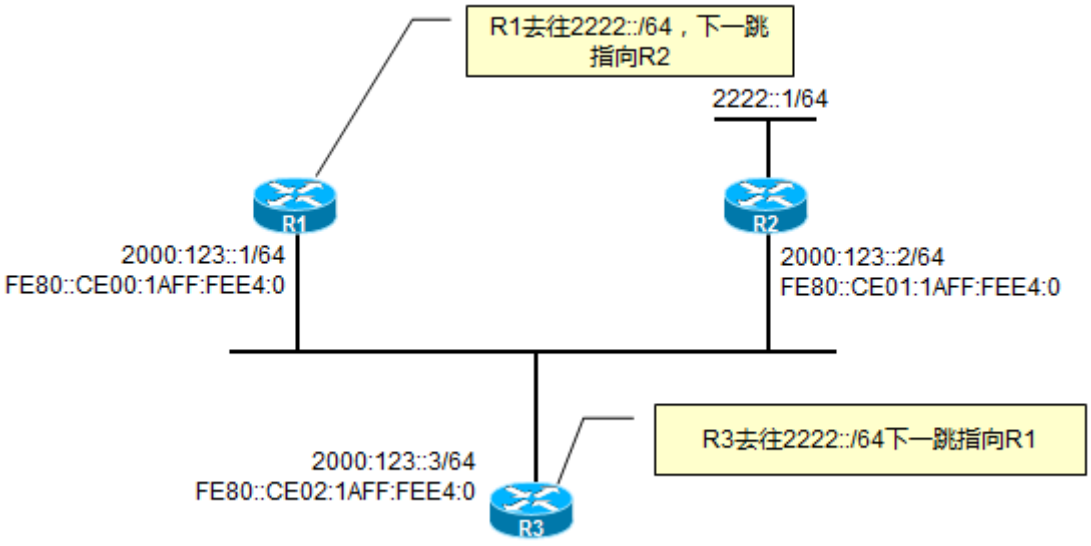
```
Ipv6 nd prefix 2001:0001::/64  43200  0                    // 老前缀
Ipv6 nd prefix 2001:0002::/64  43200  43200                // 新前缀
或者
Ipv6 nd prefix 2001:0001::/64  at Jul 31 2012 23:59  Jul 20 2012 23:59    // 老前缀
Ipv6 nd prefix 2001:0002::/64  43200  43200                // 新前缀
```

5.4.6 路由器重定向

路由器使用 ICMPv6 重定向消息（ICMP TYPE 137）通知链路上的节点，在链路上存在一个更好的前转数据包的路由器。接收到这个 ICMPv6 重定向消息的节点可以根据重定向消息中新的路由器地址修改它的本地路由选择表。基本上从机制上来将跟 IPv4 的 ICMP 重定向没啥两样。

在 IPv6 规范中，**不推荐使用可聚合全球单播或本地站点地址作为下一跳地址，如果这样做，ICMPv6 重定向消息就不会工作**。因此使用 Linklocal 地址作为下一跳，在某些场合可能更为推荐，毕竟 linklocal 地址稳定且长久不变。在配置 linklocal 地址作为下一跳 IP 时，必须关联路由器上相应的接口。

接下去做一个简单的测试：



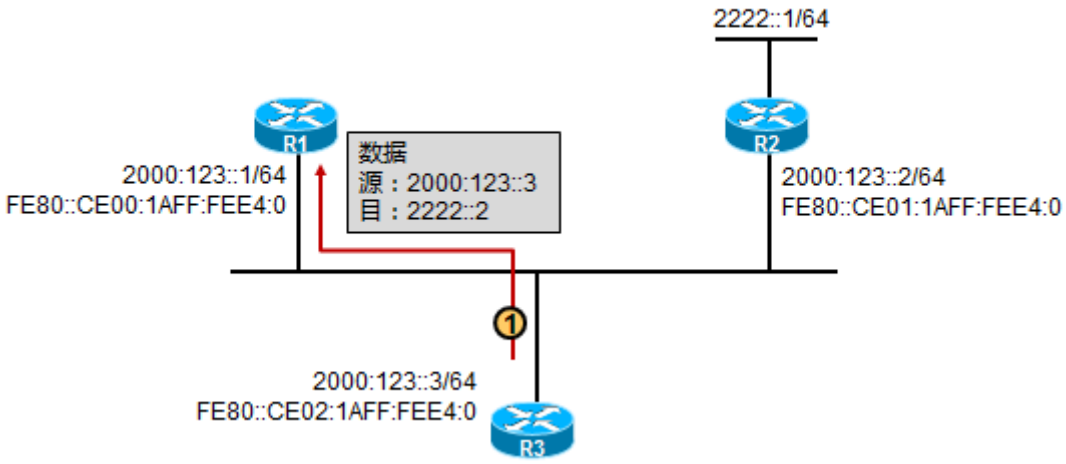
实验环境如上，配置如下：

R3 的关键配置：

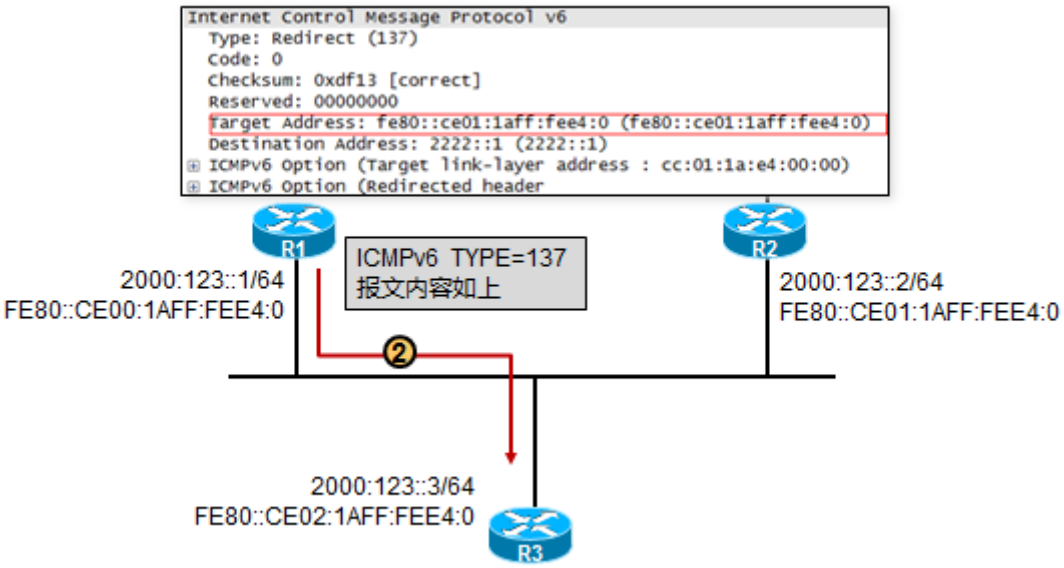
```
ipv6 route 2222::/64 FastEthernet0/0 FE80::CE00:1AFF:FEE4:0
```

R1 的关键配置：

```
ipv6 route 2222::/64 FastEthernet0/0 FE80::CE01:1AFF:FEE4:0
```



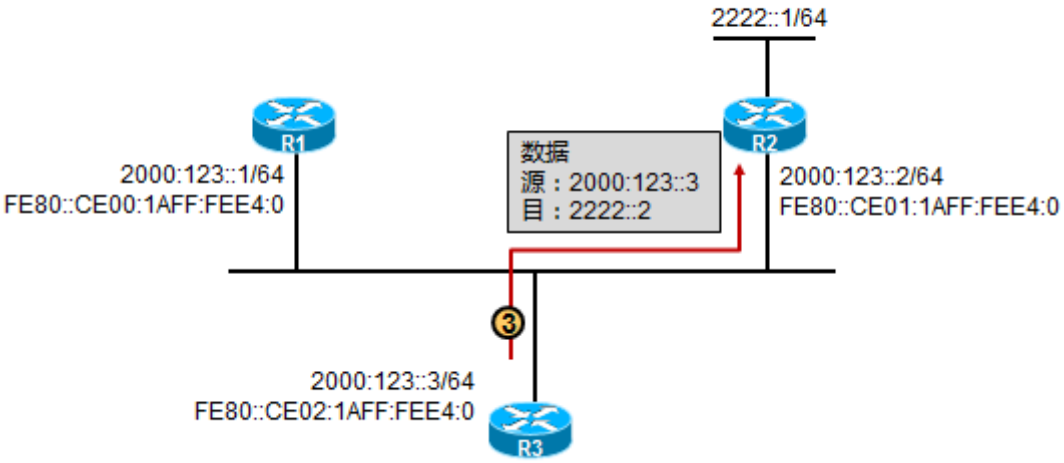
接下去，我们在 R3 上去 ping 2222::2，这个动作将使得 R3 发送一个 ICMPv6 报文，源为 2000:123::3，目的为 2222::2，报文将被发送到 R1，当 R1 接收后，他发现，这个数据包将从自己收到该报文的接口再转发出去，而下一跳 IP 与该接口在同一个网段，证明这个下一跳比自己更优（距离目的地更近）。



因此，R1 将发送一个 ICMPv6 Type=137 的重定向报文给 R3，以便告知 R3，要去往目的地 2222::3，有人比我更优，而这个 ICMPv6 消息中，包含的关键信息，就是 R2 的 Linklocal address。报文如下：

```
Internet Protocol Version 6, Src: fe80::ce00:1aff:fee4:0 (fe80::ce00:1aff:f
Internet Control Message Protocol v6
  Type: Redirect (137)
  Code: 0
  Checksum: 0xdf13 [correct]
  Reserved: 00000000
  Target Address: fe80::ce01:1aff:fee4:0 (fe80::ce01:1aff:fee4:0)
  Destination Address: 2222::1 (2222::1)
⊟ ICMPv6 Option (Target link-layer address : cc:01:1a:e4:00:00)
    Type: Target link-layer address (2)
    Length: 1 (8 bytes)
    Link-layer address: cc:01:1a:e4:00:00 (cc:01:1a:e4:00:00)
⊞ ICMPv6 Option (Redirected header
```

上面就是R1发送给R3的ICMPv6的type137报文 ,我们关键看target address ,这个就是R2的linklocal address，也就是 R1 想告知给 R3 的、距离目标 2222::1 比自己更近的下一跳。



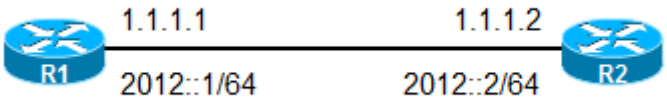
R3 收到这个 ICMPv6 重定向消息后，后续的数据包将直接发给 R2，就不用再绕过 R1 了。

6 IPv6 应用

6.1 域名系统

IPv4 中 A 记录用来将主机名称映射到一个 V4 地址，AAAA 资源记录将主机名映射到一个 IPv6 地址。

我们可以做一个测试：



R1 及 R2 为双栈路由器，R1 的配置如下（省去接口 IP 的配置）：

```
ipv6 unicast-routing
ip name-server 1.1.1.2
ip name-server 2012::2
ip domain-lookup
```

现在做一个测试，在 R1 上 ping www.ccietea.com，我们会发现：

Source	Destination	Protocol	Length	Info
1.1.1.1	1.1.1.2	DNS	72	Standard query AAAA www.sina.com
2012::1	2012::2	DNS	92	Standard query AAAA www.sina.com
2012::2	2012::1	ICMPv6	140	Destination Unreachable (Port unreachable)
1.1.1.1	1.1.1.2	DNS	72	Standard query A www.sina.com
1.1.1.1	1.1.1.2	DNS	72	Standard query A www.sina.com
2012::1	2012::2	DNS	92	Standard query A www.sina.com

R1 首先会通过 IPv4 网络去查找 AAA 记录，随后又会通过 IPv6 网络去 AAAA 记录，发现都没有回应，那么又用 IPv4 网络去查找 A 记录，没回应，又用 IPv6 网络查找了一次 A 记录。

因此对于双栈的情况下，PC DNS 解析的时候，一般会先请求 AAAA 记录。

在 CISCO IOS 路由器上，使用如下的方式配置一条主机名到 IPv6 地址的映射：

```
Ipv6 host xxx 2001::1/64
```

6.2 ACL

Ipv6 access-list x	进入 acl 配置模式
Ipv6 traffic-filter	在接口下应用

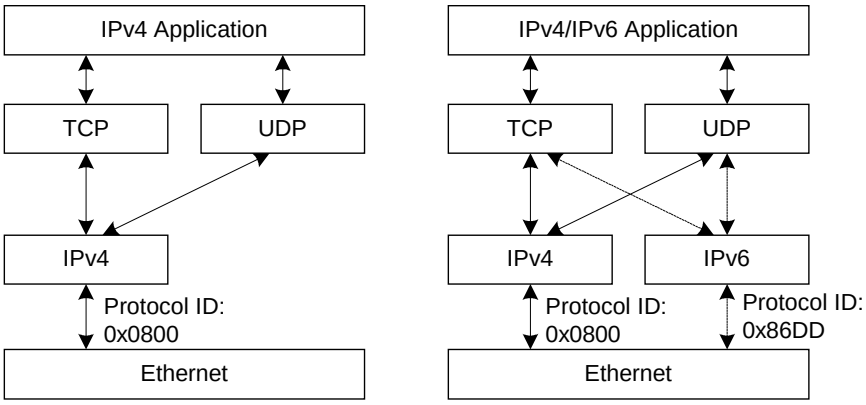
7 IPv6 过渡技术

7.1 Foundation

由于现今无论是广域网如 Internet，还是全球范围内的各种 LAN，都是以 IPv4 居多。要切换到 IPv6 不可能一气呵成，这是一个综合政治、经济、政策、方法、环境等等各种因素的大问题。IPv4 切换到 V6 需要一个相当长的时间。那么在这个过程中，就势必出现 V4 和 V6 共存的网络环境。接下去就研究研究 V4 到 V6 的平稳过渡技术。

1. 双栈（dual stack）

网络中的主机、路由器或服务器等设备如果支持双栈（IPv4 及 IPv6 协议栈），那么可以同时使用 V4 和 V6 的协议栈。在双栈设备上，上层应用会优先选择 IPv6 协议栈，而不是 IPv4。 比如，一个同时支持 v4 和 v6 的应用请求地址，会先请求 AAAA 记录，如果没有，则在请求 A 记录。



2. 隧道 tunnel

用于在现有网络（v4）中传输不兼容的协议（v6）或者特殊的数据，

- 手工隧道技术：手工 IPv6 over IP 隧道、GRE 隧道
- 自动隧道技术：6to4 隧道、IPv4 兼容 IPv6 自动隧道、ISATAP 隧道
- 6PE：6PE 技术依赖于 BGP，BGP 的 Peer 是需要手工指定的，可以算是一种半自动隧道技术。

3. v6 v4 协议转换

NAT-PT（Network Address Translation-Protocol Translation）

7.2 双栈（Dual Stack）

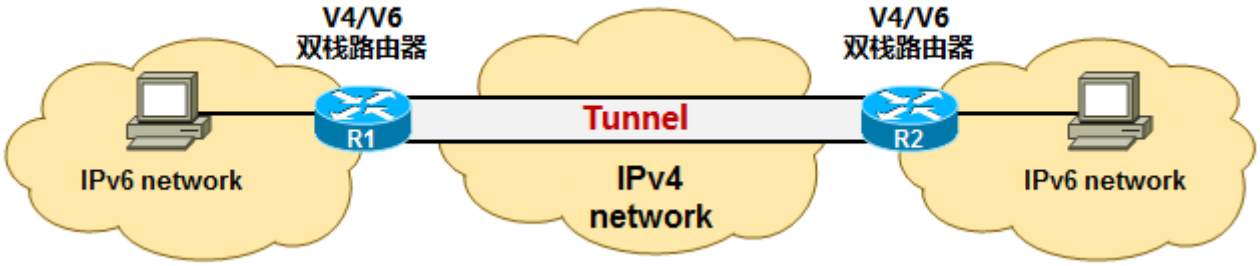
- 节点有 IPv4 及 IPv6 两个协议栈
- 缺点：每台设备都需要配置两种协议，需要占用资源，设备需要存储两个路由表（如果是路由器），需要独立处理每种协议。

```
router(config)#  ipv6 unicast-routing
router(config)#  interface fast0/0
router(config-if)#  ipv6 enable
router(config-if)#  ip address 192.168.1.254 255.255.255.0
router(config-if)#  ipv6 address 2001::0001::FFFE/64
```

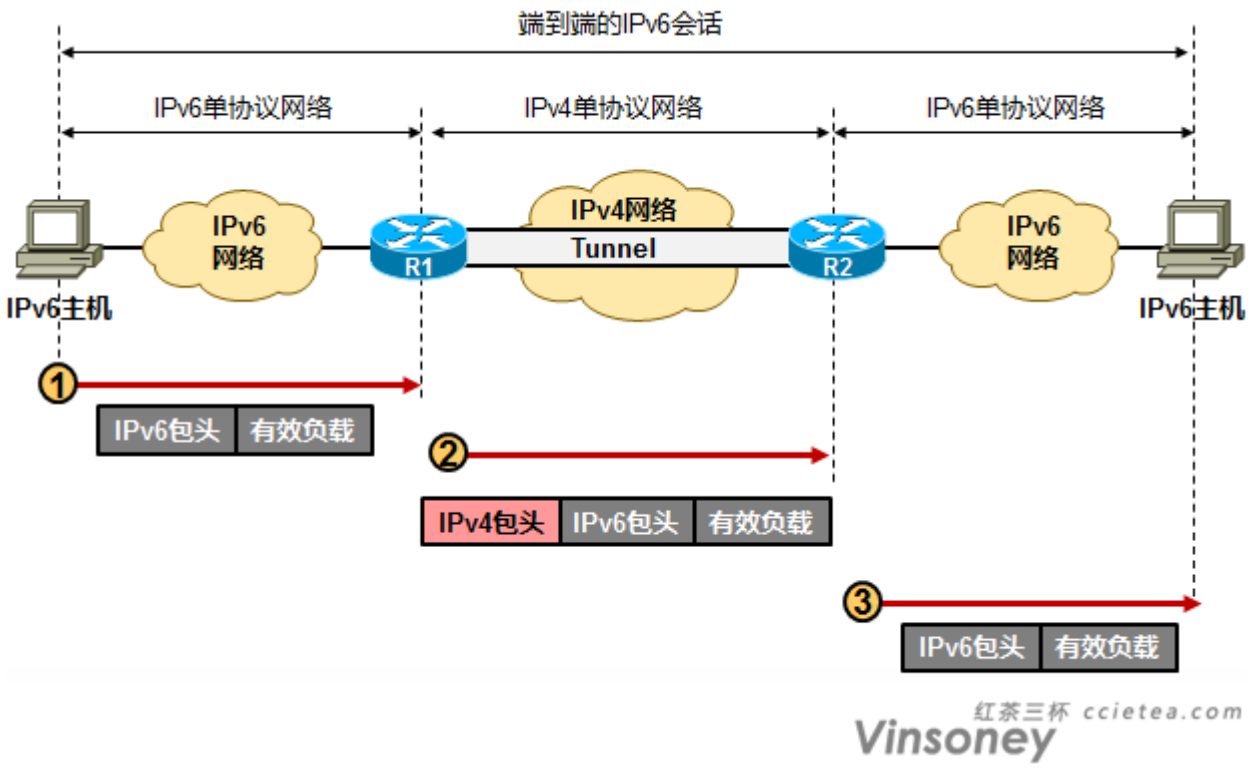
7.3 隧道（Tunnel）

7.3.1 隧道机制概述

隧道技术一般用于在现有网络中传输不兼容的协议或特殊的数据。在因特网上无处不在的现有 IPv4 基础设施中，使用隧道技术可以使得 IPv6 孤岛得以连接。当然，相对于任何过渡和并存的策略（如隧道技术），我们还是首选由纯 IPv6 连接组成的网络、链路和基础设施。实际上，只有当在网络、连接和基础设施中不可能获得纯 IPv6 连接性时，IPv4 基础设施上的 IPv6 隧道机制才被认为是一种可选择的方法。



接下来简单的了解一下最基本的隧道技术来实现在 IPv4 网络中连接 IPv6 孤岛：



- 左边的 IPv6 主机欲与右边的 IPv6 主机通信，发出原始的 IPv6 数据包
- 这些数据包到达 R1 后，R1 路由器支持 IPv4 及 IPv6 也就是双栈路由器，R1 与 R2 建立了一个隧道以连接分别挂在 R1、R2 两端的两个 IPv6 孤岛。PC 发出来的原始 IPv6 报文被 R1 添加了一个 IPv4 的隧道包头，形成了一个“外表看似 IPv4 报文的”报文。此后，这个报文在 IPv4 网络中被路由，最终到达 R2。R2 收到这个“隧道报文”后，将报文的 IPv4 头部去除，然后还原成原始的 IPv6 报文，再转发给 IPv6 网络。

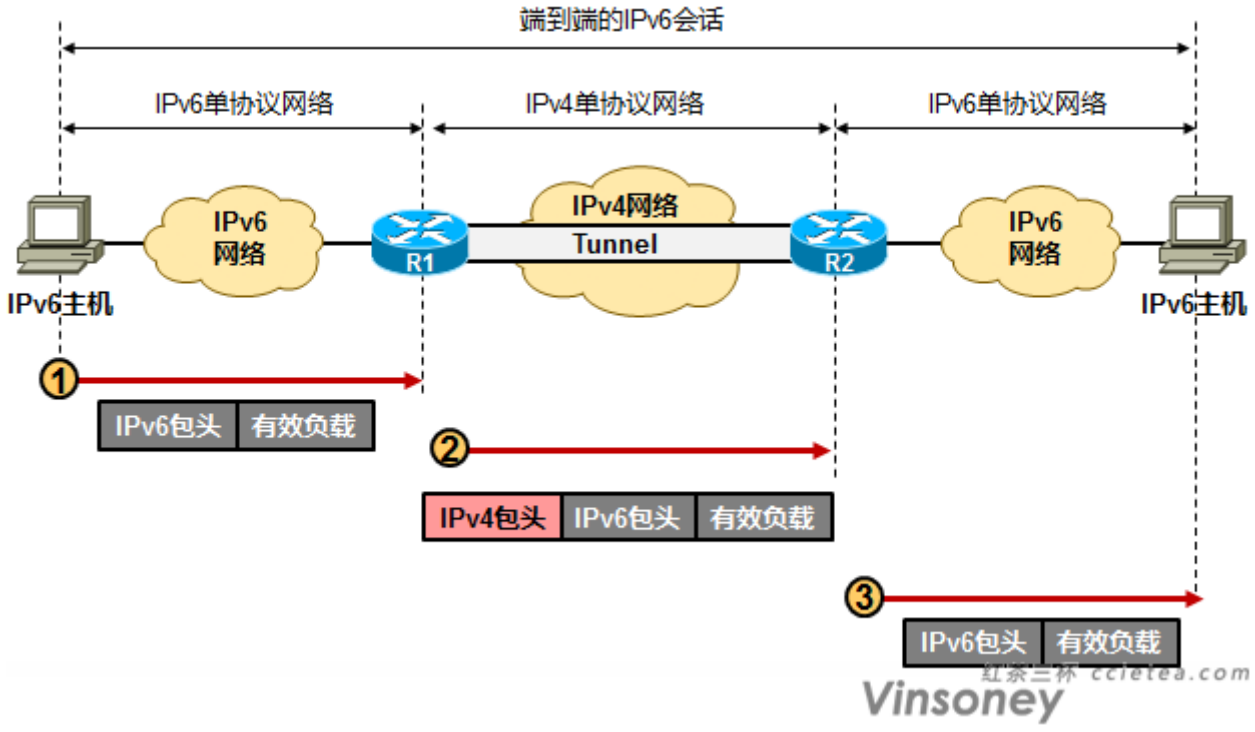
  【插嘴】　在 R1 所添加的这个 IPv4 包头中 ,protocol 字段用于指示上层封装的是 IPv6 的报文 ,protocol 字段值为 41：

```
Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
  Version: 4
  Header length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 120
  Identification: 0x0033 (51)
⊞ Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: IPv6 (0x29)
⊞ Header checksum: 0x8424 [correct]
  Source: 10.1.12.1 (10.1.12.1)
  Destination: 10.1.23.3 (10.1.23.3)
Internet Protocol Version 6                IPv6头
Internet Control Message Protocol v6       有效负载，这个是直接ping产生数据
```

- R2 将 R1 添加的 IPv4 包头去除，还原成那个原始的 IPv6 数据，然后转发给右边的 IPv6 PC。

7.3.2 手工 IPv6 over IP 隧道

1. 机制介绍



R1、R2 为双栈路由器，同时连接到 IPv6 及 IPv4 网络。

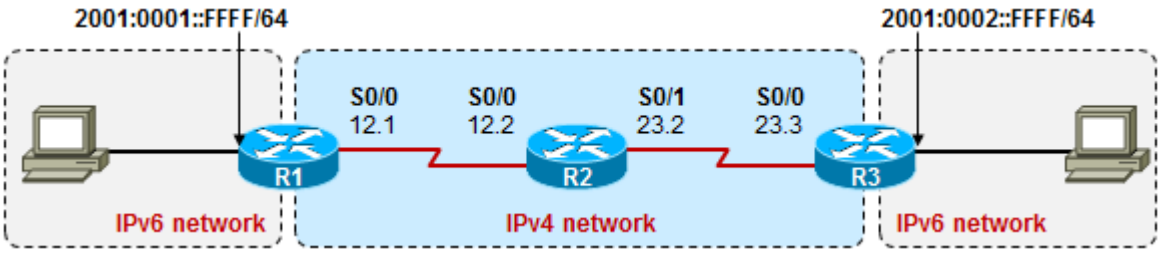
在 R1、R2 下分别挂着两个 IPv6 的孤岛，现在，通过在 R1 及 R2 间建立一个 IPv6 over IP 的 tunnel，使得两个 IPv6 的孤岛得以穿越 IPv4 的因特网而进行互联。原始的 IPv6 报文前面，被添加上一个隧道的 IPv4

头，从而形成一个外层的 IPv4 报文的报文，这个报文在 IPv4 网络中被路由。

关于配置，这里贴个简单的示例：

```
Ipv6 unicast-routing
Interface serial0/0
   ip address 10.1.12.1 255.255.255.0
Interface fa1/0
   Ipv6 enable
   Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
   Ipv6 enable
   tunnel mode ipv6ip
   tunnel source serial 0/0
   tunnel destination 10.1.23.3
ip route 0.0.0.0 0.0.0.0 10.1.12.2        !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0   tunnel 0                !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

2. 基础实验



R1 的配置如下：

```
Ipv6 unicast-routing
Interface serial0/0
   ip address 10.1.12.1 255.255.255.0
Interface fa1/0
   Ipv6 enable
   Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
   Ipv6 enable
   tunnel mode ipv6ip
   tunnel source serial 0/0
   tunnel destination 10.1.23.3
```

```
ip route 0.0.0.0 0.0.0.0 10.1.12.2          !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0    tunnel 0                 !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

R3 的配置类似；R2 只需配置接口 IP 即可；PC1 及 PC2 各自配好 IPv6 地址，指网关即可。

Tunnel 接口未必一定需要 ipv6 address，当然，也可以配置 v6 地址，这不影响穿越路由器的流量

配置完成后，PC1 即可 ping 通 PC2，抓包如下：

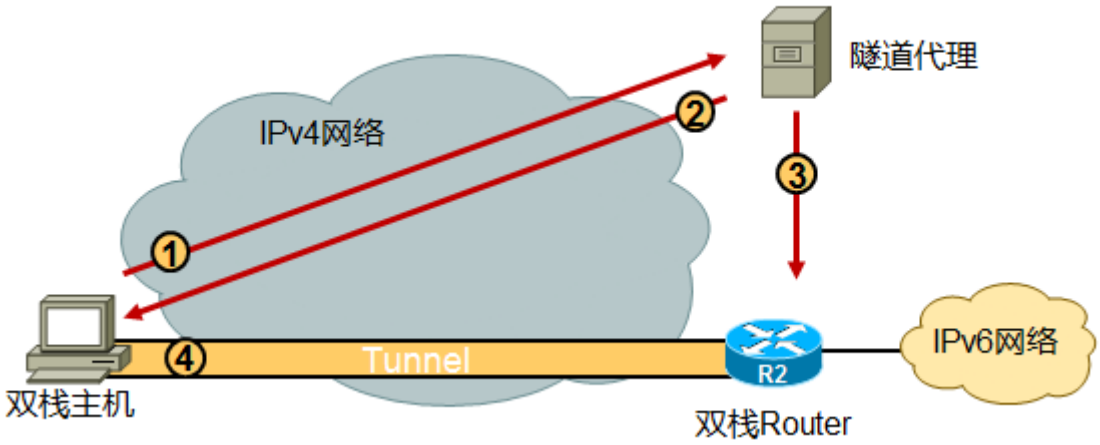
```
⊞ Cisco HDLC
⊟ Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 120
    Identification: 0x0019 (25)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: IPv6 (0x29)
  ⊞ Header checksum: 0x843e [correct]
    Source: 10.1.12.1 (10.1.12.1)
    Destination: 10.1.23.3 (10.1.23.3)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source: 2001:1::1 (2001:1::1)
    Destination: 2001:2::1 (2001:2::1)
⊞ Internet Control Message Protocol v6
```

无论是 GRE 隧道 ,或是 IPv6 over IP 隧道 ,在隧道两头都可以跑动态路由协议。例如在上面 ,可以为 tunnel 两端接口都配置上 ipv6 地址，然后激活动态路由协议即可。从封装效率来说，还是 IPv6 over IP 会比 GRE 更好，毕竟少了一层 GRE 的封装，但是当跑动态路由协议的时候，在某些场合下，GRE 封装的通用性或者兼容性好比 IPv6 over IP 的封装要好一些，例如跑 ISIS 协议，用 IPv6 over IP 隧道就会有点问题。

3. 隧道代理

IPv6 over IP 手工隧道，由于需要在隧道的两端，也就是两台双栈路由器上配置本地隧道、对端隧道的地址，因此扩展性比较差。为了方便在 IPv4 网络上手工隧道的部署，IETF 定义了隧道代理机制。

如 RFC3053 所定义，隧道代理是一个外部系统，而不是路由器。它在 IPv4 网络中作为服务器，并接受双栈节点的隧道请求。



如上图所示：

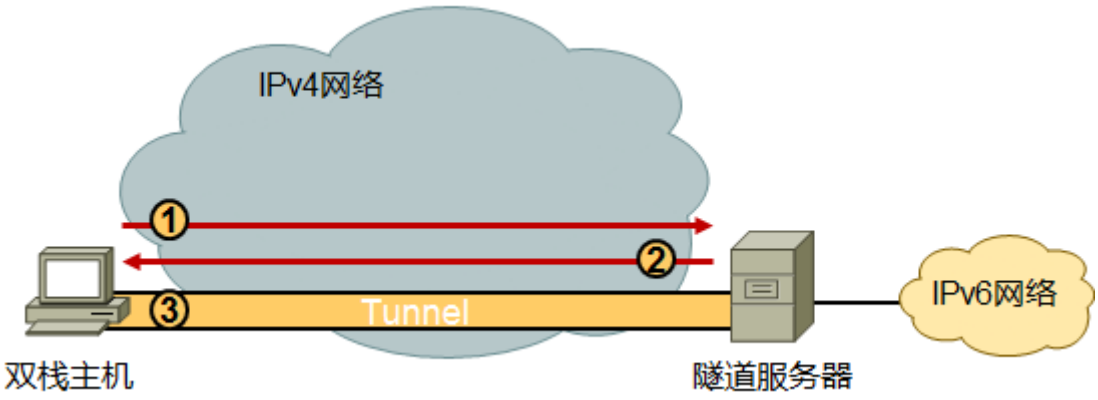
1) 双栈主机通过 IPv4 使用 HTTP 访问隧道代理。这个双栈主机可能是个终端用户，它填写一个网页。
2) 双栈主机通过 HTTP 从隧道代理处获得 IPv4 和 IPv6 地址 然后终端用户使用这些地址配置并启用隧道，其中获得到的 IPv4 地址就是隧道的 IPv4 地址。
3) 与此同时，隧道代理自动地在一个连接到 IPv6 网络的双栈路由器上应用配置隧道的远端配置，一旦配置在双栈主机和双栈路由器上得到了应用，隧道就会被正确的建立起来
4) 端到端的 IPv6 隧道建立起来之后，IPv6 流量就可以进行通讯了。

一般来说，隧道代理和双栈 Router 同属一个管理方，例如同一家公司。因为隧道代理要有双栈 Router 的配置权限。

另外，CISCO IOS 不支持隧道代理。但是其实现在公网上有许多免费的隧道代理服务器。

- **隧道服务器**

隧道服务器是隧道代理的简化模型。将隧道代理和双栈 Router 进行了整合。



1) IPv4 网络上的双栈主机首先通过 IPv4 使用 HTTP 访问隧道服务器，终端用户通过填写网页并从隧道服务器获得 IPv4 及 IPv6 地址
2) 终端用户通过双栈主机获得的地址配置隧道，而隧道服务器在本地应用隧道的远端配置
3) 最终，隧道建立起来

CISCO IOS 还不支持隧道服务器。

7.3.3 6to4 自动隧道

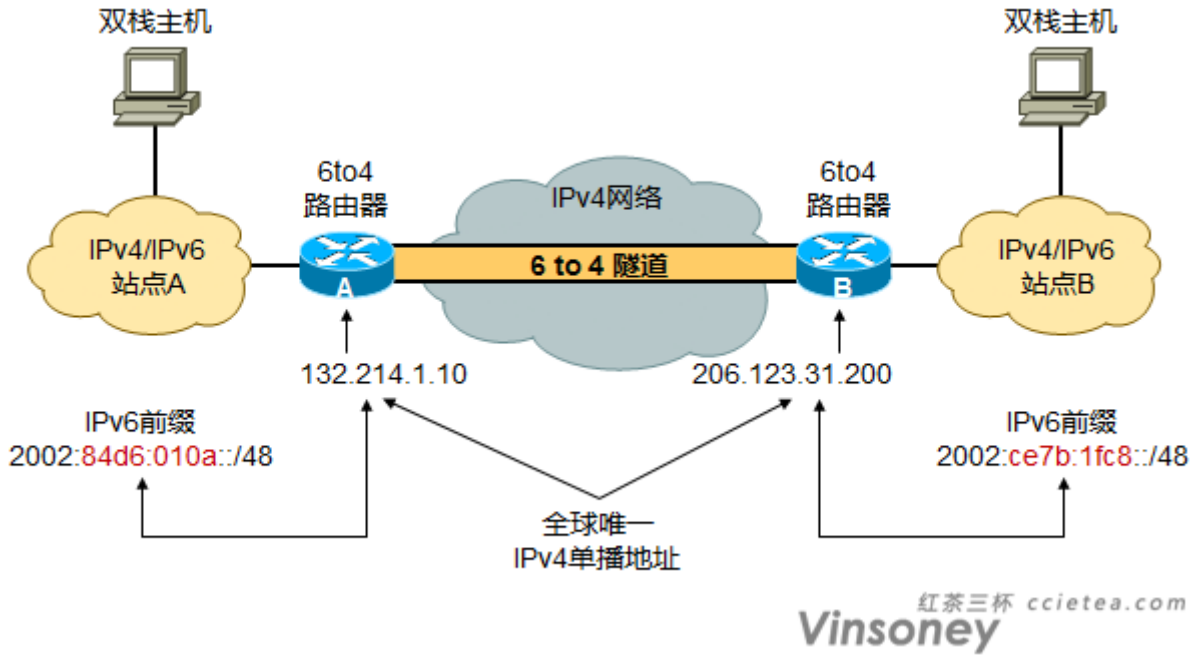
1. 机制介绍

我们来回顾一下 IPv6 over IP 手工隧道（也叫做配置隧道）的配置：

```
Ipv6 unicast-routing
Interface serial0/0
  ip address 10.1.12.1 255.255.255.0
Interface fa1/0
  Ipv6 enable
  Ipv6 address 2001:0001::FFFF/64
Interface tunnel 0
  Ipv6 enable
  tunnel mode ipv6ip
  tunnel source serial 0/0
  tunnel destination 10.1.23.3
ip route 0.0.0.0 0.0.0.0 10.1.12.2
Ipv6 route ::/0   tunnel 0
```

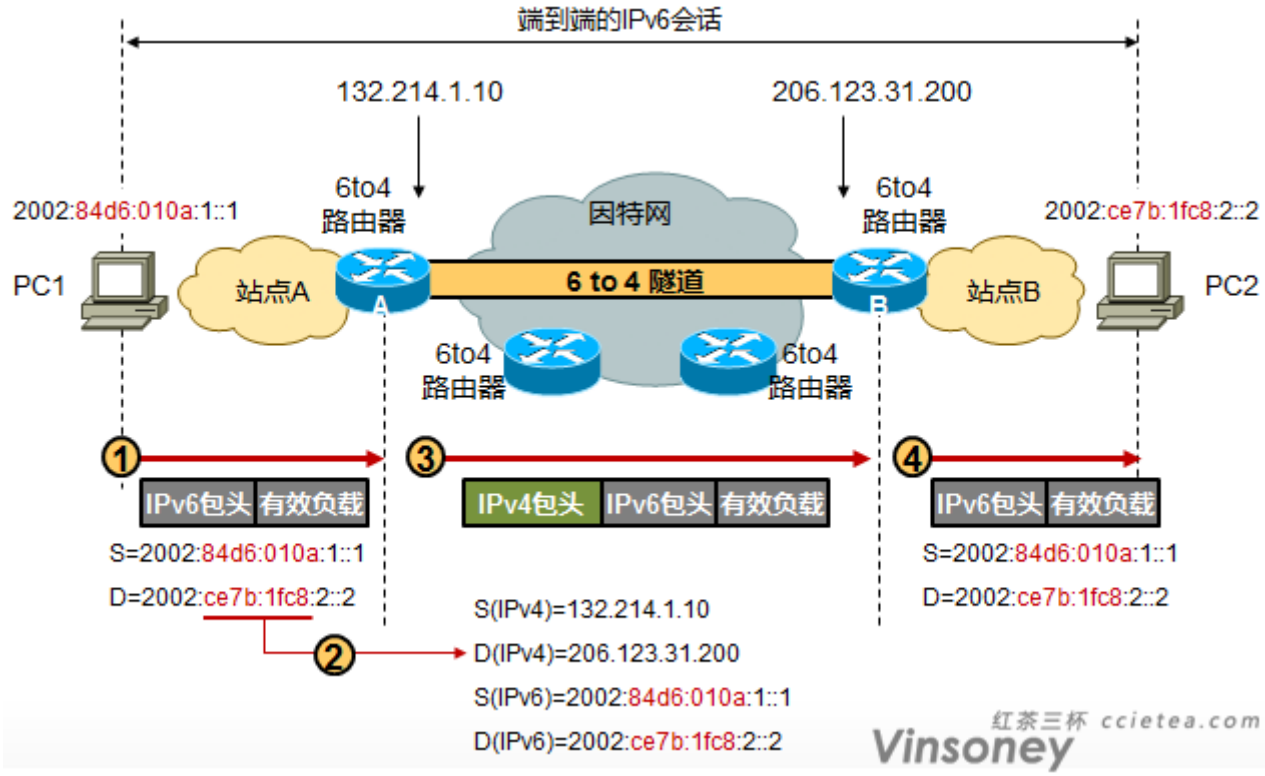
我们注意到，对于 tunnel 的配置，需要制定 tunnel 的 destination，这样一来，如果隧道比较多，这种配置就变的非常笨重了，而且可扩展性太差。因此，我们又有另外一个机制：6to4 自动隧道。

2. 机制详解



看上图：路由器 A 及 B 分别是两个站点。两边都申请到一个 IPv4 公网地址，使用这个公网 IPv4 地址进行映射得到全球唯一的 6to4 IPv6 地址，这个 6to4 IPv6 地址/48 位，空间非常大，用于站点内的 IPv6 用户。如此一来，当站点 A 内的 IPv6 用户访问站点 B 的 IPv6 用户时，IPv6 数据包发送到 A，那么 A 根据 IPv6 数据报头中的目的 IPv6 地址，得到对应的 IPv4 全局 IP，那么就将原始的 IPv6 数据包进行封装，套上 6to4 隧道的 IPv4 的头，然后将这个数据包放入 IPv4 网络进行路由，直到到达 B。

接下去，我们来看一下数据包交互的详细过程：



A 拿到的 IPv4 公网地址是 132.214.1.10，而 B 是 206.123.31.200。

A 和 B 分别使用这个 IPv4 公网地址来映射得到 6to4 地址空间，6to4 地址空间使用 2002::/16，而 6to4 地址的形成如下：

2002：**IPv4 地址**：子网 ID：：接口 ID

上面的 IPv4 地址，就是 A 或 B 拿到的那个公网 IPv4 地址，注意，这个公网 IP 非常重要，此公网 IP 的变化，有可能导致 IPv6 站点内 IPv6 不得不重新编址。而且，当然，这个 IPv4 地址不能使用私有 IPv4 地址。

得到 6to4 IPv6 地址空间后，这个地址空间是 16+32=48bits，也就是/48 的，因此空间非常大，你可以进一步的进行子网的划分，这个 IPv6 的地址空间就将用于站点内网。

1) 上面的例子，A 拿到的 IPv4 公网地址：132.214.1.10，映射得到 2002:**84d6:010a**::/48，这个地址空间的一部分最终被用在了 PC1 上。现在，PC1 要给 PC2 发送一个 IPv6 的数据。这个数据的源 IP 是 PC1 的 IPv6 全局唯一地址 2002:84d6:010a:1::1，目的 IP 为 PC2 的 IPv6 全局唯一地址 2002:ce7b:1fc8:2::2，这个 IPv6 报文被送到了 PC1 的默认网关也就是路由器 A。
2) A 是一台 6to4 路由器，它会查看这个 IPv6 数据包的包头里的目的 IPv6 地址，它发现这个 IPv6 目的地地址是一个 6to4 地址，因此，它计算得出这个 6to4 IPv6 地址对应的 IPv4 公网 IP，然后，为这个原始的 IPv6 数据进行封装，加上一个新的 IPv4 的头，而这个 IPv4 的头源地址是 132.214.1.10，目的地址刚才通过计算得出的那个 IPv4 公网地址，正是 206.123.31.200。然后，这个新的 IPv4 数据包进入因特网，最终被传送到 B。
3) B 将报文的 IPv4 头除去，得到原始的 IPv6 数据，转发给 PC2。

3. 机制补充

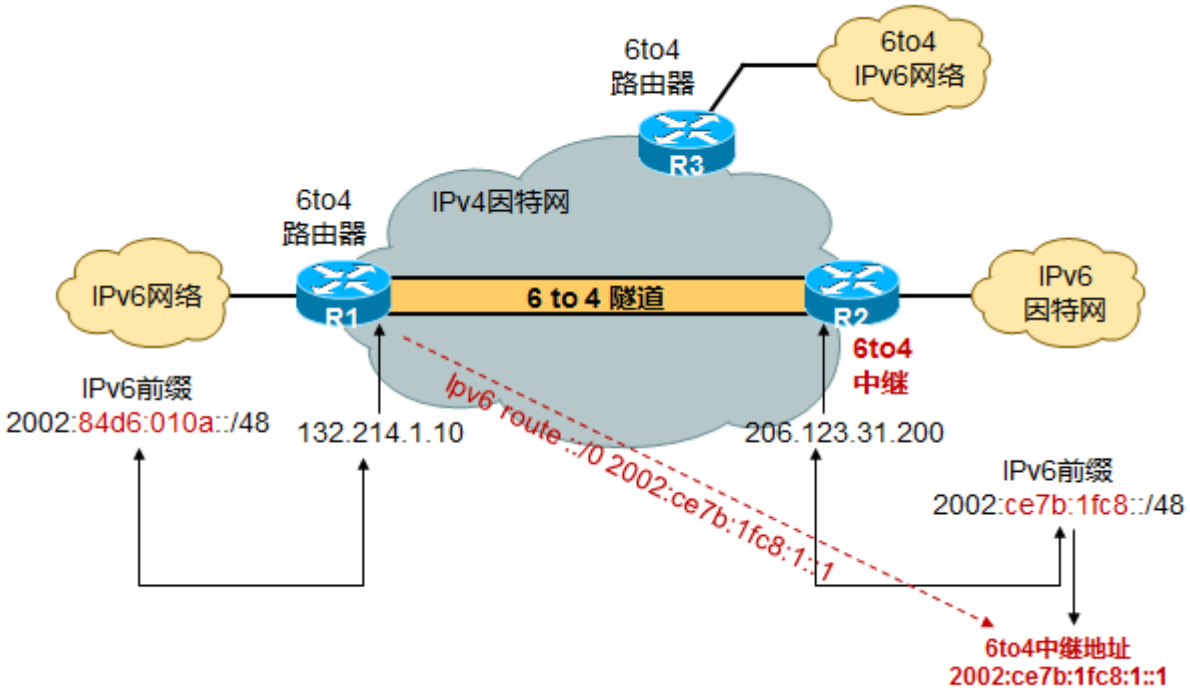
上面提到的 **IPv4 地址**是为 IPv6 孤岛申请的公有地址，在 IPv6/IPv4 边界路由器（6to4 路由器）上配置该 v4 地址，可以直接为 6to4 路由器连接公网接口的 IP，也可以是 6to4 路由器 Loopback 接口的 IP，这个 IP 同时是 6to4 tunnel 的 source 或 destination。

隧道的源 IPv4 地址手工指定（就是前面提到的 **IPv4 地址**），隧道的目标 IPv4 地址不需要显式配置（路由器动态建立隧道），根据通过隧道转发的 IPv6 报文决定，如果 IPv6 报文的目的地是 6to4 地址，则从目的 IP 中提取 IPv4 地址，如果目的地不是 6to4 地址，那么就需要使用 6to4 中继。

所以 6to4 隧道是自动完成查找和建立的，无需手工配置 tunnel 的 destination，相比配置隧道，可扩展性要更高。

4. 6to4 中继

之前我们所讲的，都是两个 IPv6 孤岛都使用的是 6to4 的 IPv6 地址空间。6to4 边界路由器在收到发送往 6to4 网络的 IPv6 数据，能够从目的 IPv6 地址中得到对应的公网 IPv4 地址从而进行转发。但是，如果 IPv6 孤岛使用的不是 6to4 地址空间，而是像 2001::/16 这样的常规可聚合全球单播地址呢？这就需要使用 6to4 中继了。



我们看上面的图，其实利用 6to4 自动隧道技术，IPv6 孤岛既可以使用 6to4 IPv6 地址空间，也可以使用普通的 IPv6 地址空间，上图中，R1 上挂的 IPv6 孤岛使用 6to4 地址空间，R3 上的 IPv6 孤岛也是 6to4 空间，而 R2 这台因特网上的 6to4 路由器同时也连接到了 IPv6 的因特网和 IPv4 因特网，因此他具有 IPv6 公网的路

由。所以，R1、R3 下的两个 6to4 孤岛的互相访问，解决方案我们前面已经讲过了，但是 R1 或 R3 如果需要访问非 6to4 的 IPv6 网络呢？就需要 R2 了，R2 此时是一台 **6to4 中继路由器**。此刻，R2 通过其自身的一个 IPv4 公网 IP，得到一个 6to4 的 IPv6 全球可聚合单播地址，这个 IPv6 地址用于响应其他 6to4 路由器的隧道建立请求。

我们拿 R1 举例，在 R1 上，配置可能如下：

```
Interface loopback0
  Ip address 132.214.1.10 255.255.255.0
Interface fast0/0
  Ipv6 enable
  Ipv6 address 2002:84d6:010a:0001:/64 eui-64        !!这是 IPv6 站点内网的网关
Interface tunnel1
  Ipv6 enable
  Ipv6 unnumbered fast 0/0
  tunnel source loopback0
  tunnel mode ipv6ip 6to4
!
Ipv6 route 2002::/16 tunnel1                !!当本地访问远端的 6to4 网络时，走 tunnel
Ipv6 route ::/0 2002:ce7b:1fc8:1::1         !!当访问的远端 IPv6 网络使用的地址空间不是 6to4 时 ,走默认路由，这时候会找下一跳。
```

经过上述配置，当 R1 所连接的 IPv6 岛屿中用户要访问其他 6to4 网络时，那么 R1 就从目的 6to4 IPv6 地址中得到 IPv4 公网 IP，然后将隧道 IPv4 数据包转发到目的地的 6to4 路由器。而如果 R1 所连接的 IPv6 岛屿中用户要访问非 6to4 的 IPv4 网络时，就需要求助于 6to4 中继，也就是 R2，那么怎么找到 R2 呢？

首先 R2 自己通过自己的 IPv4 公网地址映射得到一个 6to4 地址空间，R2 给自己分配了这个 6to4 IPv6 地址空间中的一个地址（2002:ce7b:1fc8:1::1），以便其他 6to4 路由器能够找到自己。而对于 R1 这样的 6to4 路由器，只需添加一条默认路由：Ipv6 route ::/0 2002:ce7b:1fc8:1::1 即可，这样一来 R1 所连接的 IPv6 岛屿中用户要访问非 6to4 的 IPv4 网络时，数据被这条默认路由所匹配，下一跳为 2002:ce7b:1fc8:1::1，这是一个 6to4 地址，于是 R1 首先根据这个地址得到其对应的 IPv4 地址：206.123.31.200，然后将对 2002:ce7b:1fc8:1::1 再进行路由表的递归查找，发现要从 tunnel1 接口扔出去，于是给原始的 IPv6 数据压上新的 IPv4 隧道头，源地址为隧道的 IPv4 源地址 132.214.1.10，目的地址为 206.123.31.200。这个数据包就这么到了 R2。

在因特网上，像 R2 这类 6to4 中继路由器还是有不少的。有一些公共的 6to4 中继。例如：6to4.ipv6.microsoft.com 等。

有一点要注意的是，如果这个 6to4 中继路由器距离你的网络“很远 “，那么这可能会造成你和 IPv6 因特网之间的 IPv6 流量性能低下。为了帮助一个 6to4 站点在因特网上找到可用的 6to4 中继并且给 6to4 机制更多可扩展性，RFC3068 引入了一个任意播前缀。这样 6to4 数据包可以被自动路由到 IPv4 因特网上最近的 6to4 中继。IANA 分配了 6to4 中继任意播前缀 192.88.99.0/24 专门用于自动路由 6to4 数据包到最近的 6to4 中继。在这个任意播前缀中，所定义的到达最近的 6to4 中继的 IPv4 地址若是 192.88.99.1，那么这个地址的 IPv6 表示就是 2002.c058.6301::，在你的 6to4 配置中，可以使用 ipv6 roiute ::/0 2002.c058.6301::来配置默认路由。

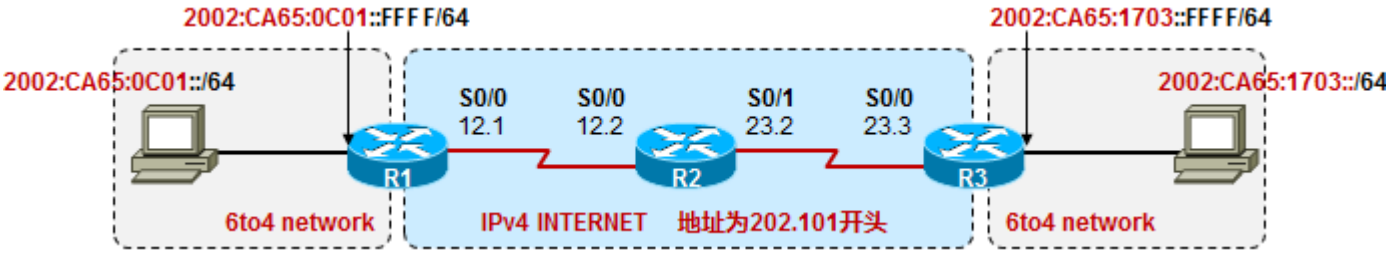
你可以配置一台 CISCO 路由器作为一个具有任意播 IPv4 前缀的 6to4 中继：

```
Interface tunnel1
   Ipv6 enable
   Ipv6 unnmbered fast0/0
   tunnel source fast0/0
   ipv6 address 2002.c058.6301::/128 anycast
   tunnel mode ipv6ip 6to4
interface fast0/0
   ipv6 enable
   ip address 132.214.1.10 255.255.255.0
   ip address 192.88.99.1 255.255.255.0 secondary
   ipv6 address 2002:84d6:010a:0001::/64 eui-64
!
Ipv6 route 2002::/16 tunnel1
```

注意，这里的 IPv4 前缀 192.88.99.0/24 应该被发布到 IPv4 因特网上的路由选择协议中。否则这个使用任意播前缀的 6to4 中继在 IPv4 因特网中是不可达的。

5. 6to4 自动隧道的配置

【实验 1】两边内网都是用 6to4 地址（双方都是 6to4 网络）



R1 路由器申请的 ipv4 公网地址为 202.101.12.1；R3 申请的公网地址为 202.101.23.3

有了公网地址，我们也就有了根据公网地址计算得来的 6to4 的地址空间：

2002:IPV4 地址映射:子网 ID::/48　　前面 48 位是 2002+ipv4 的公网地址，后面 64 位是接口 ID

例如 R1 公网地址为：202.101.12.1，那么对应的 6to4 地址空间就是：2002:CA65:0C01::/48

这个地址空间加上子网 ID ,就可以分配给内网用户了 ,这个实验我们内网用户采用无状态自动配置获取地址。

这时候，如果 R1 下有用户要访问 R3 下的 v6 网络，那么目的地址肯定是 R3 的 6to4 地址空间，数据到了 R1 后，R2 一看，发现是 2002 的 ipv6 地址，于是它就直接去读 2002 后面的 32 位，将其转换成 ipv4 地址，并作为 6to4 的 tunnel destination。

R1 的配置如下：

```
ipv6 unicast-routing
!
interface Tunnel0
  ipv6 enable
  tunnel source Serial0/0                    !! 注意，不用指 destination，因为是动态的
  tunnel mode ipv6ip 6to4
interface fast1/0
  ipv6 address 2002:CA65:0C01::FFFF/64
  ipv6 enable
interface Serial0/1
  ip address 202.101.12.1 255.255.255.0
!
ipv6 route 2002::/16 Tunnel0
```

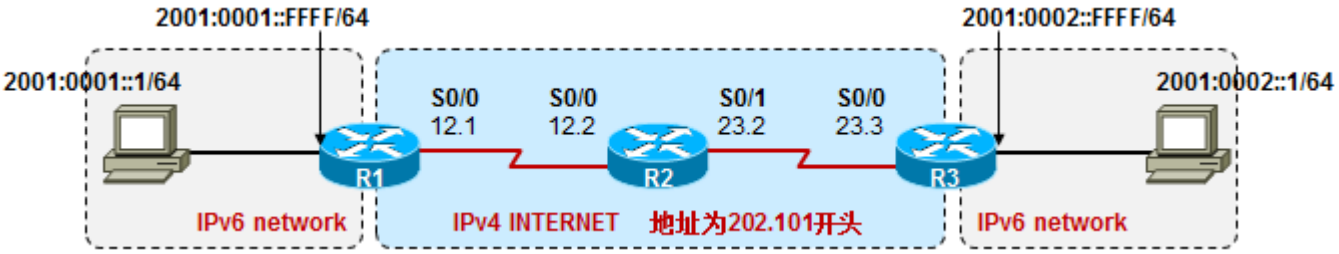
R2 的配置如下：

```
ipv6 unicast-routing
!
interface Tunnel0
  ipv6 enable
  tunnel source Serial0/0
  tunnel mode ipv6ip 6to4
interface fast1/0
  ipv6 address 2002:CA65:1703::FFFF/64
  ipv6 enable
interface Serial0/1
```

```
 ip address 202.101.23.3 255.255.255.0
!
ipv6 route 2002::/16 Tunnel0
```

PC（用路由器模拟）使用无状态自动获取地址： ipv6 address autoconfig default

【实验 2】两边的 IPv6 孤岛都是使用普通的 IPv6 全局地址



两端均为普通 IPv6 网络，使用常规的全局 IPv6 地址。

R1 及 R3 分别申请公网 Ipv4 地址，配置动态 6to4 隧道。以 R1 为例，配置 tunnel0，分配一个 IPv6 地址（6to4 地址），这个地址正是 R1 的 V4 外网地址对应的 IPv6 6to4 地址，也即 202.101.12.1 对应 2002:**CA65:0C01**::1/48 ,R3 同理。接下来其实就是静态路由的把戏了。这时 ,内网有数据去往 2001:0002::/16 网络时，数据被送到 2002:CA65:1703::FFFF（下一跳，也就是 R3 的 tunnel 6to4 地址）

路由器通过递归查询到，下一跳 2002:CA65:1703::FFFF 应该扔到 tunnel0，而 tunnel0 是 6to4 隧道，于是将 2002:CA65:1703::FFFF 翻译成对应的 V4 地址，也就是 202.101.23.3

R1	R3
ipv6 unicast-routing	ipv6 unicast-routing
interface Tunnel0	interface Tunnel0
ipv6 address 2002:CA65:C01::FFFF/64	ipv6 address 2002:CA65:1703::FFFF/64
ipv6 enable	ipv6 enable
tunnel source Serial0/0	tunnel source Serial0/0
tunnel mode ipv6ip 6to4	tunnel mode ipv6ip 6to4
interface fast1/0	interface fast1/0
ipv6 address 2001:0001::FFFF/64	ipv6 address 2001:0002::FFFF/64
ipv6 enable	ipv6 enable
interface Serial0/0	interface Serial0/0
ip address 202.101.12.1 255.255.255.0	ip address 202.101.23.3 255.255.255.0

ip route 0.0.0.0 0.0.0.0 202.101.12.2	ip route 0.0.0.0 0.0.0.0 202.101.23.2
ipv6 route 2001::/16 2002:CA65:1703::FFFF	ipv6 route 2001::/64 2002:CA65:C01::FFFF
ipv6 route 2002:CA65:1703::/48 Tunnel0	ipv6 route 2002:CA65:C01::/48 Tunnel0

要注意的是 6to4 的隧道技术，不支持动态路由协议。

7.3.4 GRE 隧道

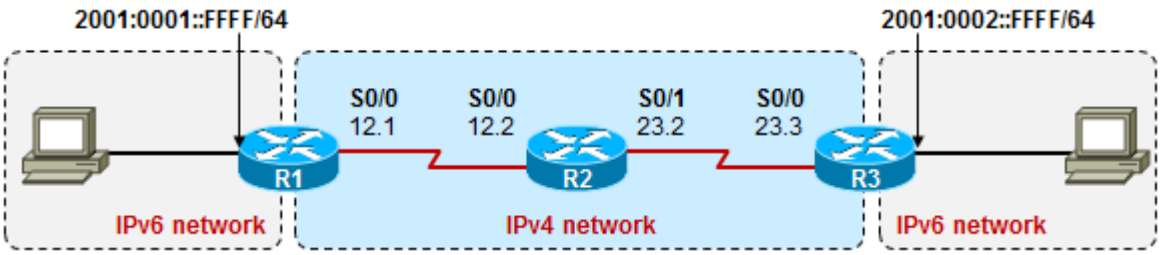
1. 机制介绍

- GRE 隧道是一种众所周知的能够保证稳定和安全的端到端链路的标准隧道技术。
- GRE 隧道同样需要在隧道的两端手工互相指定隧道的目的地
- GRE 隧道给在域内使用 IS-IS 作为 IPv6 路由选择协议的组织提供了方便。如果是 IPv6 over IP 的手工隧道，跑 IS-IS 是会出问题的，因为 IS-IS 需要在网络上相邻的路由器之间发送链路层信息，GRE 是仅有的能够在 IP 基础设施上携带这种类型流量的隧道协议。所以，同一条 GRE 隧道可以用来在广域网中同时传输 IPv6 数据包及 IS-IS 路由器间的 IS-IS 链路层信息。

2. GRE 隧道配置

```
Interface tunnel 0
Ipv6 enable
tunnel mode gre ip                    !! 注意隧道模式
tunnel source serial 0/0
tunnel destination 10.1.23.3
```

3. GRE 隧道基础实验



R1 的配置如下：

```
Ipv6 unicast-routing
Interface serial0/0
```

```
 ip address 10.1.12.1 255.255.255.0
Interface fa1/0
 Ipv6 enable
 Ipv6 address 2001:0001::FFFF/64
!
Interface tunnel 0
 Ipv6 enable
 tunnel mode gre ip                    !! 注意隧道模式
 tunnel source serial 0/0
 tunnel destination 10.1.23.3
!
ip route 0.0.0.0 0.0.0.0 10.1.12.2     !! ipv4 路由 ,使得路由能访问到 tunnel destination ,也就是 10.1.23.3
Ipv6 route ::/0   tunnel 0             !! ipv6 路由，前往 IPv6 网络的流量全部扔到 tunnel
```

R3 的配置大同小异，只不过隧道接口的 destination 修改一下即可；

如果需要在 R1、R3 之间运行动态路由协议，则在 tunnel 接口上激活路由选择协议即可。

GRE 隧道报文抓包如下：

```
⊞ Cisco HDLC
⊟ Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 10.1.23.3 (10.1.23.3)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 124
    Identification: 0x0030 (48)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: GRE (0x2f)
  ⊞ Header checksum: 0x841d [correct]
    Source: 10.1.12.1 (10.1.12.1)
    Destination: 10.1.23.3 (10.1.23.3)
⊞ Generic Routing Encapsulation (IPv6)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 60
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source: 2001:1::1 (2001:1::1)
    Destination: 2001:2::1 (2001:2::1)
⊞ Internet Control Message Protocol v6
```

7.3.5 ISATAP 隧道

1. **机制介绍**

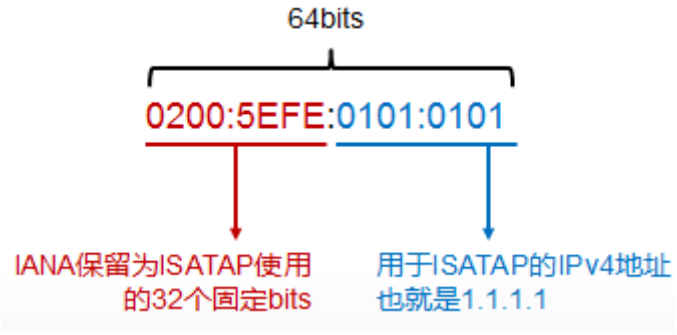
ISATAP（Intra-Site Automatic Tunnel Addressing Protocol）

ISATAP 是一种非常容易部署和使用的 IPv6 过渡机制。在一个 IPv4 网络中，我们可以非常轻松的进行 ISATAP 的部署，首先你的 PC 需是 V4/V6 双栈 PC，然后，需要有一台支持 ISATAP 的路由器，ISATAP 路由器可以在网络中的任何位置，只要 PC 能够 ping 通它（当然，你要知道路由器的 IPv4 地址）。那么接下去，我们可以通过在路由器上部署 ISATAP，这样网络中支持 ISATAP 的双栈主机，在需要访问 IPv6 资源时，可以与 ISATAP 路由器建立起 ISATAP 隧道，ISATAP 主机根据 ISATAP 路由器下发的 IPv6 前缀构造自己的 IPv6 地址（这个 IPv6 地址是被自动关联到 ISATAP 主机本地产生的一个 ISATAP 虚拟网卡上），并且将这台 ISATAP 路由器设置为自己的 IPv6 默认网关，如此一来，后续的这台主机就能够通过这台 ISATAP 路由器去访问 IPv6 的资源。

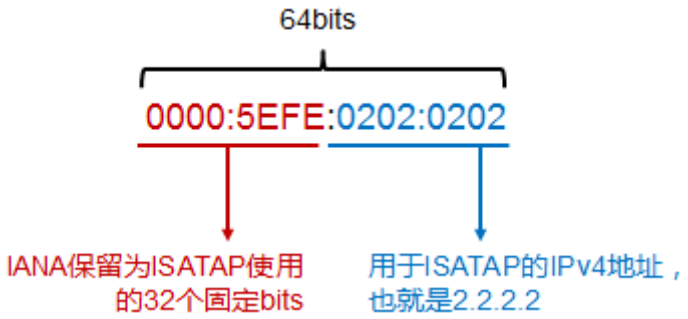
这种方法部署起来非常简单，在许多场合，客户为了节省成本，又希望网络中的 IPv6 主机能够访问 V6 资源，同时又不愿意对现有网络做大规模的变更及设备升级，那么就可以采用这种方法，购买一台支持 ISATAP 的路由器，甚至可以将 ISATAP 路由器旁挂在网络上，只要它能够访问 V6 资源并且响应 ISATAP PC 的隧道建立请求。

2. **ISATAP 的功能组件如下：**

- **自动隧道：** ISATAP 的隧道机制也是自动的，隧道在主机和 ISATAP 路由器之间被创建。主机首选需要知道 ISATAP 路由器的 IPv4 地址。
- **ISATAP 地址格式：** 分配给 ISATAP 路由器的 IPv6 地址是全局单播地址，该地址的前缀将被 ISATAP 主机用于自己的 IPv6 地址构造。ISATAP 主机通过在 IPv4 建立起来的 ISATAP 隧道从 ISATAP 路由器发送的消息中接收/64 的 IPv6 前缀，并且使用这个前缀结合“特殊的接口标识”来构造自己的 IPv6 地址。
- **接口标识：** ISATAP 在主机上启用后，会产生一个 ISATAP 虚拟网卡，该虚拟网卡会产生一个 64bits 的特殊接口标识，有点类似 EUI-64，但是产生机制不同，它是由专为 ISATAP 保留的 32 位的 **0200:5EFE** 加上主机上配置的 IPv4 地址构成，如下图，假设 ISATAP 主机配置的 IPv4 地址为 1.1.1.1，那么 ISATAP 虚拟网卡的 64bits 接口标识就是：



另一方面，在路由器上部署 ISATAP 后，路由器也会产生一个 tunnel 接口，用于响应 ISATAP 主机的隧道建立请求，这个 tunnel 接口同样会产生接口标识。地址的格式是 IANA 保留给 ISATAP 的 32 比特的 **0000:5EFE** 后追加 32 比特的 IPv4 地址。如下图，假设给 ISATAP 路由器配置的 IPv4 地址（用于隧道的）是 2.2.2.2，那么 ISATAP tunnel 的接口标识就是：

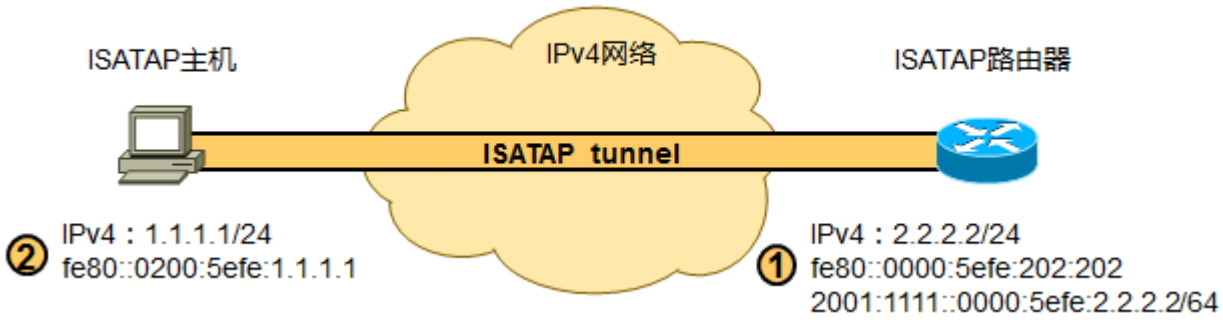


这里关于 64bits 的接口标识中“为 ISATAP 保留的”高阶 32bits 在维基百科上有这么一段描述：“The link-local address is determined by concatenating fe80:0000:0000:0000:**0200:5efe:** for global unique and fe80:0000:0000:0000:**0000:5efe**: for private addresses with the 32 bits of the host's IPv4 address.”。貌似有全局唯一和私有之分，不过在 IETF 的相关草案上找到的更多是 0200:5efe 的描述，在我所作的测试环境中，windows 主机上系统使用的是 0200:5ede，而 CISCO 路由器上用的是 0000:5efe。

ISATAP 主机和 ISATAP 路由器产生的这个 64btis 的接口标识，可进一步用于构造隧道接口的 Linklocal 地址，以及 IPv6 全局单播地址。这个下面会描述到。

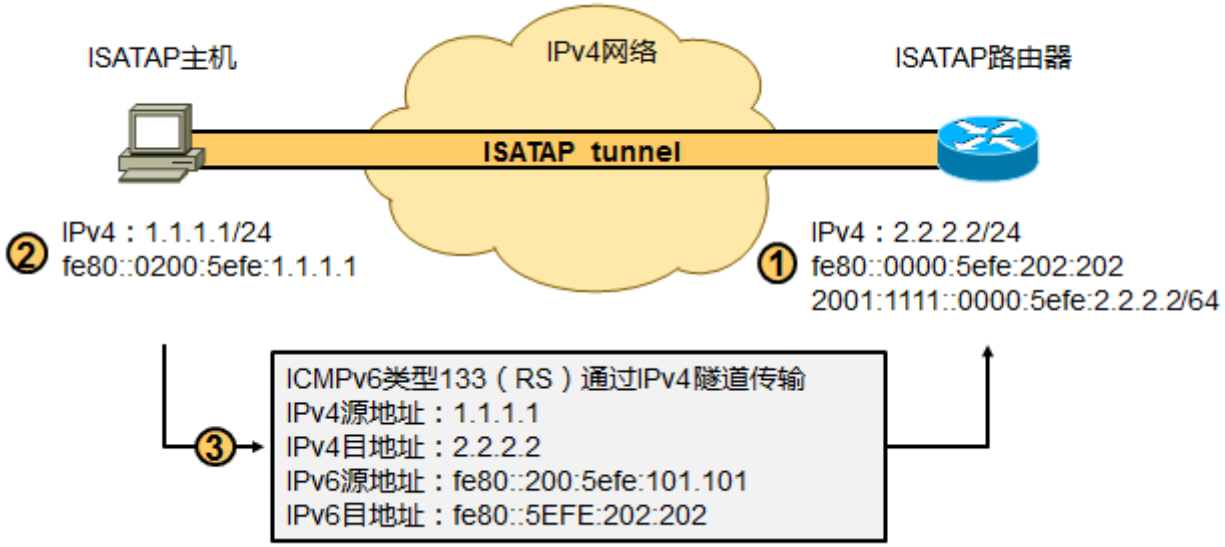
另外，因为 ISATAP 的操作范围在站点内，所以 ISATAP 主机和 ISATAP 路由器的 IPv4 地址可以是私有 IP，也可以是公有 IP。

3. 工作机制



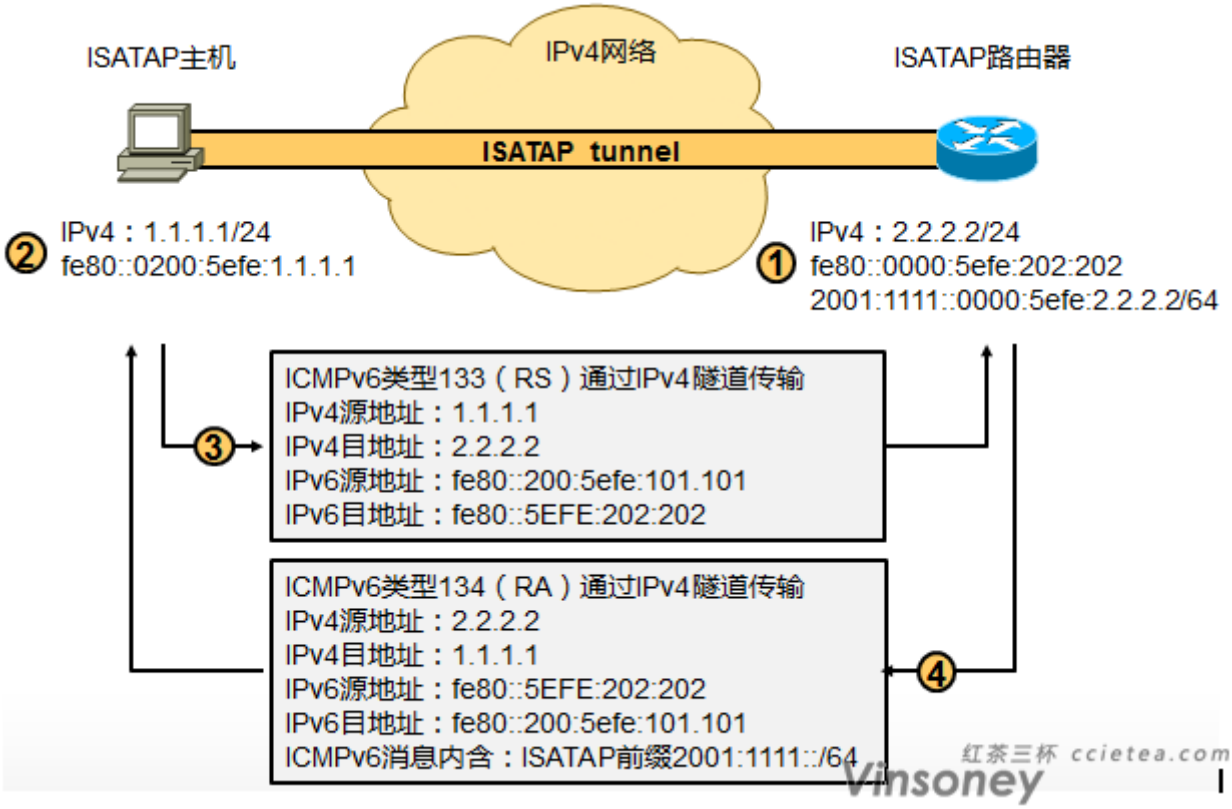
首先，我们有一个 IPv4 的网络，IPv4 网络中绝大部分网路设备都不支持 IPv6，除了终端主机，以及一台路由器，这台能够访问我们需要的 IPv6 资源。现在，一种最廉价的方式是，在这台路由器上部署 ISATAP，终端 ISATAP 主机与路由器之间建立一个 ISATAP tunnel，这样一来 PC 可以直接将 IPv6 流量放进 tunnel 传到 ISATAP 路由器从而穿越整个 IPv4 网络。

1) 现在我们在 ISATAP 路由器上进行相应的配置，给路由器分配的 IPv4 地址是 2.2.2.2/24，同时建立一个 tunnel 接口用于 ISATAP，此时 tunnel 接口会根据 IPv4 地址产生一个 64bits 的接口标识。这个接口标识搭配上高位的 fe80::就形成了 tunnel 接口的 Linklocal 地址：fe80::0000:5efe:202:202。另外，还需给 ISATAP tunnel 接口配置一个全局单播 IPv6 地址，这里可以手工配置，也可以通过前缀+EUI64 的方式来构建，这里的 EUI-64 就是上面所述的特殊的 64bits 接口标识。如上图，构建出来的 IPv6 地址就是 2001:1111::0000:5efe:0202.0202/64，因此 IPv4 的前缀为 2001:1111::/64，这个前缀稍后会通过 tunnel 下发给 ISATAP 主机，从而使它能够构建自己的 IPv6 地址。
2) 现在我们在 ISATAP 主机上，配置 ISATAP，一般来说，在 WIN7 系统上默认安装了 IPv6 协议栈，默认就会有一个 ISATAP 的虚拟网卡。在我们给 PC 的物理网卡配置 IPv4 地址如 1.1.1.1/24 后，ISATAP 虚拟网卡就会自动根据这个 IPv4 地址计算出上面所讲的特殊的接口标识：0200:5efe:1.1.1.1，注意这种格式等同与 0200:5efe:0101.0101，在 windows 系统上我们可以看到前者的简便写法。
3) 当我们在主机上配置了 ISATAP 路由器之后（指向的是 ISATAP 路由器的 IPv4 地址），ISATAP 主机开始向 ISATAP 路由器发送 RS 消息，如下图：



这个 RS 消息是通过 IPv4 隧道传输的，外层是 IPv4 的头，源地址是 ISATAP 的 IPv4 地址 1.1.1.1，目的地址是 2.2.2.2，也就是 ISATAP 的 IPv4 地址。IPv4 头里面裹着 IPv6 的报文，源地址是 ISATAP 主机的 ISATAP 虚拟网卡的 Linklocal 地址，目的地址是 ISATAP 路由器的 Linklocal 地址。

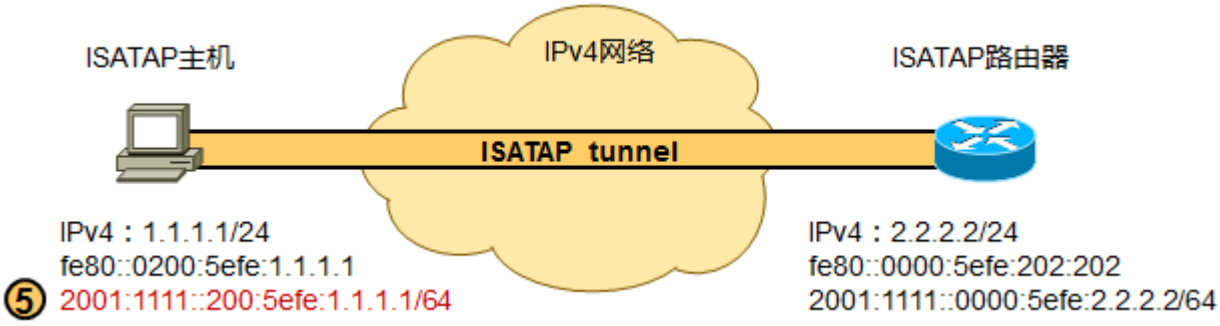
4) ISATAP 主机发出的这个 RS 消息，会在 IPv4 网络中被路由，最终转发到 ISATAP 路由器。这将使得路由器立即以一个 RA 进行回应：



而这个回应的 RA 消息里，就包含 ISATAP 上所配置的那个 IPv6 全局单播地址的/64 前缀。

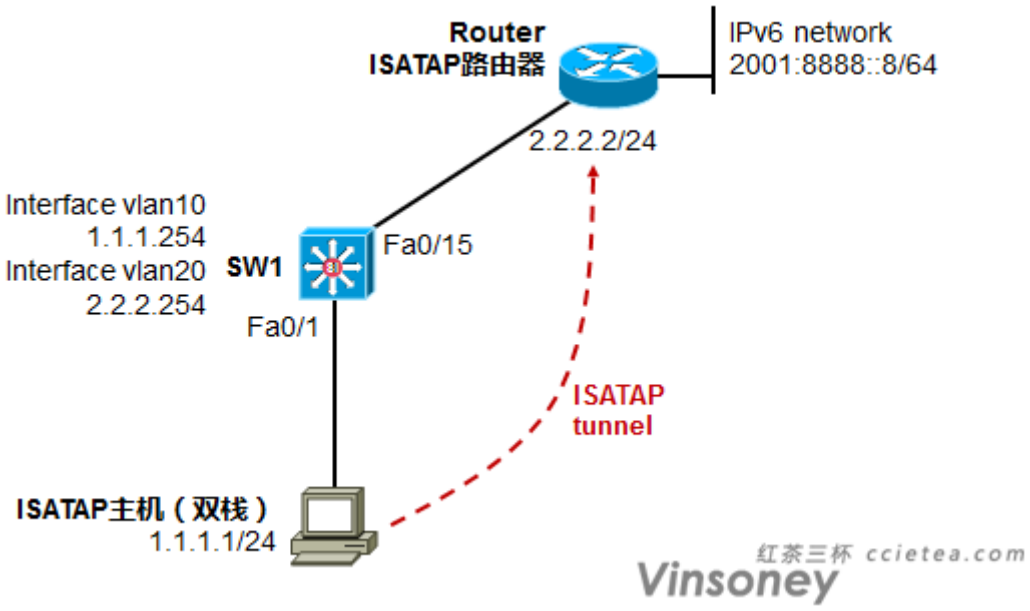
5) ISATAP 主机收到这个 RA 回应后，会拿出里头的 IPv6 前缀，随后在后面加上自己 ISATAP 虚拟网卡的

64bits 的接口标识地址，构成 128bits 的 IPv6 全局单播地址，同时会产生一条默认路由，指向 ISATAP 路由器的 Linklocal 地址：



6) 从现在起，ISATAP 主机需要访问 IPv6 资源的时候，将 IPv6 数据包封装在 IPv4 的隧道里，也就是说，套上 ISATAP 隧道的 IPv4 头，然后传给 ISATAP 路由器，再由 ISATAP 路由器解封装，再帮忙转发 IPv6 数据。

4. 典型实验



- **环境描述**

1) PC 是 ISATAP 主机，它是一台双栈 PC，这里我们使用的是一台 win7 系统的电脑做测试。电脑网卡的 IP 地址为 1.1.1.1/24，网关为 1.1.1.254，网关是 SW1 的 interface vlan10。

2) SW1 创建两个 VLAN：VLAN10 及 20，分别对应 PC 及 ISATAP 路由器。VLAN20 的 SVI 口 IP 为 2.2.2.254，是 ISATAP 路由器的默认网关。

3) ISATAP Router 的接口 IP 为 2.2.2.2。该 IPv4 地址在后续的 ISATAP 配置中使用到，ISATAP 主机就是通过这个 IP 找到 ISATAP Router 并与之建立 ISATAP tunnel。ISATAP Router 同时连接到了一个 IPv6 网络，这里我们用 loopback 模拟：2001:8888::8/64，用于后续的测试。

4) 最终的实验结果是首先 PC 要能够 ping 通 ISATAP Router 的 IPv4 地址也就是 2.2.2.2。然后 PC 与 ISATAP router 建立隧道并拿到 IPv6 地址，而且要能够 ping 通 20001:8888::8

- **设备配置**

PC1 的配置：

网卡配置 IP 地址 1.1.1.1/24，网关为 1.1.1.254

安装 IPv6 协议栈，此时 Win7 会自动产生一个 ISATAP 隧道虚拟接口：

```
隧道适配器 isatap.{0DB7233C-89B7-49DB-A8C0-D1AA005F4E6A}:
```

SW1 的配置：

```
vlan 10
vlan 20
interface fast0/1
  switchport access vlan 10
interface fast0/15
  switchport access vlan 20
interface vlan 10
  ip address 1.1.1.254 255.255.255.0
interface vlan 20
  ip address 2.2.2.254 255.255.255.0
```

Router 的配置：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ip address 2.2.2.2 255.255.255.0
  no shutdown
!
interface Tunnel1
  ip unnumbered   fastEthernet 0/0        !! 这个 IPv4 地址就是 ISATAP 隧道的目的地址
  ipv6 enable
  ipv6 address 2001:1111::/64 eui-64      !! 这个 IPv6 地址的前缀会被通告给 ISATAP 主机
  no ipv6 nd suppress-ra
  tunnel source fastEthernet 0/0
```

```
 tunnel mode ipv6ip isatap
!
interface loopback0
 ipv6 enable
 ipv6 address 2001:8888::8/64
!
ip route 0.0.0.0 0.0.0.0 2.2.2.254
```

注意 ISATAP 路由器的配置，关键部分在于 tunnel 的配置，tunnel 模式是 ipv6ip isatap 的，同时注意在 tunnel 这里配置的 IPV4 地址，就是对应的 ISATAP 主机上配置的那条 CMD 命令里 ISATAP 路由器的地址。我们这个实验演示的是 tunnel 直接用 fa0/0 的地址，当然，tunnel 也可以有自己的 IPv4 地址，只要保证 ISATAP 主机到这个 IPv4 地址路由可达就行。另外 tunnel 的 IPv6 地址，对应的前缀就是稍后要下发给 ISATAP 主机的前缀，这个实验中，我们 tunnel 的 IPv6 全局单播地址使用的是前缀+eui-64 的配置方式，这里的 eui-64 实际上指的就是前面我们介绍的那个特殊的 64bits 接口标识。

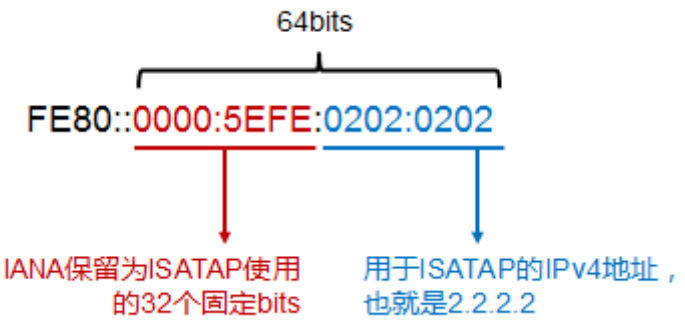
- **实验测试**

我们首先在路由器上查看一下：

R2#show ipv6 interface brief

```
FastEthernet0/0          [up/up]
Tunnel0                  [up/up]
    FE80::5EFE:202:202
    2001:1111::5EFE:202:202
```

注意，这里的 Linklocal 地址：FE80::5EFE:202:202 就是一个 ISATAP 格式的地址，最后的 64bits 是由 32bits 的 0000:5EFE 加上 32bits 的接口 IPv4 地址（这里是 2.2.2.2）构成的，如下图。而 IPv6 全局单播地址，也是使用 64bits 的接口标识构成的，当然，你也可以手工配置 IPv6 全局单播地址，不一定要使用接口标识。



接下去，我们在 ISATAP 主机上，CMD 模式下输入：

```
netsh interface ipv6 isatap set router 2.2.2.2
```

PC 就会开始发送 RS，报文如下：

```
Internet Protocol, Src: 1.1.1.1 (1.1.1.1), Dst: 2.2.2.2 (2.2.2.2)
Internet Protocol Version 6
⊞ 0110 .... = Version: 6
  .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
  .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 8
  Next header: ICMPv6 (0x3a)
  Hop limit: 255
  Source: fe80::200:5efe:101:101 (fe80::200:5efe:101:101)
  Destination: fe80::5efe:202:202 (fe80::5efe:202:202)
Internet Control Message Protocol v6
  Type: 133 (Router solicitation)
  Code: 0
  Checksum: 0xb7b8 [correct]
```

我们看到这个 RS 的 ICMPv6 报文外是 IPv6 的头，IPv6 的头外是 IPv4 的头。

注意外层 IPv4 的头，源是 1.1.1.1，目的是 2.2.2.2

然后内层 IPv6 的头，源是 ISATAP 主机的 Linklocal 地址，目的是 ISATAP 路由器的 Linklocal 地址

在路由器收到 RS 后回回应一个 RA：

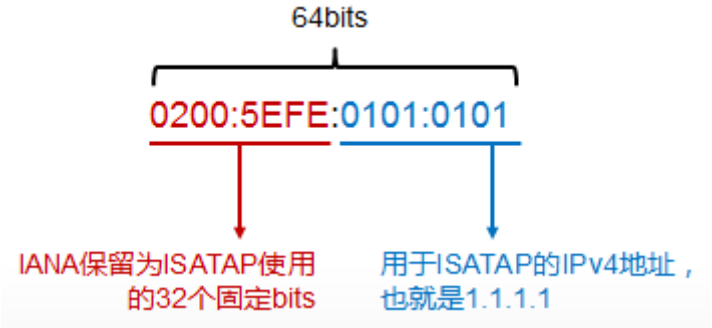
```
Internet Protocol, Src: 2.2.2.2 (2.2.2.2), Dst: 1.1.1.1 (1.1.1.1)
Internet Protocol Version 6
⊞ 0110 .... = Version: 6
  .... 1110 0000 .... .... .... .... .... = Traffic class: 0x000000e0
  .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 56
  Next header: ICMPv6 (0x3a)
  Hop limit: 255
  Source: fe80::5efe:202:202 (fe80::5efe:202:202)
  Destination: fe80::200:5efe:101:101 (fe80::200:5efe:101:101)
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x2830 [correct]
  Cur hop limit: 64
⊞ Flags: 0x00
  Router lifetime: 1800
  Reachable time: 0
  Retrans timer: 0
⊞ ICMPv6 Option (MTU)
⊞ ICMPv6 Option (Prefix information)
```

路由器回应的这个 RA 里，就有一个 ICMPv6 的 Option，其中就包含着 ISATAP 路由器的 IPv6 前缀。而 ISATAP 主机就可以根据这个前缀，结合自己的接口标识构建 IPv6 地址。

最终 PC 获取到的 IPv4 地址如下：

```
隧道适配器 isatap.{0DB7233C-89B7-49DB-A8C0-D1AA005F4E6A}:
   连接特定的 DNS 后缀 . . . . . . . :
   IPv6 地址 . . . . . . . . . . . . : 2001:1111::200:5efe:1.1.1.1
   本地链接 IPv6 地址. . . . . . . . : fe80::200:5efe:1.1.1.1%40
   默认网关. . . . . . . . . . . . . : fe80::5efe:2.2.2.2%40
```

我们看到 PC 首先根据自己本地配置的 IPv4 地址：1.1.1.1，生成 64bits 的接口 ID：



这个 64bits 的接口 ID ,与从 ISATAP 路由器获取到的 IPv6 全局单播地址前缀 2001:1111::的前 64bits，构成了 PC 的 IPv6 全局单播地址：2001:1111::200:5efe:1.1.1.1。

这个 64bits 的接口 ID，与 FE80::/10 构成了 PC 的 Linklocal 地址：fe80::200:5efe:1.1.1.1

同时，PC 将 ISATAP 路由器的 Linklocal 地址 fe80::5efe:2.2.2.2 设置为默认网关

当主机与其它 *IPv6* 主机进行通讯时，从隧道接口转发，将从报文的下一跳 *IPv6* 地址中取出 *IPv4* 地址作为 *IPv4* 封装的目的地址。如果目的主机在本站点内，则下一跳就是目的主机本身，如果目的主机不在本站点内，则下一跳为 *ISATAP* 路由器的地址。

我们最后再做一个测试，就是 ISATAP 主机去 ping 2008:8888::1。

```
⊞ Internet Protocol, Src: 1.1.1.1 (1.1.1.1), Dst: 2.2.2.2 (2.2.2.2)
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x0000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: 2001:1111::200:5efe:101:101 (2001:1111::200:5efe:101:101)
    Destination: 2001:8888::8 (2001:8888::8)
⊞ Internet Control Message Protocol v6
```

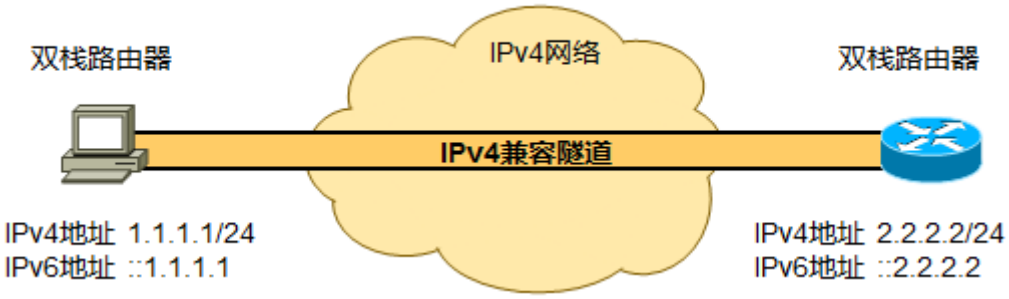
这个到达 ISATAP 的 loopback 的 IPv6 数据包，被套上一个 ISATAP 的 IPv4 隧道头，然后传给 ISATAP 路由器，由路由器进行下一步的 IPv6 转发。

7.3.6 自动 IPv4 兼容隧道

这种技术使用的比较少，是 IETF 的第一批转换机制之一。自动 IPv4 兼容隧道机制只允许两个双栈主机之间的 IPv6 数据在 IPv4 网络上进行自动隧道（无需手动配置）传输。这个机制允许 IPv4 网络上的孤立主机和另一个 IPv4 网络上的孤立主机之间自动启用隧道，从而实现 IPv6 数据在隧道中的传输。源和目的的 IPv6 地址的低 32 位地址表示隧道终点的源和目的 IPv4 地址。

自动 IPv4 兼容隧道机制使用的 IPv6 地址是“IPv4 兼容 IPv6 地址”

：：a.b.c.d/96（前面 96 个 0，后面跟上的 a.b.c.d 是 IPv4 地址）



适用于不经常性的 IPv6 节点连接需求。IPv4 兼容隧道是通过 Tunnel 虚接口实现的，如果一个 Tunnel 口的封装模式是 IPv4 兼容隧道，则只需配置隧道的源地址，而目的地址是在转发报文时，从 IPv6 报文的目的地址中取得的。从 IPv4 兼容隧道转发的 IPv6 报文的目的地址必须是 IPv4 兼容的 IPv6 地址，隧道的目的地址就是 IPv4 兼容地址的后 32 位。如果一个 IPv6 报文的目的地址不是 IPv4 兼容地址，则不能从 IPv4 兼容隧道转发出去。

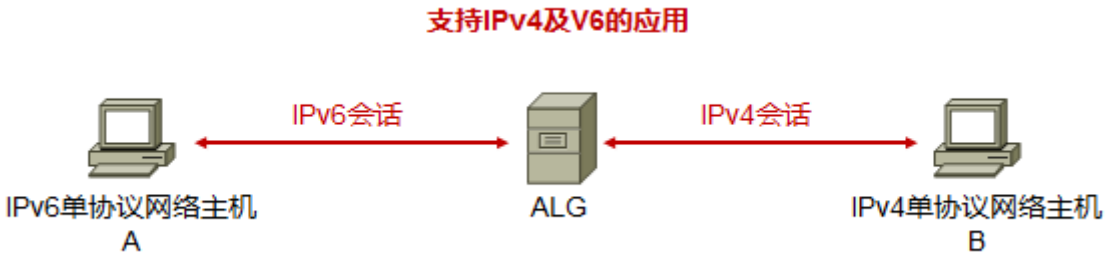
这种机制现在都不用了。

7.3.7 Teredo 隧道机制

8 IPv6 单协议网络到 IPv4 单协议网络的过渡机制

前面我们讨论的 V4 到 V6 的过渡机制，基本都是存在 V4、V6 共存的情况。那么如果是 V4 单协议和 V6 单协议网络有互访需求呢？

8.1 应用层网关 ALG



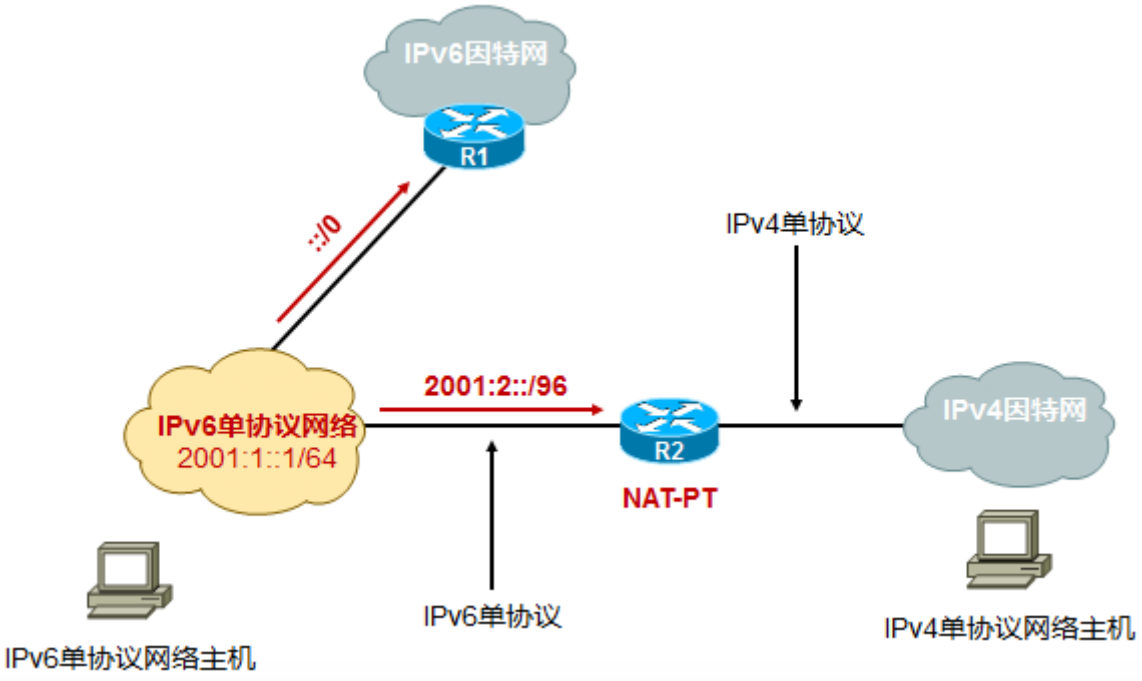
IPv6 单协议网络主机 A 与 ALG 之间维护着一条 IPV6 的会话，主机 A 可以向 ALG 发送 IPV6 数据。如此同时，ALG 与 IPV4 单协议网络主机 B 维护着一条 IPV4 会话。ALG 负责 IPV4 与 IPV6 的会话转换。这个 ALG 具有双栈的支持能力。举个例子，假设 ALG 是一台双栈，且可充当 SMTP　ALG 的服务器。那么 A 可以基于 IPV6 的 SMTP 发送电子邮件到这台邮件服务器。当然，B 可以使用 IPV4 会话去读取。

8.2 NAT-PT

8.2.1 机制概述

RFC2766、RFC2765。NAT-PT（网络地址转换-协议转换）是一种地址转换技术，它可以把 IPv6 地址转换成 IPv4 地址，反之亦然。NAT-PT 基于 RFC2766 中定义的无状态 IP/ICMP 转换器（SIIT）算法。SIIT 算法互译 IPv4 和 IPv6 数据包头部，也包括 ICMP 头部。

需要注意的是，在 IPv6 环境中，不建议像 IPv4 对待 NAT 的态度哪样，去使用 NAT。仅仅在 V4 单协议与 V6 单协议网络需要互相通信的时候，才建议使用 NAT-PT。



我们看上面的例子，对于 IPv6 单协议网络而言，首先它有访问 IPv6 因特网的需求，因此默认的 IPv6 流量全部交给 R1，另外，它可能还有访问 IPv4 因特网的需求，这时候，就需要借助 R2 这台 NAT-PT 设备。这台 NAT-PT 设备首先肯定是双栈，其次它能够将来自 IPv6 网络的流量，转换成 IPv4 流量放进 V4 网络，反之亦然。

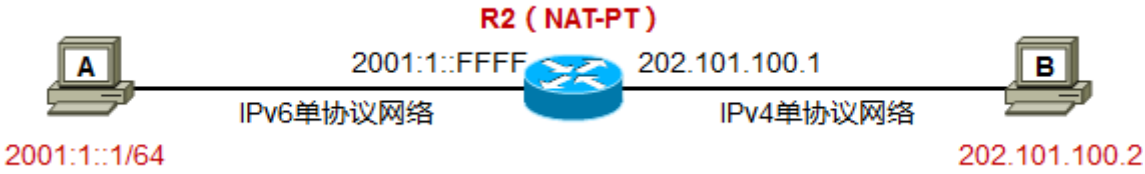
2001:2::/96，这个长度为 96 位的前缀是我们为了 NAT-PT 操作预定义的前缀，前缀可以自定义，但是长度必须是 96bits。这个预定义的前缀非常关键，相当于为 R2 右侧的 IPv4 网络预定义一个 IPv6 的空间。举个非常简单的例子，IPv6 单协议网络中，如果有 PC 想访问 IPv4 单协议网络中的 PC，那么咋访问？目的地址是啥？总不能让一个 IPv6 only 的 PC 去 ping 一个 IPv4 地址吧？因此这个/96 的预定义前缀就派上用场了，可以理解为它就是在这种场景中，为右侧的 IPv4 only 网络所服务的一个预定义的 IPv6 地址空间。

在 IPv6 单协议网络中产生的，去往 2001:2::/96 这个目的地的流量被路由到 R2 也就是 NAT-PT 设备，然后数据包中的 IPv6 地址被转换为 IPv4 地址并传送给 IPv4 因特网中的 IPv4 单协议节点。

8.2.2 NAT-PT 配置及原理

8.2.2.1 静态 NAT-PT

1. **静态 NAT-PT（单向）**

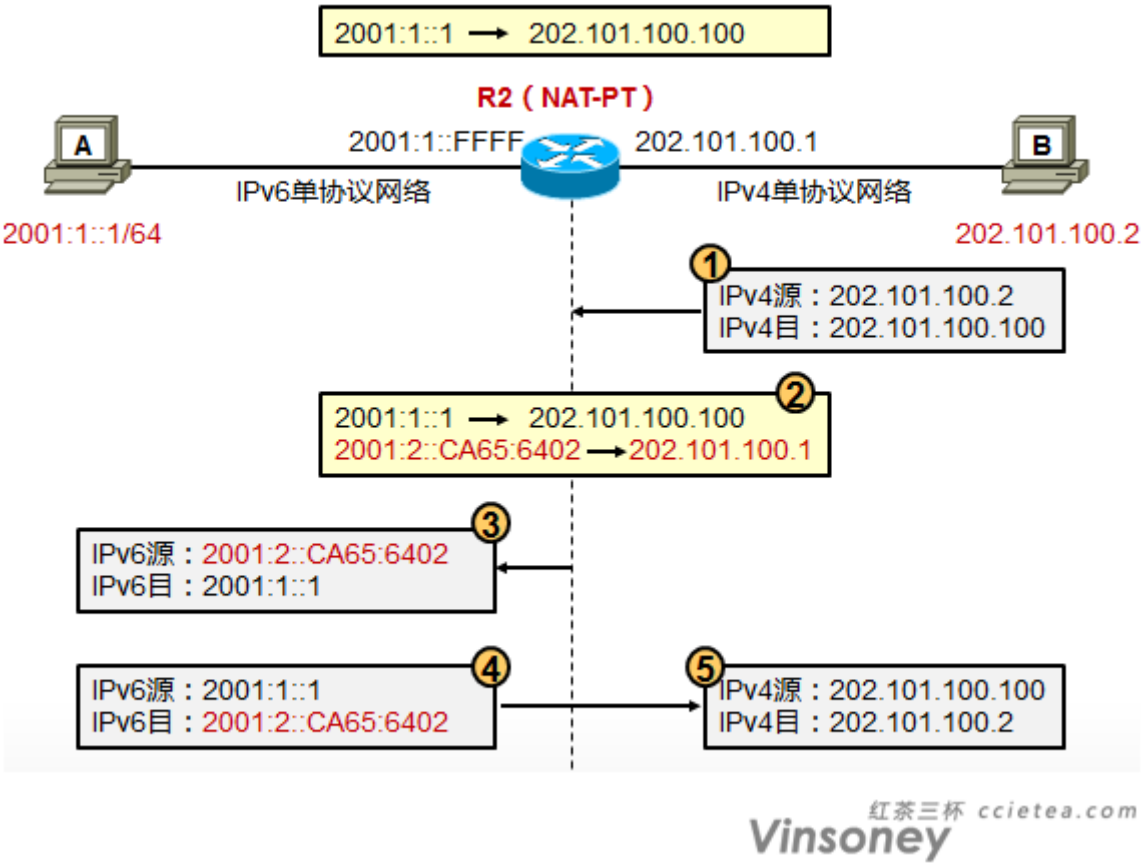


A 和 B 的配置都极其简单，这里就不说了

R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable                                  !! 连接 A 的接口
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
ipv6 nat prefix 2001:2::/96                     !! 是一个为 NAT-PT 预留的池
ipv6 nat v6v4 source 2001:1::1 202.101.100.100      !! 相当于将 2001:1::1 这个 IPv6 的节点，“告知”给 IPv4 单协议网络中的用户知道，可以以 202.101.100.100 的方式访问。
```

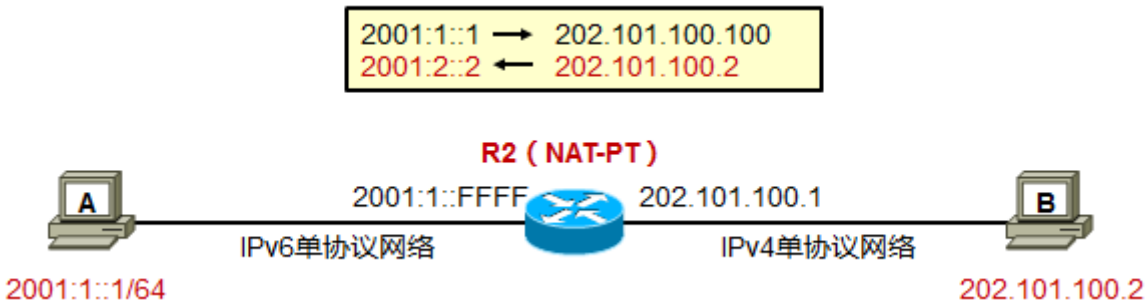
在上述配置中 ,我们将 A 节点 ,也就是 2001:1::1 映射到了 IPv4 网络 ,映射到 202.101.100.100 这个 IPv4 地址。这样一来 B 可以主动去访问 A，例如 B 去 ping 202.101.100.100，能够 ping 通，这个过程如下：



1) 首先 B 去 ping 202.101.100.100，数据包如图所示。
2) 数据包到达 R2 后，R2 本地是已经存在了一个映射：2001:1::1 映射到 202.101.100.100
   由于存在这个映射，因此 R2 将 B 发过来的这个数据包的目的地址 202.101.100.100 替换成 2001:1::1，同时 B 的源地址是个 IPv4 地址 ,咋办呢？还记得我们配置了 NAT-PT 预留的前缀么？也就是 2001:2::/96 这个玩意儿 ,R2 将 B 的 IPv4 地址映射到这个前缀上 ,构成一个临时的 IPv6 地址 :2001:2:CA65 :6402，同时在本地生成一条新的映射条目：**2001:2::CA65:6402 映射到 202.101.100.1**，如上图所示。
3) R2 将原始的 IPv4 数据包的包头替换成 IPv6 的包头，然后转给 A。
4) A 回包，数据包送给 R2，R2 由于已经有了 2001:2::CA65:6402 到 202.101.100.1 的映射，因此 R2 将 IPv6 包头替换成 IPv4 包头，然后再转发给 B。

注意在此时，A 是可以去主动访问 B 的，也就是说 A 可以主动 ping 2001:2::CA65:6402 这个临时的地址来达到访问 B 的目的。但是如果我们在 R2 上去 clear ipv6 nat translation * ，如此 2001:2::CA65:6402 到 202.101.100.1 的映射条目就被清空了，A 就无法主动访问 B 了，只能 B 先主动访问 A。

2. 静态 NAT-PT（双向）



R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
ipv6 nat prefix 2001:2::/96
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
ipv6 nat v6v4 source 2001:1::1 202.101.100.100
```

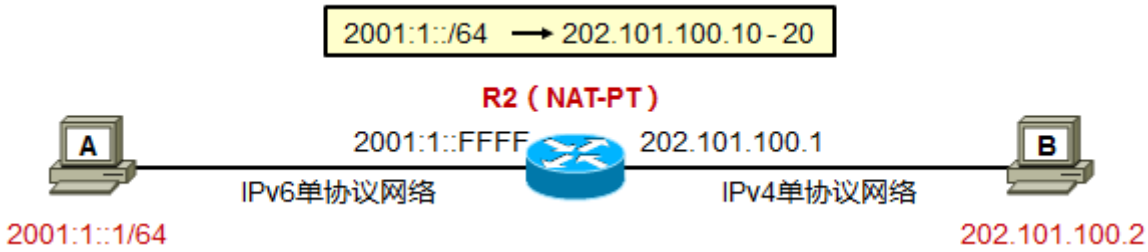
如此一来，A 主动发起访问连接到 B，或者 B 主动发起访问到 A 都可以。

NAT-PT#show ipv nat translations

Prot	IPv4 source IPv4 destination	IPv6 source IPv6 destination
---	---	---
	202.101.100.2	2001:2::2
---	202.101.100.100 202.101.100.2	2001:1::1 2001:2::2
---	202.101.100.100 ---	2001:1::1 ---

8.2.2.2 动态 NAT-PT

1. **V6 可以主动访问 V4（使用 V4 地址池）**



R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:1::FFFF/64
   ipv6 nat
!
interface FastEthernet1/0
   ip address 202.101.100.1 255.255.255.0
   ipv6 nat
!
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
ipv6 nat prefix 2001:2::/96
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
```

来分解一下关键命令：

```
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
```

上面的命令是定义允许被 IPv6 nat 的源地址，用一个 IPv6 ACL 进行匹配。

```
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
```

上面的命令是，创建一个供 v6tov4 使用的地址池，这个地址池当然是 IPv4 的地址池，当 IPv6 only 的用户，如 A 要访问 IPv4 only 网络的时候，就从池中取一个空闲的 IPv4 地址。

```
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
```

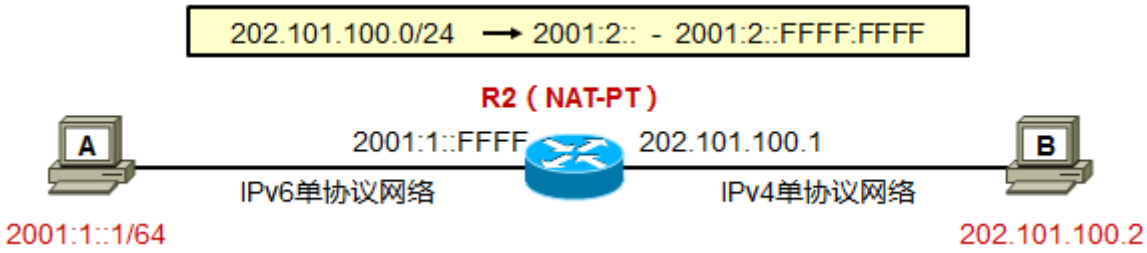
将 ACL 所允许的 IPv6 网络与这个 v6v4 地址池进行关联

```
ipv6 nat v4v6 source 202.101.100.2 2001:2::2
```

最后这条命令是将 202.101.100.2 这个 IPv4 网络的主机“放进来”到 IPv6 网络，使得 IPv6 only 的用户有访问目标，否则，你让 IPv6 only 的用户用什么目标地址去访问 IPv4 only 网络？

完成上述配置后，A 即可主动发起访问到 B 了， 使用目标地址 2001:2::2 即可访问 B

2. V4 可以主动访问 V6（使用 V6 地址池）



R2 的配置如下：

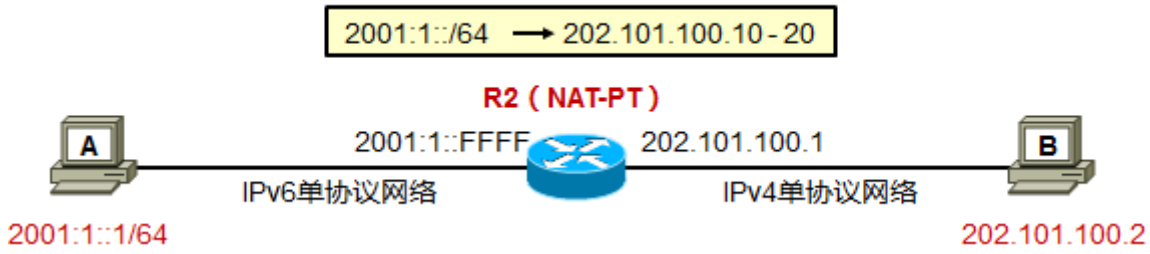
```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:1::FFFF/64
  ipv6 nat
!
interface FastEthernet1/0
  ip address 202.101.100.1 255.255.255.0
  ipv6 nat
!
access-list 1 permit 202.101.100.0 0.0.0.255
ipv6 nat v4v6 pool v4v6-pool 2001:2:: 2001:2::FFFF:FFFF prefix-length 96
ipv6 nat v4v6 source list 1 pool v4v6-pool
ipv6 nat prefix 2001:2::/96
ipv6 nat v6v4 source 2001:1::1 202.101.100.111
```

思路和 V6 访问 V4 是一样的。

这里我们用一个 access-list 1 匹配 202.101.100.0/24 网络，同时将其与 v4v6-pool 这个 IPv6 地址池做了捆

绑。使得 B 访问 IPv6 网络的时候，可以从池中拿 IPv6 地址。另外，为了让 V4 网络访问 V6 网络有目标，还做了一条静态映射，将 2001:1::1 映射到 202.101.100.111，这样，B 就能够使用 202.101.100.111 这个 IP 来访问 IPv6 主机 A。

3. IPv4-Mapped NAT-PT



回顾一下前面 IPv6 only 网络访问 IPv4 only 网络使用 V4 地址池的情况。

V4 地址池使得 V6 用户在穿越 NAT-PT 设备之后，可以从池中拿一个地址当做源地址。但是，V6 用户如何访问 V4 网络呢？ 总不能让 V6 only 主机去 ping 202.101.100.2 吧？因此，不得不在配置上增加命令：

ipv6 nat v4v6 source 202.101.100.2 2001:2::2

这条命令，将 202.101.100.2 这个 V4 主机映射到 V6 网络，使得 V6 网络用户能够使用地址 2001:2::2 去访问这个 V4 主机。那么如果我们这样的 V4 主机有很多呢？难道需要手工去添加静态映射么？

当然不用，使用 IPv4-mapped 功能即可，机制非常简单，我们在 R2 上修改配置，关键配置如下：

ipv6 access-list v4map permit 2001:1::/64 2001:2::/96

ipv6 nat prefix 2001:2::/96 v4-mapped v4map

首先用一条 IPv6 ACL 来匹配需要进行 IPv4-mapped 的流量，上面的 ACL 源是 IPv6 only 网络，目的是保留给 NAT-PT 的 IPv6 前缀。然后使用 **ipv6 nat prefix 2001:2::/96 v4-mapped v4map 命令**关联这条 ACL。这样一来当 A 访问 B 时，可以直接使用 **2001:2::CA65:6402** 这个地址来访问处于 V4 网络的 B，注意，这里前缀 2001:2::/96 是保留给 NAT-PT 的 IPv6 前缀，后面的 CA65:6402 将被翻译成 IPv4 地址，也就是 202.101.100.2。因此 R2 收到这个 IPv6 数据包的时候，发现数据包匹配上了 v4map 这个 IPv6ACL，因此将目的 IPv6 地址的最后 32bits 翻译成 IPv4 格式（这就是 IPv4 目的地址），然后从 V4 地址池中取出一个地址替换掉 IPv6 地址。完美。

R2 的配置如下：

```
ipv6 unicast-routing
!
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:1::FFFF/64
  ipv6 nat
!
```

```
interface FastEthernet1/0
  ip address 202.101.100.1 255.255.255.0
  ipv6 nat
!
Ipv6 access-list ipv6only-network permit 2001:1::/64 any
ipv6 nat v6v4 pool v6v4-pool 202.101.100.10 202.101.100.20 prefix-length 24
ipv6 nat v6v4 source list ipv6only-network pool v6v4-pool
ipv6 access-list v4map permit 2001:1::/64 2001:2::/96
ipv6 nat prefix 2001:2::/96 v4-mapped v4map
```

9 IPv6 路由选择

9.1 IPv6 路由选择基础

```
Router# Show ipv6 route
```

使用该命令查看路由器 IPv6 路由表

9.2 静态路由

```
Router(config)# ipv6 route  目的 IPv6 网络/前缀长度 { 出接口 | 下一跳 IP }
```

配置 IPv6 静态路由

在 IPv6 规范中，不推荐使用可聚合全球单播或本地站点地址作为下一跳地址，如果这样做，ICMPv6 重定向消息就不会工作。因此使用 Linklocal 地址作为下一跳 ,在某些场合可能更为推荐 ,毕竟 linklocal 地址稳定且长久不变。在配置 linklocal 地址作为下一跳 IP 时，必须关联路由器上相应的接口。例如：

```
ipv route 2222::/64 fastEthernet 0/0 FE80::CE00:1AFF:FEE4:0
```

9.3 RIPng

9.3.1 基础知识

- 使用 UDP521（源 UDP 端口及目的 UDP 端口都是 521）
- 最大跳数依然是 15 跳
- 携带的网络前缀是 128bis，而不是 32bits
- 下一跳地址是 128bits
- 使用 Linklocal 地址作为协议数据包的源地址，发送 RIPng 消息到邻接 RIPng 路由器。使用 FF02::9 作为 RIPng 更新的目标地址
- Distance 默认 120

9.3.2 配置命令

1. **激活 RIPng 进程及接口**

```
router(config)# ipv6 router rip name
router(config)# interface serial0/0
router(config-if)# ipv6 router rip name enable
```

【注意】进程名本地有效，若 Router 两接口，分别启用 RIPng 用的是两个不同的进程名，则两进程互相独立。

2. **默认路由的传递**

```
Ipv6 rip 1 default-information originate    //接口模式下，重发布默认路由进 RIP 进程( 本地无需静态默认路由 )
Ipv6 rip 1 default-information only          //接口模式下，只发布默认路由，禁止发布其他路由信息
```

3. **调整 RIPng 进程**

```
Router(config-rtr)# distance x
!! 修改 RIPng 管理距离，默认 120

Router(config-rtr)# distribute-list prefix-list xxx {in | out} [interface]
```

!! 对某个 RIPng 接口发送或接收路由更新动作执行分发列表。必须跟 in/out 方向，接口可选，如果不配置接口，则为所有接口生效。

```
Router(config-rtr)# poison-reverse
```

执行毒性逆转，默认关闭。

如果同时启用水平分割和毒性逆转，则只有水平分割有效。

```
Router(config-rtr)# split-horizon
```

执行水平分割处理

```
Router(config-rtr)# port  x  multicast-group  X:X:X:X::X
```

修改 RIP 使用的 UDP 端口及组播组地址。默认情况下是 UDP521 及 FF02::9

```
Router(config-rtr)# timers update expire holddown garbage-collect
```

修改计时器。默认 update 30S，expire 是超时时间默认 180S。

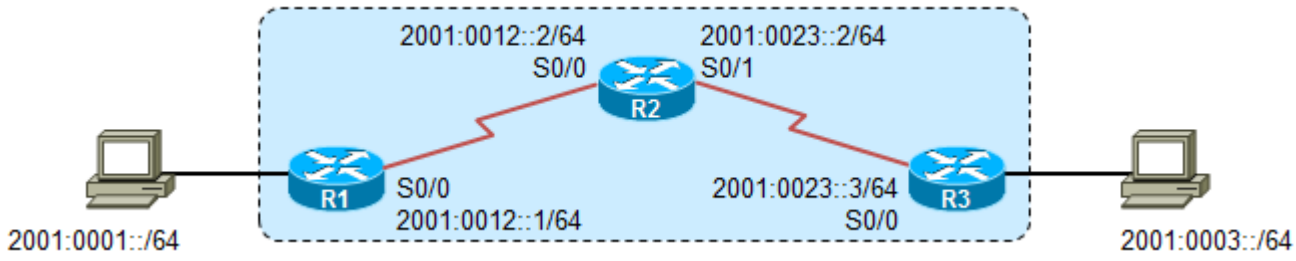
holddown 默认为 0，也就是不使用抑制

garbage-collect 默认 120S

```
Router(config-if)# ipv6 rip xxx metric-offset  ?
```

!! 注意，是在接口模式下配置的，用来设置 metric 累加量。可选值 1-16

9.3.3 配置示例



1. **实验需求**
   - 完成基本 IP 配置
   - 三台路由器完成 RIPng 的配置，要求全网互通
   - PC 无需收到多余的 RIPng 组播消息

- 假设 R1 为网络结构中的末梢节点，因此，在 R2 上，将多余的路由过滤掉，仅下放 R3 的 Fa1/0 口路由

2. 实验配置

R1 的配置如下：

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
  ipv6 rip RIPprocess enable              !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0001::FFFF/64
  ipv6 rip RIPprocess enable
```

R2 的配置如下：

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::2/64
  ipv6 rip RIPprocess enable
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:23::2/64
  ipv6 rip RIPprocess enable
```

R3 的配置如下：

```
Ipv6 unicast-routing
!
```

```
ipv6 router rip RIPprocess
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:23::3/64
  ipv6 rip RIPprocess enable              !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0003::FFFF/64
  ipv6 rip RIPprocess enable
```

完成上述配置后，全网路由就通了。

PC 的配置非常简单，使用无状态自动配置获取地址，例如 PC1（用路由器模拟）：

```
interface FastEthernet0/0
  no ip address
  ipv6 enable
  ipv6 address autoconfig default
```

加上 default 关键字会使得该接口在通过无状态自动配置拿到地址后，安装一条默认路由。

PC1#sh ipv int

```
FastEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE03:1AFF:FEC4:0
  Global unicast address(es):
    2001:1::CE03:1AFF:FEC4:0, subnet is 2001:1::/64 [PRE]
      valid lifetime 2591944 preferred lifetime 604744
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FFC4:0
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
```

```
Default router is FE80::CE00:1AFF:FEC4:10 on FastEthernet0/0
```

好那么继续 ,现在 R1 及 R3 的以太网接口都激活了 RIPng ,于是这两接口都会向 LAN 中去组播 RIPng 报文，而这个动作实际上是没有意义的，因此，我们可以将这个以太网口 passive 掉。但是很可惜，RIPng 里没有定义 passive-interface 的机制。

因此，我们可以在 R1、R3 上，将以太网口（所在网段）重发布进 RIPng。

R1 的配置修改如下（R2 的配置类似）:

```
Ipv6 unicast-routing
!
ipv6 router rip RIPprocess
  redistribute connected                !! 重发布直连
exit
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
  ipv6 rip RIPprocess enable            !! 激活 RIP
Interface fast1/0
  ipv6 enable
  ipv6 address 2001:0001::FFFF/64
  no ipv6 rip RIPprocess enable         !! 不激活 RIP，而是采用重发布的方式
```

假设 R1 为网络结构中的末梢节点，因此，在 R2 上，将多余的路由过滤掉，仅给 R1 通告 R3 的 Fa1/0 口路由。我们在 R2 上过滤掉特定的路由：

R2 的配置增加如下：

```
ipv6 prefix-list test seq 5 permit 2001:3::/64
Ipv6 router rip RIPprocess
  distribute-list prefix-list test out serial 0/0
```

为了让 R1 能够正常访问网络其他地方，一劳永逸，给 R1 下发一条 RIPng 的默认路由：

R2 的配置增加如下：

```
Interface serial0/0
  ipv6 rip RIPprocess default-information originate
```

9.4 IPv6 IS-IS

9.4.1 基础知识

现有的 CISCO IOS 的 IS-IS 支持 IPv6，相关 RFC：1195

更新内容主要是新添加的承载有关 IPv6 路由信息的两个 TLV。

- **IPv6 可达性：** 这个 TLV 定义了如 IPv6 路由选择前缀、度量值、及一些选项比特等。分配给 IPv6 可达性 TLV 的十进制值是 236,
- **IPv6 接口地址：** 这个 TLV 包含一个 IPv6 接口地址(128bits),分配给 IPv6 接口地址 TLV 的十进制值是 232，对于 IS HELLO PDU ,这个 TLV 必须包含 Linklocal 地址 ,但是对于 LSP ,TLV 必须包含非 Linklocal 地址。

9.4.2 IPv6 IS-IS 网络设计

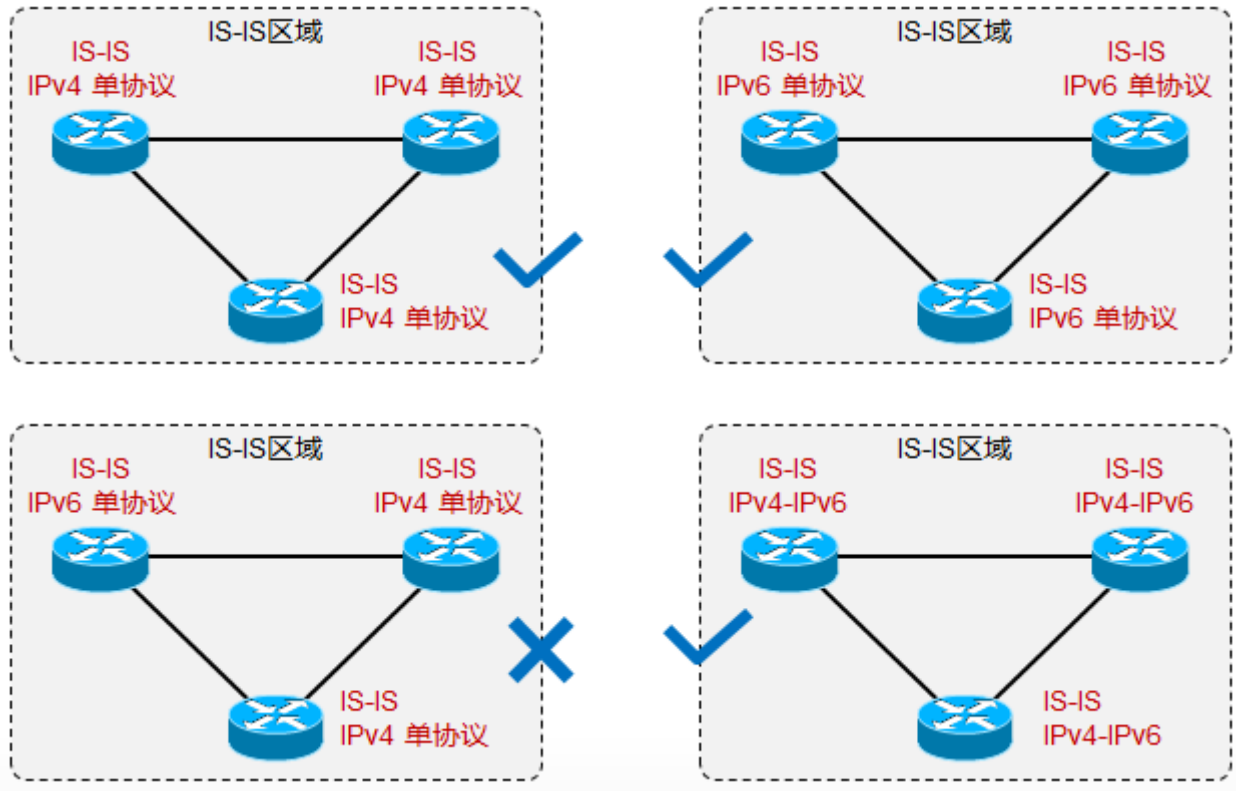
- **层次结构的考虑**

  IS-IS 域基于两层结构 , Level1 及 Level2 , 同时 L1-router、L2-router 及 L1L2-router 的概念 , 通 IPv4 IS-IS。

- **IS-IS 邻接关系的考虑**

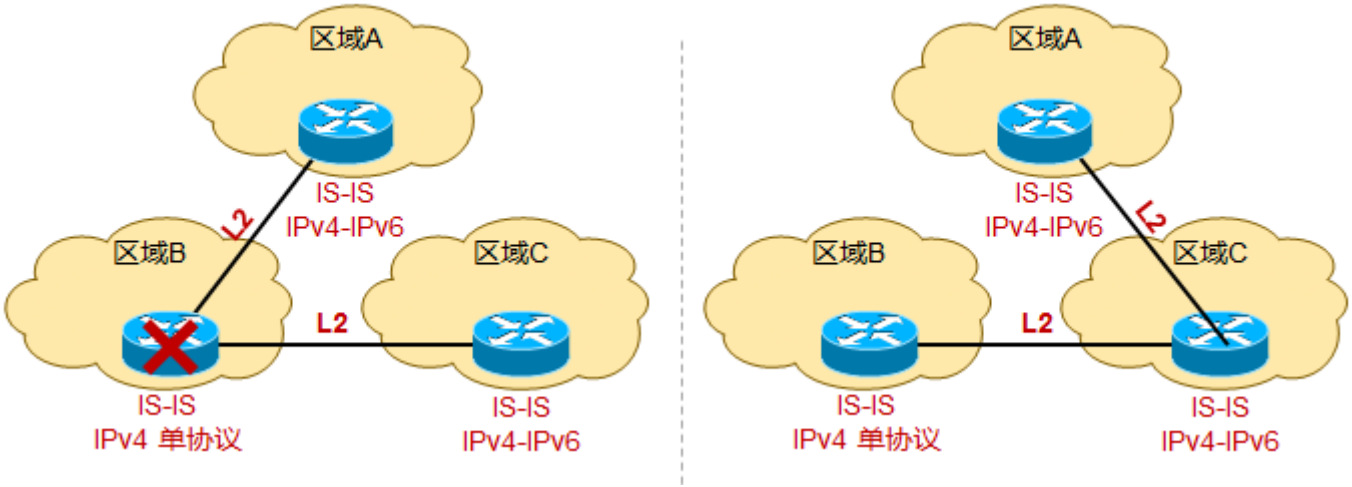
  另外 IS-IS 邻接路由器有个非常重要的因素要考虑。IPv4 或 IPv6 单协议环境下，IS-IS 的邻接关系及信息的交互都不会有问题。但是在所有邻接 IS-IS 路由器上启用两种协议（IPv4 及 IPv6）时，就要谨慎了。在这种特定的情况下，在所有的 IS-IS 邻接路由器上必须同时启用两种协议，否则 IS-IS 路由器将不再维持与其所有 IS-IS IPv4 邻居的邻接关系。事实上，当 Level1 或 Level1-2 IS-IS router 在启用邻接关系检查时，会检查从其 IS-IS 邻居发来的 Hello 消息，如果邻居使用不同的协议集，就拒绝与其形成邻接关系。

  IPv4 单协议网络 IS-IS 路由器向 IPv6-IPv4 路由器的过渡是非常关键的，如果在过渡过程中没有关掉邻接关系检查(关掉邻接关系检查 ,用 no adjacency-check),IPv4 单协议王丽萍 IS-IS 路由器将拒绝与 IPv6-IPv4 路由器形成邻接关系。成功过渡之后，当所有 IS-IS 路由器同时支持两种协议时，就可以去掉 no adjacency-check 命令了。



- **Level2 router 因素**

  在任何 IS-IS 网络中，负责区域间路由选择的 L2 router 必须是连续的，同样，对于 IPv4 单协议环境、IPv6 单协议环境或者是 IPv4-IPv6 环境，L2 router 都必须是连续的。否则会产生路由黑洞。



上面这个图，左侧的网络就有问题，IPv4 单协议的 L2 router 不支持 IPv6，因此破坏了 IPv6 L2 router 的连续性。右侧的网络则是正确的结构。

9.4.3 配置命令

```
Router(config)# router isis [tag]
```

定义 IS-IS 进程

```
Router(config-router)# address-family ipv6 [unicast]
```

进入 IPv6 地址簇

```
Router(config-router)# default-information originate [route-map x]
```

产生一条默认路由

```
Router(config-router)# no adjacency-check
```

在网络从 IPv4 单协议网络过渡到 IPv4-IPv6 IS-IS 路由器的过程中，该命令维持使用不同协议集的 IS-IS 路由器之间的 IS-IS 邻接关系。它可以防止使用不同协议集的 IS-IS 路由器执行 hello 检查而丢失 IS-IS 邻接。在网络切换或者过渡完成之后，可以将 adjacency-check 配置回去。

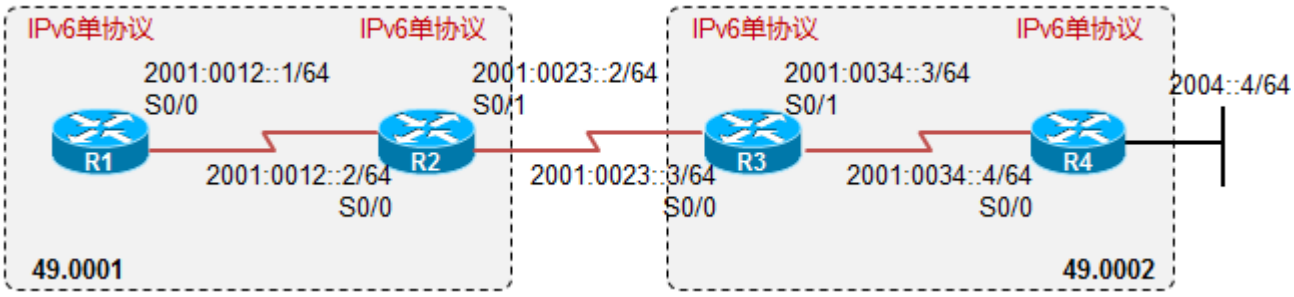
```
Router(config-router)# net XX.XXXX. ... .XXX.XX
```

为本路由器配置一个 NET 地址，这与 IPv4 IS-IS 的概念完全一样

```
Router(config-if)# ipv6 router isis
```

在接口上激活 IPv6 IS-iS

9.4.4 配置示例



1. **实验需求**
   - IS-IS 网络的规划如拓扑所示
   - R4 上的 Loopback 口使用重发布的方式引入 IS-IS
   - 实现全网互通

2. **实验配置**

注意，如果使用虚拟机配置，要主机在 serial 接口上先手工配置 linklocal 地址

R1 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0001.0000.0000.0001.00                !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:12::1/64
 ipv6 address FE80::FFFF:FE10:1 link-local      !!GNS 模拟器环境下，serial 口的 Linklocal 地址冲突，因此手工指定
 ipv6 router isis                               !! 接口激活 IS-IS
 isis circuit-type level-1
```

R2 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0001.0000.0000.0002.00                  !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:12::2/64
 ipv6 address FE80::FFFF:FE10:2 link-local
 ipv6 router isis
 isis circuit-type level-1
interface Serial0/1
 ipv6 enable
 ipv6 address 2001:23::2/64
 ipv6 address FE80::FFFF:FE10:3 link-local
 ipv6 router isis
 isis circuit-type level-2-only
```

R3 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0002.0000.0000.0003.00                !! NET 实体地址
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:23::3/64
 ipv6 address FE80::FFFF:FE10:4 link-local
 ipv6 router isis
 isis circuit-type level-2-only
interface Serial0/1
 ipv6 enable
 ipv6 address 2001:34::3/64
 ipv6 address FE80::FFFF:FE10:5 link-local
 ipv6 router isis
 isis circuit-type level-1
```

R4 的配置如下：

```
ipv6 unicast-routing
!
router isis
 net 49.0002.0000.0000.0004.00                !! NET 实体地址
 address-family ipv6
    redistribute connected level-1
 exit-address-family
!
interface Serial0/0
 ipv6 enable
 ipv6 address 2001:34::4/64
 ipv6 address FE80::FFFF:FE10:6 link-local
 ipv6 router isis
```

```
 isis circuit-type level-1
interface Loopback0
 ipv6 address 2004::4/64
 ipv6 enable
```

在 R2 上简单的查看一下：

R2#sh isis neighbors

```
System Id      Type Interface IP Address      State Holdtime Circuit Id
R3             L2   Se0/1                     UP    27       00
R1             L1   Se0/0                     UP    25       00
```

R2#sh isis neighbors detail

```
System Id      Type Interface IP Address      State Holdtime Circuit Id
R3             L2   Se0/1                     UP    22       00
  Area Address(es): 49.0002
  SNPA: *HDLC*
  IPv6 Address(es): FE80::FFFF:FE10:4
  State Changed: 00:10:04
  Format: Phase V
R1             L1   Se0/0                     UP    29       00
  Area Address(es): 49.0001
  SNPA: *HDLC*
  IPv6 Address(es): FE80::FFFF:FE10:1
  State Changed: 00:11:03
  Format: Phase V
```

继续在 R4 上进行简单的查看：

R4#sh isis database

```
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R3.00-00              0x0000000E   0xA440        835               1/0/0
R4.00-00            * 0x0000000B   0xF895        729               0/0/0
IS-IS Level-2 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R4.00-00            * 0x00000005   0x7768        850               0/0/0
```

R4#sh isis da R4.00-00 l1 detail

```
IS-IS Level-1 LSP R4.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R4.00-00              * 0x0000000B   0xF895        696               0/0/0
  Area Address: 49.0002
  NLPID:        0x8E
  Hostname: R4
  IPv6 Address: 2001:34::4
  Metric: 10         IPv6 2001:34::/64
  Metric: 10         IS R3.00
  Metric: 0          IPv6 2004::/64
```

9.4.5 高级配置

1. **在 GRE 隧道之上配置 IPv6 IS-iS**

   因为 IS-IS 运行在数据链路层之上，也就是说，运行在链路层协议之上，并且需要 CLNP 的支持。所以如果在通过配置隧道连接的远端 IPv6 网络上不能使用 IPv6 IS-IS。配置隧道就是静态的 IPv6 over IPv4 隧道，在 IPv4 的数据包之上承载 IPv6 数据包。

   如果希望用隧道技术连接两个 IPv6 IS-IS 岛屿，那么需使用 GRE 隧道。

9.5 OSPFv3（RFC2740）

9.5.1 基础知识

OSPF V2 与 V3 有相似之处：

- 使用相同的基本数据包类型，如 Hello、DBD、LSR、LSU、和 LSA
- 邻居发现和邻接关系形成机制是相同的
- 支持在遵循 RFC 的 NBMA 和点到多点拓扑模式上的 OSPV3 操作。OSPFv3 也支持 CISCO 的其他模式，例如点到点等

- LSA 泛洪和衰老机制是相同的

OSPF V2 与 V3 的不同之处：

- OSPFv3 运行在链路上，每条链路有可以有多个 OSPFv3 实例
- OSPFv3 的 RouterID 还是 32bits 的，如果路由器上没有配置 IPv4 地址，则在 OSPFv3 进程中需手工指定 RouterID
- LinkID 在 OSPFv3 中仍然是 32bits 的
- OSPFv3 使用 IPv6 Linklocal 地址标识 OSPFv3 邻接的邻居
- 新的 LSA 类型：
  - 链路 LSA（LSA 类型 0x0008）: 每条链路都有一个链路 LSA，这个新类型提供了路由器的本地链路地址，并列出了链路的所有 IPv6 前缀
  - 区内前缀 LSA（LSA 类型 0x2009）:
- OSPFv3 使用组播地址：FF02::5 及 FF02::6
- 在认证这块，不使用 OSPFv2 中定义的认证方法。而是使用 AH 及 ESP 扩展头部作为安全机制

9.5.2 OSPFv3 报文

IPv6 报头中的协议号仍然是 89；

OSPFv3 的报头将 v2 中的认证相关字段都去除了，这是因为 ipv6 报头已经有了很好的认证功能。

1. **Option 字段**

变成了 24 位，出现在 HELLO 包、DBD 包，以及 router LSA、network LSA、interAreaRouter LSA、Link LSA 中。这些字段都有特殊的用途。

```
Options: 0x000033 (DC, R, E, V6)
  .... .... .... 0... .... .... = F: F is NOT set
  .... .... .... .0.. .... .... = I: I is NOT set
  .... .... .... ..0. .... .... = L: L is NOT set
  .... .... .... ...0 .... .... = AF: AF is NOT set
  .... .... .... .... ..1. .... = DC: DC is SET
  .... .... .... .... ...1 .... = R: R is SET
  .... .... .... .... .... 0... = N: N is NOT set
  .... .... .... .... .... .0.. = MC: MC is NOT set
  .... .... .... .... .... ..1. = E: E is SET
  .... .... .... .... .... ...1 = V6: V6 is SET
```

DC	是否支持按需电路
R	路由器位，表示通告着是否为一台路由器，如果为 0 则该通告者不能路由数据。经过该通告者的路由不能纳入路由计算

N	一个接口所属区域为 NSSA 时，为 1
MC	描述路由器是否运行了 MOSPF
E	是否能处理 AS-external-lsa，一般是 ASBR
V6	表示路由器是否在运行 IPV6，运行 OSPFv3 的路由器发出来的报文，该字段一般都置 1

更多的 IPv6 OSPF 全面解析，请见《红茶三杯 OSPF 技术笔记》

9.5.3 OSPFv3 配置

Router(config)# ipv6 router ospf *process-id*

启动一个 OSPFv4 进程

Router(config-router)# router-id x.x.x.x

指定一个 32bits 的 Router-ID

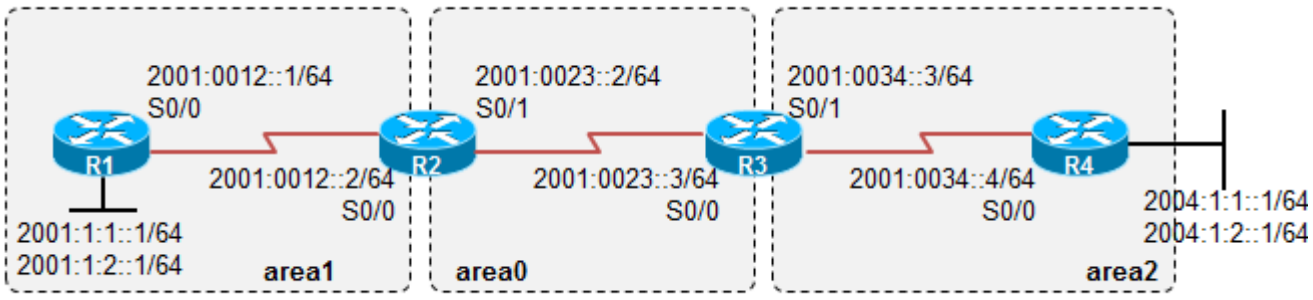
Router(config-router)# area *area-id* **range** *xxxxx*

路由汇总

Router(config-if)# ipv6 ospf *process-id* **area** *area-id*

在接口上激活 OSPFv3

9.5.3.1 基本配置示例



1. **实验需求**
   - 完成基本 IP 配置
   - 根据如果所示运行 OSPFv3
   - 对 area1 内的 IPv6 前缀进行汇总，减少路由条目

- 对 R4 重发布进来的 IPv6 前缀进行汇总，减少路由条目

2. 实验配置

R1 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
Interface loopback0
  ipv6 enable
  ipv6 address 2001:1:1::1/64
  ipv6 address 2001:1:2::1/64
  exit
!
Ipv6 router ospf 100
  router-id 1.1.1.1
  exit
interface serial0/0
  ipv6 ospf 100 area 1
interface loopback0
  ipv6 ospf 100 area 1
```

R2 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:12::2/64
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:23::2/64
!
Ipv6 router ospf 100
  router-id 2.2.2.2
```

```
  exit
interface serial0/0
  ipv6 ospf 100 area 1
interface serial0/1
  ipv6 ospf 100 area 0
```

R3 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:23::3/64
interface Serial0/1
  ipv6 enable
  ipv6 address 2001:34::3/64
!
Ipv6 router ospf 100
  router-id 3.3.3.3
  exit
interface serial0/0
  ipv6 ospf 100 area 0
interface serial0/1
  ipv6 ospf 100 area 2
```

R4 的配置如下：

```
Ipv6 unicast-routing
!
interface Serial0/0
  ipv6 enable
  ipv6 address 2001:34::4/64
interface loopback0
  ipv6 enable
  ipv6 address 2004:1:1::1/64
  ipv6 address 2004:1:2::1/64
!
Ipv6 router ospf 100
```

```
 router-id 4.4.4.4
 redistribute connected                !!重发布直连路由
 exit
interface serial0/0
 ipv6 ospf 100 area 2
```

到此为止，初步的配置都已完成了。

R1 上看看：

R1#show ipv6 ospf neighbor detail

```
 Neighbor 2.2.2.2
     In the area 1 via interface Serial0/0
     Neighbor: interface-id 4, link-local address FE80::FFFF:FE10:2
     Neighbor priority is 1, State is FULL, 6 state changes
     Options is 0x6423B8FD
     Dead timer due in 00:00:31
     Neighbor is up for 00:05:46
     Index 1/1/1, retransmission queue length 0, number of retransmission 0
     First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
     Last retransmission scan length is 0, maximum is 0
     Last retransmission scan time is 0 msec, maximum is 0 msec
```

R1#sh ipv6 route ospf

```
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI   2001:23::/64 [110/128]
      via FE80::FFFF:FE10:2, Serial0/0
OI   2001:34::/64 [110/192]
      via FE80::FFFF:FE10:2, Serial0/0
OE2   2004:1:1::/64 [110/20]
      via FE80::FFFF:FE10:2, Serial0/0
OE2   2004:1:2::/64 [110/20]
```

```
        via FE80::FFFF:FE10:2, Serial0/0
```

R4#show ipv6 route ospf

```
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI   2001:1:1::1/128 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:1:2::1/128 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:12::/64 [110/192]
       via FE80::FFFF:FE10:5, Serial0/0
OI   2001:23::/64 [110/128]
       via FE80::FFFF:FE10:5, Serial0/0
```

接下去在 R2 上对 R1 的 Loopback 路由进行汇总：

R2 的配置增加如下：

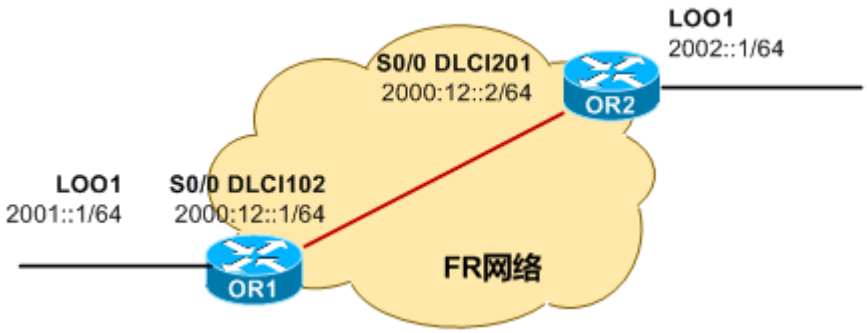
```
ipv6 router ospf 100
area 1 range 2001:1::/32
```

接下去在 R4 上，对其自身重发布进来的外部路由做汇总：

R4 的配置增加如下：

```
Ipv6 router ospf 100
  summary-prefix 2004:1::/32
```

9.5.3.2 帧中继环境下的问题



OR1 的配置

```
ipv6 unicast-routing
interface Serial0/0
encapsulation frame-relay
  no frame-relay inverse-arp
  ipv6 enable
  ipv6 address 2000:12::1/64
  ipv6 ospf network point-to-point
  ipv6 ospf 1 area 0
  frame-relay map ipv6 2000:12::2 102 broadcast
```

OR2 的配置

```
ipv6 unicast-routing
interface Serial0/0
encapsulation frame-relay
  no frame-relay inverse-arp
  ipv6 enable
  ipv6 address 2000:12::2/64
  ipv6 ospf network point-to-point
  ipv6 ospf 1 area 0
  frame-relay map ipv6 2000:12::1 201 broadcast
```

完成此步骤后，OR1 及 OR2 能建立起 OSPF 邻居关系，互相之间的全局 IPV6 单播地址也都能 ping 通

此时分别在 OR1 及 OR2 上开启 loopback 接口，并配置 Ipv6 地址，然后宣告进 OSPFv3

OR1 的配置

```
interface Loopback0
  ipv6 address 2001::1/64
```

```
 ipv6 enable
 ipv6 ospf 1 area 1
```

OR2 的配置

```
interface Loopback0
 ipv6 address 2002::1/64
 ipv6 enable
 ipv6 ospf 1 area 2
```

完成后，**OR1 及 OR2 都能学习到对方的 Loopback 接口路由，但是却无法 ping 通。**

查看 OR1 的路由表：

```
OI    2002::1/128 [110/64]
       via FE80::5C10:8CFF:FEE0:FE89, Serial0/0
```

发现去往 OR2 loopback 接口的路由，下一跳是 FE80::5C10:8CFF:FEE0:FE89，也就是 OR2 接口的链路本地地址，原来 Ipv6 环境下，一个接口往往具有多个 Ipv6 地址，而 OSPF 邻居关系的维护又以稳定为前提，每个接口都必备的链路本地地址，就成了建立邻居关系最好的一句，那么为什么 ping 不通呢？**正是由于这是个帧中继的环境，FE80::5C10:8CFF:FEE0:FE89 这个 IPV6 地址，OR1 上并没有做映射，OR2 上同理，因此分别添上各自对端的链路本地地址的帧中继映射即可。**

如 OR1，OR2 同理

```
frame-relay map ipv6 FE80::5C10:8CFF:FEE0:FE89 102 broadcast
```

9.6 MP-BGP

9.6.1 多协议 BGP 概述

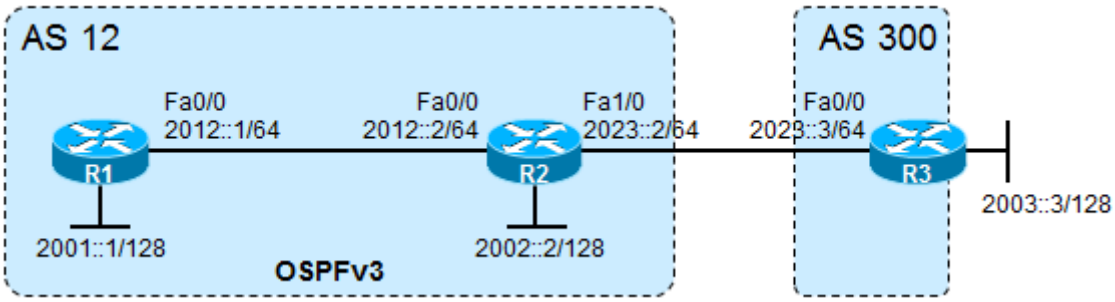
BGP4+，也叫做多协议 BGP，扩展 BGP-4 规范以支持诸如 IPv6、IPX、VPN 等这样的新地址簇。因此 BGP4+能够为 IPv6 和包括 IPv4 在内的其他协议携带路由选择信息。RFC2858 和 RFC2545 定义了在 BGP4+中处理 IPv6 而更新的属性。

下面是 BGP-4 规范中为了支持 IPv6 而进行了更新的属性：

- NEXT_HOP：表示为 IPv6 地址，或者是一个可聚合全球单播地址，或者是一个可聚合全球单播地址及其下一跳的本地链路地址
- NLRI： 是一组目的地，表示为一个 IPv6 前缀。

9.6.2 多协议 BGP 的基础配置

9.6.2.1 使用 IPv6 全局单播地址建 BGP4+邻居、传递 IPv6 路由



实验环境：

- R1、R2 处于 AS12，AS 内运行 OSPFv3，使得 R1、R2 能够学习到对方的 Loopback 路由
- R1、R2 建立 IPv6 的 IBGP 邻居关系（基于 LOOPBACK 接口）
- R2、R3 建立 IPv6 的 EBGP 邻居关系，R3 重发布直连路由，使得 R1、R2 能够学习到

实验配置：

R1 的配置如下（基本配置及 OSPFv3）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2012::1/64
   ipv6 nd suppress-ra                    !!由于 Fa0/0 连接的是路由器而非主机，为减少不必要的 ra 的
   no shutdown
interface loopback 0
   ipv6 enable
   ipv6 address 2001::1/128
!
Ipv6 router ospf 1
   router-id 1.1.1.1
!
interface fastEthernet 0/0
   ipv6 ospf 1 area 0
```

```
interface loopback 0
   ipv6 ospf 1 area 0
```

R2 的配置如下（基本配置及 OSPFv3）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2012::2/64
   ipv6 nd suppress-ra
   no shutdown
interface fastEthernet 1/0
   ipv6 enable
   ipv6 address 2023::2/64
   ipv6 nd suppress-ra
   no shutdown
interface loopback 0
   ipv6 enable
   ipv6 address 2002::2/128
!
Ipv6 router ospf 1
   router-id 2.2.2.2
!
interface fastEthernet 0/0
   ipv6 ospf 1 area 0
interface loopback 0
   ipv6 ospf 1 area 0
```

R3 的配置如下（基本配置）：

```
Ipv6 unicast-routing
interface fastEthernet 0/0
   ipv6 enable
   ipv6 address 2023::3/64
   ipv6 nd suppress-ra
   no shutdown
```

```
interface loopback 0
  ipv6 enable
  ipv6 address 2003::3/128
```

接下去开始配置 BGP4+：

R1 的 BGP4+配置如下：

```
router bgp 12
  bgp router-id 1.1.1.1
  no bgp default ipv4-unicast                    !!如果不配置此命令，则一旦指定 BGP 邻居后，默认 ipv4 邻居就会建立起来，本实验中路由器之间不传递 IPv4 前缀因此无需 IPv4 的连接。
  neighbor 2002::2 remote-as 12
address-family ipv6
  neighbor 2002::2 activate
  neighbor 2002::2 update-source loopback 0
  exit-address-family
```

R2 的 BGP4+配置如下：

```
router bgp 12
  bgp router-id 2.2.2.2
  no bgp default ipv4-unicast
  neighbor 2001::1 remote-as 12
  neighbor 2023::3 remote-as 300
address-family ipv6
  neighbor 2001::1 activate
  neighbor 2023::3 activate
  neighbor 2001::1 update-source loopback 0
  exit-address-family
```

R3 的 BGP4+配置如下：

```
router bgp 300
  bgp router-id 3.3.3.3
  no bgp default ipv4-unicast
  neighbor 2023::2 remote-as 12
address-family ipv6
```

```
 neighbor 2023::2 activate
 exit-address-family
```

简单的查看一下：

R1#show ip bgp ipv6 unicast summary

```
BGP router identifier 1.1.1.1, local AS number 12
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2002::2         4    12       3       3        1    0    0 00:00:39        0
```

R2#show ip bgp ipv6 unicast summary

```
BGP router identifier 2.2.2.2, local AS number 12
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001::1         4    12      22      22        1    0    0 00:19:44        0
2023::3         4   300       6       6        1    0    0 00:01:32        0
```

现在，在 R3 上，将直连 loopback 路由重发布进 BGP4+

```
ipv6 prefix-list loopb permit 2003::3/128
route-map test permit 10
 match ipv6 address prefix-list loopb
!
router bgp 300
 address-family ipv6
  redistribute connected route-map test
```

现在 R2 能学习到这条 IPv6 路由：

R2#sh ip bgp ipv6 unicast

```
BGP table version is 2, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 2003::3/128      2023::3                  0             0 300 ?
```

但是在 R1 上：

R1#sh ip bgp ipv6 unicast

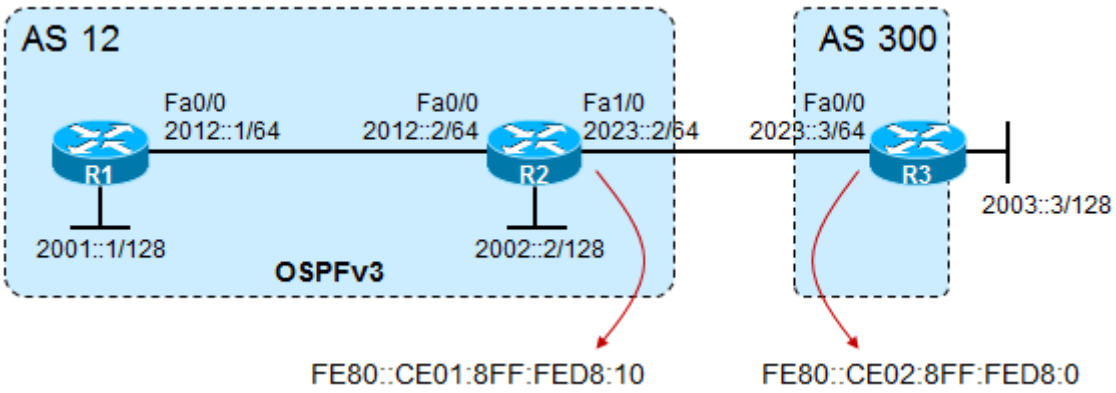
```
BGP table version is 1, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i2003::3/128      2023::3                  0    100      0 300 ?
```

我们发现在 R1 上，路由并不 best，很简单，因为 NH 不可达。解决的办法很简单，在 R3 上对 R2 做 next-hop-self 即可：

```
router bgp 12
  address-family ipv6
    neighbor 2001::1 next-hop-self
    exit-address-family
```

9.6.2.2 使用 IPv6 Linklocal 地址建 BGP4+邻居关系、传递 IPv6 路由

在某些环境下，可以使用 Linklocal 地址来建立 BGP 邻接关系，这样做的好处之一是，无需为这些链路分配可聚合全球 IP 地址。在 BGP 中使用 Linklocal 地址有一些注意事项，例如要确定目的 Linklocal 地址相对应的路由器物理接口，因为 Linklocal 地址只是本地有效。在一个是结合 route-map 修改 NEXT_HOP 属性。



如上图，我们在上一个实验的基础上，在 R2、R3 之间使用 Linklocal 地址来建立邻居关系：

其中 R2 的配置如下：

```
router bgp 12
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 neighbor 2001::1 remote-as 12
 neighbor 2001::1 update-source Loopback0
 neighbor FE80::CE02:8FF:FED8:0 remote-as 300
 neighbor FE80::CE02:8FF:FED8:0 update-source FastEthernet1/0      !!注意这条命令一定要配，因为linklocal 地址只是 link 范围内有效，因此需要指定本地与这个 linklocal 邻居地址对接的接口。
 !
 address-family ipv6
 neighbor 2001::1 activate
 neighbor 2001::1 next-hop-self
 neighbor FE80::CE02:8FF:FED8:0 activate
 exit-address-family
```

其中 R3 的配置如下：

```
 router bgp 300
 bgp router-id 3.3.3.3
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor FE80::CE01:8FF:FED8:10 remote-as 12
 neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
 !
 address-family ipv6
 neighbor FE80::CE01:8FF:FED8:10 activate
 redistribute connected route-map test
 no synchronization
 exit-address-family
```

注意，在 R2 上我们是对 R1 做了 next-hop-self，因此从 AS300 传递过来的路由 NH 发生了改变。但是在某些网络环境中，使用 Linklocal 建立 BGP 邻接关系可能导致 NH 的一些问题，毕竟 Linklocal 地址只是链路内有效，因此，可以搭配 route-map，使用 set ipv6 next-hop 来修改这些 BGP 路由的 NH 属性。

9.6.3 其他配置

1. 应用前缀列表

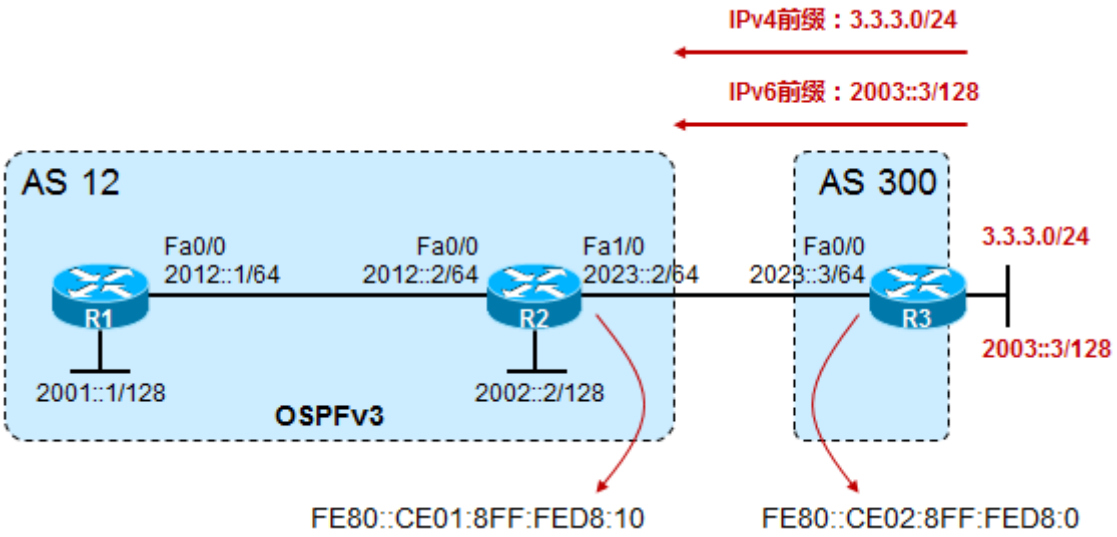
```
ipv6 prefix-list loopb permit 2003::3/128
router bgp 12
  address-family ipv6
    neighbor 2023::2 prefix-list loopb out
  exit-address-family
```

注意，过滤的命令是在 address-family ipv6 中进行配置

2. 在 IPv6 BGP 对等体之间传递 IPv4 路由

有这么一种网络环境：两个 IPv4 网络通过纯 IPv6 提供商连接，那么这时就不得不通过 BGP IPv6 对等体交换 IPv4 路由。因为 BGP4+支持 IPv4 及 IPv6 协议簇，因此这是完全没有问题的。

但是需格外关注 next-hop 的可达性问题。



上面的实验环境中，R2、R3 是使用 Linklocal 地址建立 BGP 邻居关系。R3 所在的 AS300 内 IPv6 的路由前缀，也有 IPv4 的路由前缀，要传给 R2。

那么 R2 的配置如下：

```
router bgp 12
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 2001::1 remote-as 12
```

```
neighbor 2001::1 update-source Loopback0
neighbor FE80::CE02:8FF:FED8:0 remote-as 300
neighbor FE80::CE02:8FF:FED8:0 update-source FastEthernet1/0
!
address-family ipv4
neighbor FE80::CE02:8FF:FED8:0 activate
no auto-summary
no synchronization
exit-address-family
!
address-family ipv6
neighbor 2001::1 activate
neighbor 2001::1 next-hop-self
neighbor FE80::CE02:8FF:FED8:0 activate
exit-address-family
```

R3 的配置如下：

```
router bgp 300
 bgp router-id 3.3.3.3
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor FE80::CE01:8FF:FED8:10 remote-as 12
 neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
 !
 address-family ipv4
 neighbor FE80::CE01:8FF:FED8:10 activate
 no auto-summary
 no synchronization
 network 3.3.3.0 mask 255.255.255.0
 exit-address-family
 !
 address-family ipv6
 neighbor FE80::CE01:8FF:FED8:10 activate
 redistribute connected route-map test
 no synchronization
```

```
 exit-address-family
```

完成配置后，R2 及 R3 之间的 BGP 连接就能够同时传递 IPv4 及 IPv6 的路由前缀。

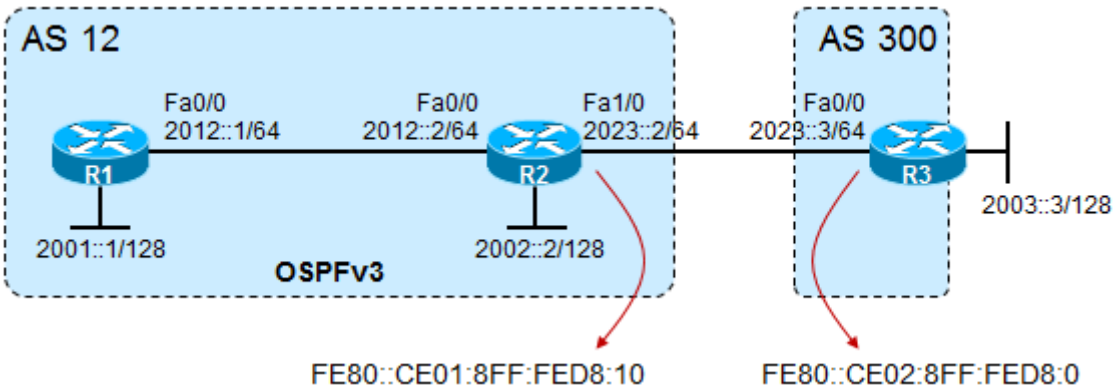
但是我们发现，R2 上，已经能够学习到 R3 传递过来的 IPv6 前缀，但是 IPv4 前缀始终没有收到。在 R2 上，还能看到如下报错：

```
*Mar  1 17:40:49.491: %BGP-6-NEXTHOP: Invalid next hop (254.128.0.0) received from FE80::CE02:8FF:FED8:0: martian next hop
R2#
*Mar  1 17:40:49.495: BGP(0): FE80::CE02:8FF:FED8:0 rcv UPDATE w/ attr: nexthop 254.128.0.0, origin i, metric 0, originator 0.0.0.0, path 300, community , extended community
*Mar  1 17:40:49.499: BGP(0): FE80::CE02:8FF:FED8:0 rcv UPDATE about 3.3.3.0/24 -- DENIED due to: martian NEXTHOP;
```

原来，R3 传递过来的 3.3.3.0/24 的 IPv4 前缀，NH 为 254.128.0.0，是一个保留的地址，该地址当然是不合法的，因此 R2 直接忽略这条路由。解决的办法，可以在 R3 上对 R2 做 out 方向的 route-map，并且设置 IPv4 前缀的 NH，如：

```
route-map setNH permit 10
 set ip next-hop 10.1.23.3
 !
router b 300
address-family ipv4
neighbor FE80::CE01:8FF:FED8:10 route-map setNH out
```

3. 和 BGP4+一起使用 MD5



R2 和 R3 之间用 MD5 认证，

R3 的配置如下：

```
router bgp 300
 bgp router-id 3.3.3.3
```

```
no bgp default ipv4-unicast
bgp log-neighbor-changes
neighbor FE80::CE01:8FF:FED8:10 remote-as 12
neighbor FE80::CE01:8FF:FED8:10 password cisco
neighbor FE80::CE01:8FF:FED8:10 update-source FastEthernet0/0
!
address-family ipv4
.....
exit-address-family
!
address-family ipv6
neighbor FE80::CE01:8FF:FED8:10 password cisco
neighbor FE80::CE01:8FF:FED8:10 activate
redistribute connected route-map test
no synchronization
exit-address-family
R3#
```

10 IPv6 基础实验

10.1 Windows 主机篇

1. **安装 IPv6 协议栈**

   Windows7 的系统，默认就已经安装了 IPv6 协议栈。

   WindowsXP 的系统，需要手工安装，方法很简单，进入 CMD 界面操作：

```
Ipv6 install                          !! 安装完后重启，默认开启无状态自动获取地址
ipv6 if                               !! 查看 ipv6 接口配置信息
ping IPv6Address [%ZoneID]            !! ping 链路本地地址和站点本地地址可能要加索引号，ping 全局单播地址不需要
```

2. **常用 IPv6 命令**

   netsh 可以用来配置、管理、重置 IPV4 V6 协议栈

netsh interface ipv6 show ?

可以查看 IPv6 配置的各项参数。

netsh interface ipv6 show interface

查看接口（网卡，包括虚拟网卡）索引及名称，等同于 WindowsXP 下的 ipv6 if x

netsh interface ipv6 show address

查看网卡 ipv6 地址信息，如有效期、prefered 生存期等等 （ipconfig 也可）

netsh interface ipv6 show neighbors

相当于看 ARP 表，其实是邻居缓存表

netsh interface ipv6 show route

查看 IPv6 路由

netsh interface ipv6 add address *接口索引号 ipv6 地址*

为接口添加 ipv6 地址

例如：

netsh interface ipv6 show interface !!得到目标接口索引号

netsh interface ipv6 add address 21 3ffe::1 unicast !!假设要配置的接口索引号为 21，则这条命令是为该接口配置一个 IPv6 单播地址。

netsh interface ipv6 add route ::/0 “本地连接” 2010::1

添加路由 或 ipv6 rtu

例如：

netsh interface ipv6 add route 3333::/64 21 nexthop=fe80::1 !!添加一条目的为 3333::/64 的 IPv6 路由，关联到索引号为 21 的接口，路由的下一跳是 fe80::1

netsh interface ipv6 reset

删除用户已配置的所有设置。需要重新启动计算机之后默认设置才能生效

3. IPv6 基本应用

IPv6 主机（xp 系统 ie6.0）访问 IPv6 WEB 站点

IE6.0 暂不支持地址栏输入 rfc2732 定义的 literal ipv6 address 的方式访问 IPv6 站点。

解决办法，通过域名来访问，修改 hosts 文件 **C:/windows/system32/drivers/etc/hosts**

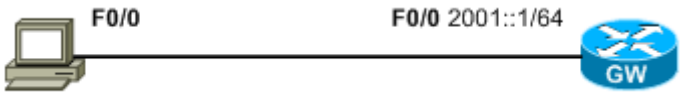
```
# 102.54.94.97      rhino.acme.com          # source server
# 38.25.63.10       x.acme.com              # x client host

127.0.0.1           localhost
::1   test.com
```

10.2 路由器基本配置

常用基本配置

```
Router(config)#ipv6 unicast-routing            !! 打开接口之间的 IPv6 数据包转发功能
Router(config)#intertface f0/1
Router(config-if)#ipv6 enable                  !! 接口激活 IPv6
Router(config-if)#ipv6 adress 2001::1/64       !! 配置一个 IPv6 地址
Router(config-if)#ipv6 adress 2002::/64 eui-64     !! 又配置了一个 IPv6 地址，并且使用 EUI-64 填充
```

10.3 无状态自动配置



RFC2462 所定义，即插即用

利用 RS 及 RA 报文。

RS 报文由主机主动发起，RA 报文由主机默认以 200S 周期性发送（收到 RS 也会立刻发送）。

GW 的配置

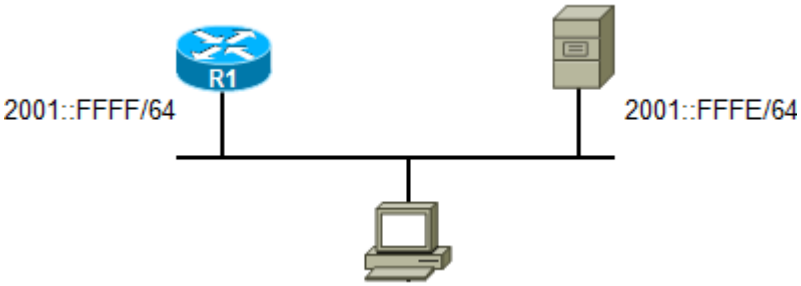
```
ipv6 unicast-routing
interface fast0/0
  ipv6 address 2001::1/64
```

```
 no ipv6 nd suppress-ra                    !! 接口地址 2001::1/64， 同时开启路由器通告（默认关闭）
```

客户端的配置

```
ipv6 address autoconfig [default]          !! 使用 EUI64 地址构建 Ipv6 地址；
                                           !! 如果加 default 关键字，则会在获取到地址的接口上添加一个默认
网关（默认路由）
```

10.4 有状态自动配置



这里的示例是 DHCPv6，使用 Router 作为 DHCPserver

DHCPserver 的配置

```
ipv6 unicast-routing                                  !! 这条必须敲
ipv6 dhcp pool DHCP-pool
 prefix-delegation pool dhcppool lifetime 1800 600    !! 调用 ipv6 local pool 并设定 lifetime
 dns-server 2000::8
 domain-name HelloWord
ipv6 local pool dhcppool 2001::/64 64                 !! 定义准备通告的 Ipv6 前缀

interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFE/64
 ipv6 nd other-config-flag
 ipv6 nd managed-config-flag
 ipv6 dhcp server DHCP-pool      !! 在接口上开启 ipv6 DHCP，并调用池
```

PC 的配置

```
interface FastEthernet0/0
```

```
ipv6 enable                          !! 接口激活 IPV6
ipv6 dhcp client pd test             !! 配置 DHCP client，并且指定获取到的前缀名称为 test（本地有效）
ipv6 address test ::/64 eui-64       !! 使用获取到的前缀（test），加上本接口的 EUI64，构成接口全局 ipv6 地址
```

当一个主机收到 RA 路由器通告的时候，它会去查看 RA 里 M 位，如果M位为 1，则开始在 LAN 上寻找 DHCP 服务器。主机将使用 linklocal 地址作为源发送 DHCP-solicit message，目的是 ALL_DHCP_Agents，

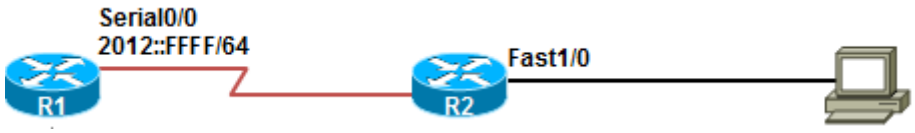
以下是 DHCPv6 可能使用到的地址及端口：

FF02::1:2 = ALL DHCP Agents（servers or relays,link-local scope）

FF05::1:3 = ALL DHCP servers（Site-local scope）

DHCP 报文 client 侦听 UDP546；servers 及 agents 侦听 UDP547

10.5 DHCP PD



R1 的配置如下：

```
ipv6 dhcp pool DHCP-pool
   prefix-delegation pool dhcppool lifetime 1800 600
   dns-server 2000::8
   domain-name HelloWord
ipv6 local pool dhcppool 2001::/64 64
interface Serial0/0
   ipv6 address 2012::FFFF/64
   ipv6 dhcp server DHCP-pool
```

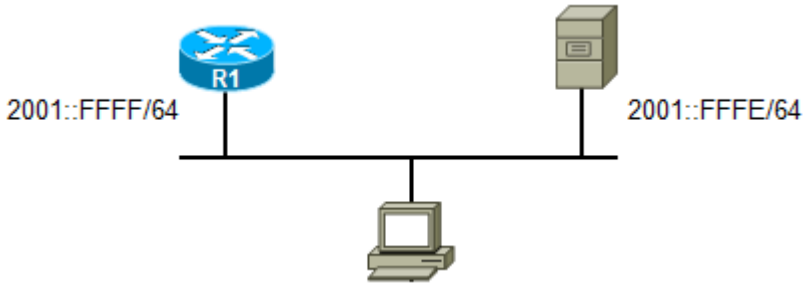
R2 的配置如下：

```
interface Serial0/0
   ipv6 address autoconfig default
   ipv6 dhcp client pd test
Interface fast1/0
   ipv6 address test    ::FFFF/64
```

R1 上有个地址池，如果 R2 希望使用 R1（IPv6 DHCPserver）的地址池分配给自己内网的用户，那么可以使

用 DHCP PD 特性。在 R2 上，这个 test，作为一个 PD 标示符本地有效。

10.6 混合的自动配置实验



R1 的配置如下

```
interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFF/64
 no ipv6 nd suppress-ra
 ipv6 nd other-config-flag
```

DHCPserver 的配置

```
ipv6 dhcp pool DHCP-pool
 dns-server 2000::8
 domain-name HelloWord
 exit
interface FastEthernet0/0
 ipv6 enable
 ipv6 address 2001::FFFE/64
 no ipv6 nd suppress-ra
 ipv6 dhcp server DHCP-pool
```

11 IPv6 Multicast

11.1 基本配置

ipv6 multicast-routing

全局命令，开启 IPv6 multicast-routing，打开该开关后，所有激活 IPv6 的接口自动打开 IPv6 PIM

Ipv6 mld join-group ff04::1

接口级命令，用于加入组播组，也就是说该接口成为该组播组的成员

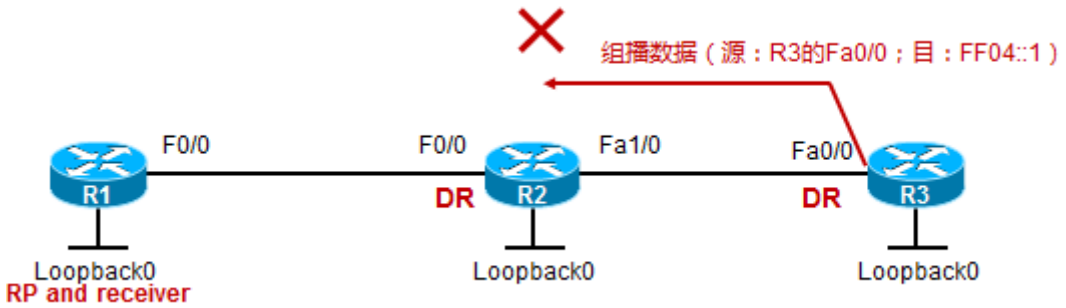
Ipv6 pim rp-address X:X:X:X::X

全局命令，静态指定 RP

关于 IPv6 multicast 的全面详解，请见《红茶三杯 Multicast 技术笔记》

11.2 关于 DR 理解的一个小实验

【测试1】



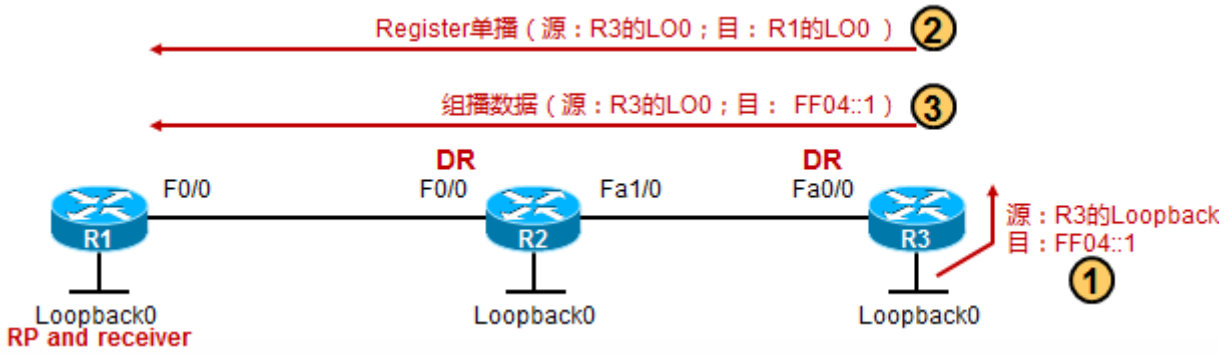
测试描述： 在 R3 上 ping FF04::1，出接口为 Fa0/0

测试现象： 无法 ping 通。

现象分析： R3 直接产生组播数据，这些组播数据的源 IP 为 R3 Fa0/0 的接口 IP，目的 IP 为组播组地址 FF04::1。这些组播数据是直接发送给了 R2，此时此刻，R2 是 First-hop 路由器。那么当 R2 收到这些组播数据的时候，由于是在一个多路访问网络中收到的（在自己的 Fa1/0 口），而自己的 Fa1/0 口又不是该 LAN 内的 DR，既然不是 DR，就无权向 RP 发起 register。因此，R2 只是简单的创建一个 IPv6 组

播转发表项而已（且 outgoing interface list 为空），不会做进一步的处理。于是这些组播数据在 R2 处被丢弃。

【测试2】



测试描述： 在 R3 上 ping FF04::1，出接口为本地的 Loopback0

测试现象： 能够 ping 通

现象分析： 这次环境跟前面就不大一样了。首先组播数据是来自 R3 的 Loopback0 口，那么 First-hop 路由器就变成了 R3，而不是 R2 了。那么 R3 作为第一跳路由器，在收到组播源（也就是自己的 LO0）发出的组播数据后，R3 将触发 Register 过程，register 过程的详细内容，大家都非常熟悉了，这里就不罗嗦了，最终的结果是，R3 到 R1 构建了一个 SPF，于是乎来自 R3 Loopback0 口、发向组播组 FF04::1 的组播数据可以沿着这可途径 R2 到 R1 的 SPF 进行转发。

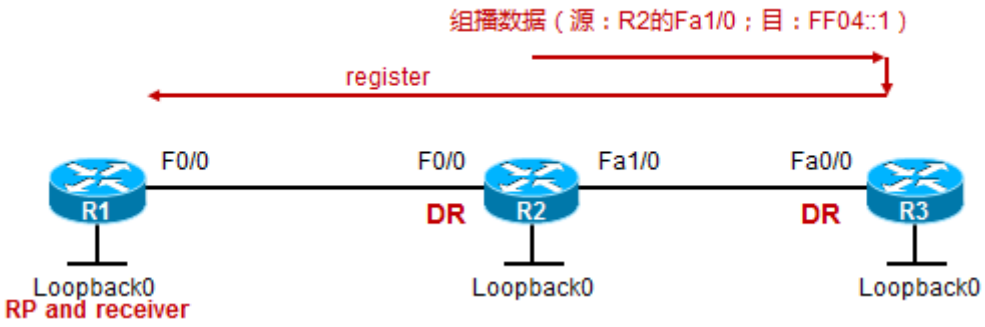
R2 的表项（2033::3 是 R3 的 Loopback0 口 IP 地址）：

```
(2033::3, FF04::1), 00:00:04/00:03:24, flags: ST
  Incoming interface: FastEthernet1/0
  RPF nbr: FE80::CE02:1BFF:FE54:0
  Immediate Outgoing interface list:
    FastEthernet0/0, Forward, 00:00:04/00:03:24
```

R3 的表项：

```
(2033::3, FF04::1), 00:00:16/00:03:13, flags: SFT
  Incoming interface: Loopback0
  RPF nbr: FE80::CE02:1BFF:FE54:0
  Immediate Outgoing interface list:
    FastEthernet0/0, Forward, 00:00:15/00:03:14
```

【测试3】

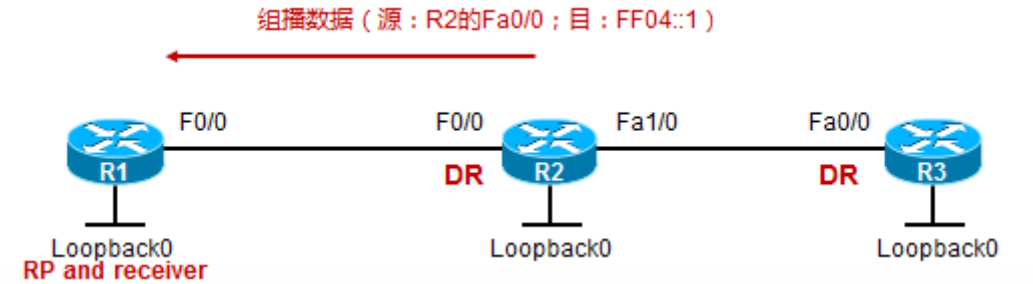


测试描述： 在 R2 上 ping FF04::1，出接口为 Fa1/0

测试现象： 能够 ping 通

现象分析： R2 产生的这些组播数据包，源地址为 R2 的 Fa1/0 口地址，目的地就不说了铁定是组播组的 IP。这些组播数据到了 R3，由于 R3 是第一跳路由器，并且 R3 的 Fa0/0 口又是本 LAN 的 DR，因此 R3 将向 RP 发起 Register，组播数据被送到了 R1。这时候实际上组播数据是到了 R1 也就是到了接收者了，但是实际上，底下还在发生着一些更复杂的事情。R1 在收到 Register 之后，会有意愿与源形成 SPT，因此它向 R2 发了一个 SPT 的 join 消息，R2 收到之后将自己的 Fa0/0 口添加到对应组播表项的 outgoing interface list 中。但是，由于组播数据始终是从 R2 的 Fa1/0 口发出（因为我们测试的时候，指定的 outgoing 接口为 Fa1/0），因此 R1 不断的收到 R3 发来的、封装了组播数据的 Register，却没有收到沿着 SPT 传下来的组播数据，因此，R1 不会向 R3 发送 register-stop，R3 的注册过程只能反复的进行，组播数据将一直被封装在 Register 消息中传给 R1。

【测试4】



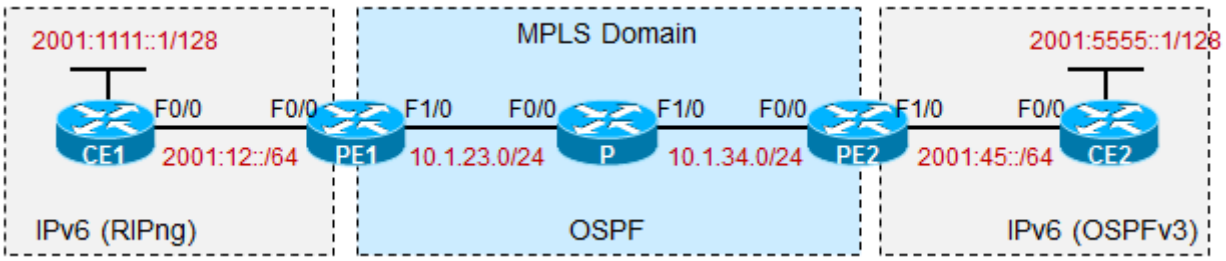
测试描述： 在 R2 上 ping FF04::1，出接口为 Fa0/0

测试现象： 能够 ping 通

现象分析： 这个应该说没任何难度了，此时此刻 R2 就是一个组播源，将组播数据直接发送给 R1。

12 6PE 及 6VPE

12.1 6PE 基础实验



1. 实验环境

1) IPv4 及 IPv6 地址规划如上图所示。

2) MPLS Domain 由三台路由器：PE1、P、PE2 构成，域内运行的 IGP 协议是 OSPF，进程号用 100。三台路由器各自将自己的 Loopback 口宣告进 Core 的 OSPF。Loopback 的编址为 x.x.x.x/32，x 为路由器编号，如 PE1 为 2.2.2.2、P 为 3.3.3.3、PE2 为 4.4.4.4。

3) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。为了方便观察实验现象，将 PE1 的标签范围定为 200-299；P 的标签空间为 300-399；PE2 的为 400-499。

2. 实验需求

1) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。

2) PE1 及 PE2 之间激活 MPBGP 的邻居关系，为传递 IPv6 路由做准备。

3) CE1 将 IPv6 路由放给 PE1，经由 PE1、PE2 间的 MPBGP 邻接传递给 PE2，最终传递给 CE2。CE2 到 CE1 的 IPv6 路由传递类似。实验的最终目的就达到了，通过 MPLS Backbone 为我们传递 IPv6 的路由。那么两个 CE 都能学习到对方的 IPv6 路由 我们的 IPv6 数据能否穿越 Core 呢？毕竟 Core 是非 IPv6 的而且更加没有 IPv6 路由的。这时候，MPLS 的威力就发挥出来了。站点之间的数据在 MPLS Backbone 内传输的时候，多是两层标签，外层是 LDP 生成的，内层是 MPBGP 为 IPv6 路由生成的。

3. 实验步骤及配置

1) 完成 CE1、CE2 的配置

CE1 的配置如下：

```
ipv6 unicast-routing
!
```

```
ipv6 router ospf 1
   router-id 1.1.1.1
!
interface Loopback0
   ipv6 enable
   ipv6 address 2001:1111::1/64
   ipv6 ospf 1 area 0
interface FastEthernet0/0
   ipv6 enable
   ipv6 address 2001:12::1/64
   ipv6 ospf 1 area 0
```

CE2 的配置如下：

```
ipv6 unicast-routing
!
ipv6 router ospf 1
   router-id 5.5.5.5
!
interface Loopback0
   ipv6 enable
   ipv6 address 2001:5555::5/64
   ipv6 ospf 1 area 0
interface FastEthernet0/0
   ipv6 address 2001:45::5/64
   ipv6 ospf 1 area 0
```

2) 完成 Core 区域内 IGP 协议的配置

PE1 的配置如下：

```
ip cef
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
interface FastEthernet0/0
```

```
 ipv6 enable
 ipv6 address 2001:12::2/64
 ipv6 ospf 1 area 0
!
mpls label range 200 299
mpls ldp router-id loopback0
!
interface FastEthernet1/0
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100                          !! Core 内的 OSPF
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
```

P 的配置如下：

```
ip cef
mpls label range 300 399
mpls ldp router-id Loopback0
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
interface FastEthernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
interface FastEthernet1/0
 ip address 10.1.34.3 255.255.255.0
 mpls ip
router ospf 100
 router-id 3.3.3.3
 network 3.3.3.3 0.0.0.0 area 0
 network 10.1.23.3 0.0.0.0 area 0
 network 10.1.34.3 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip cef
!
mpls label range 400 499
mpls ldp router-id Loopback0
!
interface Loopback0
   ip address 4.4.4.4 255.255.255.255
interface FastEthernet0/0
   ip address 10.1.34.4 255.255.255.0
   mpls ip
interface FastEthernet1/0                    !! 连接 CE2 的接口
   ipv6 enable
   ipv6 address 2001:45::4/64
   ipv6 ospf 1 area 0
 !
router ospf 100                              !! core 内的 IGP
   router-id 4.4.4.4
   network 4.4.4.4 0.0.0.0 area 0
   network 10.1.34.4 0.0.0.0 area 0
```

3) 完成 PE1 及 PE2 上，PE-CE 路由协议的配置，以及 MP-BGP 的配置

PE1 的配置如下：

```
ip cef
ipv6 unicast-routing          !! 注意，要支持 IPv6 单播路由协议，该开关必须打开
ipv6 cef                      !! 注意，支持 IPv6 的 PE 上，IPv6 cef 必须打开
!
ipv6 router ospf 1            !! PE-CE 间的 OSPFv3
   router-id 2.2.2.2
   redistribute bgp 234
!
interface FastEthernet0/0
   ipv6 ospf 1 area 0
!
router bgp 234
```

```
 bgp router-id 2.2.2.2
 no bgp default ipv4-unicast
 neighbor 4.4.4.4 remote-as 234              !! 4.4.4.4 也就是 PE2
 neighbor 4.4.4.4 update-source Loopback0
!
 address-family ipv6
    neighbor 4.4.4.4 activate                !! 激活与 PE2 的 IPv6 地址族
    neighbor 4.4.4.4 send-label              !! 向 PE2 发送为 IPv6 前缀分配的标签
    redistribute connected                   !! 重发布直连
    redistribute ospf 1                      !! 将从 CE1 学习到的 IPv6 路由前缀重发布进 BGP
 exit-address-family
```

PE2 的配置如下：

```
ip cef
ipv6 unicast-routing                 !! 注意，要支持 IPv6 单播路由协议，该开关必须打开
ipv6 cef                             !! 注意，支持 IPv6 的 PE 上，IPv6 cef 必须打开
!
interface FastEthernet1/0
 ipv6 ospf 1 area 0
 !
router ospf 100
 network 4.4.4.4 0.0.0.0 area 0
 network 10.1.34.4 0.0.0.0 area 0
 !
router bgp 234
 no bgp default ipv4-unicast
 neighbor 2.2.2.2 remote-as 234
 neighbor 2.2.2.2 update-source Loopback0
 !
 address-family ipv6
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 send-label
    redistribute connected
    redistribute ospf 1
```

```
  exit-address-family
 !
ipv6 router ospf 1
  router-id 4.4.4.4
  redistribute bgp 234
```

4. 实验分析

首先在 PE1 上，看一下：

PE1# show bgp all neighbors 4.4.4.4

```
…部分省略. ..
For address family: IPv6 Unicast
BGP neighbor is 4.4.4.4,   remote AS 234, internal link
  BGP version 4, remote router ID 4.4.4.4
  BGP state = Established, up for 01:05:47
  Last read 00:00:44, last write 00:00:47, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv6 Unicast: advertised and received
    ipv6 MPLS Label capability: advertised and received
…部分省略. ..
```

可以看出，PE1 及 4.4.4.4 的 PE2 之间是具有 IPv6 MPLS 的标签分配和分发能力的。

PE1#show bgp ipv6 unicast

```
BGP table version is 5, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 2001:12::/64	::	0		32768	?
*>i2001:45::/64	::FFFF:4.4.4.4	0	100	0	?
*> 2001:1111::1/128	::	1		32768	?
*>i2001:5555::5/128	::FFFF:4.4.4.4	1	100	0	?

我们看到，PE1 上已经学习到来自 CE2 的 IPv6 路由了，继续看看

PE1#show bgp ipv6 unicast labels

Network	Next Hop	In label/Out label
2001:12::/64	::	204/nolabel
2001:45::/64	::FFFF:4.4.4.4	nolabel/404
2001:1111::1/128	::	203/nolabel
2001:5555::5/128	::FFFF:4.4.4.4	nolabel/403

我们重点看 2001:5555::5/128 这条路由，下一跳是个奇怪的地址，不过从这个地址的最后 32bits 也就是 4.4.4.4 不难看出这条路由是谁传给我的，另外，前缀的 out label 是 403，而这个 403 正是 PE2 传递给我的。也就是说，PE2 为 2001:5555::5/128 这条 IPv6 前缀分配了 403 标签，并且将这个映射传递给了 PE1，这个标签其实是 MP-BGP 产生的。那么当 PE1 有数据要去往 2001:5555::5 是不是直接打上标签 403 就行了呢？当然不是，因为标签 403 对于 MPLS Backbone 内的 P 路由器来说，根本不理解，因此，我们还需要一个外层标签，这个标签由 LDP 产生，使得我们的数据包能够在 MPLS Backbone 内传输。

进一步查看这条路由：

PE1#show bgp ipv6 unicast 2001:5555::5/128

```
BGP routing table entry for 2001:5555::5/128, version 3
Paths: (1 available, best #1, table Global-IPv6-Table)
  Not advertised to any peer
  Local
    ::FFFF:4.4.4.4 (metric 3) from 4.4.4.4 (4.4.4.4)
      Origin incomplete, metric 1, localpref 100, valid, internal, best
      mpls labels in/out nolabel/403
```

我们发现路由的下一跳是 4.4.4.4。

PE1#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	10.1.34.0/24	0	Fa1/0	10.1.23.3
201	301	4.4.4.4/32	0	Fa1/0	10.1.23.3
202	Pop tag	3.3.3.3/32	0	Fa1/0	10.1.23.3
203	Untagged	2001:1111::1/128	570	Fa0/0	FE80::CE00:10FF:FE58:0
204	Aggregate	2001:12::/64	2204		

进一步查看，发现 P 为 4.4.4.4 分配的了标签 301，因此，当 PE1 收到数据包，去往 2001:5555::5 的时候，数据包会被压上两层标签，外层标签值为 301，内层标签值为 403。

为了再次证明这一点，我们在 PE1 上：

PE1#sh ipv cef 2001:5555::5

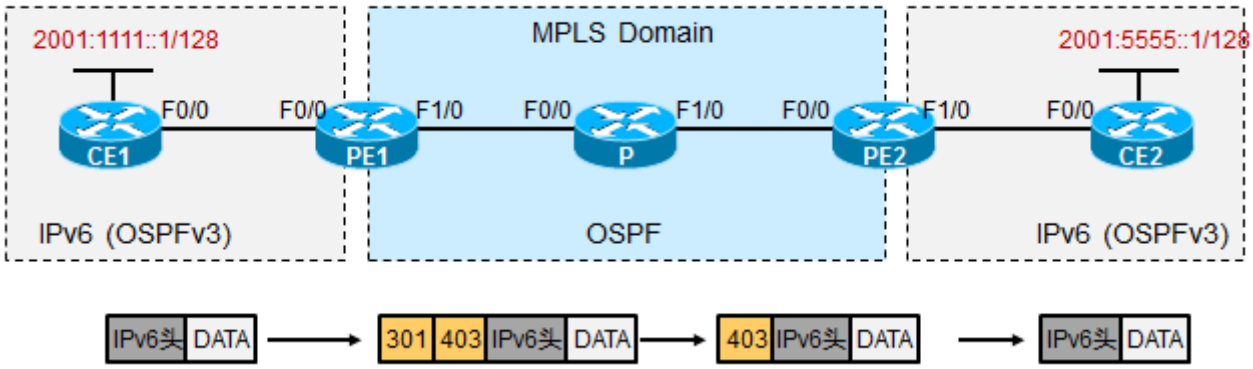
```
2001:5555::5/128
```

```
    nexthop ::FFFF:4.4.4.4
  fast tag rewrite with Fa1/0, 10.1.23.3, tags imposed: {301 403}
```

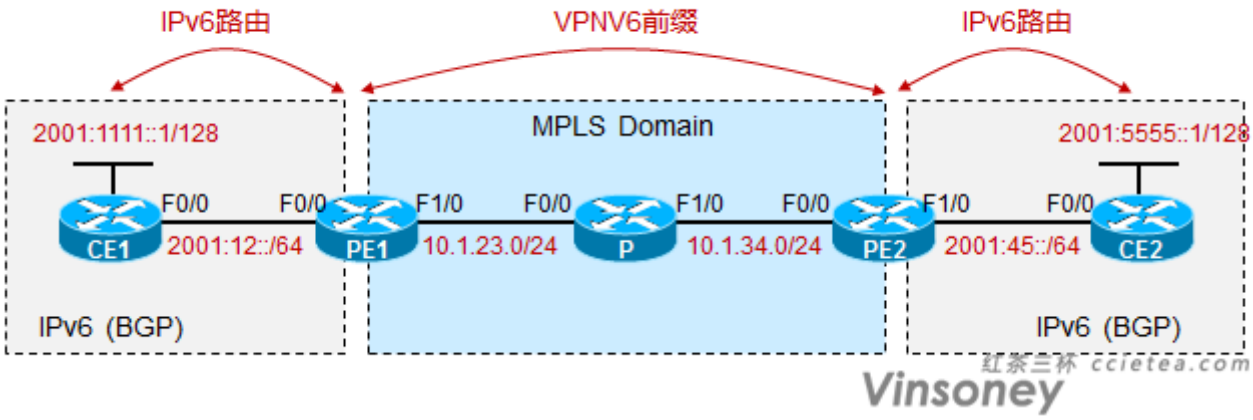
以下是我们在 CE1 上，ping 2001:5555::5，然后在 PE1 的 Fa1/0 口上抓包：

```
Ethernet II, Src: cc:01:10:58:00:10 (cc:01:10:58:00:10), Dst: cc:02:10:58:00
MultiProtocol Label Switching Header, Label: 301, Exp: 0, S: 0, TTL: 63
  MPLS Label: 301
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 0
  MPLS TTL: 63
MultiProtocol Label Switching Header, Label: 403, Exp: 0, S: 1, TTL: 63
  MPLS Label: 403
  MPLS Experimental Bits: 0
  MPLS Bottom Of Label Stack: 1
  MPLS TTL: 63
Internet Protocol Version 6, Src: 2001:12::1 (2001:12::1), Dst: 2001:5555::5
Internet Control Message Protocol v6
```

从抓包的结果能看到，有两层 label，这个标签包到了 P 路由器后，由于有 PHP 机制，顶层的 LDP 标签被弹出，接着这个标签包又被传到 PE2，PE2 弹出 MP-BGP 的标签，还原成原始的 IPv6 数据然后转发给 CE2。



12.2 6VPE 基础实验



1. 实验环境

4) IPv4 及 IPv6 地址规划如上图所示。

5) MPLS Domain 由三台路由器：PE1、P、PE2 构成，域内运行的 IGP 协议是 OSPF，进程号用 100。三台路由器各自将自己的 Loopback 口宣告进 Core 的 OSPF。Loopback 的编址为 x.x.x.x/32，x 为路由器编号，如 PE1 为 2.2.2.2、P 为 3.3.3.3、PE2 为 4.4.4.4。

6) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。为了方便观察实验现象，将 PE1 的标签范围定为 200-299；P 的标签空间为 300-399；PE2 的为 400-499。

2. 实验需求

4) PE1、P、PE2 形成 LDP 邻居关系，传递 IGP 标签。

5) PE1 及 PE2 之间激活 MPBGP 的邻居关系，**为传递 VPNv6 前缀做好准备**

6) CE1 将 IPv6 路由放给 PE1，PE1 上创建 IPv6 的 VRF，将 Fa0/0 口放进 VRF。PE1 通过 BGP 学习到 CE1 的 IPv6 前缀，并形成 VPNv6 前缀，通告给自己的 MP-BGP 邻居 PE2，PE2 将 VPNv6 前缀对应的 IPv6 前缀最终传递给 CE2。

3. 实验步骤

1) CE1 及 CE2 的配置

CE1 的配置如下：

```
ipv6 unicast-routing
interface Loopback0
  ipv6 enable
  ipv6 address 2001:1111::1/128
```

```
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:12::1/64
router bgp 100
  bgp router-id 1.1.1.1
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 2001:12::2 remote-as 234
 !
  address-family ipv6
    neighbor 2001:12::2 activate
    network 2001:1111::1/128
  exit-address-family
```

CE2 的配置如下：

```
ipv6 unicast-routing
interface Loopback0
  ipv6 enable
  ipv6 address 2001:5555::5/128
interface FastEthernet0/0
  ipv6 enable
  ipv6 address 2001:45::5/64
router bgp 100
  bgp router-id 5.5.5.5
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 2001:45::4 remote-as 234
 !
  address-family ipv6
    neighbor 2001:45::5 activate
    network 2001:5555::5/128
  exit-address-family
```

2) 完成 MPLS Backbone 的 IGP 及 LDP 配置

PE1 的配置如下：

```
ip cef
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 200 299
!
interface FastEthernet0/1
 ip address 10.1.23.2 255.255.255.0
 mpls ip
!
router ospf 100
 router-id 2.2.2.2
 network 2.2.2.2 0.0.0.0 area 0
 network 10.1.23.2 0.0.0.0 area 0
```

P 的配置如下：

```
ip cef
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 300 399
!
interface FastEthernet0/0
 ip address 10.1.23.3 255.255.255.0
 mpls ip
interface FastEthernet0/1
 ip address 10.1.34.3 255.255.255.0
 mpls ip
!
```

```
router ospf 100
    network 3.3.3.3 0.0.0.0 area 0
    network 10.1.23.3 0.0.0.0 area 0
    network 10.1.34.3 0.0.0.0 area 0
```

PE2 的配置如下：

```
ip cef
interface Loopback0
  ip address 4.4.4.4 255.255.255.255
!
mpls ldp router-id Loopback0
mpls label range 400 499
!
interface FastEthernet0/0
  ip address 10.1.34.4 255.255.255.0
  mpls ip
!
router ospf 100
  router-id 4.4.4.4
  network 4.4.4.4 0.0.0.0 area 0
  network 10.1.34.4 0.0.0.0 area 0
```

3) 完成 VRF 的配置、MP-BGP 的配置

PE1 的配置如下：

```
ip cef
ipv6 unicast-routing                !! 再次强调下这几天命令的必要性
ipv6 cef
!
vrf definition cisco                !! 定义 vrf
  rd 1:1
  !
  address-family ipv6               !! 在 IPv6 地址族中定义 RT 值
    route-target export 234:2
    route-target import 234:4
```

```
  exit-address-family
!
interface FastEthernet0/0
  vrf forwarding cisco                    !!接口关联到 VRF cisco
  ipv6 enable
  ipv6 address 2001:12::2/64
!
router bgp 234
  bgp router-id 2.2.2.2
  no bgp default ipv4-unicast
  neighbor 4.4.4.4 remote-as 234
  neighbor 4.4.4.4 update-source Loopback0
  !
  address-family vpnv6                    !!邻居 4.4.4.4 也就是 PE2，激活 VPNv6 支持能力
    neighbor 4.4.4.4 activate
    neighbor 4.4.4.4 send-community extended
  exit-address-family
  !
  address-family ipv6 vrf cisco           !! 用于从 CE1 学习 IPv6 路由前缀
    neighbor 2001:12::1 remote-as 100
    neighbor 2001:12::1 activate
  exit-address-family
```

PE2 的配置如下：

```
ip cef
ipv6 unicast-routing                      !! 再次强调下这几天命令的必要性
ipv6 cef
!
vrf definition cisco
  rd 1:1
  !
  address-family ipv6
    route-target export 234:4
    route-target import 234:2
```

```
  exit-address-family
!
interface FastEthernet0/1
  vrf forwarding cisco
  ipv6 enable
  ipv6 address 2001:45::4/64
!
router bgp 234
  bgp router-id 4.4.4.4
  no bgp default ipv4-unicast
  neighbor 2.2.2.2 remote-as 234
  neighbor 2.2.2.2 update-source Loopback0
  !
  address-family vpnv6
    neighbor 2.2.2.2 activate
    neighbor 2.2.2.2 send-community extended
  exit-address-family
  !
  address-family ipv6 vrf cisco
    neighbor 2001:45::5 remote-as 500
    neighbor 2001:45::5 activate
  exit-address-family
```

4. **实验分析**

我们在 PE1 来看看：

PE1#show bgp vpnv6 unicast all 2001:5555::5/128

```
BGP routing table entry for [1:1]2001:5555::5/128, version 32
Paths: (1 available, best #1, table cisco)
  Advertised to update-groups:
        1
  500
    ::FFFF:4.4.4.4 (metric 3) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:234:4
```

```
        mpls labels in/out nolabel/403
```

查看 PE1 上的 VPNv6 前缀 2001:5555::5 ,可以看到 out label 是 403 ,很明显是 R4 也就是 PE2 分配的标签，这个标签其实就是 VPNv6 的标签，为了让 PE2 知道，这个数据所归属的 VRF。也可以使用如下命令查看 VPNV6 前缀关联的标签：

PE1#show bgp vpnv6 unicast all labels

```
   Network          Next Hop        In label/Out label
Route Distinguisher: 1:1 (cisco)
   2001:1111::1/128 2001:12::1        204/nolabel
   2001:5555::5/128 ::FFFF:4.4.4.4    nolabel/403
```

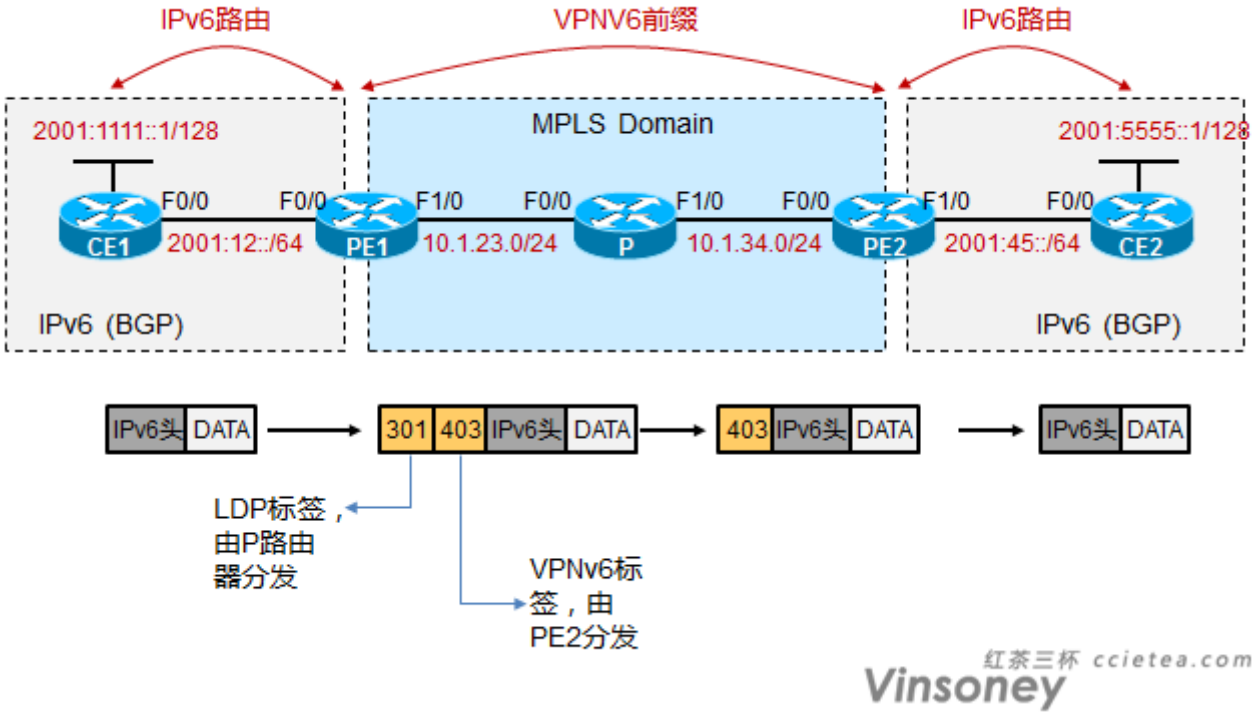
现在 2001:5555::5 有了 VPNV6 的标签，但是，还数据层面是无法穿越 MPLS BACKBONE 的，因为 P 路由器都不识别这个标签。因此，还是需要再增加一个标签，使得内网标签及数据负载能够在 MPLS BACKBONE 中传输。这个外层标签由 LDP 分配。

PE1#show ipv6 cef vrf cisco 2001:5555::5/128

```
2001:5555::5/128
  nexthop 10.1.23.3 FastEthernet0/1 label 301 403
```

从 CEF 表可以看出，当 PE1 收到 CE1 去往 CE2 的 2001:5555::5 的流量时，标签是压两层的，外层是 301，是 P 路由器分发的，内层是 403，是 PE1 分发的。

因此，当 CE1 从源 2001:1111::1 访问 2001:5555::5 时，数据层面上是这样的：



5. 其他命令

Show vrf ipv6	查看 vrf ipv6
Show vrf ipv6 detail	查看 vrf 详细信息

13 参考书籍

- CISCO IPv6 网络实现技术：CISCO self-study : Implementing Cisco IPv6 Networks(IPv6)