MPLS 技术笔记

红茶三杯 CCIE 学习文档

文档版本：	1.5	更新时间：	2013-04-02
文档作者：	红茶三杯	文档地址：	http://ccietea.com
文档备注：	请关注文档版本及更新时间。		

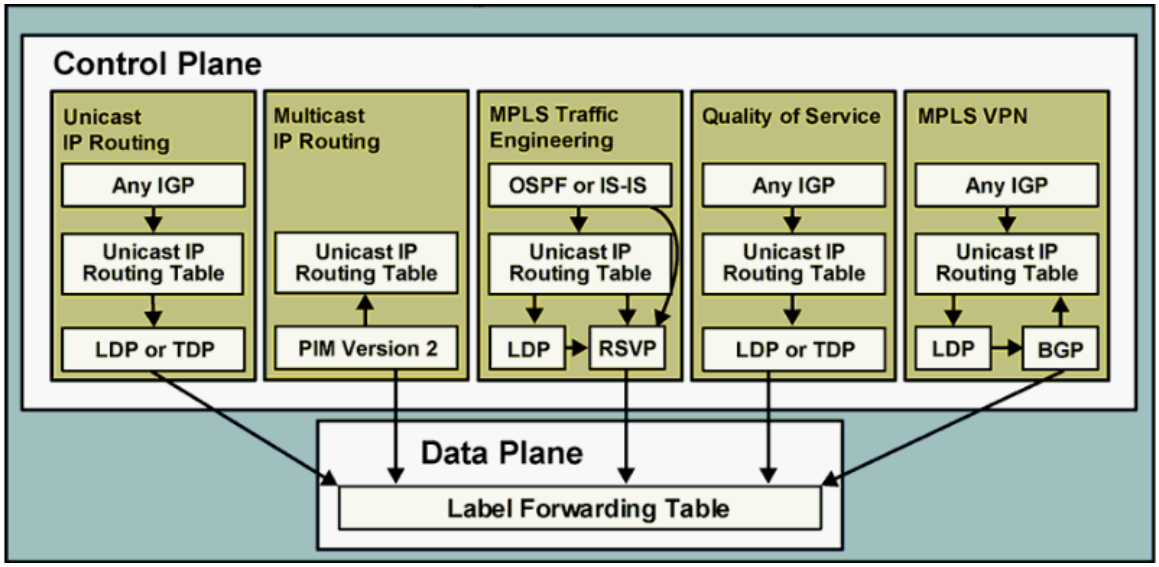
红茶三杯
学习 沉淀 成长 分享
微博：http://weibo.com/vinsoney
博客：http://blog.sina.com.cn/vinsoney
站点：http://ccietea.com

1 Basic

1.1 基础知识点

1. **MPLS 的特性**
   - MPLS 依赖 IP 路由和 CEF 交换
   - MPLS 是一种依据数据包标签的转发技术
   - MPLS 支持多种三层协议
   - MPLS 涉及的目标网段与传统的 IP 转发是一样的

2. **MPLS 的应用**



3. **当一个数据包进入 PE 后，是查路由表（FIB）还是查标签表（LFIB）？**

   具体查哪个表，主要取决于所收到的数据的二层封装，例如如果二层封装是以太网（数据帧），则看帧头的“类型/长度”字段的值：

   Type：0x8847（单播）上层承载的是 MPLS，查找 LFIB

   Type：0x8848（组播）上层承载的是 MPLS，查找 LFIB

   Type：0x0800 承载的是 IPv4 报文，查找 FIB

4. **在初始路由器上，是查找 FIB 还是 LFIB 呢？**

   由于在初始路由器上，产生的是 IP 报文，查找的是 CEF 表，用“show ip cef <ip address> detail”命令，可以从输出中看到对于此路由是否压入标签，分配标签是根据 CEF 表中的前缀来分配的。

   R1#show ip cef 4.4.4.4

```
4.4.4.4/32, version 12, epoch 0, cached adjacency 10.1.12.2
0 packets, 0 bytes
  tag information set
    local tag: 104
    fast tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
  via 10.1.12.2, FastEthernet0/0, 0 dependencies
    next hop 10.1.12.2, FastEthernet0/0
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
```

例如这条 CEF 显示，去往 4.4.4.4 的数据，要压上 203 的标签，然后发送给 10.1.12.2。

1.2 基本名词

1. **A label switch router (LSR)**

   一台支持并激活了 MPLS 的设备。It is capable of understanding MPLS labels and of receiving and transmitting a labeled packet on a data link. Three kinds of LSRs exist in an MPLS network:

   - **Ingress LSRs**—Ingress LSRs receive a packet that is not labeled yet, insert a label (stack) in front of the packet, and send it on a data link.
   - **Egress LSRs**—Egress LSRs receive labeled packets, remove the label(s), and send them on a data link. Ingress and egress LSRs are edge LSRs.
   - **Intermediate LSRs**—Intermediate LSRs receive an incoming labeled packet, perform an operation on it, switch the packet, and send the packet on the correct data link.

2. **A label switched path (LSP)**

   is a sequence of LSRs that switch a labeled packet through an MPLS network or part of an MPLS network. Basically, the LSP is the path through the MPLS network or a part of it that packets take. //单向路径。

   实际上，LSP 是报文在穿越 MPLS 网络或者部分 MPLS 网络时的路径。

3. **A Forwarding Equivalence Class (FEC)**

   is a group or flow of packets that are forwarded along the same path and are treated the same with regard to the forwarding treatment.

   在转发过程中，具有相同处理方式的一组数据，可通过地址、隧道、COS 等方式来标识，通常在一台设备上，对于一个 FEC 分配相同的标签。属于一个 FEC 的流量具有相同的转发方式、转发路径和转发待遇。

但是并不是所有拥有相同标签的报文都属于一个 FEC，因为这些报文的 EXP 值可能不相同，执行方式可能不同，因此他们可能属于不同的 FEC。

决定报文属于哪一个 FEC 的路由器是入站 LSR，因为是它对报文进行分类和压入标签。

下面是一些 FEC 的范例：

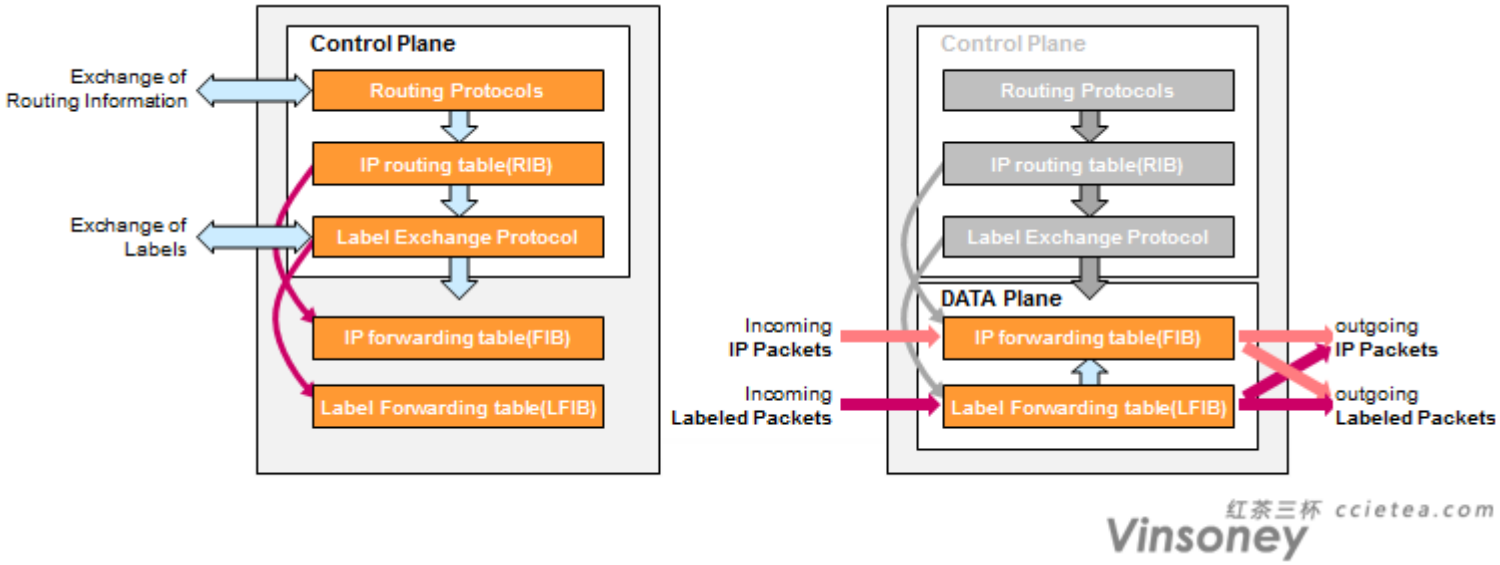
- 第三层目的 IP 匹配同一个特定前缀的报文
- 属于某个特定组播组的组播报文
- 根据进程或者 IP　DSCP 字段有相同处理方式的报文

注：一条 FEC 可以包含多个流，但不是一个流一个 FEC，比如一台主机在看新浪的网页，这是一个流，又在看新浪的视频，这又是一个流，这两个流在新浪发给远程主机时，走的路径应该是相同的，所以一个 FEC 有多个流，但是每个流并没有属于单独的 FEC

1.3 基本架构

Control plane： 交换三层**路由信息**（如 OSPF、ISIS、BGP 等）及**标签**（如 TDP、LDP、BGP 及 RSVP 等）；

Data plane： 基于标签进行数据转发

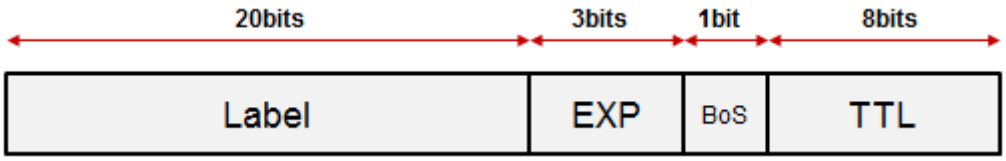


注意：LDP 或 TDP 只会对 IGP 协议的路由条目产生标签，不会对 BGP 的路由条目产生标签

1.4 MPLS 标签

1. 标签描述

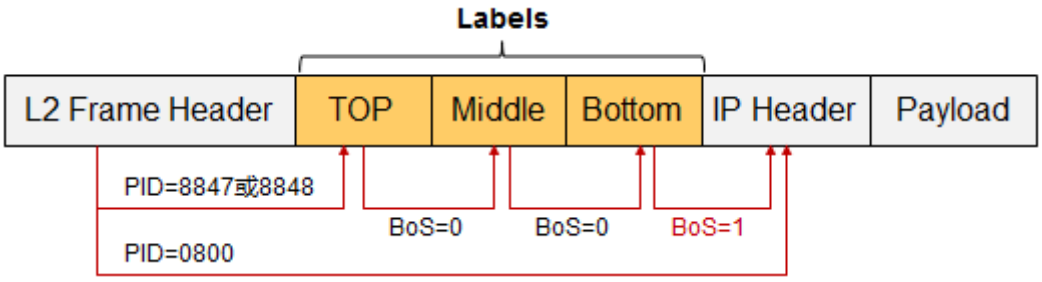
两种模式：frame mode ；ATM mode（cell mode），我们主要研究的是 frame mode



标签头一共 32 位，包含的字段如下：

LABEL	20bits
BoS	1bit 为栈底位（这一位 置 1，则为最后一个标签。 可以给一个数据包压多层标签，最后一个标签的 Bos 位则置 1，当处理到这层，上层数据包则为普通数据包。）
TTL	8bit 最大 255，通常在加标签的时候，是把普通 ip 报文的 TTL 直接 copy 进来
EXP	3bit 实验位用于 QoS

2. 标签的位置



路由器如何知道一个报文是普通报文还是一个标签包呢？ MPLS 报文会在二层头和三层包间插入一个 MPLS 标签头。同时二层数据链路层帧头会做相应的指示，例如以太网数据帧，MAC 层的 TYPE 字段会指示上层数据是否为 mpls 标签帧（如果是 IPv4 报文的话值为 0x0800，如果是标签包则为 8847-单播或 8848-组播）

上图是封装了三层标签包，对于一个 LSR 来说，只处理第一个标签。每一层标签都有个 BoS 栈底位，用来表示是否已经是标签栈的栈底，最后一个标签的 BoS=1。

3. MPLS 的编码

标签栈放置在第三层包头之前，也就是说，被传输的协议头部之前，同时在第二层包头之后。

对于第 2 层链路层封装可以是 CISCO IOS 所支持的所有封装类型，PPP、HDLC、Ethernet 等等。那么在这些第 2 层的帧头，就需要有相应的协议字段来指示上层是一个标签头。

第2层协议类型	第2层协议标识名称	值（16进制）
PPP	PPP协议字段	0281
以太网/802.3 LLC/SNAP封装	TYPE	8847（单播） 8848（组播）
HDLC	协议	8847
帧中继	NLPID（网络级别协议ID）	80

4. **标签的处理方式**
   - Insert（impose or push）
   - Swap
   - Remove（PoP）

   关于标签处理方式的更详细内容，见下文。

2 转发带标签的报文

2.1 转发带标签的报文

The first label is imposed on the ingress LSR and the label belongs to one LSP. The path of the packet through the MPLS network is bound to that one LSP. All that changes is that the top label in the label stack is swapped at each hop. The ingress LSR imposes one or more labels on the packet. The intermediate LSRs swap the top label (the incoming label) of the received labeled packet with another label (the outgoing label) and transmit the packet on the outgoing link. The egress LSR of the LSP strips off the labels of this LSP and forwards the packet.

For this to work, adjacent LSRs must agree on which label to use for each IGP prefix. Therefore, each intermediate LSR must be able to figure out with which outgoing label the incoming label should be swapped

MPLS 与 Tag switching

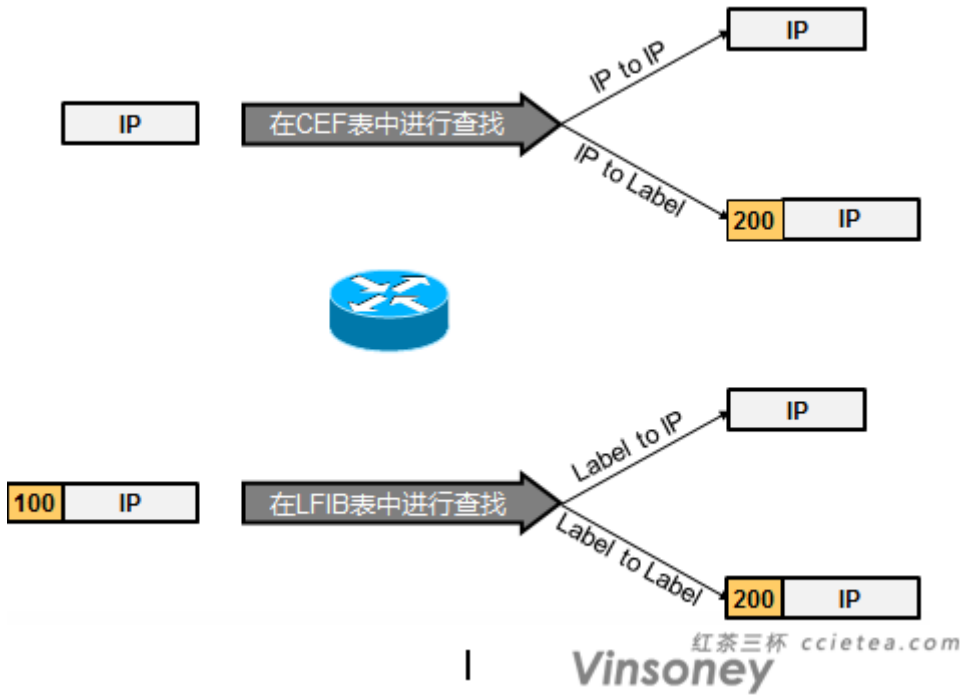
MPLS 是 CISCO 私有协议，后来被 IETF 借用，在 IETF 叫做 tag switching，可以理解为名字不同而已

两者在启用时，用的协议不一样

LDP　　IETF 标准　　（TCP/UDP 端口号 646）

TDP　　CISCO 私有　（TCP/UDP 端口号 711）

2.2 IP 查找和标签查找



如果 LSR 收到一个 IP 报文，则查看 FIB 表，假设 FIB（CEF 表）相关表项输出如下：

```
R1#show ip cef 4.4.4.0 detail
4.4.4.0/24, epoch 0
  local label info: global/16
  nexthop 10.1.12.2 FastEthernet0/0 label 204
```

那么 LSR 将这个 IP 包压入一个标签头，打上标签 204，再从 F0/0 接口转发出去。在 CISCO IOS 中，CEF 交换是唯一的一种可用于标记报文的 IP 转发模式，因此在启用 MPLS 的时候必须在路由器上开启 CEF。

而当一台路由器收到一个带标签的报文的时候，则在 LFIB 表中进行查找，LFIB 表中相关的匹配条目会有针对该入站标签的出站动作或标签，以及下一跳信息。

【注意】如果路由器收到一个带标签的数据包，并且顶部标签不能在本地 LFIB 中找到，那么 CISCO IOS 将丢弃

这个数据包。

2.3 三张表

1. **FIB**

   也就是 CEF 表

2. **LIB 标签信息库**

   路由器为每一个 IGP 前缀在本地生成一个标签并分发给 LDP 邻居，同时也从其他 LDP 邻居收到为特定前缀分发的标签。路由器将本地标签和远程标签（LDP 邻居发给我的）存储在 LIB 中。

3. **LFIB 标签转发信息库**

   LSR 路由可能收到某个特定前缀的多个标签（多个 LDP 邻居分发的），但他只需要使用其中一个，IP 路由表用来确定这个 Ipv4 前缀的下一跳（LFIB 的构造需要一来 IGP 路由表，因为 LFIB 的下一跳就是 IGP 表算来的）。LSR 用这样的信息来创建它的 LFIB。在 LFIB 中本地捆绑的标签作为入站标签，LDP 邻居通告的标签作为出站标签。

   构成 LFIB 的标签可以不是 LDP 分发的 ,在 MPLS 流量工程中使用 RSVP 分配标签 ,在 MPLS VPN 中，VPN 标签由 BGP 分发。在任何情况下，LFIB 总是用来转发入站的带标签的报文。

- **查看 MPLS 转发表（LFIB）**

R1#show mpls forwarding-table

R1#show mpls forwarding-table

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
16	204	4.4.4.0/24	0	Fa0/0	10.1.12.2
17	205	3.3.3.0/24	0	Fa0/0	10.1.12.2
100	Pop Label	2.2.2.0/24	0	Fa0/0	10.1.12.2
103	202	10.1.34.0/24	0	Fa0/0	10.1.12.2
104	Pop Label	10.1.23.0/24	0	Fa0/0	10.1.12.2

R1#show mpls forwarding-table 4.4.4.0 detail

Local Label	Outgoing Label or VC	Prefix or Tunnel Id	Bytes Label Switched	Outgoing interface	Next Hop
16	204	4.4.4.0/24	0	Fa0/0	10.1.12.2
	MAC/Encaps=14/18, MRU=1500, Label Stack{204}				
	CA014FEC0008CA004FEC00088847 000CC000				

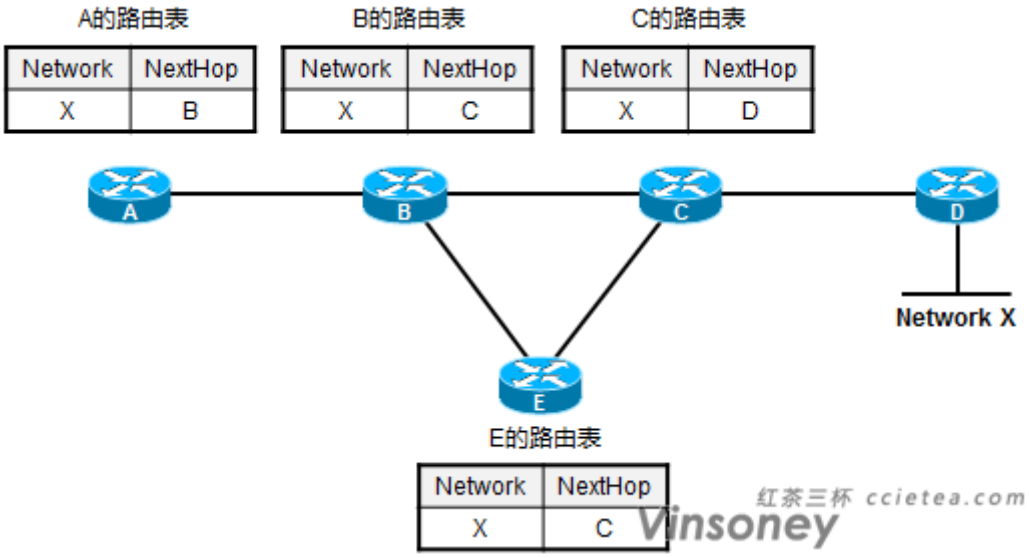
```
No output feature configured
```

使用 detail 关键字能查看到 LFIB 转发条目的详细信息，包括二层的信息，以及所有的标签内容，如果不加则只能看到顶层标签。

2.4 标签分配过程概述

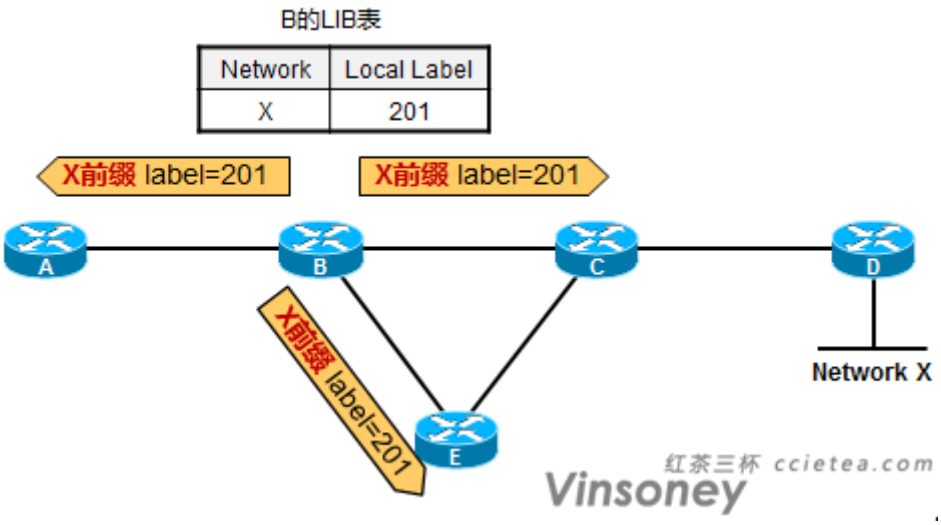
注意，我们研究的是 frame-mode 的标签分配。

1. 构建 IP 路由表

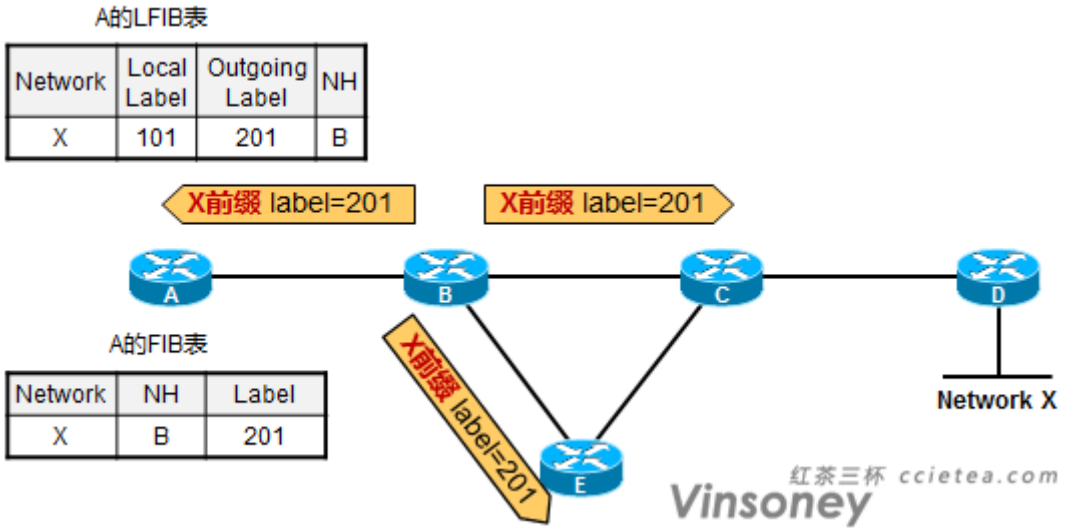


首先呢，所有的路由器都运行路由协议，全网路由互通。接下去在路由器的接口上激活 LDP，那么路由器两两之间就会形成 LDP 邻接关系。

2. 分配并分发标签



激活 LDP 后，每台路由器会为自己本地路由表中的路由前缀分配并捆绑标签，例如本图中的 X 路由，所有路由器都会自己为 X 路由分配一个标签 ,然后将为 X 路由捆绑的标签保存在本地的 LIB 表里。随后 ,所有的 LDP 路由器，将自己为各个路由前缀捆绑的标签发给它的 LDP 邻居，如图所示，B 将捆绑的标签发给了 A、E 和 C。这里有个小细节，注意，标签的分发没有水平分割的概念，也就是说虽然 B 可能是从 C 学习到路由 X，但是 B 仍然会将为前缀 X 捆绑的标签传递给 C。C 也会将传递自 B 的标签放在 LIB 中，不过不用担心会有环路，因为 LDP 可以借助 IGP 路由协议来防止环路。



当 LDP 路由器从他的 LDP 邻居收到邻居的标签捆绑，它会将捆绑的结果存储在自己的 LIB 表中，LIB 表中有 local label ,这是自己为特定路由前缀分发的标签 ,remote label 这是邻居为该路由前缀分发的标签。LDP 路由器会将邻居分配的 remote label 都放在 LIB，但是只会从 LIB 中所有可能的标签中选择出一个可能的出站标签放入 LFIB 中。LDP 路由器根据 LIB 表，结合路由表，构成 LFIB 表，如上图中 A 的 LFIB 表。

这里有一点要注意，我们看 E 路由器，实际上，它是会收到自己 LDP 邻居 B 和 C 的标签捆绑，其中都有关于路由 X 捆绑的标签，那么 E 在自己的 LFIB 表里，存的是哪位邻居的 remote label 呢？答案是 C，因为 C 是 E 路由器去往 X 的下一跳，E 会借助路由表，来判断谁的标签捆绑 “更优”。

3. LIB and LFIB 表的建立

A的LFIB表

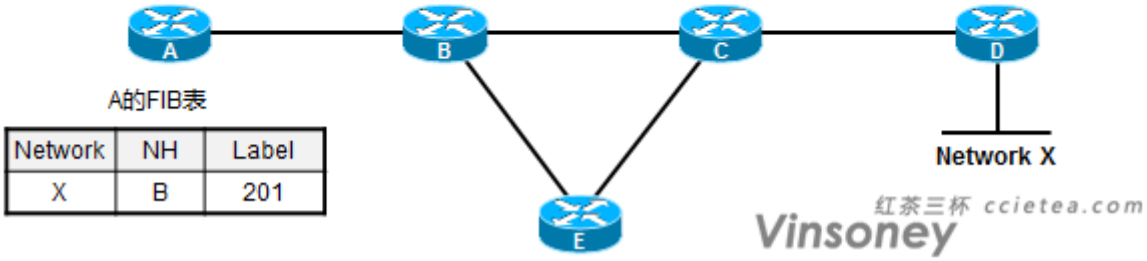
Network	Local Label	Outgoing Label	NH
X	101	201	B

B的LFIB表

Network	Local Label	Outgoing Label	NH
X	201	301	C

C的LFIB表

Network	Local Label	Outgoing Label	NH
X	301	PoP	D



在所有的 LDP 路由器都互相发送标签捆绑后，大家逐步构建自己的 LFIB。

A的LFIB表

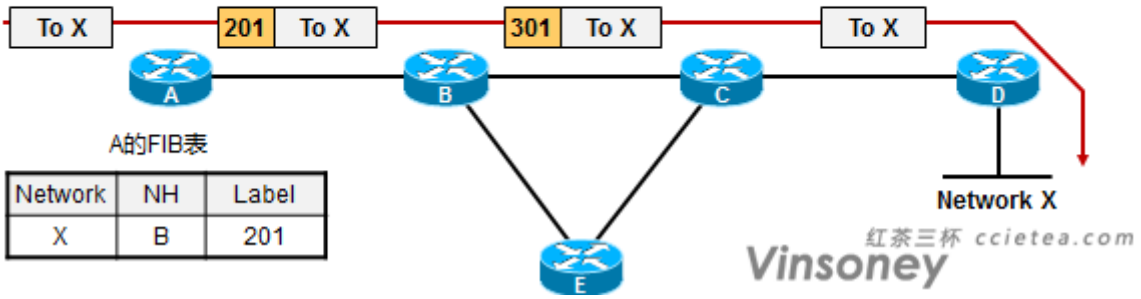
Network	Local Label	Outgoing Label	NH
X	101	201	B

B的LFIB表

Network	Local Label	Outgoing Label	NH
X	201	301	C

C的LFIB表

Network	Local Label	Outgoing Label	NH
X	301	PoP	D

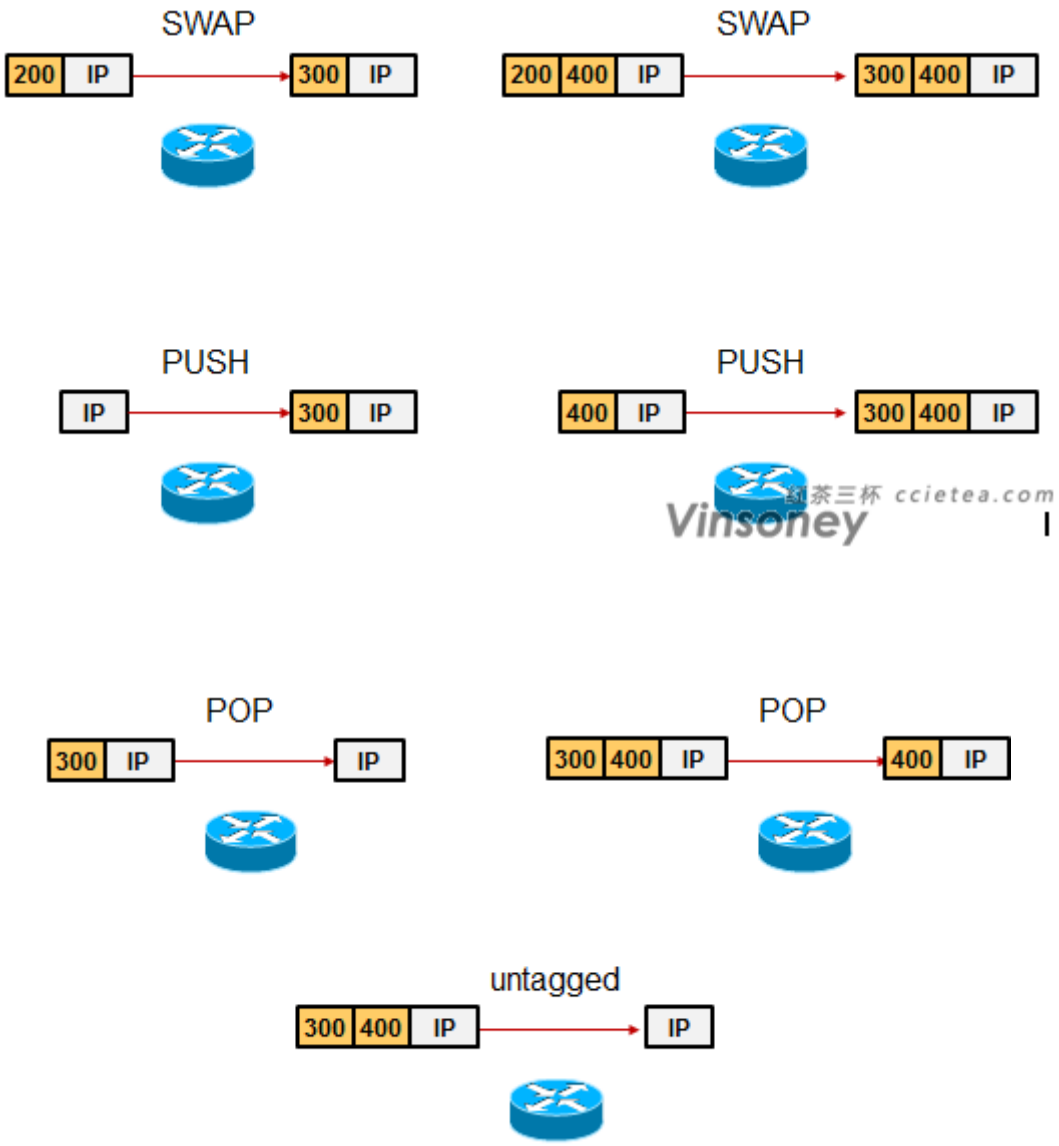


大家的 LFIB 构建完成后，我们就能看到，当 A 收到一个 IP 数据包，要访问 X，那么 A 查看自己的 CEF 表（为什么是查看 CEF 还记得么？），发现要给数据包压标签，于是它压上标签头，标签值为 201，然后下一跳是 B ,也就是发送给 B。那么 B 收到这个标签包 ,发现标签头里的 label 字段为 201 ,于是查看自己的 LFIB，发现 201 标签的 outgoing label 是 301 并且下一跳是 C，于是它将 201 标签替换成 301，然后交给 C。到了 C 它也去查 FIB 表，发现 301 标签对应的 outgoing label 是 PoP 并且下一跳是 D，于是它将标签头弹出，这就露出了原始的 IPv4 数据，C 将这个数据包交给 D。

2.5 标签处理动作的小结

- **移除：** 顶部标签被移除。报文的转发依靠标签栈中余下的标签，或者将其作为无标签的报文进行转发。
- **交换：** 顶部标签被移除，同时使用一个新的标签代替被移除的标签。
- **添加：** 顶部标签被一个新的标签所替代（交换），并且在代替的标签上层会再增加一个或者多个标签。

- **未标记/无标签：** 标签栈整个被移除，并且报文按照无标签的方式转发。
- **聚合：** 标签栈被移除，并且对该 IP 报文进行 IP 查找



1. **Outgoing tag(Label)中的 POP 与 Untag 有什么区别?**
   - POP　仅会将顶层标签头部弹出，经过该动作转发后的报文可以是 IP 报文也可以是 MPLS 标签报文
   - Untag　会将所有标签头部移除，变成一个纯 IP 报文转发出去

2. **产生 PoP 和 untag 的情况**

   PoP：　收到了下游发来的分配给某特定前缀的空标签，这个标签的值是 3，那么该 LSR 向这个下游 LSR 发送去往该前缀的数据的时候，他会弹出顶层标签（POP）进行转发，注意，这时候对于本 LSR 来说只需要查

找一次 LFIB，LFIB 中有下一跳的信息，因此弹出顶层标签，然后交给下一跳即可，而不用再次查找 FIB 表(如果弹出标签后是 IP 包)。

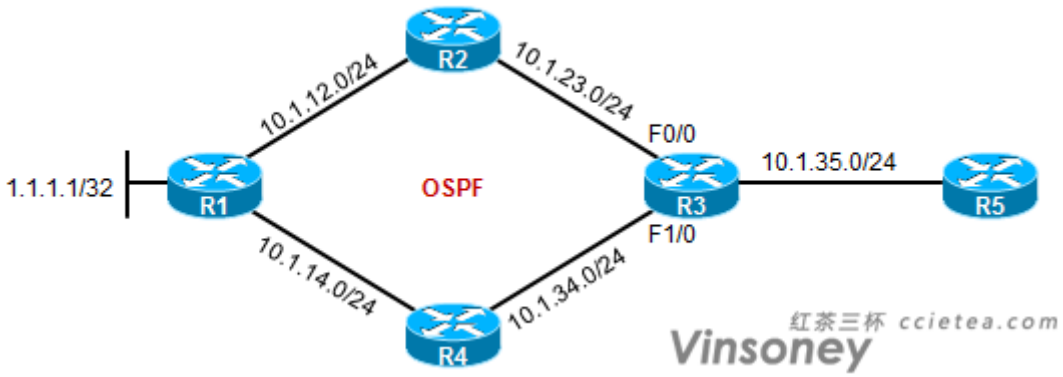
Untag： 弹出所有标签，随后根据下一跳（查找 FIB 表）转发。出现 untag 有以下 3 种原因：

- 下游不能分配标签，没有启用 MPLS
- 下游分了标签但传不出来，是因为 LDP Neighbor 没有建立
- 分配标签错误 ,这种情况仅出现在 IGP 是 OSPF 的情况下 ,因为如果用环回口作为 ldp 的 router-id，并且不是 32 位的，OSPF 会自动以 32 位的环回地址发布，这样会导致标签就分配错误。

3. 关于 untagged 、聚合等的详细介绍和区别、产生原因，请见《红茶三杯 MPLS VPN 技术文档》

2.6 标签报文的负载均衡

1. CEF 中的等价负载均衡



1) 环境描述

- 所有路由器运行 OSPF，全网互通，IP 规划如图所示，所有设备的 loopback0 为 x.x.x.x/32 编址，x 为设备编号。
- R1 通告 Loopback 路由.1.1.1.1/32
- R5 接口上配置两个地址：10.1.35.5，以及 10.1.35.55，用于模拟两个不同的源主机

2) 实验现象

在 R3 上看看路由表：

```
O        1.1.1.1 [110/3] via 10.1.34.4, 00:53:36, FastEthernet1/0
                         [110/3] via 10.1.23.2, 00:53:36, FastEthernet0/0
```

路由表中，R3 去往 1.1.1.1，使用 10.1.34.4 及 10.1.23.2 等价负载均衡。

我们在 R3 上开启了 CEF，实际上在 CEF 中，默认的负载均衡方式是基于目的地的负载均衡。CEF 的

基于目的地负载均衡实际上是通过对目的和源 IP 地址进行 HASH 后实现的。也就是说 ,实际上是基于源、目地址对进行负载均衡的。

R3#show ip cef exact-route 10.1.35.5 1.1.1.1

10.1.35.5 -> 1.1.1.1 : FastEthernet1/0 (next hop 10.1.34.4)

通过上述命令，可以查看 R3 上，源为 10.1.35.5，目的为 1.1.1.1 的报文的实际转发出接口及下一跳，我们看到，其实这个源目地址对的流量，使用的实际下一跳是 10.1.34.4，出接口是 F1/0 口，也就是说，只要是来自 10.1.35.5，去往 1.1.1.1 的流量，恒定走 Fa1/0 口出去。

R3#show ip cef exact-route 10.1.35.55 1.1.1.1

10.1.35.55 -> 1.1.1.1 : FastEthernet0/0 (next hop 10.1.23.2)

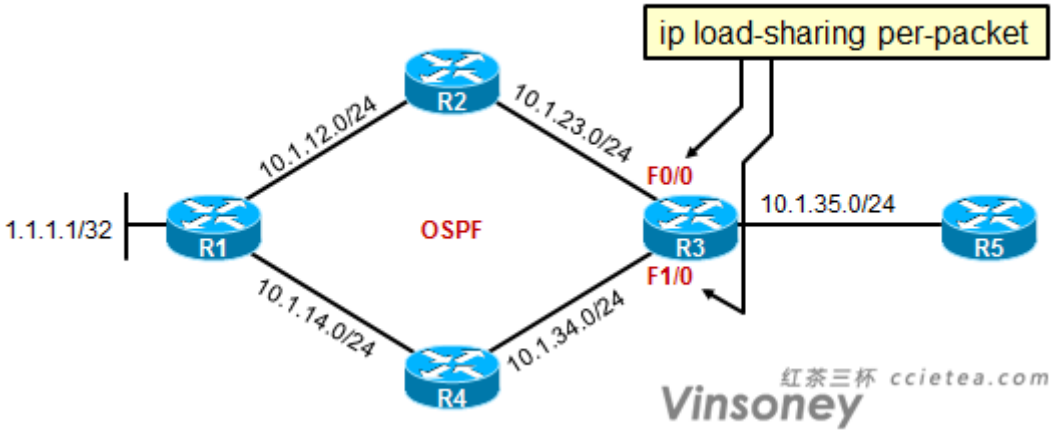
再看，现在看的是源为 10.1.35.55 目的为 1.1.1.1 的流量，我们发现 R3 使用了另一条等价路径，也即是从 F0/0 口出去。这就是基于源目地址对的含义。

在 R5 上可以使用：

traceroute 1.1.1.1 source 10.1.35.55

traceroute 1.1.1.1 source 10.1.35.5

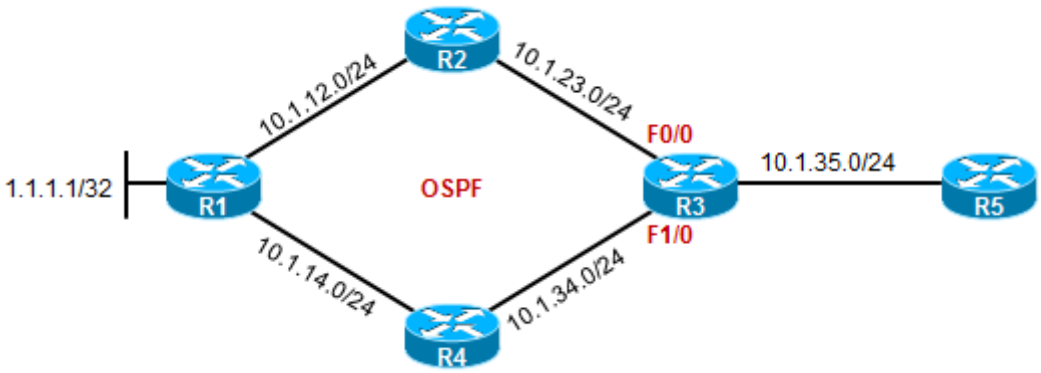
进行测试，看看数据的转发路径。那么如果希望**基于报文做负载均衡**呢？



在 R3 的 F0/0 及 Fa1/0 口上做如图配置即可。如此一来 ,即使是源目的地址不变 ,R3 也会在 F0/0 及 F1/0 接口上平均的分配流量。

2. 带标签报文的负载均衡

如果 MPLS 的有效负载是一个 IPv4 或者 IPv6 的报文的话，在基于目的的负载均衡环境中，CISCO IOS 使用 CEF 的哈希算法来选择出战接口。注意，如果有一条 IP（无标签）路径及一条带标签的路径拥有相同的度量值的话，只有带标签的路径才用于报文转发。这是因为在某些情况下，经过不带标签路径的流量无法到达目的地。例如 MPLS VPN 环境中的 P 路由器上。



现在，我们激活所有接口的 LDP。

然后在 R3 上：

R3#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
300	400	1.1.1.1/32	0	Fa1/0	10.1.34.4
	200	1.1.1.1/32	0	Fa0/0	10.1.23.2

我们看到，在 R3 的 LFIB 中，去往 1.1.1.1 这个目的地，有两条标签路径负载均衡。

那么实际上 ,这里默认的仍然是基于源目地址对的负载均衡。从测试的结果来看 ,10.1.35.55 访问 1.1.1.1 走的是 f.0/0，用的出站标签是 200，而 10.1.35.5 访问 1.1.1.1 走的是 F1/0，用的出站标签是 400。

在没有 IPv6 能力的 CISCO IOS 路由器中，对带标签的报文实施负载均衡的通用规则如下：

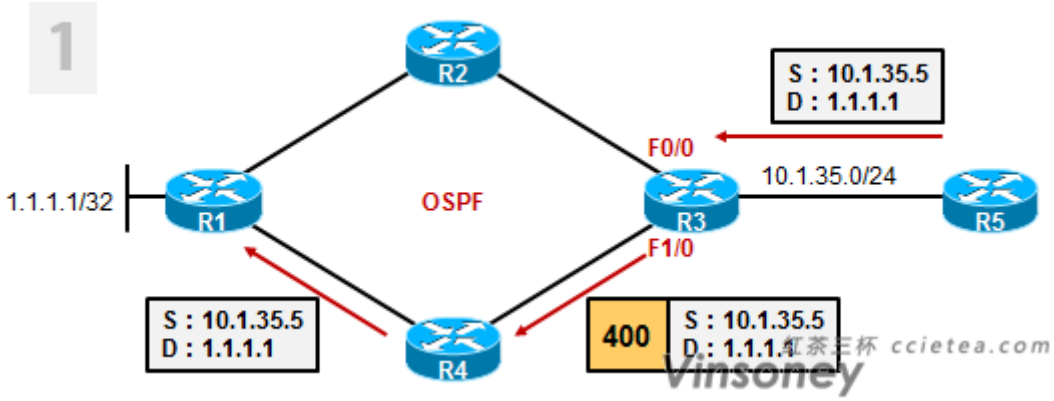
- 如果 MPLS 的有效负载是一个 IPv4 报文的话 ,负载均衡将通过对 IPv4 报文头部的源和目的 IP 地址进行哈希来实施；
- 如果 MPLS 的有效负载不是一个 IPv4 报文的话 ,负载均衡将通过查看 bottom 底部标签的值来进行。

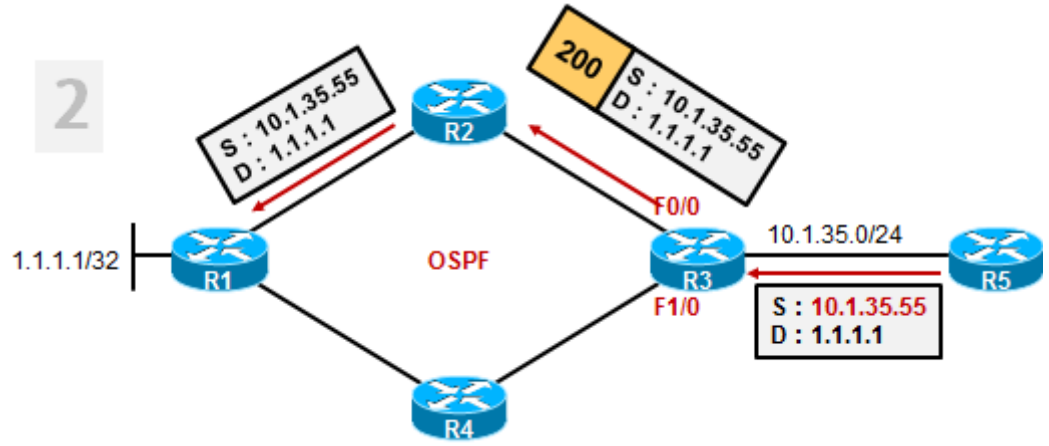
而对于支持 IPv6 的路由器，对于 MPLS 报文执行负载均衡的算法如下：

- 如果 MPLS 的有效负载是一个 IPv4 报文的话 ,负载均衡将通过对 IPv4 报文头部的源和目的 IP 地址进行哈希来实施；
- 如果 MPLS 的有效负载是一个 IPv6 报文的话，那么负载均衡就根据 IPv6 头部中的源和目的地址来进行
- 如果 MPLS 的有效负载不是一个 IPv4 报文也不是一个 IPv6 报文的话，负载均衡将通过查看底部标签的值来进行。

如果执行负载均衡的路由器是 IP 报文的入站 LSR，那么他很容易就能知道，收到的这个 IP 报文是 IPv4 还是 IPv6 的，因为它收到的是个 IP 报文，而不是标签包。

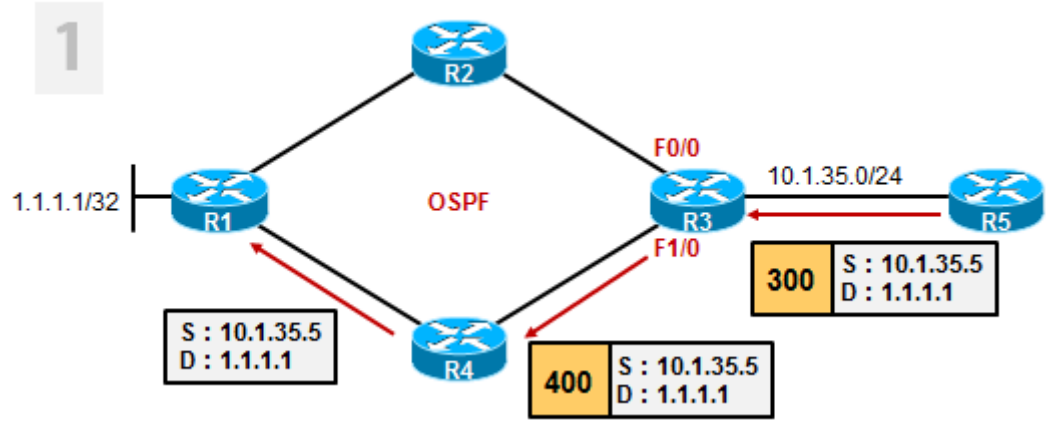
但是如果执行负载均衡的路由器是 P 路由器，它收到的是一个有多个标签的标签包，咋办？我们知道不管你有多少个标签，标签栈的后面，跟着就是 IP 的头部，而 IP 报文的第一个字段就是 version，如果为 4 就是 IPv4，如果为 6 就是 IPv6，因此 LSR 通过这个值来判断报文是 V4 还是 V6，再根据不同的 IP 版本来选择复杂均衡的算法。

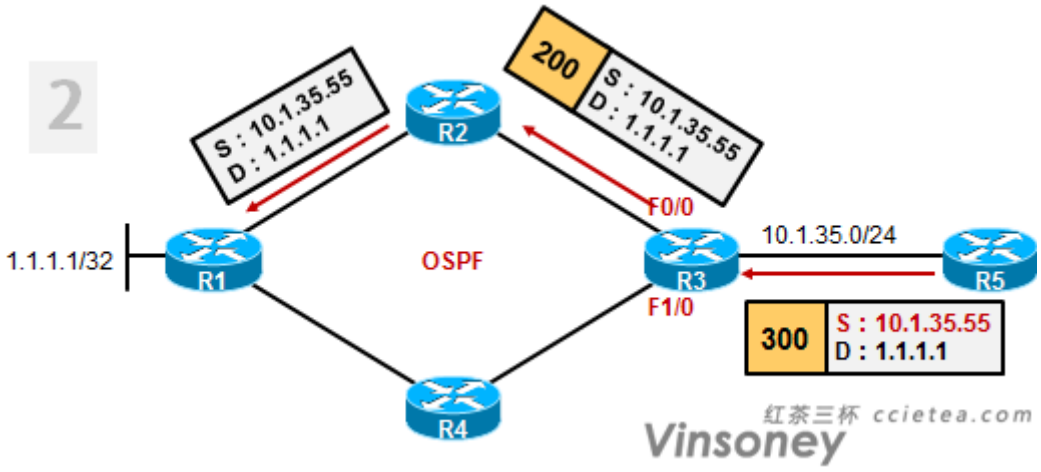




所以上图中，如果 R5 发给 R3 的是 IPv4 的数据包，那么 R3 能直接识别，并对 Ipv4 包头的源和目的地址进行哈希来实施负载均衡。

而如果 R3-R5 之间，跑起 LDP，让 R5 发送标签包，R3 收到这个标签包，查看标签栈后的字节，发现值为 4，因此确定是 IPv4 报文，于是将 IPv4 包头的源和目的地址拿出来做哈希执行负载均衡。





3 LDP

3.1 协议概述

MPLS 的基本特点就是所有的报文都是有标签的，那么如果让 OSPF、EIGRP、RIP、ISIS 这样的协议来分发标签是不可能的了。那么就需要一个新的协议，独立于所有的路由协议并且能够结合所有 IGP 一起使用，LDP 就是一个这样的协议。当然 BGP 因为运载的是外部路由，所以认为用 BGP 本身来分发标签更为有效，甚至非常完美，所以 BGP 可以实现多协议，用它来分发标签信息所产生的影响是非常小的。而且 BGP 是唯一一种在 AS 自治系统之间分发前缀的协议。

对 IP 路由表中的每一条 IGP 的 IP 前缀来说，每一台运行 LDP 协议的 LSR 都会进行本地捆绑，也就是说，为 IPv4 前缀分配标签，然后 LSR 再将该分配的标签分发给所有的 LSR 邻居。这些接收到的标签转换为远程标签 remote label，之后邻居将该远程标签和本地标签存储于一张特殊的表中，这个表就是**标签信息库 LIB**。通常一台 LDP 路由器会有多个 LDP 邻居，那么这些邻居都会给路由分配标签然后将这些标签传给自己。

在所有捆绑某一特定前缀的 remote label 中，LSR 只使用其中一个标签来确定该前缀的出站标签。RIB，也就是路由表来决定 IPv4 前缀的下一跳是什么。而 LSR 从下游 LSR 收到的远程标签中选择其路由表中到达该前缀的下一跳的标签。LSR 用这样的信息来创建它自己的**标签转发信息库 LFIB**。

注意在 CISCO IOS 中，LDP 不会为 BGP 的 IPv4 前缀捆绑标签。

3.2 LDP 邻接建立过程

1. **两个过程：邻居发现过程、会话建立过程**

2. 除了 HELLO 报文基于 UDP646 外，其他报文基于 TCP 端口号 646

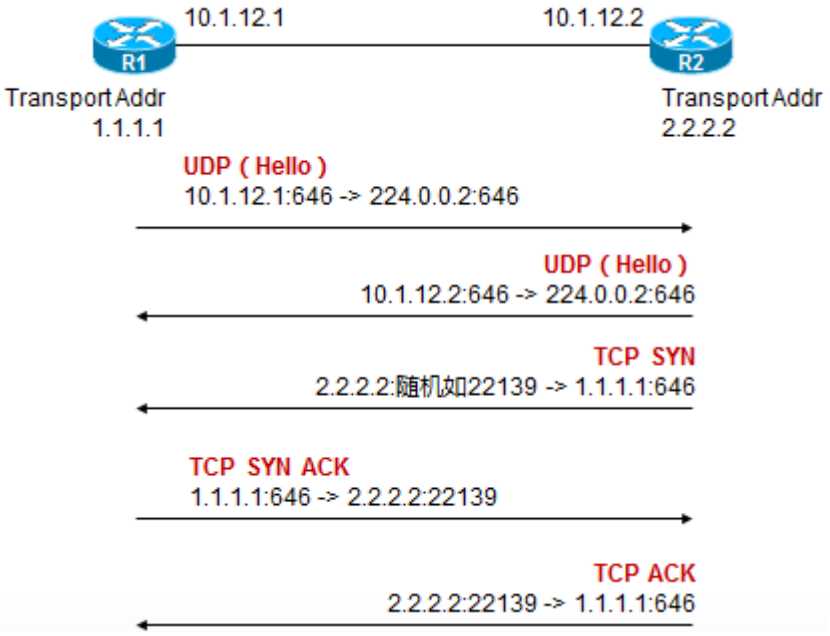
HELLO 包发向组播地址 224.0.0.2，源地址为**接口 IP**

3. 查看是否收到对方的 HELLO

验证是否收到对方的 LDP Hello

```
r1#sh mpls ldp discovery
    Local LDP Identifier:
    1.1.1.1:0
        Discovery Sources:
        Interfaces:
        FastEthernet0/0.12 (ldp): xmit/recv
        LDP Id: 2.2.2.2:0        // 2.2.2.2 为 router-id；0 为标签间隙
```

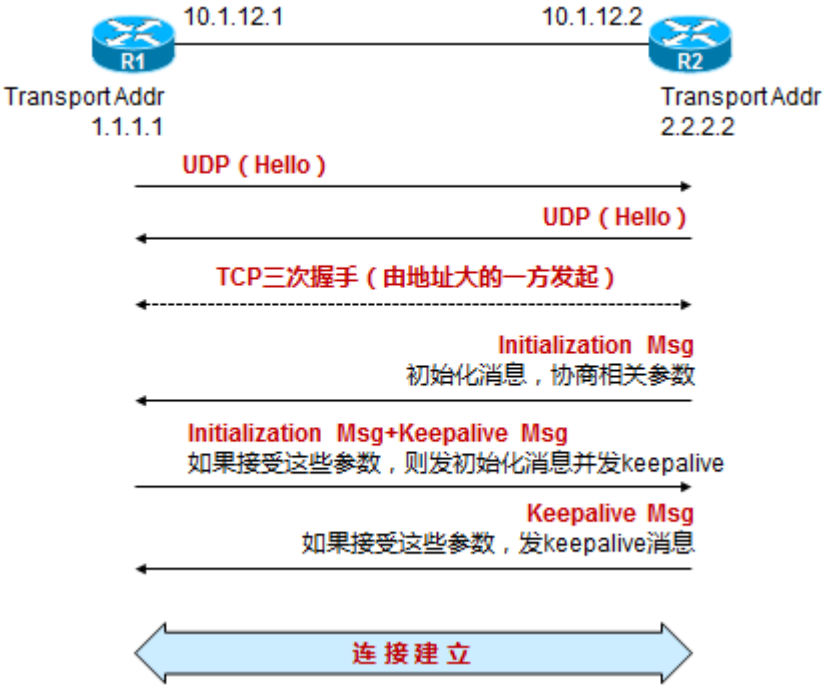
4. LDP 邻居发现



- 邻居发现是借助 UDP 的 Hello 包来进行的，这个 Hello 包源和目的端口都是 UDP646。
- 发现到邻居后，传输地址大的一方为主动发起方；注意，这里 TCP 握手报文，源地址是本地的 Transport Address，目的地址是对端的 Transport Address。传输地址为默认情况下为路由器的 LDP Router-ID。所以必须保证两个 Transport Address 之间是路由可达的。

  否则可考虑使用接口下的 mpls ldp discovery transport-address interface，将 Transport Address 配置为直联接口的 IP 地址。

5. LDP 会话建立



连接建立成功后，开始交互初始化消息，初始化消息中包含各种参数。对方也发送自己的初始化消息，并且如果接收前者的初始化消息中的各项参数，则发送一个 keep alive 消息表示接受。到此 LDP 的邻居关系就建立起来了，接下去就可以互相传递标签映射消息了。

6. LDP 非直连邻居

LDP 允许非直连邻居，这样一来邻居发现无需借助组播的 HELLO 包，而是采用单播包。

3.3 LDP ID

LDP ID 是一个 6B 的字段，是 LSR 的 LDP 标示符。包含 4B 的 LSR 标示符和 2B 的标签空间。

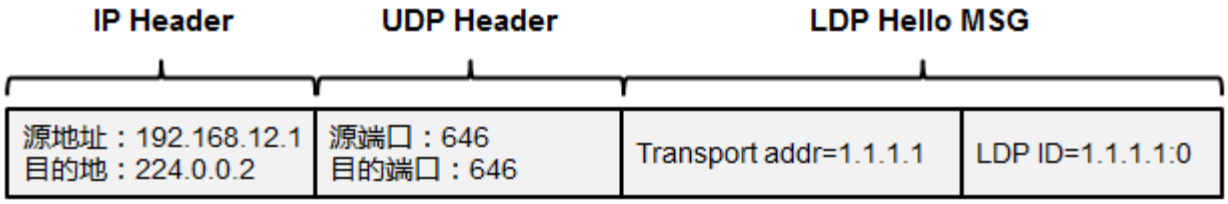
- LDP IP 的前 4B 是当存在 loopback 接口时，loopback 接口的最大 IP，如果没有 lp 接口，则是活动物理接口的最大 IP，使用如下命令可以更改：

  mpls ldp router-id *interface* [force]

  加 force 关键字则为马上生效，否则只有当前面选作 routeriD 的接口 DOWN 掉的时候，才会进行重新选择。注意：LDP 邻居的 routerID，必须在本地有可达的路由。
- 如果 2B 的标签空间都是 0 ,那么就是基于设备的标签空间 ,如果是非 0 ,就说明用的是基于接口的标签空间。那么在这种情况下，可以使用多个 LDP ID，这些 LDP ID 的前 4 字节都是相同的，但是后 2 个字节标识不同的标签空间，基于每个接口的标签空间用于 LC-ATM。

3.4 消息格式

1. HELLO 消息



LDP 邻居建立首先发送 HELLO 包（HELLO 包使用 UDP，源目端口都是 646）

LDP ID 为 6 个字节（4 字节的 IP 加 2 字节的 LABEL SpaceID）

两个路由器建立 LDP 邻居，要保证双方到对方的 LDP ID 三层可达

Transport addr 除非手工指定（mpls ldp discovery transport-address），否则一般等于 LDP ID。

注意这里（上面所描述的）其实是有三种地址：接口 IP、Transport addr、LDP ID

LDP ID 的选举和 OSPF routerID 一样

```
Internet Protocol, Src: 10.1.12.1 (10.1.12.1), Dst: 224.0.0.2 (224.0.0.2)
User Datagram Protocol, Src Port: ldp (646), Dst Port: ldp (646)
Label Distribution Protocol
    Version: 1
    PDU Length: 30
    LSR ID: 1.1.1.1 (1.1.1.1)
    Label Space ID: 0
  Hello Message
      0... .... = U bit: Unknown bit not set
      Message Type: Hello Message (0x100)
      Message Length: 20
      Message ID: 0x00000000
    Common Hello Parameters TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: Common Hello Parameters TLV (0x400)
        TLV Length: 4
        Hold Time: 15
        0... .... .... .... = Targeted Hello: Link Hello
        .0.. .... .... .... = Hello Requested: Source does not request periodic hellos
        ..00 0000 0000 0000 = Reserved: 0x0000
    IPv4 Transport Address TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: IPv4 Transport Address TLV (0x401)
        TLV Length: 4
        IPv4 Transport Address: 1.1.1.1 (1.1.1.1)
```

使用如下命令可以看到相关信息：

R1#show mpls ldp discovery detail

```
Local LDP Identifier:
```

1.1.1.1:0

Discovery Sources:

Interfaces:

FastEthernet0/0 (ldp): xmit/recv

Enabled: Interface config

Hello interval: 5000 ms; Transport IP addr: 1.1.1.1

LDP Id: 2.2.2.2:0; no host route to transport addr

Src IP addr: 10.1.12.2; Transport IP addr: 2.2.2.2

Hold time: 15 sec; Proposed local/peer: 15/15 sec

Reachable via 2.2.2.0/24

Password: not required, none, in use

2. Initialization 消息

```
Transmission Control Protocol, Src Port: 22139 (22139), Dst Port: ldp (646), Seq: 1, Ack: 1
Label Distribution Protocol
  Version: 1
  PDU Length: 32
  LSR ID: 2.2.2.2 (2.2.2.2)
  Label Space ID: 0
⊟ Initialization Message
    0... .... = U bit: Unknown bit not set
    Message Type: Initialization Message (0x200)
    Message Length: 22
    Message ID: 0x00000001
  ⊟ Common Session Parameters TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Common Session Parameters TLV (0x500)
      TLV Length: 14
    ⊟ Parameters
        Session Protocol Version: 1
        Session KeepAlive Time: 180
        0... .... = Session Label Advertisement Discipline: Downstream Unsolicited proposed
        .0.. .... = Session Loop Detection: Loop Detection Disabled
        Session Path Vector Limit: 0
        Session Max PDU Length: 0
        Session Receiver LSR Identifier: 1.1.1.1 (1.1.1.1)
        Session Receiver Label Space Identifier: 0
```

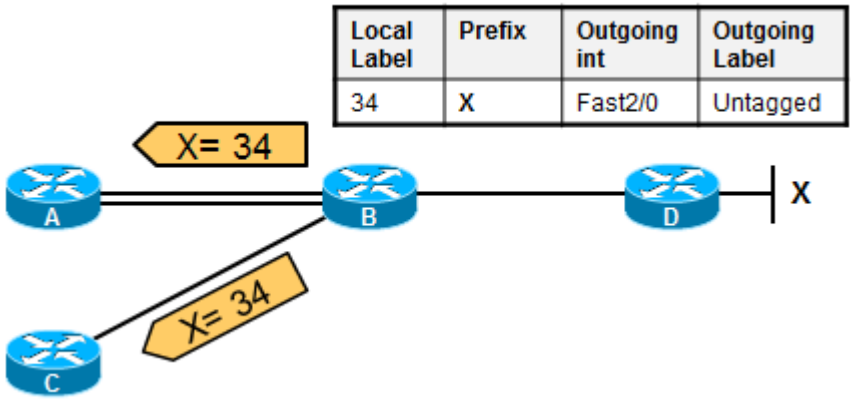
3. Label Mapping 消息

一个 LDP 报文中会承载多个标签映射消息

每个标签映射消息包含两要素:FEC TLV 和 Label TLV

```
Label Mapping Message
    0... .... = U bit: Unknown bit not set
    Message Type: Label Mapping Message (0x400)
    Message Length: 24
    Message ID: 0x00000029
  Forwarding Equivalence Classes TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Forwarding Equivalence Classes TLV (0x100)
      TLV Length: 8
    FEC Elements
      FEC Element 1
          FEC Element Type: Prefix FEC (2)
          FEC Element Address Type: IPv4 (1)
          FEC Element Length: 32
          Prefix: 9.9.0.1
  Generic Label TLV
      00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
      TLV Type: Generic Label TLV (0x200)
      TLV Length: 4
      Generic Label: 3
```

3.5 标签空间



默认的 LDP 标签空间是基于平台的 per-platform 或者说基于设备的。什么意思呢，我们看 B 路由器，它为前缀 X 捆绑了标签 34，并且将这个标签捆绑信息发布给所有的 LDP 邻居，给大家的都一样，而且人人有份，都是标签 34。这就是基于平台。那么除此之外还有基于接口的标签空间。

Label space:Per-Platform

- Label Space ID 一般为 0，表示我们的标签是基于平台（Per-platform）的标签空间
- LFIB 表不包含入接口信息
- 为前缀分配的标签在本地任意 MPLS 接口可用并且会分发给所有 LSR 邻居
- 本地分配的标签会分发给邻居，如果与单个邻居有多条连接，则该标签在所有连接上均有效。那么不管本地

从哪个接口上收到一个标签包，只要有这个标签，都会对其进行交换。

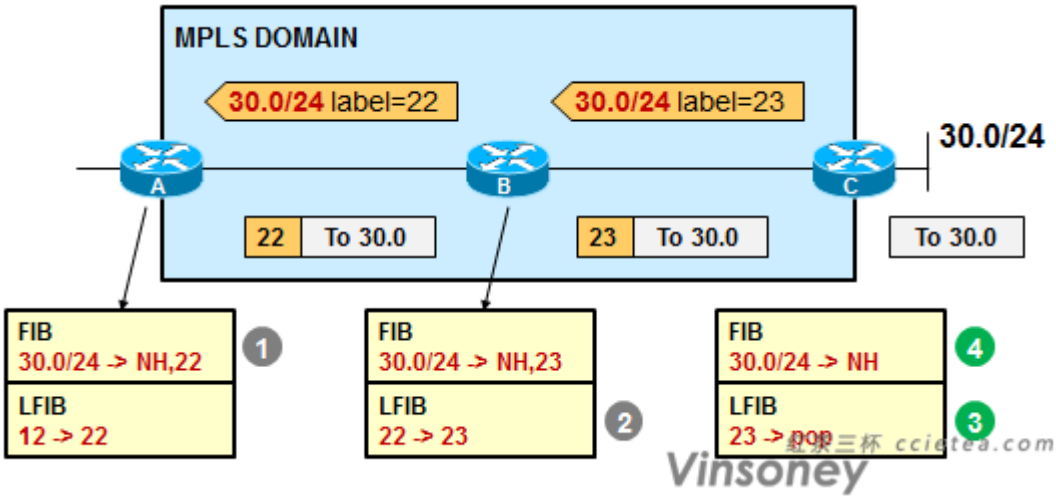
- 基于平台（per-platform）的标签空间相比于基于接口（per-interface）的标签空间安全性要更低

Negotiating Label space

- **LSRs 为每个标签空间只维护一个 LDPsession**
  基于平台（Per-platform）的标签空间只需要一条 LDPsession，即使在 LDP 邻居间存在多条冗余链路
- **基于平台（Per-platform）的标签空间 label space ID=0**
  如 LDP ID = 1.1.1.1:0

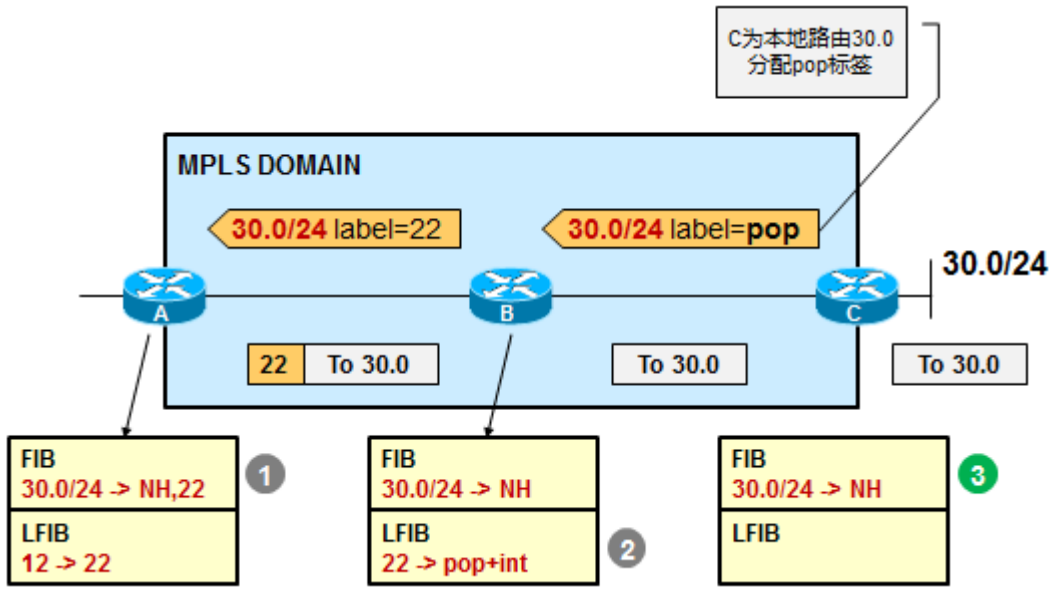
3.6 PHP 倒数第二跳弹出机制

如果没有 PHP 机制：



- 关于 30.0/24 这个前缀，C 分配的标签是 23，这个映射传递给了 B；B 本地给 30.0/24 分配的标签是 22，这个映射传递给了 A。
- 现在 A 下面有用户发送数据到 30.0 网络，A 将数据压上标签头，标签值为 22。标签包到了 B，B 将标签替换成 23，然后传递给 C。
- C 上，先查找 LFIB 表，发现要将标签弹出，于是它将标签弹出，弹出后发现是个 IP 报文，于是又去查 FIB 表，最终将这个 IP 数据包转发出去。C 进行了两次查找。这降低了转发效率。
- 标签可以在（倒数第二跳）上弹出，C 只需查找 FIB 表将收到的 IP 报文进行转发

有了 PHP 机制：



- 有了 PHP 倒数第二跳弹出机制的话，C 为本地的直连的前缀分配 POP 标签并通告给其他 LDP 邻居
- 如此一来，B 收到一个 A 发送过来的标签值为 22 的标签包，会将标签弹出得到 IP 包，再转发给 C，则 C 仅需对 IP 包进行 FIB 表的查找和转发。

LDP 在帧模式（Frame Mode）下，LSR 会为每一条路由分配一个标签；而为本地的直连路由分配的是 POP 标签。

倒数第二跳弹出机制有两种标签，一是 POP 或 implicit null，在 LDP 中标签值为 3；另一个是 explicit null，在 LDP 中标签值为 0。如果收到邻居发送来的关于某条路由分配的标签值为 3，则我发送数据给该邻居时，我会将该标签弹出，再将内层数据转给邻居。而如果邻居关于某条路由分配的标签值为 0，那么本地在转数据给邻居时，会带上标签（为 0 的），一并发给邻居。

这里要留意的是，如果收到一个标签包，标签为 0，则直接弹出标签，并将数据交给 FIB 进行查找，不会有两次查找的损耗。标签为 0 的标签包，为什么不干脆将标签去掉（分配个 13 值给路由下一跳让下一跳将标签去掉啊），为什么还要保留这个为 0 的标签头呢？这是为了在某种情况下保持网络规划的统一性，例如部署了 MPLS 的 QoS，则需使用标签包中的 EXP 字段，那么就需要有标签。在实施 QoS 时，最后一跳必须携带 exp 位，因此标签不能被弹出，需配置 mpls ldp explicit-null,此时分配给特定路由的标签值为 0 并传递给 LDP 邻居（如倒数第二跳）。

3.7 保留的标签

标签 0-15 都是被保留的标签。以下是一些有特定作用的保留标签：

- 标签 0　　显式空标签
- 标签 3　　隐式空标签

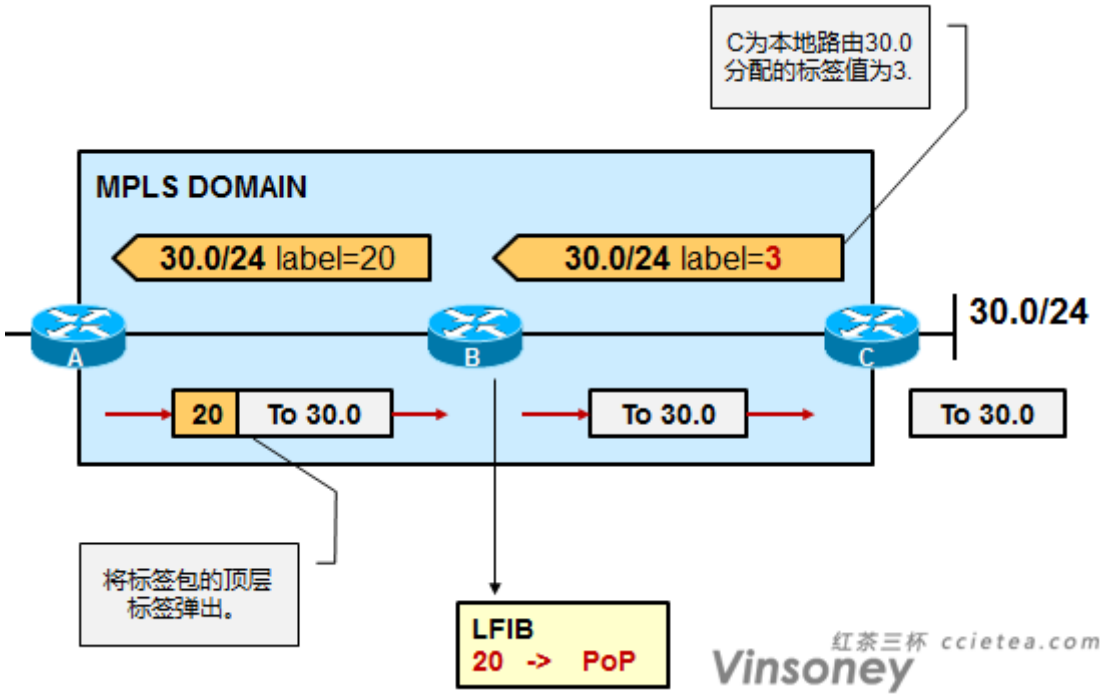
- 标签 1　　路由器报警标签
- 标签 14　　OAM 报警标签

其他 0-15 之间的被保留标签的功能目前暂时没有定义。因此我们的可用标签是 16 到 1048575（2^{20}-1）

1. 隐式空标签

在 PHP 中，我们已经了解了隐式空标签的作用，当然，隐式空标签不局限在 PHP 中。它还可以运用在标签栈中有 2、3 个或者更多的标签的报文中。在出站 LSR 上使用隐式空标签（在 LDP 中，值为 3）将会通知倒数第二跳路由器移除顶层标签，而向出站 LSR 传递的带标签报文其标签数量就会少一个，这样的话，出站 LSR 就不需要执行两个标签的查找了。注意，使用隐式空标签并不是必须将标签栈中的所有标签都弹出，而是弹出顶层标签。

尽管隐式空标签也使用了一个标签值为 3 的标签，但是标签 3 永远不会出现在 MPLS 报文的标签栈中，这也正是其叫隐式空标签的原因。

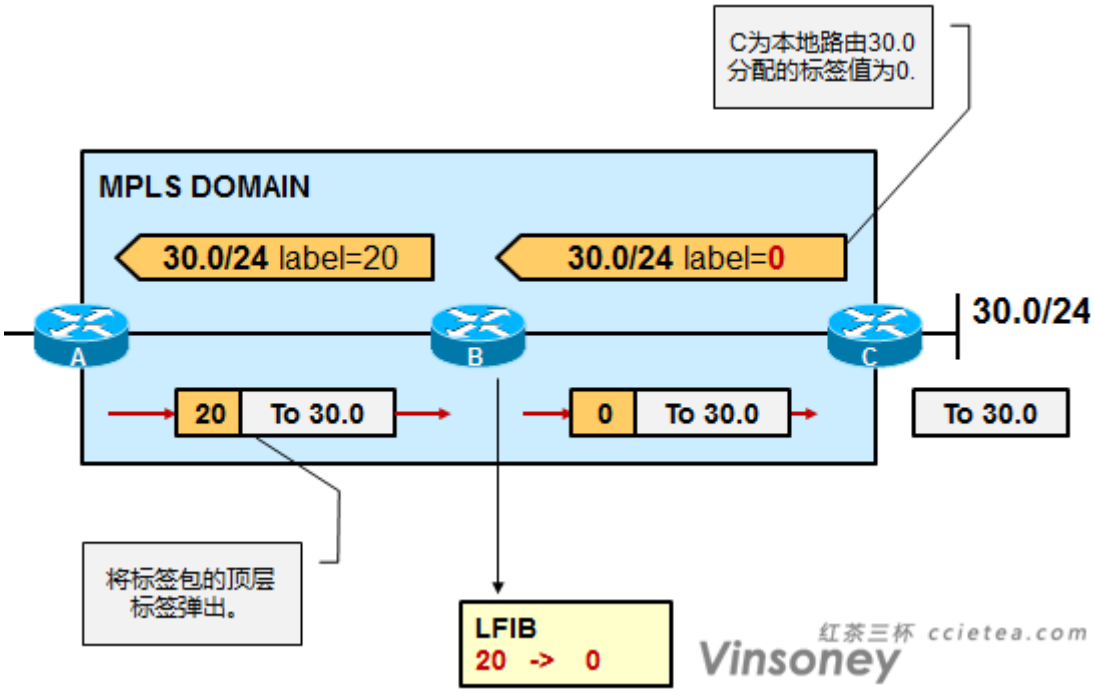


2. 显式空标签

在 IPv4 中，显示空标签为 0，ipv6 中为 2。

上面的隐式空标签已经介绍过了，它确实可以增加效率，但是也有一个问题，因为如果我收到一个下游邻居发送过来的关于某个特定前缀捆绑的隐式空标签，那么我在转发标签数据给该邻居之前，我会先将顶层标签弹出，那么这个弹出的动作，实际上是将整个顶层标签头都弹出了，也就是连带着标签字段、EXP 等字段都弹出了，而 EXP 我们知道，用于做 QoS 的，它也被弹出了，意味着这里就丢失了用于 QoS 的部分信息。

因此我们又定义了显式空标签，用于应对上面描述的场景。



我们看上图，C 针对 30.0/24 的前缀捆绑了标签 0，也就是显式空标签，然后将标签映射发给 B，B 也产生自己的标签映射然后发给 A。那么这时候，如果 B 收到来自 A 的一个标签包，顶层标签的值为 20，那么 B 查找自己的 LFIB，发现要标签要转换成 0。于是，B 将顶层标签替换成 0，然后转发给 C，那么这个时候对于 C 来说，它就收到了一个标签值为 0 的标签包，C 不能通过在 LFIB 中查找标签值 0 来转发这样的报文，因为这个标签值可以分配给多个 FEC，C 只是仅仅弹出 0 标签也就是显式空标签，之后不得不进行另外一种查找 ,虽然这里不得不进行两次查找 ,但是 C 就可以通过查看标签头的 EXP 位来获得该报文的 QoS 信息了。

3. 路由器报警标签

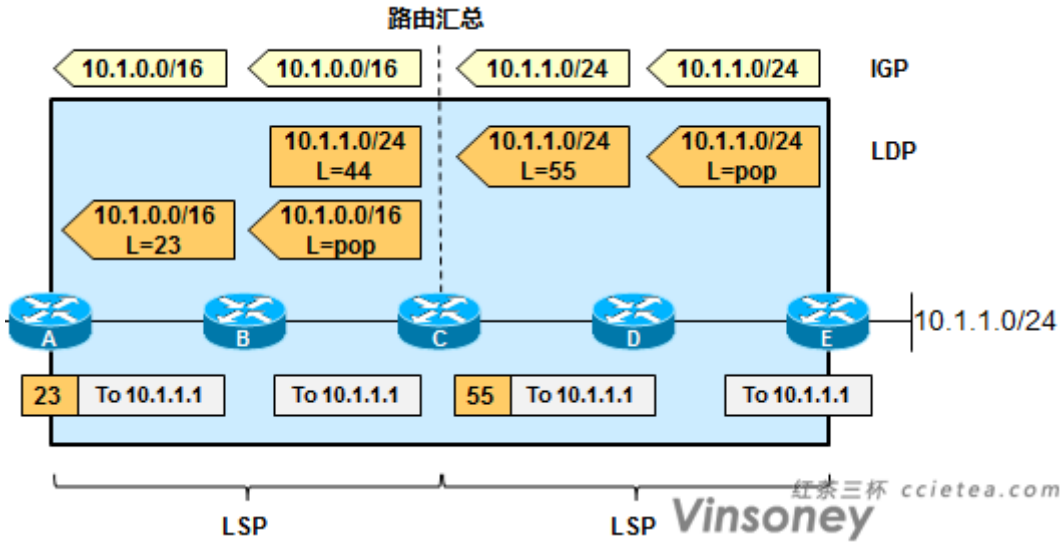
标签值为 1，这个标签可以出现在标签栈的任何位置，除了栈底位外。

当路由器报警标签位于栈顶时，它向 LSR 发出警告说该报文需特别注意。这样一来该报文就不会通过硬件传输，而是通过软件进程传输。一旦这个报文开始被转发，标签 1 首先被移除，接下来 LSR 在 LFIB 中对标签栈中的下一个标签进行查找然后执行相应的标签操作(添加、移除、交换) ,标签 1 又会被添加到标签栈的顶部，最后才被转发出去。

4. OAM 报警标签

OAM 基本上用于错误检测、定位和监控实施。该标签将普通报文和 OAM 报文区分开来。CISCO IOS 不使用标签 14，它会执行 MPLS OAM 但不是通过标签 14 来实现。

3.8 路由汇总对 MPLS 的影响



E 上有条路由 10.1.1.0/24 被通告了出来，现在在 C 上做了汇总，汇总路由是 10.1.0.0/16 并向 B 通告。

- 路由汇总将原先的一段 LSP 分割成了两段
- A 将数据压上标签 23，到了 B，将标签头弹出交给 C（PHP 机制）；C 收到这个 IP 包去查找 FIB 表，又将包压上标签 55 交给 D，D 最后查找 LFIB 并将标签弹出将 IP 包丢给 E，E 将该 IP 包转发到目的地。
- 在这个环境中看似没有什么问题，但是路由汇总在点到点的 LSP 环境下就有问题了，例如 MPLS VPN、TE 等。

3.9 MPLS 模式

1. **分配模式:Label Allocation**

// 本地为一条路由前缀绑定一个标签的前提条件

独立控制模式:Independent Control

只要本地通过 IGP 学习到路由前缀，就会为这条路由前缀分配标签(本地也会为直连路由分配 POP 标签)

有序控制模式:Odered Control

本地通过 IGP 学习到路由前缀，但必须该路由前缀的下一跳路由器将该前缀所对应的标签映射消息通告给本地，本地才会为该前缀分配标签

2. **分发模式:Label Distribution**

// 本地将一个标签映射消息通告给邻居的前提条件

下游主动模式:Downstream Unsolicited

本地会主动将所生成的标签映射消息通告给所有 LDP 邻居

下游按需模式:Downstream On Demand

只有邻居向本地请求某条前缀的标签映射消息时,本地才会通告标签映射消息给邻居

3. **保留模式:Label Retention**

// 本地是否会在数据库中保留从邻居接收到的所有标签映射消息

自由模式: Liberal Retention

本地将从邻居接收的所有标签映射消息都保存在数据库中

保守模式: Conservative Retention

本地仅保存最优路由下一跳邻居所通告的该路由前缀的标签映射消息

- **标签空间:Label Space**

// 本地所通告出去的标签是对局部(接口)有意义还是对全局有意义

基于平台: Per-Platform

本地通告出去的标签映射消息对全局有意义,从不同的接口通告出去的同一 FEC 所对应的标签相同

基于接口: Per-Interface

本地通告出去的标签映射消息对局部有意义,从不同的接口通告出去的同一 FEC 所对应的标签不同

【注意】帧模式(Frame Mode)的标签行为

分配模式(Label Allocation)： 独立控制模式(Independent Control)

分发模式(Label Distribution)： 下游主动模式(Downstream Unsolicited)

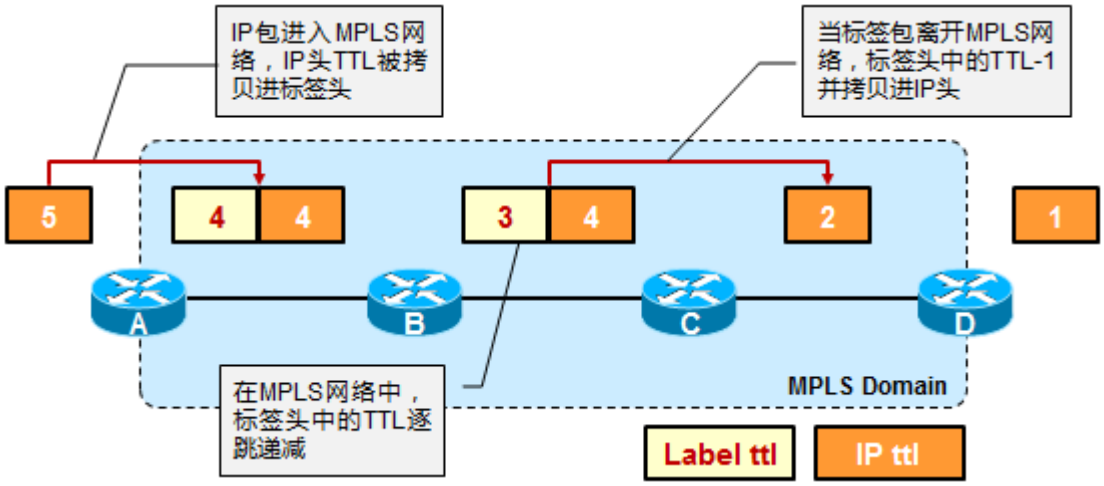
保留模式(Label Retention)： 自由模式(Liberal Retention)

标签空间(Label Space)： 基于平台(Per-Platform)

3.10 LOOP Detection

- LDP 的环路检测机制依赖于 IGP 协议
- 如果出现的环路（一般是 IGP 出了问题，如静态路由的配置错误），标签头中的 TTL 将防止标签包无止尽的被转发
- 标签头中的 TTL 与 IP 头中的 TTL 是一样的 ,通常拷贝自 IP 头中的 TTL 值(当一个 IP 包进入 MPLS 网络时), 这是 TTL propatation

1. 传统的 TTL 操作



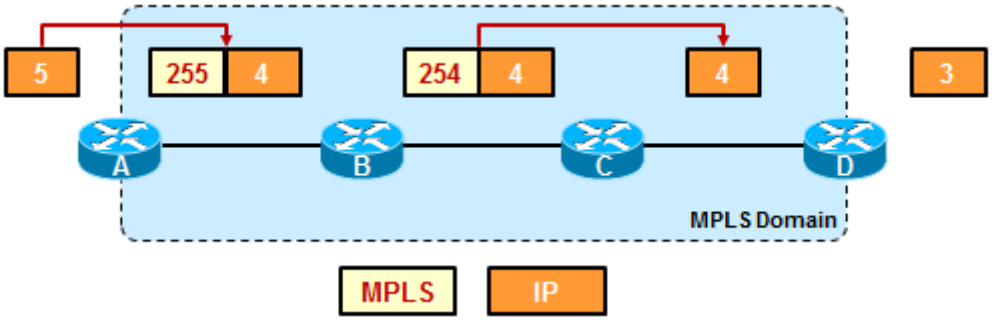
数据包被 A 压上标签 ,标签的头的 TTL 值拷贝 IP 头中的 TTL(当然要先减 1),并且随着数据帧在 MPLS 网络中传输，MPLS 头中的 TTL 值在递减，到 0 则丢弃，直到数据帧出了 MPLS 域，那么 IP 头中的 TTL 才开始工作。在出站时，标签头中的 TTL 减 1 后拷贝到 IP 头中的 TTL。

这里实际上存在一定的隐患，例如使用 traceroute，可能会暴露网络内部结构（TTL=0 后，路由器会返回差错消息，就可能会暴露网络信息）。

我们来看一下，为什么会暴露网络信息？假设 A 下面有个路由器 E，E 发了一个数据包要到 D 下的 F，数据包被 A 打上标签转给 B ,假设在 BC 之间出现了环路随后标签 TTL 递减至 0 ,由于 TTL 值设置的非常小，因此很快，TTL 就递减到了 0，这时候 B（报文丢弃者），就会发送 ICMP 出错消息回 E，那么这时候如果 B 有 E 的路由（或标签），B 就会直接将 ICMP 出错消息发回 E，如果没有，B 会继续将出错信息打上标签然后逐渐传递到 D，D 作为 PE，有关于 E 的路由，于是再装上标签送回给 E。那么无论什么情况，最终 E 都能收到由 B 产生的 ICMP 差错消息。

这时可以关闭 TTL propagation（CISCO 路由器默认开启）

2. 关闭 TTL propagation



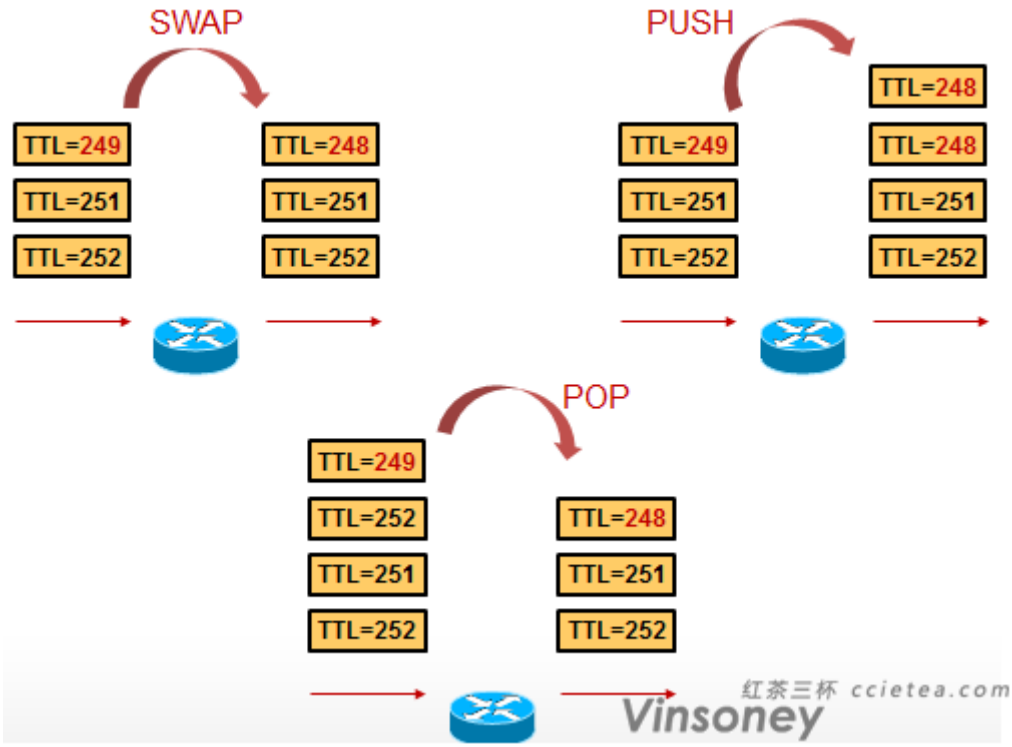
在 A 设备（一般在边界设备）上配置该特性 no mpls ip propagate-ttl

这样一来在边界设备上将 IP 包加标签头的时候，就不去拷贝原 IP 头里的 TTL 值了，而是用一个如 255 的 TTL 放入标签头。关闭 TTL propagation 可以避免 MPLS 网络被暴露（通过 traceroute 方式）。

```
no mpls ip propagate-ttl [ forwarded | local ]
```

forwarded 关键字表示这条命令针对穿越本路由器的流量生效。Local 关键字表示针对本地产生的流量生效

3. 补充知识点：在 SWAP、PUSH、POP 操作中标签到标签的 TTL 扩散行为



前面描述的是 IP-标签包的 TTL 扩散过程，现在来看一下标签到标签的 TTL 扩散过程。

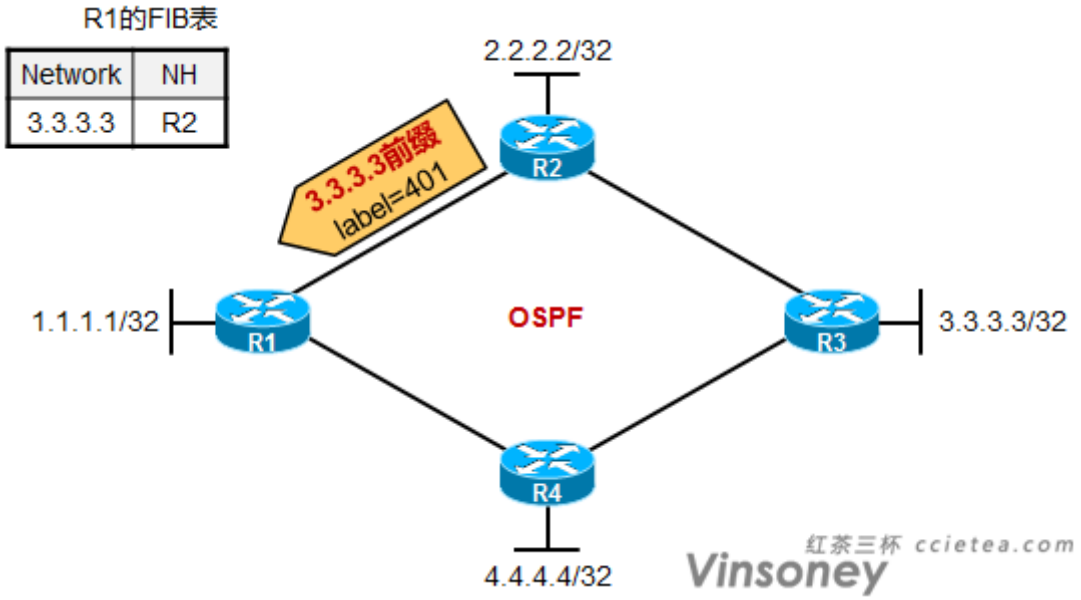
SWAP 这个过程很好理解，注意，LSR 在处理标签栈中有多个标签的标签数据时，只会处理顶层标签。因此 SWAP 这个过程，标签在交换后，入站标签 TTL -1，然后拷贝到出站标签 TTL。

PUSH 也是类似的理解，只对顶层标签操作，首先入站顶层标签的 TTL249 先减 1，然后新压入的标签头 TTL 拷贝这个值。

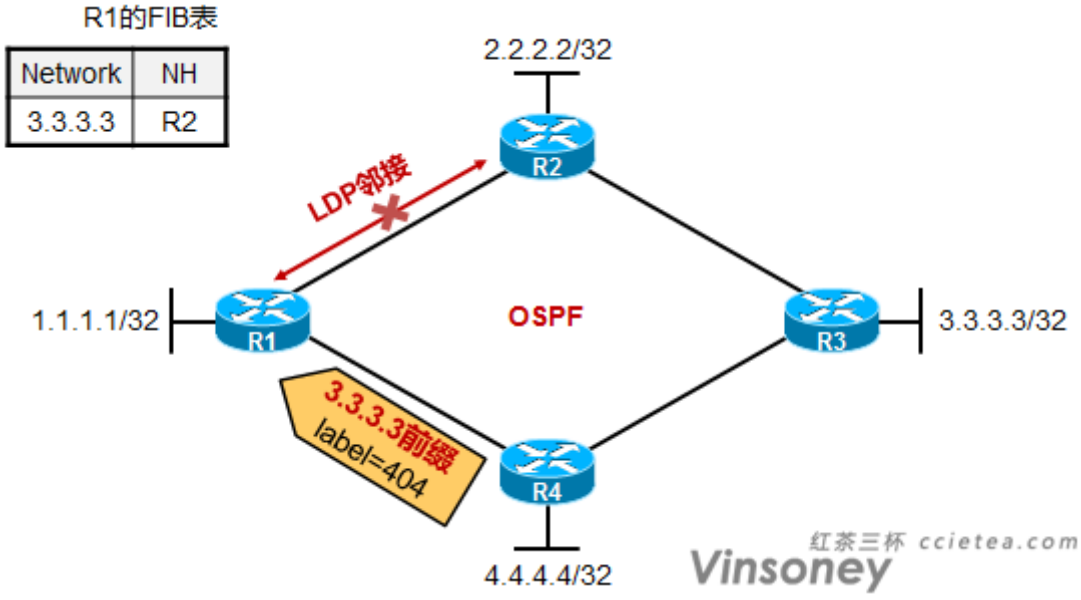
POP 则是顶层标签的 TTL 先减去 1，然后弹出，新的 TTL 值被写入到出站数据的顶层标签上。

3.11 LDP 与 IGP 的同步

1. 关于 LDP 与 IGP 的同步



MPLS 网络中的一个很重要的问题就是 LDP 和 IGP 的同步，所谓的**同步的是意思是，IGP 和 LDP 都认可某条链路为待转发报文的出站链路的**。例如上图中，四台路由器都运行 OSPF，同时激活 LDP。那么在 R1 上，如果其去往 3.3.3.3/32 的路由，在路由表中下一跳为 R2（假设我们调了 cost），同时又收到了 R2 发过来的针对 3.3.3.3/32 前缀捆绑的标签 401，那么这时候，IGP 和 LDP 就同步了，R1 可以正常的使用 R2 作为下一跳来转发 MPLS 标签报文。



然而可能出现这样一个问题，R1、R2 之间的 LDP 连接，由于某种原因断掉了，但是 R1、R2 之间的 OSPF 邻接没 DOWN，那么 R1 的路由表中，关于 3.3.3.3/32 依然选择 R2 作为下一跳，这就出问题了。R1 上的 LFIB 里，关于 3.3.3.3/32 的条目可能是这样的：

R1#sh mpls for

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop

104 **Untagged** 3.3.3.3/32 0 Fa0/0 10.1.12.2

这样，R1 发送数据去往 3.3.3.3 就有直接以 IPv4 报文的形式发送出去了，这在本拓扑中看似没什么问题，确实能够通，但是，在 MPLS VPN 环境中就可能出问题了。因为报文在 MPLS VPN Backbone 中传输往往是需要带标签的。

另一个造成 IGP 和 LDP 不同步的例子是，LSR 重启的时候，IGP 可以很快就建立起邻接关系，而 LDP 的会话建立起来可能就慢点，也就是说，在 LFIB 装载必须的信息来进行正确的标签交换之前，IGP 的转发就已经开始了。于是报文可能就错误的转发，或者在进入 MPLS 网络后在某处被丢弃。

可以使用 MPLS LDP-IGP 同步来解决这个问题。注意，MPLS LDP-IGP 的解决方案不能用于 BGP 的标签分发。

2. MPLS LDP-IGP 同步如何工作

当在一个接口上激活了“MPLS LDP-IGP 同步“之后，IGP 将会通告该链路的度量值为最大，一直到同步完成或者 LDP 会话在该接口上成功建立。在 OSPF 中，metric 最大为 65535。这样一来，可以保证在 LDP 处于断开状态的时候不会有路径会经过该接口。在 LDP 会话成功建立并且标签捆绑已经开始进行交换以后，IGP 才会用正常的 IGP 度量值来通告该链路。这个时候穿越这个接口的流量就是进行标签交换的流量了。基本上说，如果 LDP 会话没有建立，OSPF 是不会在这个链路上建立邻接关系的，压根不发 HELLO（当然，这里有个基本上说的字眼，也就说，还有二般情况）。

一直到 LDP 会话成功建立，或者“同步保持时间”超时，否则 OSPF 邻接关系是不会建立的。这里的同步表示本地标签捆绑已经通过 LDP 会话发送给了 LDP 对等体。但是如果路由器上 A 激活了“MPLS LDP-IGP 同步 “，并且 A 和 B 之间只有一条链路，而没有其他可用路径到 B 的话，OSPF 邻接关系将永远不会建立。因为 OSPF 会等待 LDP 会话的建立，但是 LDP 会话根本无法建立，因为 A 没办法在它的路由表中学习到 B 的 LDP routerID 路由，这样就进入了一个死循环，OSPF 和 LDP 的邻接关系将永远无法建立。对于这种情况，LDP-IGP 同步会无条件激活 OSPF 的邻接关系，这样一来链路将通告最大的 metric 值，直到同步完成。

然而在某些环境中，LDP 会话的这个问题可能永远无法解决，因此可能并不希望永远等待 IGP 邻接关系的建立。那么可以通过配置“同步保持时间”。

3. MPLS LDP-IGP 同步的配置

MPLS LDP-IGP 同步是在 IGP 进程中启用的，它会应用到所有运行了该 IGP 的接口上

```
router ospf 1
  mpls ldp sync
```

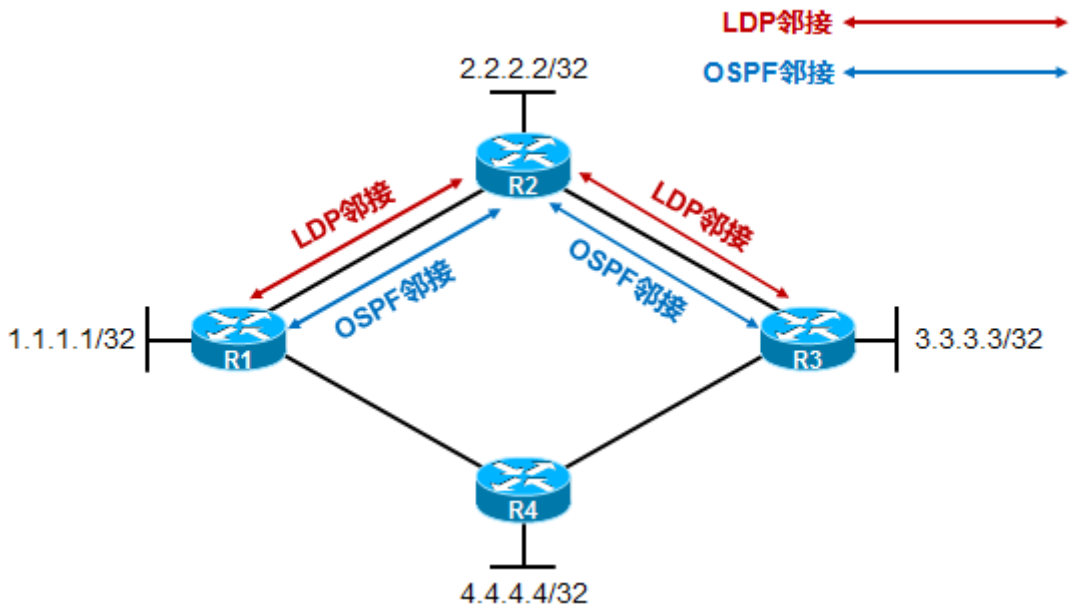
可以在某个特定的接口上使用 no mpls ldp igp sync 命令来关闭接口的同步功能。

默认情况下，如果同步没有完成，IGP 并没有明确在建立邻接关系之前所需等待的时间，可以通过：

mpls ldp igp sync holddown *msecs*

这条全局命令来进行修改。在该计时器到期之后，IGP 将在该链路上建立邻接关系，一旦 IGP 的邻接关系建立成功同时 LDP 会话还未同步的话，IGP 将通告该链路的度量值为最大。

4. MPLS LDP-IGP 同步示例



为了先保证实验环境的简洁 ,我先忽略 R4 以及 R4 直连链路的存在 ,大家当他不存在就好 ,等会儿丫才上场。

R1、R2、R3 运行 OSPF，通告直连及自己的 Loopback 口。并且在直连接口上都激活 LDP。

现在是一个正常的情况，

R1#show mpls forwarding-table （这是 R1 的 LFIB 表）

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
101	204	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.12.2
104	**201**	3.3.3.3/32	0	Fa0/0	10.1.12.2
105	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2

R1 的路由表如下（直连路由忽略了）:

```
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/2] via 10.1.12.2, 00:00:37, FastEthernet0/0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/3] via 10.1.12.2, 00:00:37, FastEthernet0/0
O       10.1.23.0 [110/2] via 10.1.12.2, 00:00:37, FastEthernet0/0
O       10.1.34.0 [110/3] via 10.1.12.2, 00:00:39, FastEthernet0/0
```

现在我们将 R2 上，完成如下配置：

```
access-list 100 deny    udp any any eq 646
access-list 100 permit ip any any
```

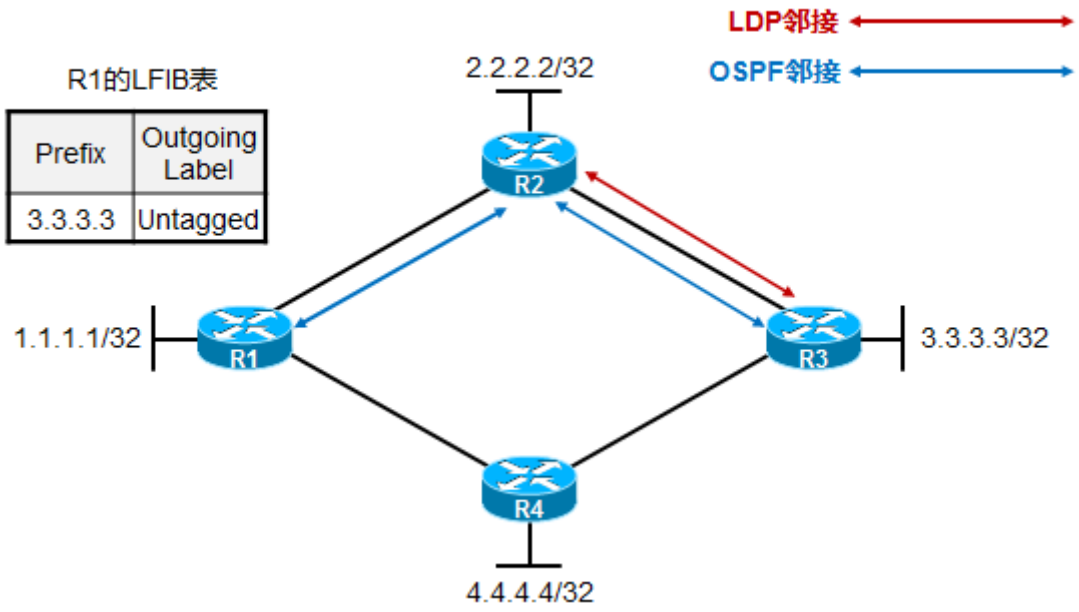
然后将 ACL 应用在 R2 上，连接 R1 的接口上，in 方向

这样，R2 将忽略掉接口上收到的，来自 R1 的 LDP hello 包，然后 R1-R2 之间的 LDP 邻接过一会就 DOWN 了。这时候：

R1#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
101	**Untagged**	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	**Untagged**	2.2.2.2/32	0	Fa0/0	10.1.12.2
104	**Untagged**	3.3.3.3/32	0	Fa0/0	10.1.12.2
105	**Untagged**	10.1.23.0/24	0	Fa0/0	10.1.12.2

我们看到 R1 上，相关路由前缀的 outgoing label 都是 untagged。而 R1 的路由表暂时没有任何变化。在现在这个环境中，虽然 R1 仍然能够 ping 通 3.3.3.3，但是实际上已经出问题了，因为这是直接走的 IPv4 包，而不是标签包。



现在我们在 R1、R2 上配置 MPLS LDP-IGP 同步：

```
router ospf 1
  mpls ldp sync
```

注意，这时候，虽然 R1、R2 之间的 LDP 邻接已经断了，原则上说，LDP 邻接不起来，在开启同步的情况下，

OSPF 邻接关系是无法建立的，但是这里 R1 只有一条可用路径到达 R2，因此 OSPF 邻接无条件建立。与此同时 R1 及 R2 对外通告 10.1.12.0/24 这段直连链路，将以最大的 metric 65535 来通告。这样做的目的是，如果网络环境是冗余链路的环境，那么可以让从 IGP 的角度让这条链路的 metric 最差，从而使得 LDP 路径绕开这段链路。我们在 R3 上来瞄一眼：

R3#show ip route

```
      1.0.0.0/32 is subnetted, 1 subnets
O        1.1.1.1 [110/65537] via 10.1.23.2, 00:09:07, FastEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
O        2.2.2.2 [110/2] via 10.1.23.2, 00:09:07, FastEthernet0/0
      10.0.0.0/24 is subnetted, 3 subnets
O        10.1.12.0 [110/65536] via 10.1.23.2, 00:09:07, FastEthernet0/0
```

我们看到，由于 R2 更新出来的 LSA 中，关于 10.1.12.0/24 这个直连网段，metric 设置成了 65535，因而，在 R3 的路由表里，我们看到 1.1.1.1/32 及 10.1.12.0/24 的路由 metric 都离奇的大。再看一下 R2 产生的 1 类 LSA：

R3#show ip ospf database router 2.2.2.2

```
            OSPF Router with ID (3.3.3.3) (Process ID 1)
                Router Link States (Area 0)
  LS age: 705
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 2.2.2.2
  Advertising Router: 2.2.2.2
  LS Seq Number: 8000001F
  Checksum: 0x580A
  Length: 60
  Number of Links: 3

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 2.2.2.2
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1                          !! 2.2.2.2 链路的 metric 不变

    Link connected to: a Transit Network
```

```
    (Link ID) Designated Router address: 10.1.23.2
    (Link Data) Router Interface address: 10.1.23.2
     Number of TOS metrics: 0
      TOS 0 Metrics: 1                          !! 10.1.23.2 的也不变

   Link connected to: a Transit Network
    (Link ID) Designated Router address: 10.1.12.2
    (Link Data) Router Interface address: 10.1.12.2
     Number of TOS metrics: 0
      TOS 0 Metrics: 65535                      !!出问题的链路,metric 被设置成了 65535
```

R1 的路由表如下：

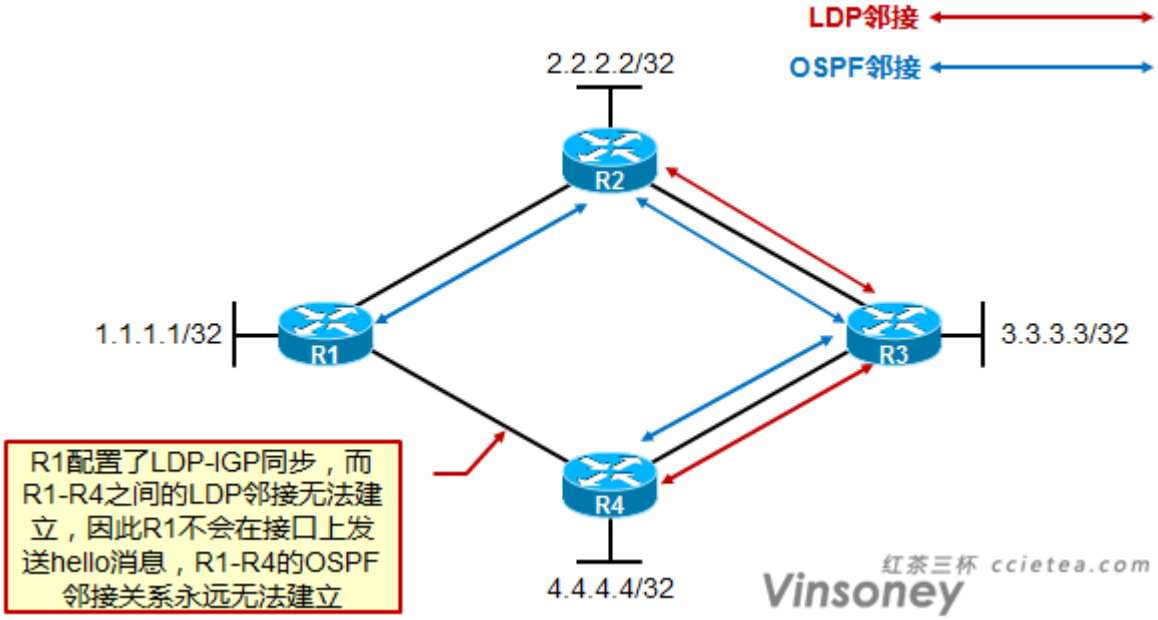
```
O        2.2.2.2 [110/65536] via 10.1.12.2, 00:00:47, FastEthernet0/0
O        3.3.3.3 [110/65537] via 10.1.12.2, 00:00:47, FastEthernet0/0
O        10.1.23.0 [110/65536] via 10.1.12.2, 00:00:48, FastEthernet0/0
```

看到没 R1 将直连链路 10.1.12.0 的 cost 调成了 65545。

R1#sh ip os mpls ldp interface fa0/0

```
FastEthernet0/0
  Process ID 1, Area 0
  LDP is not configured through LDP autoconfig
  LDP-IGP Synchronization : Required
  Holddown timer is not configured
  Interface is up and sending maximum metric
```

现在我们将 R4 加进来，注意，R4 在这个时候只配 OSPF，先不在连接 R1 的接口上配置 mpls ip（或者使用 ACL 过滤掉 LDP 报文），这样使得 R1、R4 之间的 LDP 邻接无法建立，我们来观察一下现象。

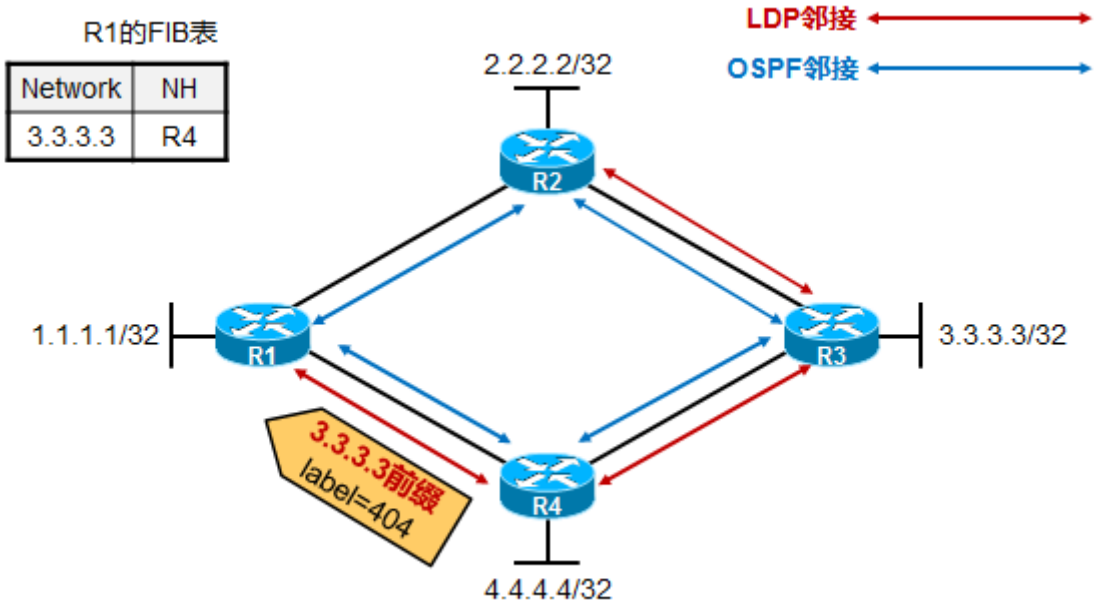


注意，由于 R1 激活了 MPLS LDP-IGP 同步，因此，在 R1-R4 之间的 LDP 邻接关系建立起来之前，R1 上连接 R4 的接口是不会发送 OSPF HELLO 包的，也就是说 R1-R4 的 OSPF 邻接关系是永远无法建立的。当然，我们也不希望看着 R1-R4 这么拧巴下去，对谁都不好，是吧？ 所以在 R1 上来配置个：

```
mpls ldp igp sync holddown 5000
```

将同步 holddown 计时器设置为 5S，这样一来 5S 超时后，R1-R4 就建立起来了 OSPF 邻接关系。

又或者，我让 R1-R4 之间的 LDP 邻接建立起来，那么 R1-R4 之间的 OSPF 自然也就起来了。好了，我们现在让 R1-R4 之间的 LDP 邻接关系起来，那么随之 OSPF 邻接关系马上也会自动建立起来。



现在一来，由于 R1 上有 MPLS LDP-IGP 同步，R1 将 10.1.12.0/24 这段链路 metric 调到 65535（其实这时候 R1-R2 之间的 OSPF 邻接关系应该 DOWN 掉的，保存配置将 R1 重启，会发现 R1-R2 之间的 OSPF 邻接关系起不来了），使得 R1 优选 R4 作为去往 3.3.3.3 网络的下一跳，同时 R1-R4 之间又维持着 LDP 邻接关系，

因此，R1 将只采用 R4 关于 3.3.3.3 的标签映射，也就是使用标签 404 来发送标签包到 3.3.3.3。

3.12 MTU 问题

```
nterface fast0/0
 mpls mtu mtu-size
```

接口级命令，修改该接口的 MPLS mtu，要注意 mpls mtu 不能大于接口 mtu

4 MPLS 配置（frame mode）

4.1 基础命令

1. **基础配置**

```
lp cef
!
mpls ldp router-id loopback0                    !! ldp 的 routerid 使用 loopback 口 IP
mpls label protocol ldp                         !! 标签协议使用 LDP（默认就是 LDP）
mpls label range 100 199                        !! 指定本地标签的范围，这个在实验中可以极大的方便观
察现象及排错
!
Interface loopback
   Ip address 10.1.255.1 255.255.255.255
Interface fast 0/0
   Ip address 10.1.12.1 255.255.255.0
   mpls ip                                      !! 接口上激活 mpls，实际上是激活 ldp
```

2. 邻居关系建立

Router(config)# mpls ldp discovery hello interval *seconds*

修改 ldp hello 消息发送间隔，默认 5S

Router(config)# mpls ldp discovery holdtime *seconds*

修改 ldp holdtime，默认 15S

如果两个 LDP 对等体的 LDP 保持时间配置的不同，那么其中较小的那个值将被用作 LDP 发现的保持时间。注意 LDP HELLO 消息中没有携带 hello interval 时间，只携带了 holdtime，因此如果两端配置了不同的 holdtime，那么小的那么值将被用作 LDP 发现的保持时间。Cisco IOS 可能会对已配置的 LDP Hello 间隔进行重写。

Router(config)# mpls ldp backoff *initial-backoff maximum-backoff*

initial-backoff 默认 15s、maximum-backoff 默认 120s。

当两台 LDP 邻居对等体在交换参数时发现又不匹配的话，这条命令可以减缓两台 LDP LSR 之间的尝试性 LDP 会话建立时间，如果会话建立失败，那么下一次再尝试的时间间隔会成倍数增长，直到 maximum-backoff 计时器超时。

4.2 Show 及 debug

```
Show ip cef detail
Show tcp brief
Show mpls ldp discovery detail
Show mpls ldp parameters                    //查看协议的基本参数
Show mpls ldp neighbor
Show mpls ldp discovery
show mpls forwarding-table
show mpls ldp bindings
```

```
show mpls ldp parameters
Protocol version: 1
Session hold time: 180 sec; keep alive interval: 60 sec
Discovery hello: holdtime: 15 sec; interval: 5 sec
Discovery targeted hello: holdtime: 90 sec; interval: 10 sec
```

Downstream on Demand max hop count: 255

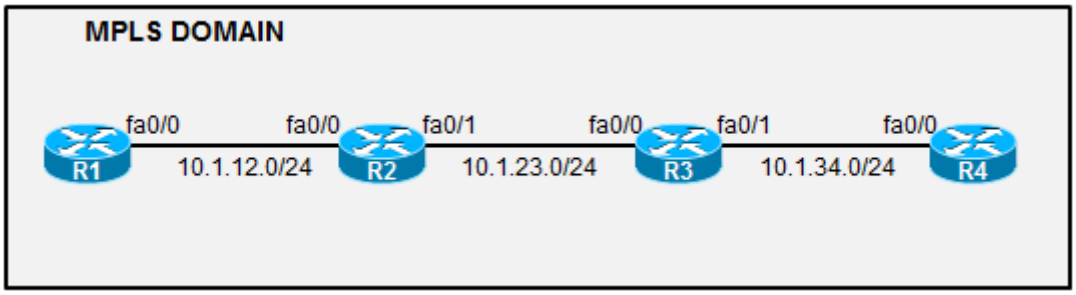
Downstream on Demand Path Vector Limit: 255

LDP for targeted sessions

LDP initial/maximum backoff: 15/120 sec

LDP loop detection: off

4.3 基础实验 1



1. **实验环境**
   - R1、R2、R3、R4 运行 OSPF，宣告直连接口，以及 Loopback 接口，Loopback 口 IP 为 **x.x.x.x/32**，x 为设备编号，该 IP 同时为 LDP routerID。
   - 设备互联网段如图所示 ,例如 10.1.23.0/24 ,这是 R2-R3 互联地址段 ,那么 R2 的接口 IP 就是 10.1.23.2，R3 的接口 IP 就是 10.1.23.3、
   - 在所有设备上激活 LDP，为了方便观察现象，为每台设备指定 label range，如 R1 的 label range 为 100 199，R2 的为 200 299，其他设备依此类推。

2. **实验需求**
   - 认识 FIB、LIB、LFIB 表
   - 了解 LDP 邻居关系建立过程
   - 了解数据在 MPLS 域中的转发过程

3. **实验配置**

R1 的配置如下：

```
Ip cef                                    !! 注意，运行 MPLS，IP cef 必须打开
Interface fas0/0
  Ip address 10.1.12.1 255.255.255.0
Interface loopback0
```

```
   Ip address 1.1.1.1 255.255.255.255
!
router ospf 1
   router-id 1.1.1.1
   network 10.1.12.1 0.0.0.0 area 0
   network 1.1.1.1 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 100 199
interface fast0/0
   mpls ip
```

R2 的配置如下：

```
Ip cef
Interface fas0/0
   Ip address 10.1.12.2 255.255.255.0
Interface fas1/0
   Ip address 10.1.23.2 255.255.255.0
Interface loopback0
   Ip address 2.2.2.2 255.255.255.255
!
router ospf 1
   router-id    2.2.2.2
   network 10.1.12.2 0.0.0.0 area 0
   network 10.1.23.2 0.0.0.0 area 0
   network 2.2.2.2 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 200 299
interface fast0/0
   mpls ip
interface fast1/0
   mpls ip
```

R3 的配置如下：

```
ip cef
Interface fas0/0
   Ip address 10.1.23.3 255.255.255.0
Interface fas1/0
   Ip address 10.1.34.3 255.255.255.0
Interface loopback0
   Ip address 3.3.3.3 255.255.255.255
!
router ospf 1
   router-id 3.3.3.3
   network 10.1.23.3 0.0.0.0 area 0
   network 10.1.34.3 0.0.0.0 area 0
   network 3.3.3.3 0.0.0.0 area 0
!
mpls ldp router-id loopback0
mpls label range 300 399
interface fast0/0
   mpls ip
interface fast1/0
   mpls ip
```

R4 的配置如下：

```
ip cef
Interface fas0/0
   Ip address 10.1.34.4 255.255.255.0
Interface loopback0
   Ip address 4.4.4.4 255.255.255.255
!
router ospf 1
   router-id 4.4.4.4
   network 10.1.34.4 0.0.0.0 area 0
   network 4.4.4.4 0.0.0.0 area 0
!
```

```
mpls ldp router-id loopback0
mpls label range 400 499
interface fast0/0
  mpls ip
```

4. **实验现象**

完成上述配置后，我们可以检验一下，现在是路由全网是互通的。

R1#show mpls ldp neighbor

```
    Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 2.2.2.2.31044 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 16/16; Downstream
        Up time: 00:05:38
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 10.1.12.2
        Addresses bound to peer LDP Ident:
          2.2.2.2         10.1.12.2       10.1.23.2
```

上面是 R1 上显示的 LDP 邻居，有一个 LDP 邻居，它的 LDP routerID 是 2.2.2.2，label spaceID=0，说明是基于平台的标签空间。

TCP connection: 2.2.2.2.31044 - 1.1.1.1.646，表示这个 LDP 连接是建立在 TCP 的 1.1.1.1 源端口 646，到目的地 2.2.2.2 的 31044 端口。因为 2.2.2.2 地址大，所以它是发起方。

R1#show mpls ldp bindings （查看 R1 的 LIB 表）

```
tib entry: 1.1.1.1/32, rev 2
        local binding:   tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
tib entry: 2.2.2.2/32, rev 6
        local binding:   tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 3.3.3.3/32, rev 13
        local binding:   tag: 103          !! 本地为前缀 3.3.3.3/32 分配的标签
        remote binding: tsr: 2.2.2.2:0, tag: 202   !! 邻居 R2 为前缀 3.3.3.3/32 分配的标签
tib entry: 4.4.4.4/32, rev 14
        local binding:   tag: 104
        remote binding: tsr: 2.2.2.2:0, tag: 203
```

tib entry: 10.1.12.0/24, rev 4

local binding: tag: imp-null

remote binding: tsr: 2.2.2.2:0, tag: imp-null

tib entry: 10.1.23.0/24, rev 8

local binding: tag: 101

remote binding: tsr: 2.2.2.2:0, tag: imp-null

tib entry: 10.1.34.0/24, rev 12

local binding: tag: 102

remote binding: tsr: 2.2.2.2:0, tag: 201

一旦 LDP 激活后，LSR 会为路由表中的前缀在本地产生一个标签，然后和前缀捆绑在一起，将这个标签映射消息发送给所有的 LDP 邻居。当我收到 LDP 邻居发来的（remote binding），针对某些前缀的标签捆绑后，我会将这些标签，以及我本地为特定前缀捆绑的标签（local binding），放置于 LIB 中。

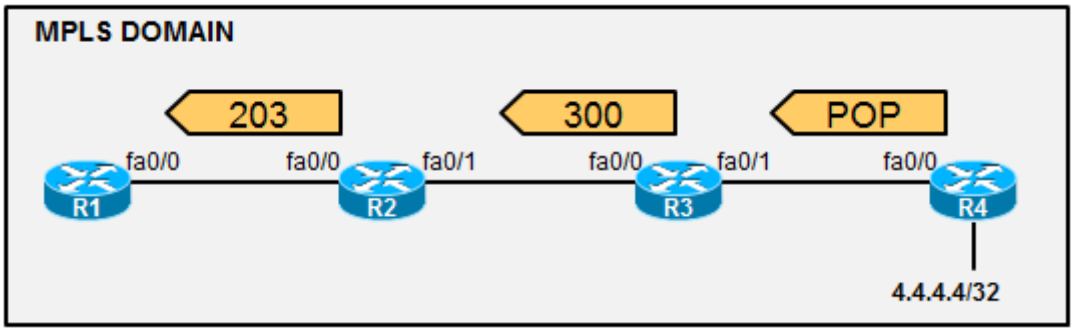
当然，并不是 LIB 中的 remote 标签都会被用上，我们还需结合 FIB 表，来获得有关前缀的下一跳信息。最后形成 LFIB 表：

R1#show mpls forwarding-table　　R1 的 LFIB 表

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.12.2
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2
104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

好，现在我们来分析一下，当 R1 要发送数据去往 R4 的 Loopback 4.4.4.4，数据是如何传送的。

首先分析一下控制层面：



由于大家都通过 OSPF 学习到了 4.4.4.4/32，那么所有的 LSR 都会为 4.4.4.4/32 在本地产生一个标签，然后将这个标签捆绑在前缀上传递给其他 LDP 邻居，如图所示。

好，那么现在当 R1 要去 ping 4.4.4.4 时，R1 得查自己的 FIB，也就是 CEF 表，注意，这是一个 IP 查找：

R1#show ip cef 4.4.4.4

```
4.4.4.4/32, version 12, epoch 0, cached adjacency 10.1.12.2
0 packets, 0 bytes
  tag information set
    local tag: 104
    fast tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
  via 10.1.12.2, FastEthernet0/0, 0 dependencies
    next hop 10.1.12.2, FastEthernet0/0
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.12.2, tags imposed: {203}
```

CEF 的条目指示，要去往 4.4.4.4，需要给 IP 报文压上一层标签，值为 203，然后将数据包丢给下一跳 10.1.12.2，从 Fa0/0 口扔出去。

接下来 R2 收到这个标签包，R2 从这个数据包的二层以太网帧头的类型字段，知道了这是一个标签包，因此它去查找自己的 LFIB 表：

R2#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
200	Pop tag	1.1.1.1/32	0	Fa0/0	10.1.12.1
201	Pop tag	10.1.34.0/24	0	Fa1/0	10.1.23.3
202	Pop tag	3.3.3.3/32	0	Fa1/0	10.1.23.3
203	**300**	**4.4.4.4/32**	**0**	**Fa1/0**	**10.1.23.3**

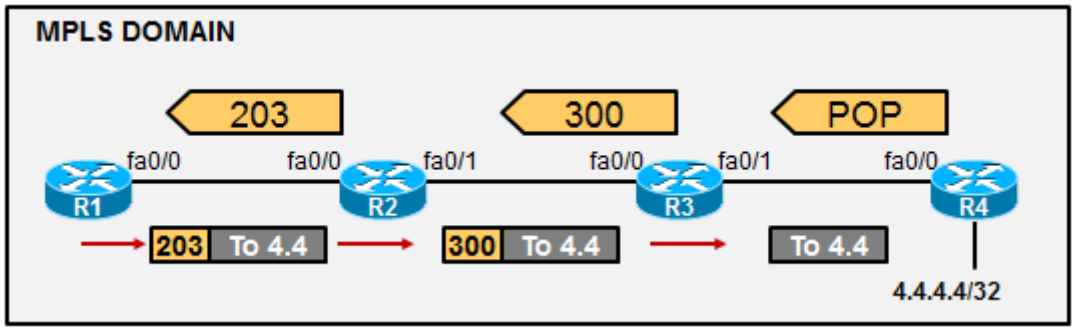
这个入站的标签包，标签值为 203，那么在 R2 的 LFIB 表中指示，203 需要交换成 300，然后丢给下一跳 10.1.23.3 从 Fa1/0 口送出去。于是，R2 将标签替换成 300，然后丢给了 R3。

接下来 R3 收到了这个标签包，同样，查看自己的 LFIB：

R3#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
300	**Pop tag**	**4.4.4.4/32**	**0**	**Fa1/0**	**10.1.34.4**
301	Pop tag	10.1.12.0/24	0	Fa0/0	10.1.23.2
302	200	1.1.1.1/32	0	Fa0/0	10.1.23.2
303	Pop tag	2.2.2.2/32	0	Fa0/0	10.1.23.2

R3 发现，入站标签 300 的标签包，出站标签是个 POP，于是他将顶层标签弹出（实际上就一层），然后直接将弹出后的数据丢给 10.1.34.4 ,注意 ,这时候它无需再次查找 FIB 表 ,因为 LFIB 表中已经有下一跳信息了。最终这个数据被传到了 R4。



我们可以验证一下：

R1#traceroute 4.4.4.4

Type escape sequence to abort.

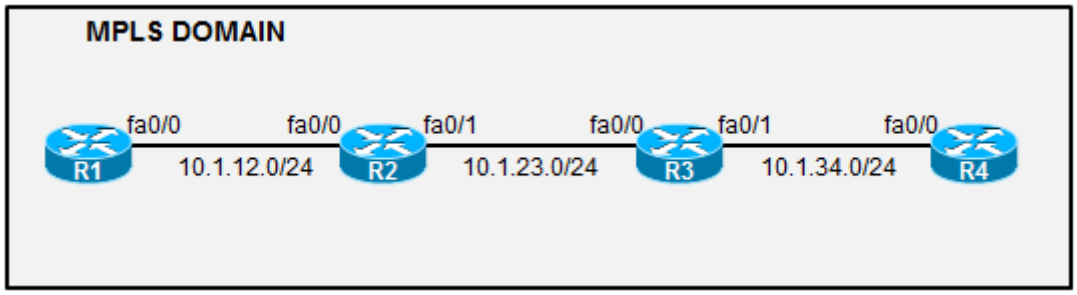
Tracing the route to 4.4.4.4

1 10.1.12.2 **[MPLS: Label 203 Exp 0]** 200 msec 84 msec 136 msec

2 10.1.23.3 **[MPLS: Label 300 Exp 0]** 108 msec 116 msec 64 msec

3 10.1.34.4 52 msec * 120 msec

5. 关于 MPLS 环境下 OSPF 的一点小话题



前面的配置，中所有设备的 Loopback 都是/32 的，因此没出啥问题。现在我们再看看 R1.

R1#show mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	**Pop tag**	**2.2.2.2/32**	**0**	**Fa0/0**	**10.1.12.2**
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2

104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

重点关注一下 2.2.2.2/32 这个条目，现在的 outgoing label 是 pop，也就是空标签。

那么现在这样的，我们将 R2 的 loopback 口地址改为 2.2.2.2/24，改成/24 的，来看看会有什么现象。

R1#sh mpls forwarding-table

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface	Next Hop
100	**Untagged**	**2.2.2.2/32**	**0**	**Fa0/0**	**10.1.12.2**
101	Pop tag	10.1.23.0/24	0	Fa0/0	10.1.12.2
102	201	10.1.34.0/24	0	Fa0/0	10.1.12.2
103	202	3.3.3.3/32	0	Fa0/0	10.1.12.2
104	203	4.4.4.4/32	0	Fa0/0	10.1.12.2

变成 untagged 了，之前是 POP，为什么现在是 untagged 呢？分析一下，我们修改了 R2 的 loopback 口，改成 2.2.2.2/24，那么对于 R2 自己，这个直连路由就是 2.2.2.0/24 对吧？ R2 会为这个 2.2.2.0/24 分配标签，由于这是直连，所以 R2 给这条前缀分了个空标签 3。然后将标签映射消息发给其他 LDP 邻居包括 R1 和 R3：

```
⊟ Forwarding Equivalence Classes TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Forwarding Equivalence Classes TLV (0x100)
    TLV Length: 7
  ⊟ FEC Elements
    ⊟ FEC Element 1
        FEC Element Type: Prefix FEC (2)
        FEC Element Address Type: IPv4 (1)
        FEC Element Length: 24
        Prefix: 2.2.2.0
⊟ Generic Label TLV
    00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
    TLV Type: Generic Label TLV (0x200)
    TLV Length: 4
    Generic Label: 3
```

那么 R1 和 R3 收到了个标签映射，前缀是 2.2.2.0/24，标签是 3。与此同时，他们也收到了 R2 更新过来的路由，由于 loopback 接口路由被 OSPF 更新默认是采用/32 的方式更新的，因此 R1、R3 上学习到的关于 2.2.2.2 的路由，是/32 位的。那么这就出问题了，我这的路由条目是 2.2.2.2/32，但是你发给我的标签消息里却是 2.2.2.0/24，不匹配啊，于是 R1 就认为，走标签，是到不了 2.2.2.2 了，干脆就给了个 untagged，就像我们看到的 R1 的 LFIB 表。这样一来，当 R1 收到标签包去往 2.2.2.2，R1 会将该标签包的标签栈整个弹出，然后去查找自己的 FIB 表，将报文转发出去。这在本拓扑中，貌似没什么问题，但是，在许多环境下，却会出问题，譬如 MPLS VPN 等等。那么怎么解决呢？很好办，R2 loopback 口来个 ip ospf network point-2-point，或者改掩码都成。

4.4 高级特性

1. **TTL-propagate**

no mpls ip propagate-ttl [forwarded | local]

可以防止由于 TTL=0 返回差错消息而暴露核心传输网的结构

2. **targeted 邻居关系建立**

通常 LDP 会话是建立在直连的 LSR 之间的，但是在某些情况下，可能需要一个远程的、或者基于目的的 LDP 会话，例如 MPLS VPN 中的 TE 隧道。当存在链路翻动的时候，一个基于目的的 LDP 邻居相对于直连的 LDP 对等体来说，可以提高标签收敛的时间。因为如果是直连 LDP 邻居，两台 LSR 之间的链路如果 DOWN 掉了，那么 LDP 会话也就 DOWN 了，然而基于目的的 LDP 会话，另一条可选路径就可以用来承载 LDP TCP 报文，这样的话，就算就算链路 DOWN 掉了，LDP 会话仍然有效，所有的标签都会被保留，在链路重新恢复的时候可以很快将 LIB 中的标签更新到 LFIB 中去。

Router(config)# mpls ldp neighbor [vrf vpn-name] **ip-addr targeted [ldp | tdp]**

建立 targeted 邻居

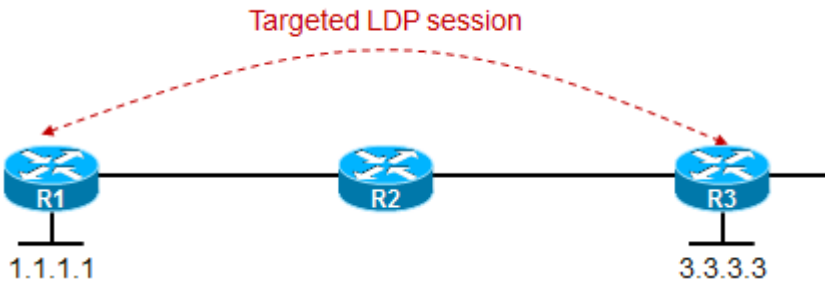
Router(config)# mpls ldp targeted-hello accept from *acl*

配置 targeted-hello accept acl

Router(config)# mpls ldp targeted-hello holdtime x interval y

修改 targeted ldp 会话相关参数

- **配置范例 1：**



R1 的配置如下：

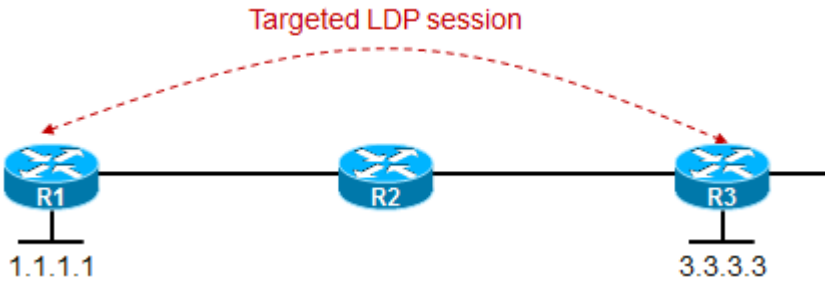
mpls ldp neighbor 3.3.3.3 targeted ldp

R3 的配置如下：

mpls ldp neighbor 1.1.1.1 targeted ldp

这种配置适合有明确的会话端点的情况。

- **配置范例 2：**



R1 的配置如下：

```
mpls ldp neighbor 3.3.3.3 targeted ldp
```

R3 的配置如下：

```
ip access-list standard accept-ldp
 permit 1.1.1.1
mpls ldp discovery targeted-hello accept from accept-ldp
```

R3 是一个被动的会话接受者。

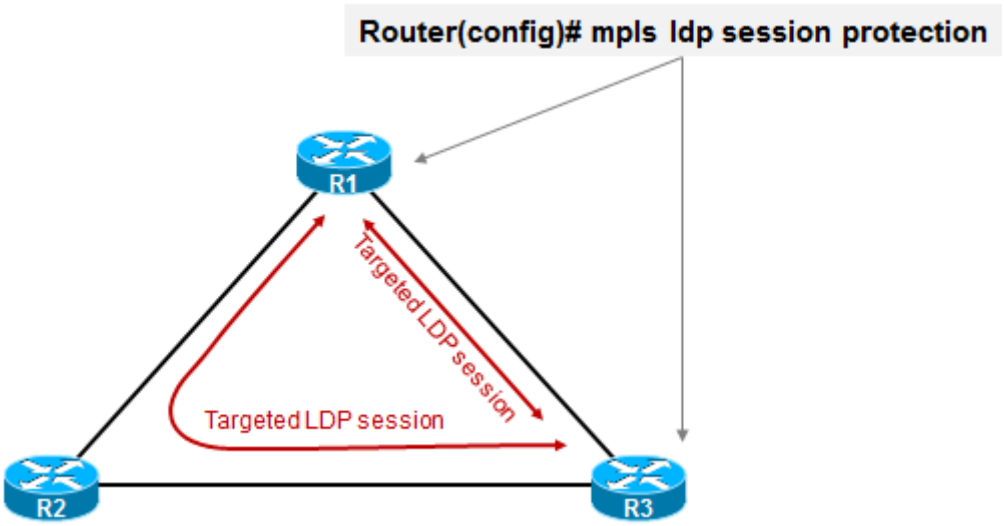
3. LDP 会话保护

链路出现翻动，将导致 LDP 会话重建，并且必须再次交换标签捆绑，要避免重建 LDP 会话，这是非常低效的，我们可以对会话进行保护，在两台直连 LSR 之间的 LDP 会话实施保护之后，将会在这两台 LSR 之间建立基于目的的 LDP 会话，也就是 targeted LDP session，当这两台 LSR 之间的直连链路 DOWN 掉后，只要在这两台 LSR 之间存在可替代的路径，基于目的的 LDP 会话将会得到维持，因此当链路再次恢复后，LSR 就不需要重新建立 LDP 会话了，收敛的效率也就提高了。

Router(config)# mpls ldp session protection for *acl-peer* **duration** *sencondes*

其中 acl-peer 是关联 ACL 用于匹配需要保护的 LDP peer，注意这里 ACL 中匹配的需是邻居的 LDP routerID

Duration 指的是在 LDP 链路的邻接关系 DOWN 掉后，需要得到持续保护（基于 target LDP session）的时间，默认为永久。



为了保护特性正常工作，你需要在两端的 LSR 上都启用该特性。如果有一端不支持该特性，那么可以在一段配置该特性，另一端配置 mpls ldp discovery targeted-hello accept 命令来接收 target-hello。

上图中，R1、R2、R3 运行 OSPF，宣告直连及 Loopback，Loopback 编址为 x.x.x.x/32，x 为设备编号；同时所有的直连接口激活 LDP。

R1#show mpls ldp neighbor

Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
TCP connection: 2.2.2.2.61914 - 1.1.1.1.646
State: Oper; Msgs sent/rcvd: 13/13; Downstream
Up time: 00:04:05
LDP discovery sources:
FastEthernet0/0, Src IP addr: 10.1.12.2
Addresses bound to peer LDP Ident:
10.1.12.2 10.1.23.2 2.2.2.2
Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 1.1.1.1:0
TCP connection: 3.3.3.3.33664 - 1.1.1.1.646
State: Oper; Msgs sent/rcvd: 13/13; Downstream
Up time: 00:03:47
LDP discovery sources:
FastEthernet1/0, Src IP addr: 10.1.13.3
Addresses bound to peer LDP Ident:
10.1.23.3 10.1.13.3 3.3.3.3

接下去我们在 R1 上 shutdown Fa1/0 口，然后在 R1 上看一看：

R1#sh mpls ldp bindings

```
tib entry: 1.1.1.1/32, rev 2
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
tib entry: 2.2.2.2/32, rev 4
        local binding:    tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 3.3.3.3/32, rev 6
        local binding:    tag: 101
        remote binding: tsr: 2.2.2.2:0, tag: 201
tib entry: 10.1.12.0/24, rev 10
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
tib entry: 10.1.13.0/24, rev 8(no route)
        local binding:    tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 202
tib entry: 10.1.23.0/24, rev 12
        local binding:    tag: 102
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
```

很明显由于丢失了与 R3 的直连链路，R1-R3 之间的 LDP 邻接挂掉了，自然而言 LIB 表里，之前 R3 传递过来的标签捆绑也就丢失了，现在，如果 R1-R3 之间的链路回复，他们又要重新建 LDP 邻接，重新发送标签捆绑。

现在我们先 no shutdown R1 的 Fa1/0 口，然后在 R1 及 R3 上开启 LDP 会话保护。

R1#show mpls ldp neighbor

```
    Peer LDP Ident: 2.2.2.2:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 2.2.2.2.61914 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 26/22; Downstream
        Up time: 00:09:33
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 10.1.12.2
        Addresses bound to peer LDP Ident:
          10.1.12.2       10.1.23.2       2.2.2.2
    Peer LDP Ident: 3.3.3.3:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 3.3.3.3.41954 - 1.1.1.1.646
        State: Oper; Msgs sent/rcvd: 9/9; Downstream
```

```
        Up time: 00:00:33
        LDP discovery sources:
          FastEthernet1/0, Src IP addr: 10.1.13.3
          Targeted Hello 1.1.1.1 -> 3.3.3.3, active, passive
        Addresses bound to peer LDP Ident:
          10.1.23.3       10.1.13.3       3.3.3.3
```

发现 R1-R3 之间建立起了 targeted session，现在，我们再来 shutdown R1 的 Fa1/0 口

R1#sh mpls ldp bindings 我们仍然在 R1 的 LIB 表里能看到之前 R3 传递过来的标签捆绑：

```
  tib entry: 1.1.1.1/32, rev 2
        local binding:   tag: imp-null
        remote binding: tsr: 2.2.2.2:0, tag: 200
        remote binding: tsr: 3.3.3.3:0, tag: 300
  tib entry: 2.2.2.2/32, rev 4
        local binding:   tag: 100
        remote binding: tsr: 2.2.2.2:0, tag: imp-null
        remote binding: tsr: 3.3.3.3:0, tag: 301
(.....省略.....)
```

而且，由于默认的 duration 是无限期，因此这些条目会一直保留着。当然，如果在 R1 上修改配置：

```
access-list 1 permit 3.3.3.3
mpls ldp session protection for 1 duration 30
```

那么 R1 将限定建立 targeted session 的 peer，为 3.3.3.3，这是 R3 的 LDP routerID，另外，LDP 会话保护的时间为 30S，也就是说，30S 后如果 LDP 邻居还没起来，targeted session 将失效。自然的，LIB 里之前存储的被保护的标签信息也就没了。

4. LDP 认证

LDP 会话是一种 TCP 会话，TCP 会话可能遭受 TCP 碎片欺骗的攻击。可以使用 LDP 认证来进行保护。MD5 将添加一个签名—称为 MD5 摘要，到 TCP 分段中。

mlps ldp neigbor *[vrf vpn-name] ip-addr* **password** *[0-7] pswd*

5. 通过 LDP 控制标签通告（条件通告）

router(config)# mpls ldp advertise-labels [for *prefix-access-list* [to *peer-access-list*]]

Parameters:

- **For prefix-access-list** – the IP access list that selects the destinations for which the labels will be

generated

- **To peer-access-list** - the IP access-list that selects the MPLS neighbors that will receiver the labels

注意：要使用上述命令完成条件通告，需先使用“no mpls ldp advertise-labels”，否则默认情况下，LSR 通告所有的标签捆绑。

6. LDP 入站标签捆绑过滤

过滤 LDP 邻居通告过来的入站标签捆绑

router(config)# mpls ldp neighbor *x.x.x.x* labels accept *acl*

neighbor 是邻居的 LDP routerID

accept acl 用来匹配前缀

7. LDP autoconfig

通常的做法是在每个运行 IGP 协议的接口上使用 mpls ip 来激活 LDP 协议，有一种更加简便的方法：

```
router ospf 1
  mpls ldp autoconfig area 0
```

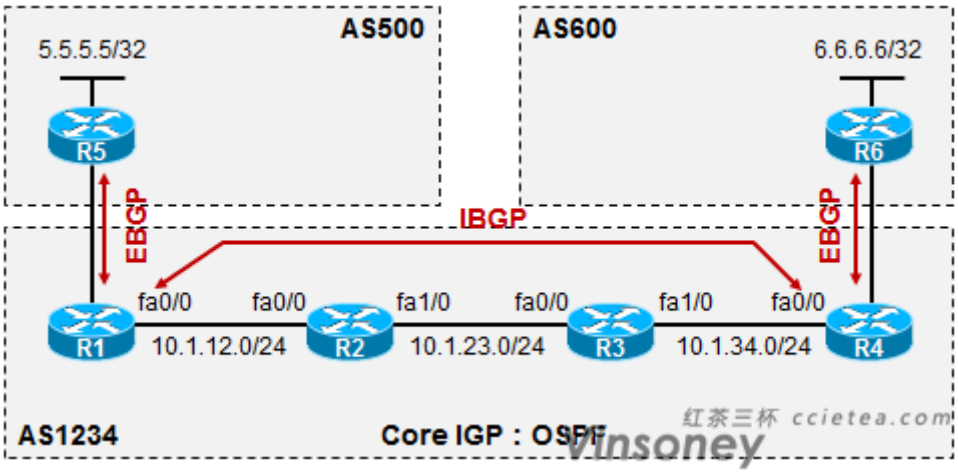
则所有 ospf area0 的接口都将自动激活 LDP

如果某个特定接口不希望激活 LDP，可使用 no mpls ldp igp 来关闭 LDP。

```
Router# show mpls interface detail
Interface FastEthernet0/0:
   ip labeling enable(LDP):
   interface config          <-- 在接口上激活LDP
   LSP tunnel labeling not enabled
   BGP labeling not enabled
   ... ...
Interface FastEthernet0/1:
   IP labeling enabled(LDP):
   IGP config                <-- 使用autoconfig方式激活LDP
   LSP tunnel labeling not enabled
```

5 MPLS 环境下的 BGP 路由传递

1. MPLS 不会为 BGP 路由分配标签，但为 BGP 路由的下一跳分配标签



拓扑环境描述：

- R1、R2、R3、R4 处于 Transit AS 1234。在 AS 内运行的 IGP 协议是 OSPF
- 所有的互联 IP 如图所示
- 所有设备的 Loopback0 口地址为 x.x.x.x/32，x 为设备编号
- R1 与 R4 之间建立 IBGP 邻接关系，IBGP 邻接关系建立在物理接口上。R1 与 R5、R4 与 R6 之间建立 EBGP 邻接关系，也是建立在物理接口上。
- 在这个实验测试中，我们在 OSPF 中宣告 R1-R5 和 R4-R6 的直连网段。
- R5 及 R6 各自在 BGP 进程中宣告自己的 Loopback 路由

实验结果：

由于 R2、R3 没有运行 BGP 协议，并且 Core OSPF 内也没有 5.5.5.0 及 6.6.6.0 的路由，因此最终的结果是 R5 及 R6 虽然能够学习到彼此的路由，但是却无法互访，因为在 R2 及 R3 上出现了路由黑洞。

解决的办法就是用 MPLS，我们将 Core 变成 MPLS 域：

R1 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 100 199
interface fa0/0
  mpls ip
```

R2 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 200 299
```

```
interface fa0/0
  mpls ip
interface fa1/0
  mpls ip
```

R3 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 300 399
interface fa0/0
  mpls ip
interface fa1/0
  mpls ip
```

R4 的配置如下：

```
mpls ldp router-id loopback0
mpls label rangempls label range 400 499
interface fa0/0
  mpls ip
```

R1、R2、R3、R4 运行 LDP 协议。

要注意，LDP 默认是不会为 BGP 路由分配标签的，但是我们也知道，BGP 路由都有 next-hop，而这个下一跳是 IGP 路由可达的，LDP 则会为这条（下一跳地址所在的）路由分配标签，这一点非常之重要。

这样一来我们 5.5.5.5 和 6.6.6.6 之间就能够互访了，在 R6 上

R6#traceroute 5.5.5.5 soure 6.6.6.6

```
Type escape sequence to abort.
Tracing the route to 5.5.5.5
  1 10.1.46.4 128 msec 152 msec 120 msec
  2 10.1.34.3 [MPLS: Label 303 Exp 0] 856 msec 1012 msec 1072 msec
  3 10.1.23.2 [MPLS: Label 203 Exp 0] 964 msec 1124 msec 1008 msec
  4 10.1.12.1 1060 msec 984 msec 1080 msec
  5 10.1.15.5 1564 msec 1124 msec 1484 msec
```

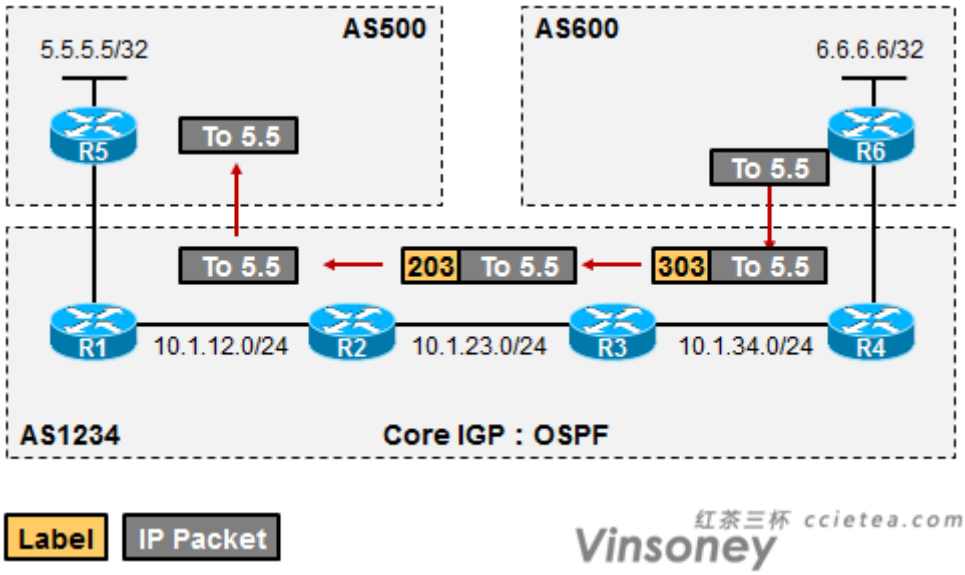
我们发现 ,R6 始发的报文是 IP 的 ,到了 R4 ,R4 查 CEF 表 ,发现目的地 :5.5.5.5 的条目 ,关联了一个 Label：303，于是 R4 将 IP 包压上标签 303，然后丢给下一跳 10.1.34.3 也就是 R3。下面就是 R4 的 CEF 表项：

```
5.5.5.5/32, version 22, epoch 0, cached adjacency 10.1.34.3
0 packets, 0 bytes
  tag information from 10.1.15.0/24, shared
    local tag: 403
```

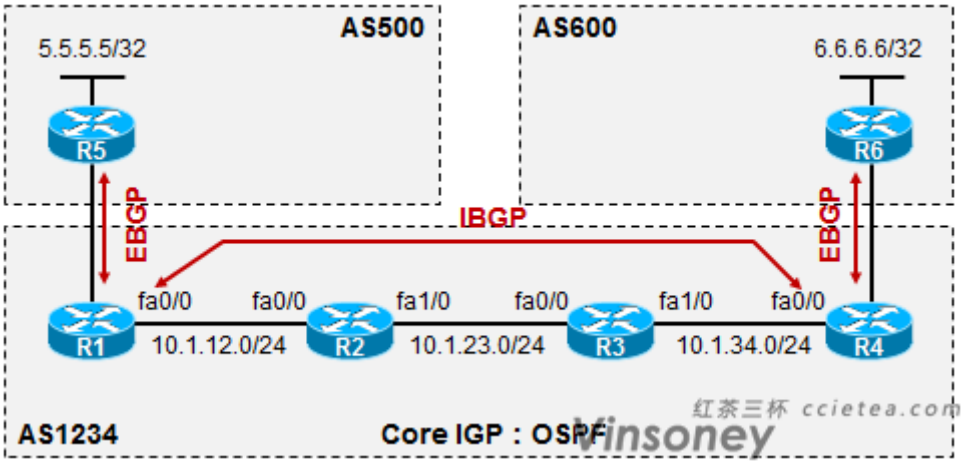
```
    fast tag rewrite with Fa0/0, 10.1.34.3, tags imposed: {303}
via 10.1.15.5, 0 dependencies, recursive
    next hop 10.1.34.3, FastEthernet0/0 via 10.1.15.0/24
    valid cached adjacency
    tag rewrite with Fa0/0, 10.1.34.3, tags imposed: {303}
```

注意这里这个 303 很明显，是 R3 分配的，然后将这个分配结果给到了 R4，R4 是在用 R3 分配的标签去压到 IP 包前面。那么这个 303，实际上是为 BGP 路由 5.5.5.5 的下一跳 10.1.15.0 这条路由分配的标签。还是那句话，LDP 不会为 BGP 路由分配标签，但是会为 BGP 路由的下一跳（路由）分配标签。

接下去数据到了 R3，R3 查自己的 LFIB 表后，将 303 标签置换成 203，然后丢给下一跳 R2。R2 收到这个标签包后，查看自己的 LFIB 表，发现 outgoing 动作是一个 PoP，于是将标签弹出，变成最原始的那个 IP 包，然后丢给 R1，最后 R1 将这个 IP 包转发到了 R5。那么这里为什么 R2 这里会 PoP 呢？答案是这条 BGP 路由，前面我们讲过了，其实是用的是它的下一跳 10.1.15.0 的标签，10.1.15.0 是 R1 的直连网段，因此 R1 在为这条路由分配标签时，给了个 PoP，然后将这个结果分发给 R2，这就是原因。

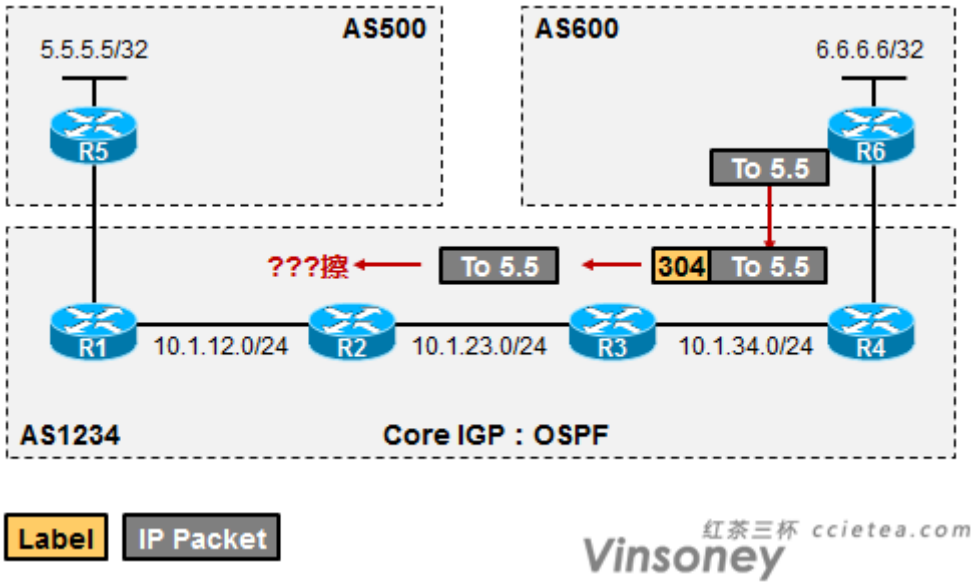


2. （承上）不宣告 R1-R5 及 R4-R6 之间的直连网段进 OSPF



在上面的实验中，我们在 Core OSPF 中宣告了 R1-R5 及 R4-R6 的直连网段，然而我们知道，在实际的场合中，往往不会在 Core IGP 中宣告这条 AS 外的链路，因此接下去我们来看看，如果不在 OSPF 中宣告这两个直连链路，会有什么现象：由于 OSPF 没有了这两个直连网段，那么 5.5.5.5 及 6.6.6.6 学习进来就不优了，需在 R1 上对 R4 去使用 next-hop-self，R4 对 R1 也同理。

那么这样一来，R4 上去往 5.5.5.5 下一跳就是 10.1.12.1 了，那么当 6.6.6.6 访问 5.5.5.5 的时候，IP 包到达 R4，R4 查 CEF 并且为 IP 包压上标签，关键在这个标签上，R4 会使用 10.1.12.0 这条路由的标签（12.0 是去往 5.5.5.5 的下一跳），而这个标签是 R3 分配的（我给出口数据打上的标签永远是 LDP 邻居给我的标签）。那么这个标签包被传递到了 R3，R3 会怎么处理呢？实验现象表明，R3 会将标签 PoP，变成 IP 包然后丢给 R2。可是为什么？原因是 R2 为路由 10.1.12.0 分配的标签就是 PoP，因为 12.0 是 R2 的直连网段，这里有次末跳弹出机制。那么这个 IP 包到了 R2 就歇菜了，因为 R2 根本没有 6.6.6.6 和 5.5.5.5 的路由啊。怎么办？



解决的办法：R1 及 R4 使用 loopback 接口建立 IBGP 邻居关系

前面的实验之所以在 R1 和 R4 上使用物理接口建立 IBGP 邻接 ,是为了帮助大家理解一下 MPLS 环境下 BGP 的路由问题，好了，现在我们在 R1 和 R4 上，使用 loopback 接口来建立 IBGP 邻接关系，问题就解决了。 这时候 R4 上，去往 5.5.5.5 的下一跳就是 R1 的 Loopback 口地址，所以 R4 在给去往 5.5.5.5 的 IP 包压标签的时候，用的就是 1.1.1.1/32 路由的标签，R3、R2 也一样，那么标签包就能通过 1.1.1.1/32 打通的 LSP 一路传到 R2，并在 R2 这弹出标签变成 IP 包再转给 R1，R1 再将 IP 包转发给 R5，搞定。

因此，路由器不会对 BGP 路由直接分配标签，而是为 BGP 路由通过递归后的下一跳路由分配标签，这样做是很有好处的。BGP 中的路由条目是相当多的，如此一来我们通过 MPLS，可大大简化路由器的性能损耗，BGP 的 Transit AS 的路由黑洞问题也得到了很好的解决。

6 参考书籍

- MPLS 技术架构
- MPLS VPN 体系结构 CCSP 版
- MPLS VPN 体系结构 卷二
- MPLS 设计与实现
- MPLS VPN configuration