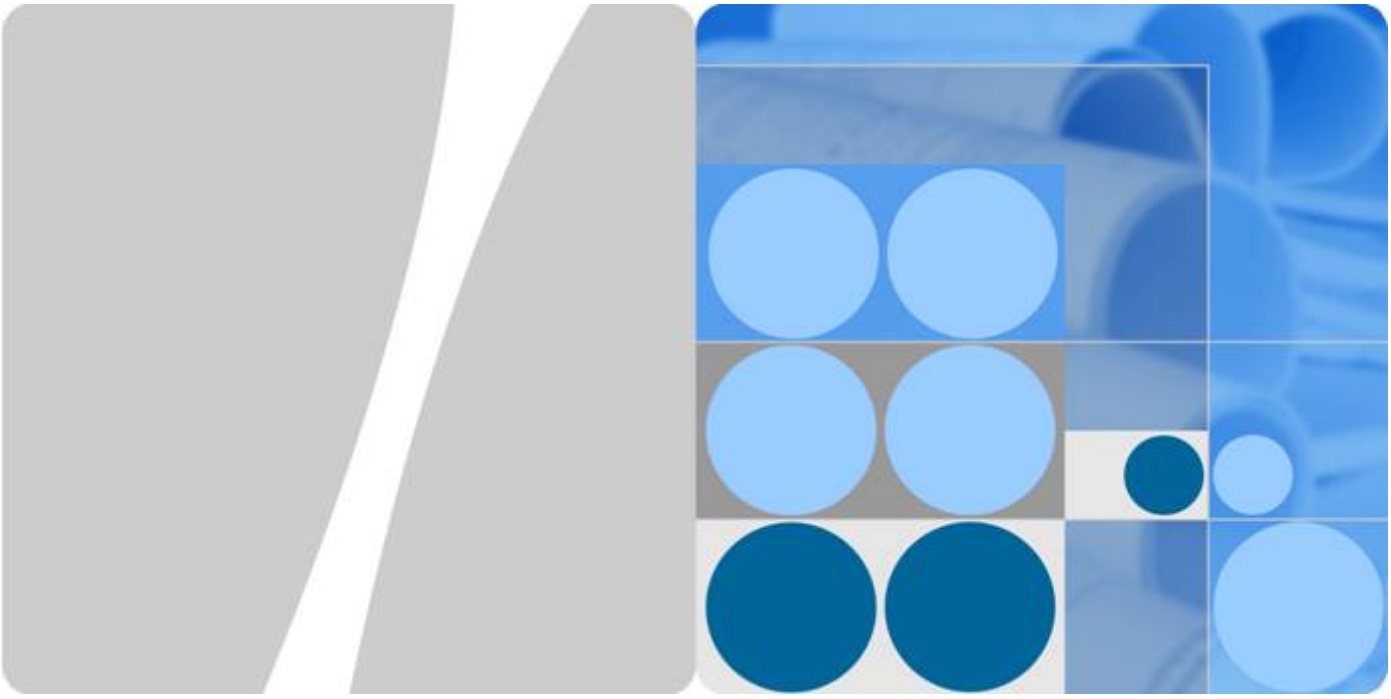


一个报文的路由器之旅

文档版本　01

发布日期　2016-03-28

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

修订记录

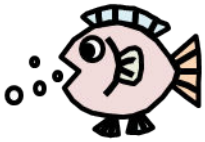
文档版本	发布时间	修订记录
01	2016-03-28	首次发布。

引 言

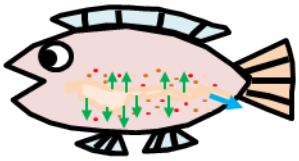
如今，在互联网的各种级别的网络中都随处可见路由器，各种低、中、高端的，种类繁多，所具备的功能和内部实现不完全一样。为此，本文档将为您揭晓华为高端路由器（NE40E/80E/5000E）上的实现。

路由器“吃”进去的数据去哪儿了？

路由器不断的在吞吐通信数据，就像鱼儿一样。通信数据像是路由器的食物。



鱼儿吃进去的食物，有的会被吸收了，进入血液，最终转换成能量或变为身体的一部分；没被吸收的经过肠道排泄到体外。



那么，路由器“吃”进去的数据，上哪去了呢？

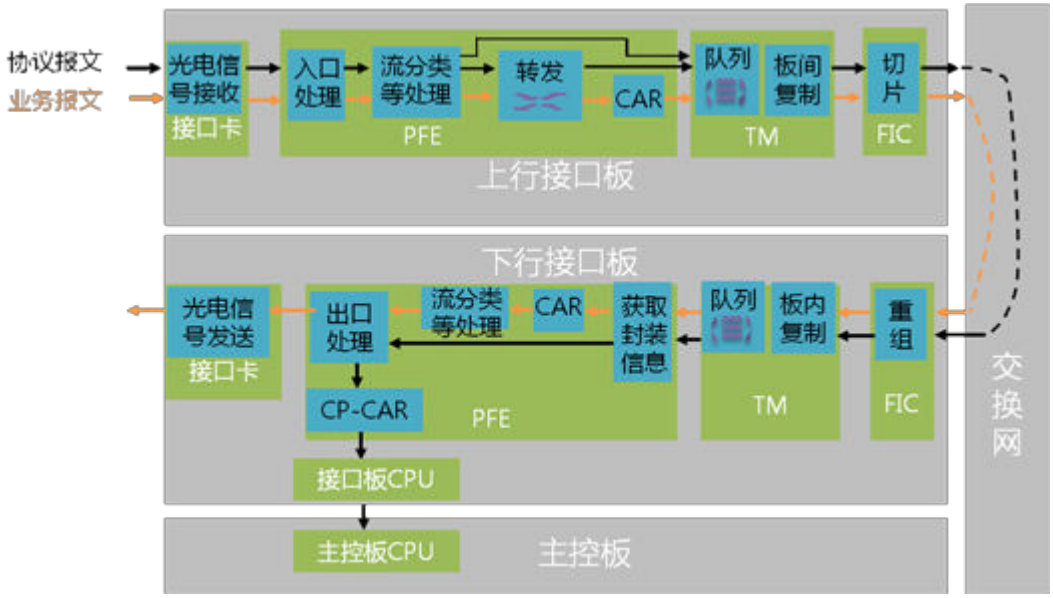
鱼儿吞进去的大部分是水，这些水基本都吐出来了，并没有被吸收。同样的，进入路由器的数据，大部分从一个接口进去，从另一个接口出来，它们只是“过路”的业务报文，也有人称之为“过路”报文。有一小部分数据被“吸收”了（被上送CPU处理，或者因为各种原因中途被丢弃）。

说明

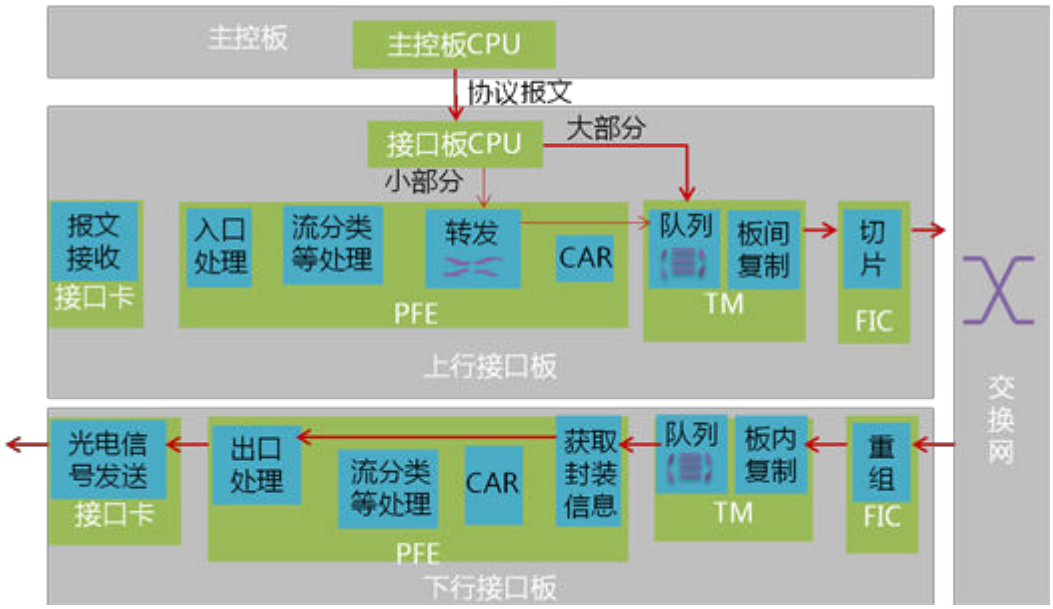
华为高端路由器采用的是“硬转发”，业务报文不经过CPU处理。如果是“软转发”，所有报文都经过CPU处理，那么CPU将成为瓶颈，不利于转发性能的提升。

路由器转发全景图

下图是路由器收到的业务报文和协议报文在转发层面的处理流程。



下图是路由器CPU发送的协议报文的在转发层面的处理流程。



是不是觉得上图太复杂、概念太多，看不懂？

上述图片只是给一个总的概念，以便更好的理解后续章节的描述。

为了方便理解，让读者知其然，并知其所以然，后续章节并非按设备执行的顺序进行介绍，而是按由主到次，由总到分的逻辑来介绍。

目 录

1 交换和寻址转发

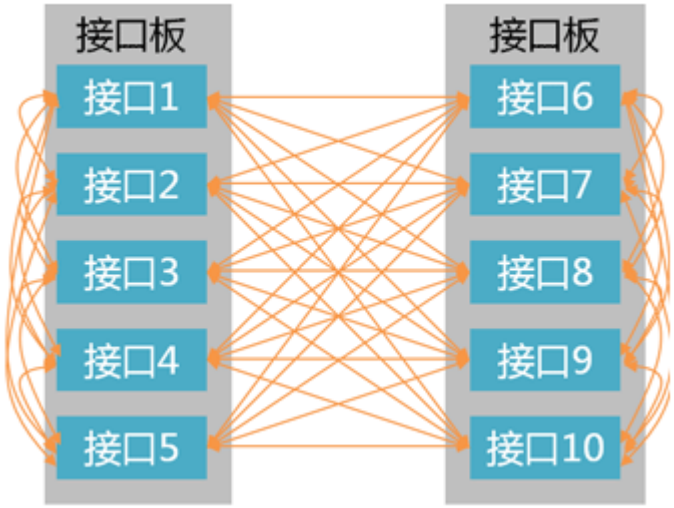
关于本章

本章从路由器最核心、最基本的功能（交换和寻址转发）讲起，主要知识点为：

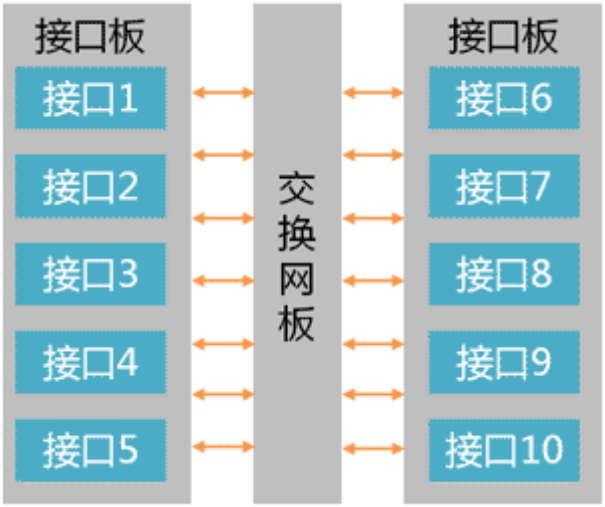
- 为什么要有交换网
- 上行和下行的概念
- 路由表和转发表的关系，以及它们在路由器内部存放的位置
- 生成转发表的2种方式（“预路由”和“流触发”）

1.1 从“交换”谈起

数据是通过接口板接收和发送，通信线缆都要插接到接口板的接口上。那么，把某一个接口来的数据包送到另一个接口发出去，这两个接口需要连起来。但实际上，数据包可能从任意接口进来，从任意接口出去，都这么点到点连接的话，则需要N*(N-1)/2根线互联，太多了。



为了解决这种大量连接的问题，接口板和接口板之间需要通过交换网（Switch Fabric）板衔接起来，接口板只要通过若干连线跟交换网板连接，就能完成任意接口的互通。



说明

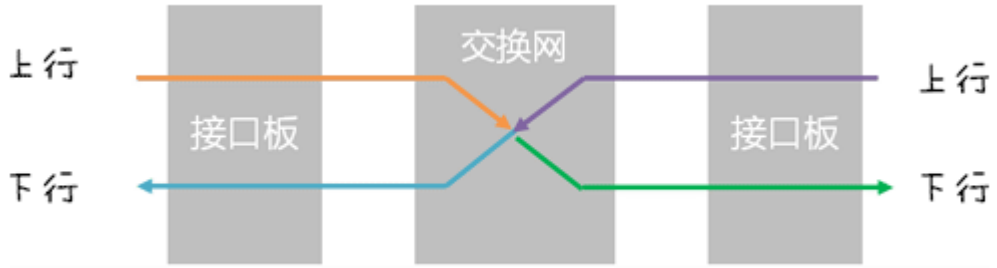
有的设备如NE40E-X1/X2，没有交换网板，但其接口板上有交换模块（Switch Module），其完成的功能都是一样的。

交换网属于“三无”部件，即与设备配置无关、与协议无关、与数据包类型无关。交换网专注于在入接口和出接口之间建立连接，完成数据的交换。

关于交换网更详细的介绍，请参见《**路由器硬件基础知识-交换网**》。

1.2 上行和下行

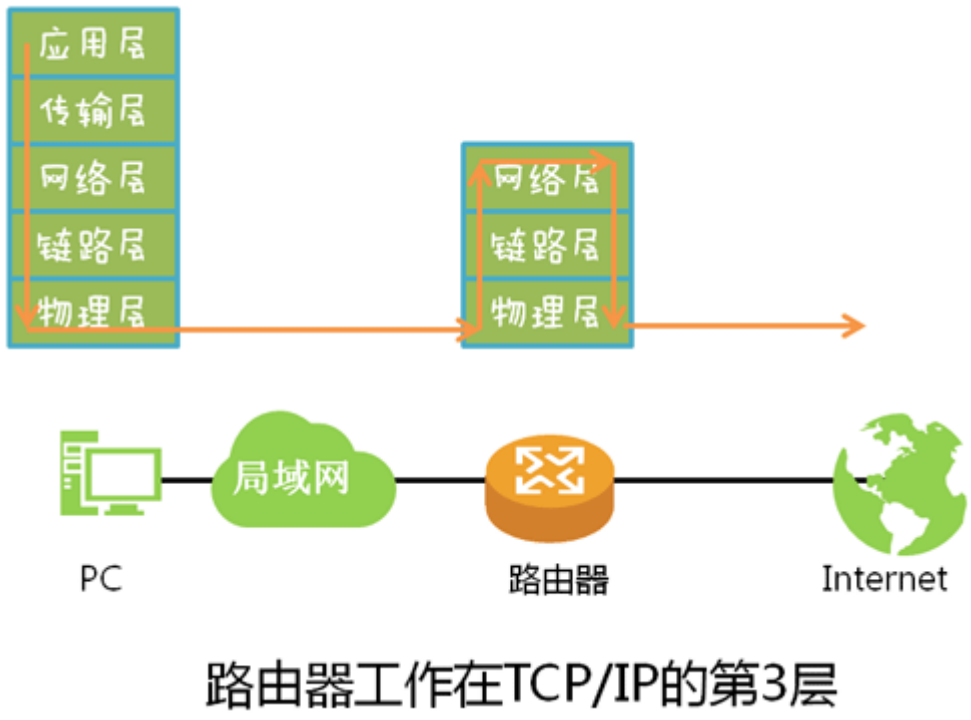
以交换网为中心，可将报文在路由器的行程一分为二，前半程称为“上行”，下半程称为“下行”。



1.3 寻址转发

可能有人会问，报文从一个接口进来，经过“交换”，从另一个接口出去，这个交换机也会做啊，何必用路由器？是的，交换机也有交换功能。但是，在互联网中，从一个节点到另一个节点，有许许多多的路径，路由器可以选择通畅的最短的路径，从而提高通信速度，减轻网络负荷，节约网络资源，这是交换机所不具备的能力。为数据包选择一条合适的（通常指最短的）传输路径，然后从对应的接口发送，这个过程就称为“寻址转发”。

路由器所在的网络几乎都是遵循TCP/IP体系的，路由器是工作在该体系的第三层，即网络层。



所以，刚才提到的“寻址”的”址“是指根据数据包的网络层地址——IP地址。为了寻址，路由器需要一张“地图”，以目的IP地址为索引的“地图”，也就是路由表。每个路由器中都有一张路由表。

路由表长什么样

下图是一张实际的地铁出口地图。

实际的路由表跟上图有些相似。路由表的索引是目的IP地址/掩码，每个表项中都有对应的下一跳IP地址和出接口信息，如下图。

目的IP地址/掩码					下一跳IP	出接口
Destination/Mask	Proto	Pre	Cost	Flags	NextHop	Interface
10.0.0.0/8	Static	60	0	RD	10.136.120.1	GigabitEthernet1/0/0
10.136.120.0/23	Direct	0	0	D	10.136.120.107	GigabitEthernet1/0/0
10.136.120.107/32	Direct	0	0	D	127.0.0.1	GigabitEthernet1/0/0
10.136.121.255/32	Direct	0	0	D	127.0.0.1	GigabitEthernet1/0/0
127.0.0.0/8	Direct	0	0	D	127.0.0.1	InLoopBack0
127.0.0.1/32	Direct	0	0	D	127.0.0.1	InLoopBack0
127.255.255.255/32	Direct	0	0	D	127.0.0.1	InLoopBack0
192.1.1.0/30	Direct	0	0	D	192.1.1.2	GigabitEthernet2/0/0
192.1.1.2/32	Direct	0	0	D	127.0.0.1	GigabitEthernet2/0/0
192.1.1.3/32	Direct	0	0	D	127.0.0.1	GigabitEthernet2/0/0
255.255.255.255/32	Direct	0	0	D	127.0.0.1	InLoopBack0

有了这张表，路由器接在收到数据包时就能做到心中有数了。比如收到一个目的地址为10.0.0.1的报文，路由器就可以查表得知需要将该报文发送到GE1/0/0这个接口。

这个路由表怎么得来的呢？一种办法是手工制作，对路由器进行手工设置固定的路由。但是这种路由不能对网络的改变作出反映，如果网络拓扑变化了，需要人工去修改设置。还有一种办法就是运行动态路由协议，让路由器之间相互传递路由信息，利用收集到的路由信息进行计算，生成路由表，这样就可以让路由表实时跟进网络拓扑的变化。在实际应用中，这两个办法都用上了，当动态路由与静态路由发生冲突时，以静态路由为准。当然，路由表还有一类路由，不是人工配置的，也不是路由协议的学习，而是由链路层协议发现的，称为直连路由。

路由表放在哪

有了路由表，接下来要考虑的是，路由表放哪合适呢？

前面说过，数据包是从某个接口进来，经过交换网，再从另一个接口出去。那路由表能不能放交换网？答案是不行，因为交换网要完成整个设备所有报文的交换，为了让交换网完成高速交换，不成为瓶颈，不能再让交换网去运行路由协议、维护路由表、做寻址转发。

那路由表能不能放下行接口板？答案也是不行，交换网做交换的时候，就需要知道要送往哪块目的单板，所以寻址转发需要在上行完成。然而，如果把路由表放上行接口板，由于报文可能从任意接口板进来，那么所有的接口板都需要放一个路由表。其实，还有更好的办法，就是将路由表放在一个公共的地方，比如主控板上，由主控板的CPU运行路由协议，计算路由，生成和维护路由表。

转发表与路由表

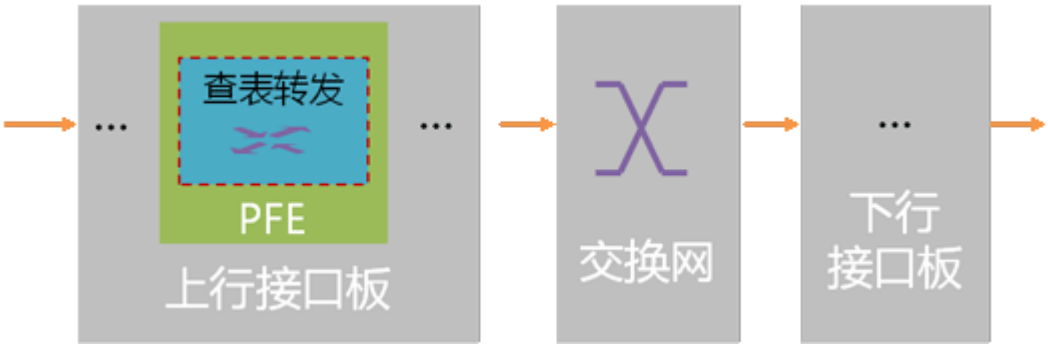
华为高端路由器采用的是“硬转发”，业务报文不经过主控板CPU处理，不能直接用主控板上的路由表，接口板上也需要有供寻址转发的信息。所以，主控板CPU生成路由表之后，还要将相关信息下发给各个接口板。这些相关的信息就是转发信息，存放在各个接口板的转发信息表FIB（Forwarding Information Base）中。各个接口板上的转发信息都是相同的，因为它们具有相同的来源，都来自主控板。

实际上，现代高性能路由器在架构上都是转发和控制分离：把转发层面和控制层面分配在不同的组件，控制层面运行路由协议，维护路由表，并下发转发表FIB到转发层面，由转发层面负责数据包转发。这样做的最基本的好处就是不会相互影响：如果流量很高导致转发层面高负荷，但是其不会影响控制层面进行正常的路由学习；相反的，如果控制层面对路由信息的处理比较繁忙，也不会影响转发层面进行其高速的数据包转发。

Destination/Mask	Nexthop	Flag	TimeStamp	Interface	TunnelID
192.1.1.0/30	192.1.1.2	U	15:54:32	GE2/0/0	0x0
192.1.1.3/32	127.0.0.1	HU	15:54:32	GE2/0/0	0x0
192.1.1.2/32	127.0.0.1	HU	15:54:32	GE2/0/0	0x0
10.136.120.0/23	10.136.120.107	U	00:00:00	GE0/0/0	0x0
127.0.0.0/8	127.0.0.1	HU	00:00:00	InLoop0	0x0
10.136.121.255/32	127.0.0.1	HU	00:00:00	GE0/0/0	0x0
10.136.120.107/32	127.0.0.1	HU	00:00:00	GE0/0/0	0x0
127.255.255.255/32	127.0.0.1	HU	00:00:00	InLoop0	0x0
255.255.255.255/32	127.0.0.1	HU	00:00:00	InLoop0	0x0
127.0.0.1/32	127.0.0.1	HU	00:00:00	InLoop0	0x0
10.0.0.0/8	10.136.120.1	GSU	00:00:00	GE0/0/0	0x0

细心的读者会发现，路由表和转发表看起来差不多，都有目的IP地址/掩码、下一跳、出接口这三个信息。实际上，转发表是根据路由表生成的。路由表中可能包含到达目的地址的多条路由，但是转发表里面只取其中的最优路由。而且，路由表的下一跳是原始的下一跳，不一定是直接可达的，FIB是用于指导转发的，它的下一跳必须是直接可达。根据“原始下一跳”找到“直接下一跳”的过程就称为“路由迭代”。

路由器上电启动之后，就会运行路由协议学习网络拓扑，生成路由表，如果接口板注册成功，主控板就可以根据路由表生成转发表项并下发给接口板，这样路由器就可以根据转发表转发数据包了。执行数据包转发的部件是位于接口板上的一个被称为包转发引擎PFE（Packet Forwarding Engine）部件，通常是NP或ASIC芯片。



找不到路怎么办？

上述这种在转发报文前，提前准备好转发表，待收到报文时再查表转发的方式称为“预路由”，“先铺路，后通车”。现在路由器都采用这种方式进行IP单播转发。在这种方式中，查表转发时，如果没有匹配上（如果有默认路由，最终会匹配上默认路由，默认路由不存在“不匹配”的情况），意味着这台路由器没有到这个目的地址的路由（或者还没有学习到这个路由），也就是找不到路，迷路了。数据包迷路了怎么办，原路返回？想象下，如果迷路了就被原路返回给源端，那源端重发的还是同样的目的地址，那这个报文还是会在同一个地方迷路，再原路返回，死循环了。所以，数据包迷路了只能被丢弃。出于可维护方面的考虑，包转发引擎PFE会记录丢弃原因和统计丢弃的报文数。

预路由与流触发

刚才说到路由器都采用“先铺路，后通车”的预路由方式。相对的，“先通车，后铺路”的方式，被称为“流触发”。流触发方式中，设备收到报文，查转发表，如果转发表中不存在对应的表项，就根据这个报文生成一个转发表项。这样，该用户流的下一个报文就可以命中转发表进行转发了。

目前，路由器和交换机在进行二层转发时所使用的MAC表，就是采用MAC地址学习方式，类似于“流触发”方式。

从安全性角度上，流触发显然容易造成流量攻击，为攻击者提供了一个合理合法的攻击路径。攻击者可以使用各种未知目的报文对系统进行遍历扫描攻击，形成对路由器的流量攻击。所以，华为高端路由器上，除了有MAC学习方式机制外，为了预防流量攻击，还提供了限制MAC地址学习的功能，即限制最多允许学习多少个MAC地址，并限制每次学习的时间间隔；而且还允许去使能MAC地址学习，允许人们像配置静态路由一样去手工配置MAC表项。

2 报文的收发、解析和封装

关于本章

报文在通信线路上只是一些光/电信号，从光/电信号的接收到转发、到交换，再到发送，这个过程中，还经过了什么处理？本章将为您揭晓答案。

本章主要知识点为：

- 光/电信号和数据帧之间的转换
- 数据帧的“合法性”检查
- 报文解析过程
- 报文封装过程

2.1 报文的接收和发送（接口卡的处理）

数据在通信线缆上传输时还只是光/电信号（对应于物理层的比特流）。为了让路由器读懂这些信号，以便获取数据包的目的地址用于寻址转发，在路由器上插线缆的接口里边有一块物理接口卡——PIC（Physical Interface Controller）卡，能感知这些光/电信号，把信号转换成数据帧（比如以太帧、PPP帧、ATM信元）。

接口卡有两个重要的功能，其中一个就是完成上述的物理层功能，光/电信号的收、发；另一个重要功能，就是进行数据帧的“合法性”检查。数据经过物理线路的传递后，有可能发生畸变，变成错包，无法被包转发引擎PFE正确解析，因此在接口卡需要进行一些必要的检查。

比如，一个以太帧格式如下：

以太网标准中规定如下帧为无效帧：

- 帧的长度不是整数个字节；
- 用收到的帧检验序列FCS（Frame Check Sequence）查出有错误；
- 收到的帧的负荷长度不在46~1500字节之间。

对于检查出的无效帧就简单的丢弃，以太网不负责重传丢弃的帧。而这些检查，是在接口卡上执行的。

说明

每个以太帧之间都要有帧间隙（Interframe Gap），即每发完一个帧后要等待一段时间才能再发另外一个帧，以便让帧接收者对接收的帧作必要的处理（如调整缓存的指针、更新计数、通知对报文进行处理等等）。

在以太网标准中规定最小帧间隙是12个字节，其数据为全1。对于个别的接口，可减少到64(GE)或40比特(10GE)，其他的接口都不应该小于12字节。

以太网标准中规定前导码为10101010 10101010 10101010 10101010 10101010 10101010 10101010（二进制），共7字节；帧开始定界符为10101011，共1字节。

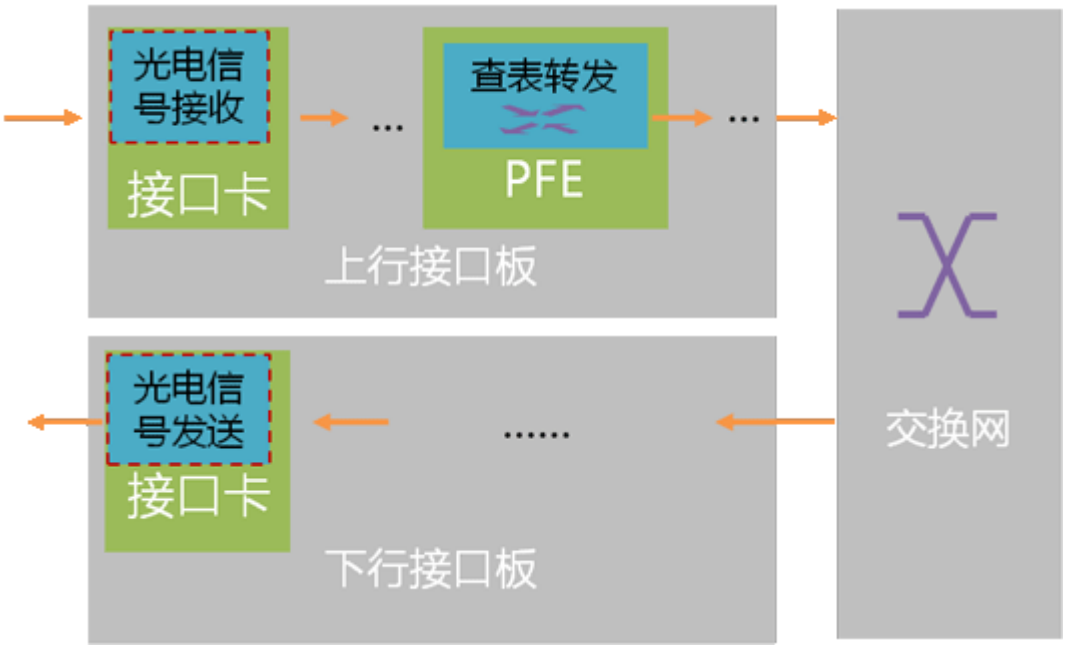
那么，如果PIC卡实际收到的帧间隙、前导码、帧开始界定符，如果跟协议规定的不一样，是不是这个数据帧也会被丢弃？答案是，PIC卡在处理帧间隙时，帧间隙一般可以容忍跟协议规定的不一样（比如不是全1）；但前导码、帧开始界定符必须符合协议规定的值，否则当做帧间隙处理，也就是帧被丢弃了。

PIC卡把光/电信号转换成数据帧，并检查“合法性”之后，把数据帧的内容（不包含帧间隙、前导码、帧开始界定符和FCS）发送给包转发引擎PFE。

说明

PIC卡的类型决定了接口板的业务类型，比如把4*2.5G PoS PIC卡集成在某接口板上，则该单板就支持4*2.5G 的PoS业务，把10*GE PIC卡集成在某接口板上，则该单板支持提供10*GE的以太业务。将某PIC卡集成在接口板上，对应的PFE便可得知该PIC卡的类型，因此按对应业务类型来解读PIC卡送过来的数据。

数据包经过PFE转发，经过交换网板交换，从下行接口发送时，下行的对应位置也有接口卡。



下行接口卡的作用是，用待发送的数据帧内容计算帧检验序列FCS，然后对数据帧加封装帧间隙、前导码、帧开始界定符和FCS，并将数据帧转换成光/电信号，再发送到出接口线路上。

2.2 报文解析

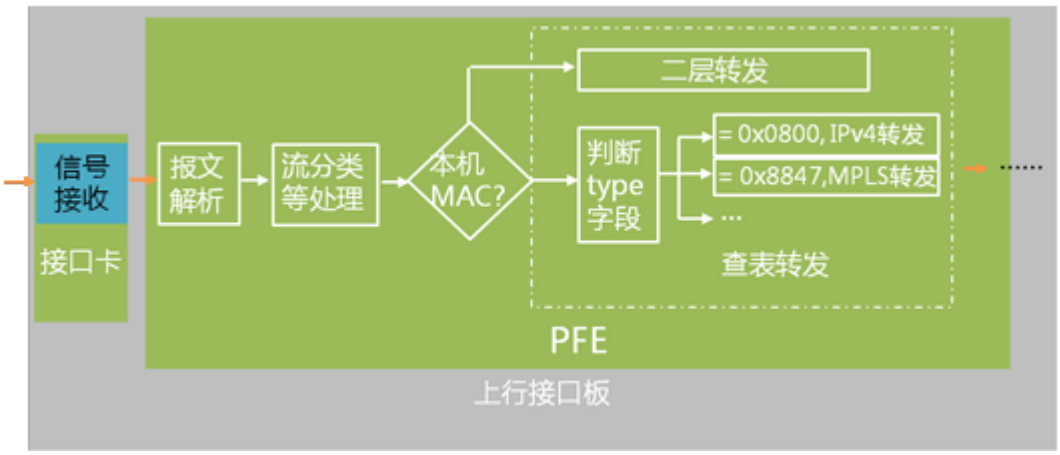
当转发引擎PFE从PIC卡收到报文时，PFE首先做的就是解析报文的二层帧头，并根据配置做一些检查和处理。那么，如何根据配置做检查和处理呢？

路由器支持丰富多彩的功能特性，以满足各种业务需求。其中，大部分功能特性允许人们通过命令行进行配置，实现使能或去使能，并允许通过命令行来调整一些参数。那么，路由器各转发部件是如何依据命令行的指示来工作的呢？

其实，所有的配置命令行都是通过主控板解读，再下发到接口板。在接口板上存有许多的表项，除了转发所使用的转发信息表之外，还有各种入接口属性表、出接口属性表、ACL表、流分类表等等。接口板上电或重启时会触发主控板向其下发配置，也就是根据配置设置接口板上的各种表项的值。之后，接口板上的各部件查找各类表项，根据对应的值进行对应的处理。当然，如果有配置的增删改，大部分情况下会实时更新到接口板。

举个例子，路由器某个以太接口上接入VLAN10～20范围内的用户，人们希望在该接口上配置允许VLAN10～20的报文通过，如果不在VLAN10～20范围内的要被丢弃。那么，人们需要在该接口上配置**portswitch**命令使得该接口为二层接口，并配置接口为Trunk类型，允许通过（allow-pass）的VLAN ID范围是10～20。当主控板下发配置后，该接口的属性表对应的二层桥接转发状态为“使能”，端口类型为Trunk，且VLAN ID范围是10～20。当报文到来时，PFE首先对照报文的入接口属性表，发现接口为Trunk类型，于是检查报文的二层帧头是否携带VLAN。如果没有则丢弃，有则检查帧头携带的VLAN ID值是否在接口属性表对应的VLAN ID范围内，如果不在则丢弃该报文。

除了做检查外，PFE还需要根据报文的入接口属性表来做流分类、包过滤、重定向等处理（这些处理将在后续章节中介绍）。



之后，PFE根据入端口属性表对应的转发动作做后续的查表转发处理。以以太报文为例，PFE检查MAC地址，如果不是本机MAC，则后续做二层桥接转发；如果是本机MAC，则根据以太帧头协议类型（type字段）做IP、MPLS或其他类型的转发。值得注意的是，如果入接口属性表对应的转发状态与报文解析后的协议不匹配，则报文会被丢弃。比如某接口并未使能IPv6，则属性表的IPv6转发能力为”未使能“，如果收到的报文是IPv6报文，则报文会被丢弃。

说明

有些报文是不需要进行查表转发的。比如，PFE解析二层帧头时，从二层帧头的协议字段就可以直接判断出某些协议报文是需要上送本机CPU处理的，如ARP、RARP、IS-IS、LLDP、LACP、PPP控制报文等等；还有一些协议报文，其目的地址为特定的保留组播IP地址（标准中定义，组播地址224.0.0.1～224.0.0.255供路由协议使用），可直接判断其需要上送本机CPU处理，因此这类协议报文也不需要查表转发。

2.3 报文封装

不同报文，需要做的封装不同。以大家熟悉的以太帧为例吧，先来回顾下IP转发流程，看看IP转发过程中要封装哪些信息。下图是个最简单的IP转发场景，某局域网的主机A发送报文给另一局域网的主机B，中间经过一台路由器，那么这台路由器就是PC-A的网关。

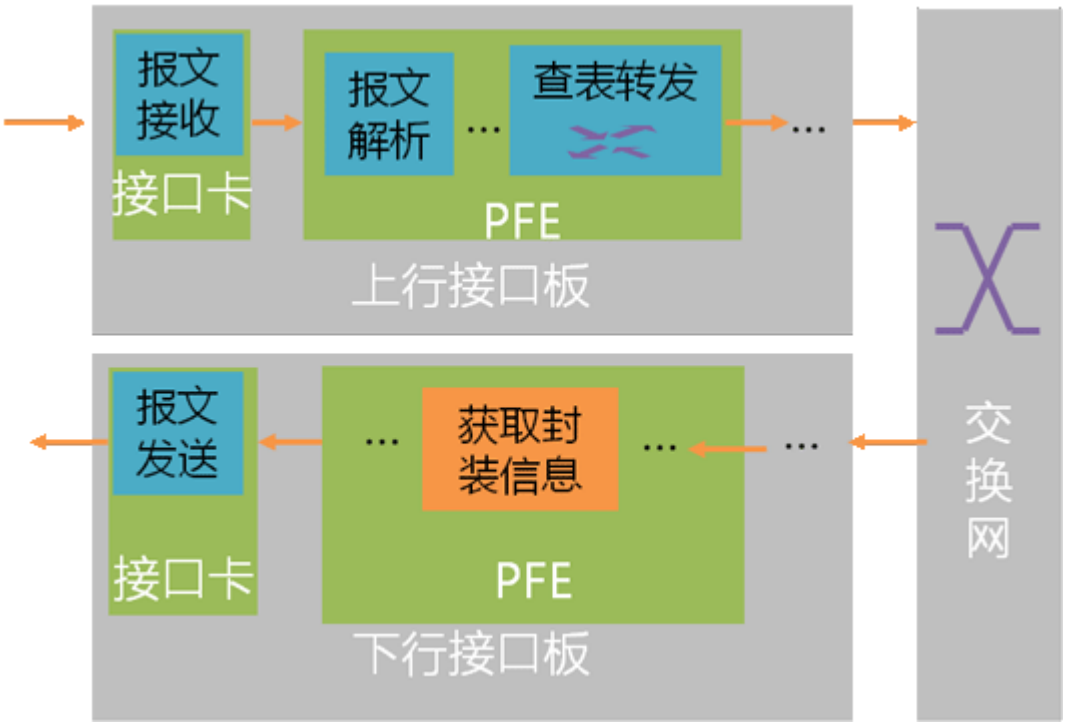
由主机PC-A向主机PC-B发送IP报文，那么该报文的目的IP地址就是PC-B的IP地址，源IP地址就是主机PC-A的IP地址，目标MAC地址就是其网关路由器Port1的MAC地址，源MAC地址就是PC-A的MAC地址。

路由器转发过程：

1. 路由器收到这个报文，发现其目的MAC为本机Port1端口的，表明需要本机来进行进一步解析（如果目的MAC不是本机，表明直接进行二层转发，不需要再解析帧的其他内容了）；
2. 路由器进一步解析报文，得知帧所承载的协议类型为IPv4（协议类型值=0x800），即需要进行IPv4转发；
3. 查转发表（FIB表），得知该报文并不是发给自己的，而是需要送往出端口Port2，因此，路由器不再继续分析IP头后面的内容。
4. 路由器将目的MAC更换成PC-B的MAC，将源MAC更换成出接口Port2的MAC，并将报文发给PIC卡，PIC卡将报文从Port2发送出去。

在上述过程中，将报文原来的源/目的MAC更换成新值的过程，称为“封装”；往报文里添加新的字段，也称为“封装”。需要被封装到待发送报文的信息称为“封装信息”。

那么，路由器是如何得到上述的封装信息（源MAC和目的MAC）呢？答案是：在路由器的下行接口板也对应有包转发引擎PFE，里面存有2张重要的表项，一张是IP地址与MAC地址的映射表（即ARP表），另一张是出接口属性表（该表有出接口对应的MAC地址）。

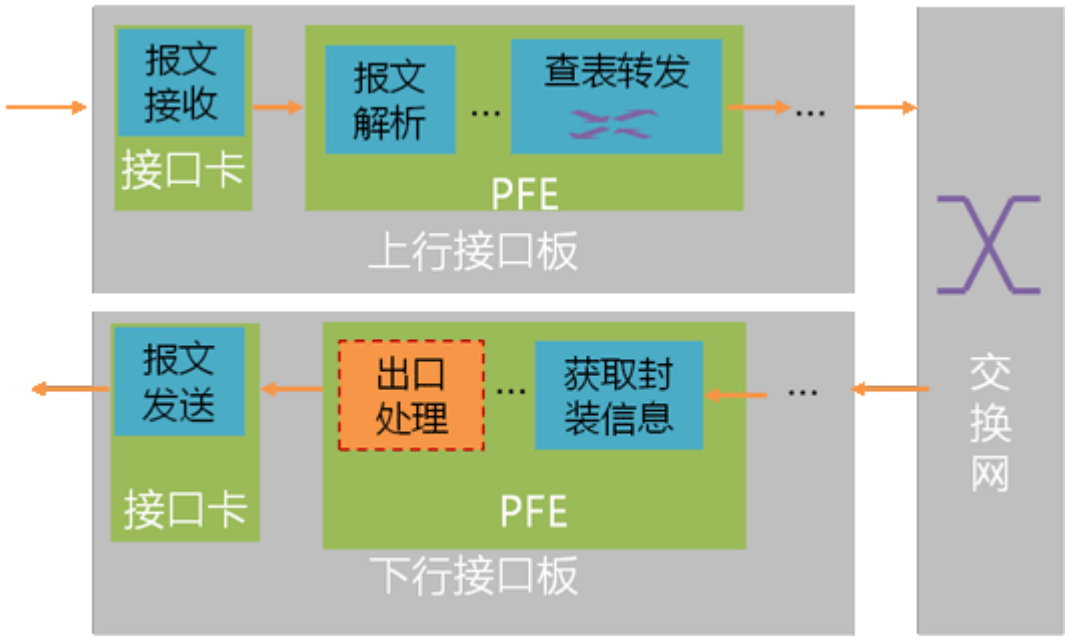


数据包经过上行PFE的查FIB表后，得到了报文的出接口。到了下行，下行PFE再根据报文的目的IP查找ARP表得到目的MAC，根据报文出接口查找出接口属性表便可得到源MAC。

上述是IP转发场景，封装信息包括链路层的源MAC和目的MAC。对于其他场景，除了这两个封装信息外，还需要获取其他的封装信息，例如，对于QinQ场景，需要加封装VLAN Tag；再如，对于MPLS场景，需封装MPLS标签。这些处理都是在下行接口板的PFE上进行。

2.4 出口处理

同上行，封装完成后，数据帧送往下行接口卡之前，也要根据出口属性表做出口检查和处理。例如，检查报文长度是否超出接口的MTU值，如果超出则进行分片或其他处理（关于MTU的详细介绍请参见《**MTU专题**》）。



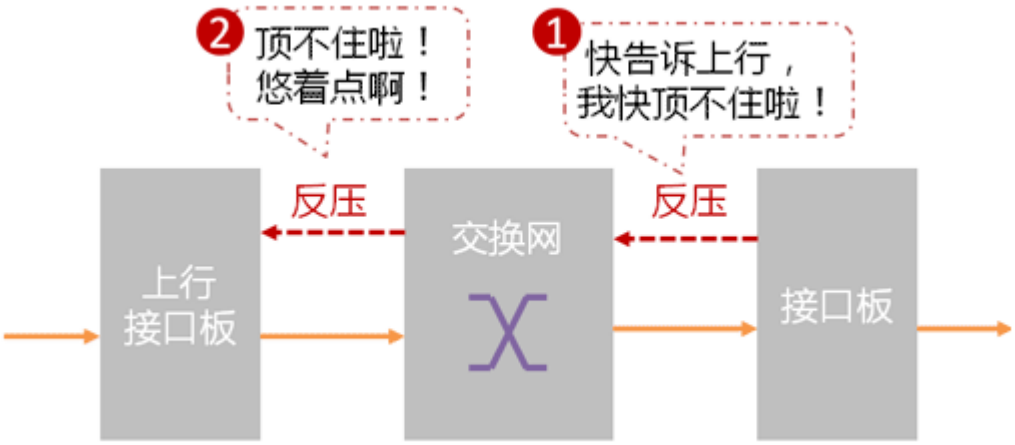
3 流量控制

关于本章

前两贴介绍了报文的接收、解析、转发、交换、封装、发送的操作过程。除了这些操作，路由器还有一个重要功能，便是流量控制。

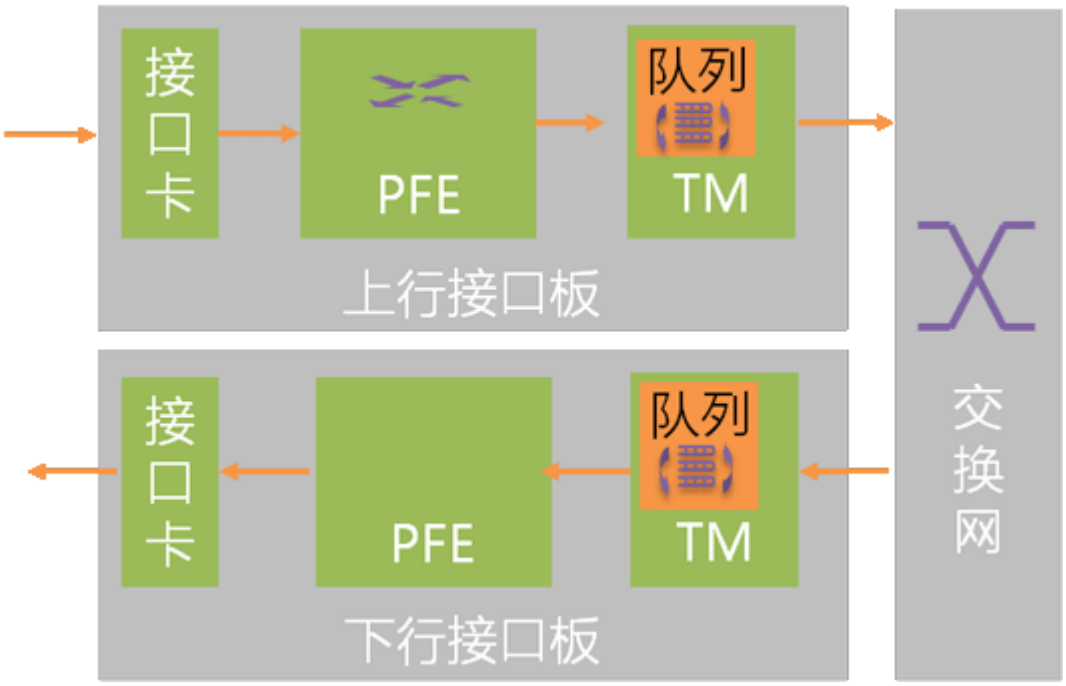
3.1 反压机制

报文经过上行接口板的处理之后，送往交换网板进行交换。所有报文都要经过交换网的交换，所以交换网是决定路由器性能的最核心单元，交换网应该是“无阻塞”的，其转发容量是所有接口板的转发容量之和，很强大。然而，下行接口板可能没那么强大，万一扛不住怎么办？为了避免这种现象，路由器上有“反压机制”，下行接口板扛不住，会发生反压，这样交换网板知道了，会通知上行接口板暂停发包，将报文进行缓存。



3.2 队列机制

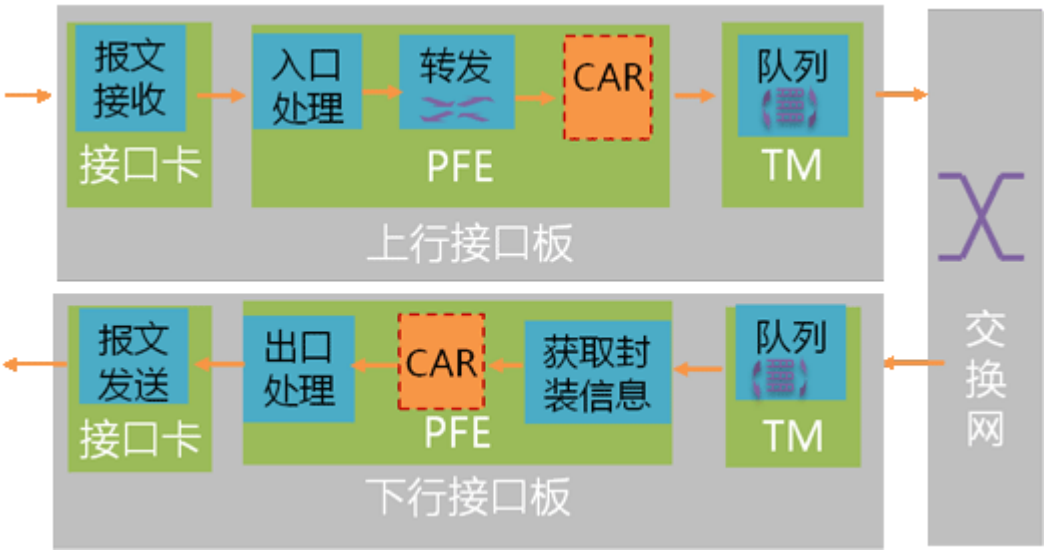
配合反压机制，华为高端路由器的接口板上有一块被称为TM（Traffic Management）的部件，里面有一些高速缓存，在拥塞的情况下，数据包以队列的形式暂存在缓存里，TM再按一定的规则把数据包调度出队，送往交换网。如果装不下，还会按一定规则做报文丢弃。



同理，报文经过交换网板送到下行接口板时，流量有可能超过出接口的发包能力，因此在下行接口板上也需要有TM芯片进行缓存和队列管理。在拥塞的情况下，数据包以队列的形式暂存在缓存里，下行TM再按一定的规则把数据包调度出队，送往出接口发送。

3.3 流量监管（CAR）

流量控制除了上述处理机制外，还有一种机制——流量监管，使得流量不超过入/出接口允许的带宽，超出限制的那部分数据包会被直接丢弃。目前流量监管使用的技术是CAR（Committed Access Rate）。CAR是由包转发引擎PFE完成的，可以在上行PFE上执行，限制流量不超过入接口带宽；也可以在下行PFE上执行，限制流量不超过出接口带宽。



注意：上送CPU以及CPU下发的协议报文不经过CAR处理，这是为了避免当流量突发时协议报文在因CAR被丢弃（不过，为了避免对CPU的攻击，上送CPU的协议报文会经过CP-CAR处理，详细请参见“协议报文之旅”）。

可能有人会有疑问，为了避免协议报文因拥塞被丢，不对其做CAR，那上文介绍队列机制时提到了丢弃策略，队列机制里有什么特殊措施避免协议报文被丢吗？为了避免协议报文在队列中被丢弃，协议报文一般都设置为高服务等级，入高优先级队列被优先调度，通常不会被丢弃。

说明

上述的队列机制、CAR、流分类和服务等级的概念及详细介绍，请参见下一章**4 QoS基础**。

4 QoS 基础

关于本章

路由器的转发层面，涉及到简单流分类、复杂流分类、流量监管CAR、队列和缓存，这些都是QoS（Quality of Service，服务质量）里的概念。因此，为了更好的理解这些概念，本章介绍QoS的基础知识。

4.1 为什么需要 QoS

通常，新商品刚出来时，用户和厂家不怎么关心质量问题。但出现了同类产品以后，用户就开始挑了，于是，商品就和质量挂钩了。IP网络也不例外。IP网络刚出现的时候，是“尽力而为”的，没有服务质量保证。你只知道报文发出去了，至于能不能到，什么时候到，就靠你运气了。这时的IP QOS，被称为“尽力而为”模型。

但随着技术的进步、竞争的加剧，客户的要求越来越高，IP网络提供有质量保证的服务是大势所趋。为此，出现了许多IP QoS服务模型，其中比较突出的有2种：集成服务（IntServ）模型和差分服务（DiffServ）模型。

4.2 集成服务模型

集成服务模型，要求用户要事先申请，声明想要什么样的服务，网络在资源满足的情况下，预留资源以满足该请求。就像旅游巴士一样，能保证每个人都有座位，但是有的车没坐满也照常开，座位空着就空着了。

而且，车辆服务方也要为此维护大量订车信息：

订车单

租车人：

联系电话：

住址：

取车地点：

退车地点：

使用时间： 年 月 日 点 分至

年 月 日 点 分止

共计 天

预定车型： 租金计算：

订金： 尾款：

承租方式： 附带驾驶人 自驾

所以，集成服务模型从上世纪90年代中叶提出至今，没有在IP网络中商用过。目前IP网络广泛使用的是差分服务模型。

4.3 差分服务模型

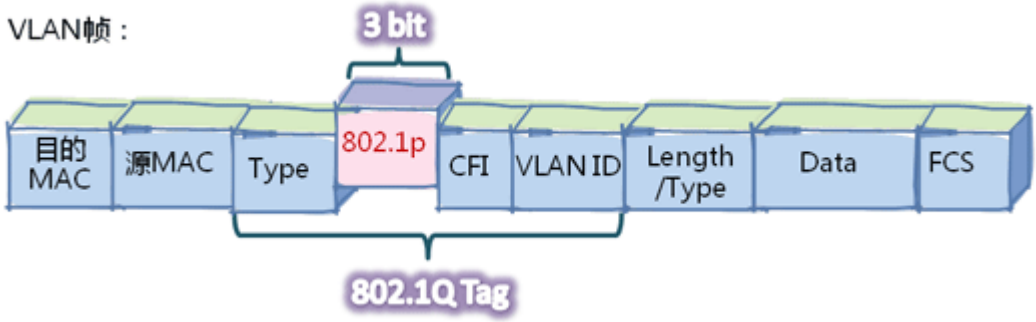
差分服务，其实就是将网络中的流量分成多个类，不同的类采用不同的处理。也就是说，先对流量分类，然后把类别标记在报文头中，通过报文携带到网络上，网络各节点只需要简单地识别报文中的这些标记，进行相应的处理。好比你去坐火车，火车票就是用来标记你所要享受的服务，是软卧、硬卧、硬座还是无座。火车票你随身携带，上车时你根据火车票进入对应的车厢，享受相应的服务：进入卧铺车厢的，可以舒服的躺着，进入硬座车厢，表示要坐着或站着。报文头里的那些标记就相当于“火车票”。

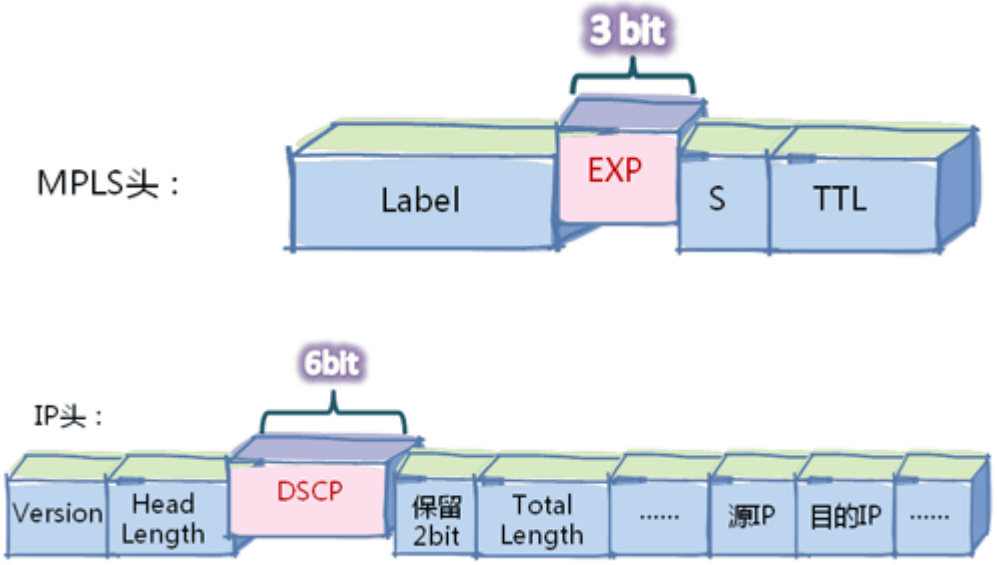
4.4 DSCP 与 PHB

DSCP和PHB是差分服务里两个很重要的概念。理解差分服务，先从这两个概念开始。

DSCP

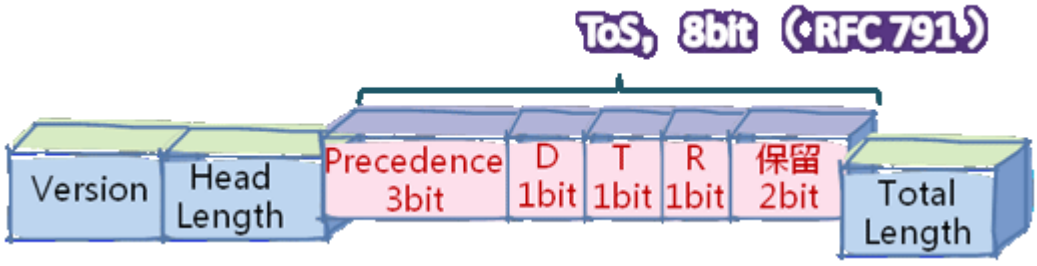
前面说过，报文头里的那些标记就相当于“火车票”。那么，报文头有很多，二层的帧头、2.5层的MPLS头、3层的IP头，那些标记到底是标在哪层里呢？其实，每层都有标记，对应的标记分别被称为802.1P，EXP、DSCP。



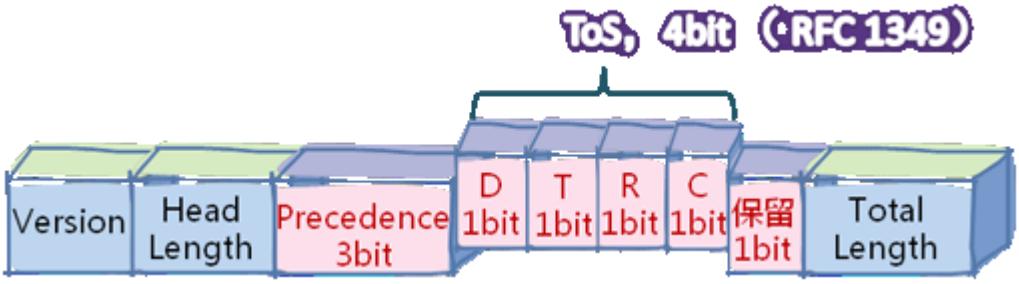


为啥每个层都要有这个标记呢？因为报文在网络中传输，中间可能要经过二层以太网、MPLS网络、IP网络等，不同网络转发行为不一样，比如二层，网络节点只分析以太帧头，不会再去分析里面的MPLS头或IP头了。

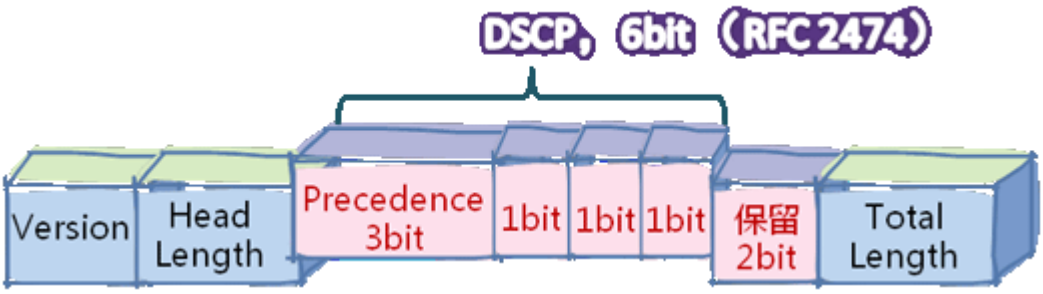
除了802.1P，EXP、DSCP，书上说还有一种标报文头里的那些标记就相当于“火车票”记叫ToS，它又是虾米东东呢？其实，ToS也是IP头里的字段。IP头的经历也比较坎坷，最初，标准定义的IPv4报文头，并没有DSCP，而是有个8bit的ToS域，且前三位才是表示优先级的，即Precedence，一共才8种等级。



后来，标准重新定义了这个域，将保留位的其中一比特定义为C比特位，并将D、T、R和C共4个比特位称为ToS域。所以提ToS，得区分是哪个标准的。华为路由器遵从的是RFC 1349。



再后来，标准又重新定义了这个8比特，把前6位改名为DSCP。



PHB

PHB是Per Hop Behavior的缩写，即每跳行为，即设备对报文的处理。可能有人理解为PHB就是调度、丢包、监管、整形、重标记等动作，实际上不完全是，这些动作是设备具体的行为，而PHB只定义了一些外部可见的转发行为，并没有指定特定的实现方式。说白了，PHB就是将各种行为，概括成几类。就如同星级酒店的标准，可以有3、4、5这样的星级。RFC定义了四类标准的PHB，并用CS、EF、AF、BE这些符号来表示，每类PHB都对应一组DSCP。其实，PHB这样分类，是根据那些可见的服务特征，如时延、抖动或丢包率。

CS、EF、AF、BE！没错，我们经常见到它们。

- BE，没有质量保证，一般对应于传统的IP分组投递服务，只关注可达性，其他方面不做任何要求。IP网络中，缺省的PHB就是BE。任何路由器都必须支持BE PHB。
- AF，代表带宽有保证、时延可控的服务，适用于视频、语音、企业VPN等业务。
- EF，低时延、低抖动、低丢包率，对应于实际应用中的视频、语音、会议电视等实时业务。
- CS，因为现网有些存量设备不支持差分服务，只解析DSCP前3位，为了后向兼容，标准预留了所有格式为XXX000的DSCP值，这类值就对应为CS PHB。

可是，大家经常看到的是，AF带有后缀的，比如AF11、AF21等，CS也有CS6、CS7等，而BE、EF都不带后缀。这怎么回事呢？那是因为，BE和EF对应的只有唯一的一个DSCP值，CS和AF有多个DSCP值与之对应。例如AF，被细分为4个等级，且每个等级有3个丢弃优先级，其表达形式为：AF1x~AF4x（x代表丢弃优先级，取值为1~3）。

举个例子来说明AF怎么用。假设有4个小区的网络，接入到ISP的同一台边缘路由器。如果某个小区发送了大量的FTP数据，可能导致拥塞，干扰其他小区的FTP传输。为了公平，约定每个小区FTP总速率不能超过500M。但有时他们可随意发送，甚至会超过1Gbps。怎么办呢？可以在每个入接口上（用CAR）监测FTP速率，并重标记报文的DSCP。如果速率小于等于500M，标记为AF11，如果速率在500Mbps~1Gbps之间，标记为AF12，如果速率超过1Gbps，标记为AF13。当拥塞发生时，优先丢弃AF13，其次AF12，AF11就会在最后被丢弃。皆大欢喜。

PHB = 服务质量？

PHB体现的是等级的高低，而不是服务质量(QoS)的好坏，不能说CS的等级最高，服务质量最好；BE等级最低，质量最差。PHB只是逐跳行为，而服务质量是端到端的。服务质量，通常用如下参数来衡量：

- 带宽/吞吐量
- 时延

- 时延差（抖动）
- 丢包率
- 可用性

除了PHB，还有很多个因素会影响服务质量，例如链路带宽、设备处理性能、网络的稳定性、传输距离，……

PHB = 队列?

华为路由器上的队列名称，也叫BE、AF1、AF2、AF3、AF4、EF、CS6、CS7。是否就是PHB？

其实，队列不等同于PHB，即队列的名称不代表服务等级的高低。举个例子，假如某车厢是硬座车厢，一开始被称为“硬座”车厢，如果把该车厢里的座椅全改装成硬卧铺位，那么该车厢虽然叫“硬座”车厢，实际的PHB是“硬卧”。 队列也可以做类似改装，例如将BE队列配置为SP调度（当然一般不会这么做），其余队列配置为WFQ调度，则BE队列的服务等级比其余7个都高，其PHB不再是BE了（SP、WFQ的概念下面的“队列机制”会介绍）。是不是已经很晕了，既然队列名称不代表PHB，那为啥要这么要跟PHB取相似的名字呢？其实，这是为了更方便、更直观看出各队列优先级的相对高低。这总比队列1、队列2、……队列8这样命名强一些吧。

如何根据 DSCP 入队列?

可能有人会有疑问，802.1P和EXP都是3比特，刚好8个值，对应8个PHB或8个队列，那DSCP有64个值，怎么和8个PHB/队列对应呢？这个是“简单流分类”考虑的事。

简单流分类（入映射与反映射）

简单流分类，其实一点都不简单。之说以叫“简单流分类”，是因为它的分类规则，相对于“复杂流分类”而言更简单而已。简单流分类，是采用简单的规则（例如，只根据IP报文的优先级标记字段，如IP的DSCP、MPLS的EXP或VLAN的802.1p），对报文进行粗略的分类。而复杂流分类，采用的分类规则稍微“复杂”一点，可以根据除了优先级标记之外的字段（例如，五元组，MAC、协议号、label、TTL等等），对报文进行精细的分类。

前面说过，差分服务是先分类，打标记，然后再进行各种PHB动作。然而，无论哪种流分类，都是根据报文头的字段进行的分类，而报文头的这些字段值需要解析报文才能获得。不可能每次QoS动作，比如入队列、丢包，调度出队等，都去解析报文头吧。

因此，设计师们想了个巧妙的办法：在设备内部为每个报文设置了两个内部标记：Service-Class和Color，对应服务等级（也称为调度优先级）和丢弃优先级。设备在解析报文头的时候，根据报文头的优先级来设置这两个内部标记。这样，设备在执行各种QoS动作时，读取这两个标记就行了。也就是说，设备的各种QoS动作都是围绕这两个内部标记进行的。

IP网络中，缺省的PHB就是BE，所以这两个内部标记初始值设为<BE, Green>。初始化之后，如果入接口配置了trust upstream命令，则设备上行单板（接口入方向）解析报文时，会根据报文优先级，对照入映射表来重新设置<Service-Class, color>，这一过程称为“入映射”。

如果对报文配置重标记（remark），或者car后remark，无论在入接口或出接口，设备都会去修改<Service-Class, color>。之后，设备根据<Service-Class, color>来对报文进行队列调度等一系列QoS动作。这些QoS动作做完之后，设备下行单板（接口出方向）还要做一个动作：根据内部优先级修改报文优先级，这个过程称为“反映射”。当然，反映射也是可选的配置，毕竟有些场景是不希望改变报文优先级的。

复杂流分类（流策略）

复杂流分类，可以采用复杂的规则，如由五元组，对报文进行精细的分类。分类之后，要和动作关联起来才有意义。将流分类和对应的流动作关联，就是流策略。流策略在配置上采用“模板化”方式，“模板化”的最大优点是可以节省配置，支持批量修改。

流策略“模板”分为三部分：

- 流分类（Classifier）：用if-match语句设定流分类的匹配规则。
- 流动作（Behavior）：定义针对该类流量实施的流动作，例如重标记、重定向、负载分担、报文分片、流量限速、流量统计等等。
- 流策略（Policy）：将流分类Classifier和流动作Behavior关联，设置完毕后还需要应用到流量的入接口/出接口。

4.5 拥塞管理（队列机制）

除了流分类和标记，差分服务比较优秀的地方还有强大的队列技术。当网络拥塞，也就是进来的报文太多，设备处理不过来，就让报文排队。日常生活中，求大于供时，基本也用“排队”解决的。不过差分服务的队列技术比现实生活的强大多了。就拿医院排队挂号做例子吧，华为路由器每个端口都设有8个队列，相当于8个挂号窗口。

队列调度算法

对于同一个窗口，先来先服务，也就是先进先出FIFO（First In First Out）。但是对于设备的一个接口，同一时间只能处理一个队列，不像医院每个窗口都坐着一个人。也就是说，对于8个窗口，同一时间只有一人，那么哪些窗口的先处理呢？可以有很多“猫腻”——队列调度算法。

比较著名的队列调度算法，比如严格优先调度SP（Strict Priority），相当于医院挂号处设一些VIP窗口，让一些腿脚不便的人能尽快办完，只有VIP窗口的办完了才办理其他窗口。如果哪天来一堆腿脚不便的，其他窗口就办不了了。所以路由器对于高优先级的队列，都限制了带宽，防止其他队列“饿死”。

其实，比较公平的做法是每个队列轮着来，即轮询调度RR（Round Robin）算法，但有些人一次挂几十个号，所以最好是每人每次限挂一个号。但是，这对一次挂好几个号的非常不方便了，轮一圈才挂一个号，那要挂几个号就得轮几圈才能办完，等的时间太长了。所以科学家又发明了一种算法，给队列设立权重。假设有3个非VIP队列，权重分别设为5:3:1，你多交点钱，到权重为5的队列去，一次给你挂5个号，很省事。这种算法很常用，叫加权公平队列WFQ（ Weighted Fair Queuing）算法。

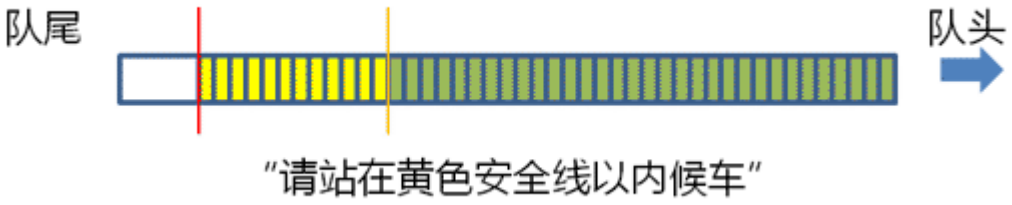
队列调度算法还有很多，不逐一介绍了……（此处省略5万字），这是路由器比医院挂号机制强的地方。

拥塞避免（丢弃策略）

在医院，为了预防医生处理不完，就限定数量和时间，比如每天的挂号200个，而且限定早上7点~11点，下午2点~4点，其他时间不受理。而路由器处理的是网络报文，虽然路由器无时不刻地劳作，但网络突发性很强，经常拥塞，眼看着队列都快排到大门口了，怎么办呢？这时候，路由器就会动用他的秘笈——“丢包”策略。

目前有两种丢弃策略，一种是“尾丢弃（Tail-drop）”。“尾丢弃”就是在网络拥塞时把新到的报文丢弃。这么做有个问题，如果这些被丢的报文属于多个TCP连接，那这些TCP连接会同时减少发包，这样发往队列的报文将同时减少，而后它们又慢慢尝试增加发包数，于是又会在某个时间同时出现流量高峰……如此反复。这就是著名的“TCP全局同步”现象。这可如何是好？TCP/IP两大传输协议（TCP和UDP）都得罪了，那还怎么玩？

没事，路由器拿出了飞机中的战斗机，另一种丢包策略——加权随机早期检测WRED（Weighted Random Early Detection）技术。WRED原理很简单，就是为每个队列都画两条线，如下图。



当队列短，还没排到黄色线，不丢报文；当排到黄色线就开始随机丢掉一些新来的报文，队列越长，丢得越多；当排到红色线时，新来的都丢弃。

对于需要提供最大限度的带宽保证的业务，一般进入SP队里，建议采用尾丢弃，队列满了再丢，这样可以最大限度的保证其带宽。而WFQ队列，是按权重分享带宽，容易发生拥塞，采用WRED策略有效的避免了TCP全局同步现象。

但是这样做，你可能还是感觉不是很公平。比如咱俩都在上网，我发了2M，你发200M，结果拥塞了，其实是你用得多导致拥塞的，应该先丢你的啊。为了解决这个问题了，科学家们又造出了一个词——丢弃优先级。IETF标准定义了三个丢弃优先级值，为了方便记忆，用红、黄、绿三种颜色来表示报文已经入队列后，拥塞时被丢弃的优先顺序。比如刚才的问题：你发2M我发200M，那就设置一个上限100M，你发的没超过，全是绿的，不会被丢弃，而我发的超过100M后，就将超过的那部分报文标红优先丢弃。

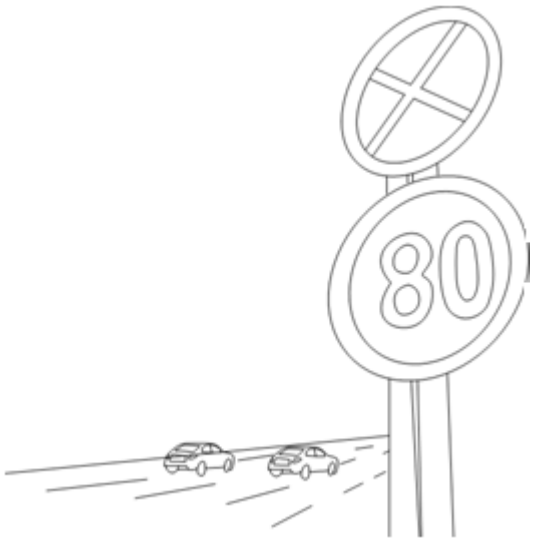
4.6 流量限速（CAR 和流量整形）

流量限速也是QoS中的一个重要的概念。对于一台路由器而言，流量限速，是指限制进/出这台路由器的数据流的速率。

流量限速有两种技术：一种是CAR（Committed Access Rate），另一种技术就是流量整形。

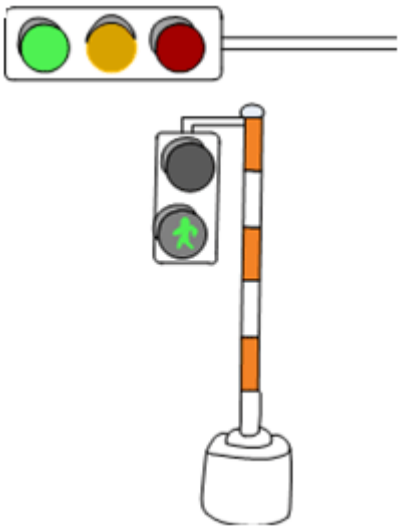
CAR

CAR是用来限速的，将进入网络某一流量的速率限制在约定的范围之内。



- 没有超速，奖励绿卡，可畅通无阻（转发）。
- 稍微超速，黄牌警告，一般降级（remark后再发）。
- 过分超速，红牌罚下，禁止通行（丢弃）。

跟红绿灯规则挺像，“红灯停，绿灯行，黄灯等一等”。



为啥叫“CAR”？那是因为，过分超速，就咔咔咔，将超速的那部分报文卡掉。

那如何判定是不是超速呢？答案是：使用令牌桶算法。

令牌桶算法

所谓令牌桶，就是用来装令牌的桶！那么，令牌桶怎么测速的呢？

令牌桶有三种测速方法，对应三种机制：

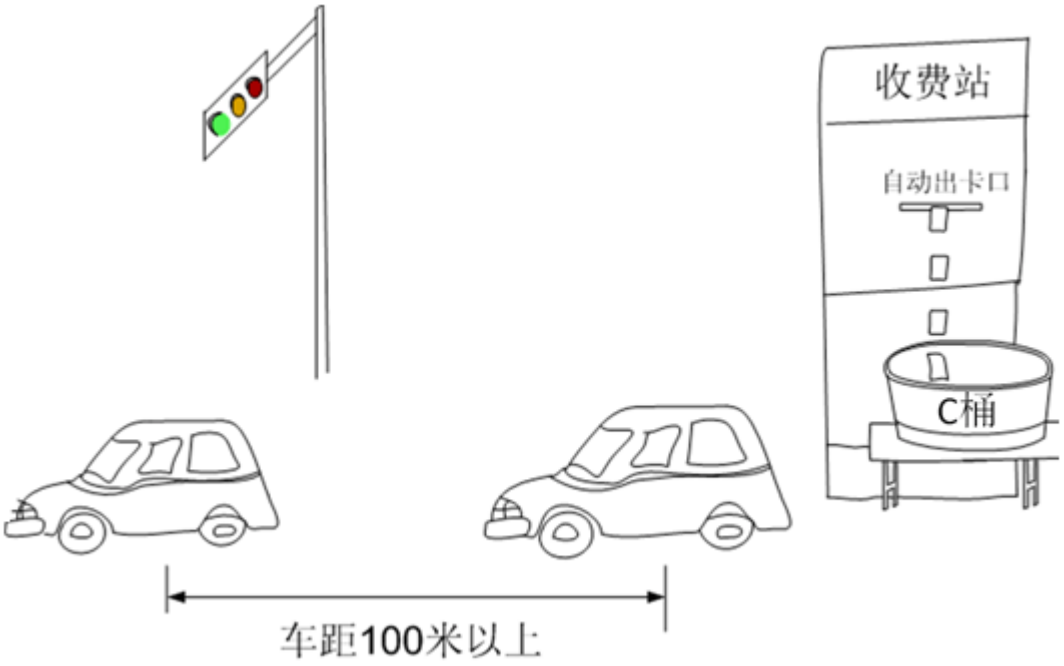
- 单速单桶
- 单速双桶
- 双速双桶

先介绍最简单的单速单桶。首先，单速单桶算法以CIR（Committed Information Rate，承诺信息速率）的恒定速率往令牌桶放令牌。

接着，规定报文要从令牌桶领到令牌才能转发。就像车要进入高速公路，需要在入口领张通行卡。

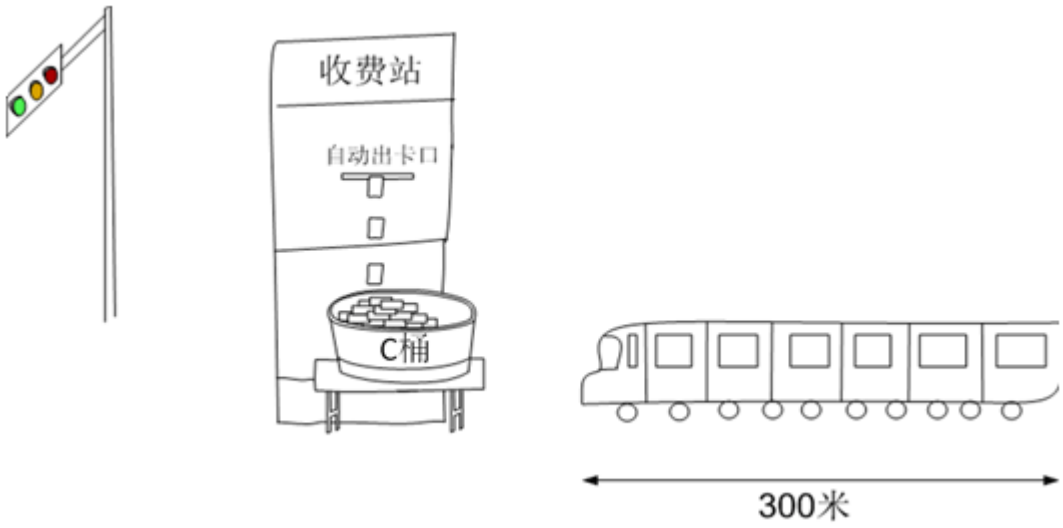
想象一下：在高速入口没有收费员，而是设台自动出卡机，出卡口下放个桶，称之为C桶吧。出卡机以恒定速率往C桶放通行卡，过路司机自己取通行卡。

为了保持100米车距（《道路交通安全法》规定，高速公路行驶的车辆，车速超过每小时100公里时，应与同车道前车保持100米以上距离。），间隔3.6秒以上才能放行一辆车，因此出卡机每3.6秒出一个通行卡。



如果平均每3.6秒来了不止1辆车，通行卡很快就耗光，触发红灯亮，新到的车禁止进入高速。这就起到了限速作用，将速率限制在放卡速率范围内。

如果平均每3.6秒来了不到1辆车，桶里就可以累积一些通行卡。累积通行卡的好处是，可以应付这种突发情况：某时刻突然来了长车（几百米长，有很多节车厢）。 不过需要规定，有几百米就拿几个通行卡。因为通常每100米最多一辆车，而长车长几百米，相当于一下子来了几辆车（高速路上每100米最多一辆车,那一辆300米的车，相当于3辆普通车）。

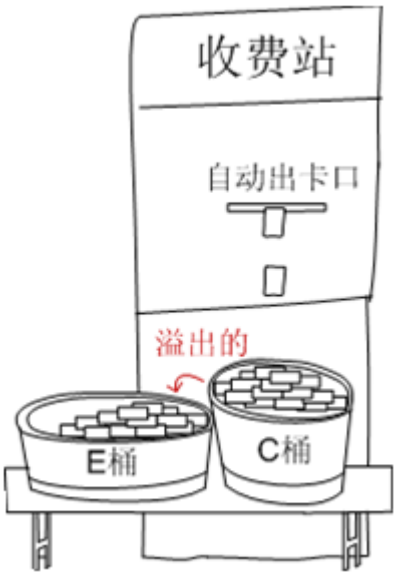


假设某时刻来了600米的长车，C桶里只有5个卡，那得等积累到6个卡才能过。

那么，能通行的长车，长度最多是几百米呢？取决于C桶里最多一次能容纳几个卡。C桶的容量称为CBS（Committed Burst Size，承诺突发尺寸）。

但是，如果一直没有来车，桶满了，但出卡机一直在出卡，岂不浪费？单速单桶就是这么浪费。其实可以拿个大桶来接溢出来的卡，单速双桶就是这么设计的，这样就不那么浪费了。既然讲到单速双桶，那接下来就介绍单速双桶吧。

设想，有个高速路系统，设置了2个桶（C桶和E桶）接通行卡，C桶在出卡口下接通行卡，等C桶接满，溢出的放E桶。



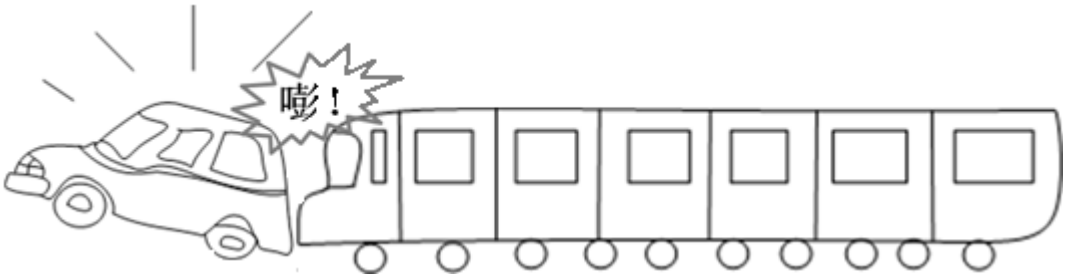
并规定，必须先拿C桶的，C桶不够把卡放回C桶再拿E桶的，不能同时拿两个桶的：

- 如果C桶足够，绿灯亮，通行，不扣分；
- 如果C桶不够E桶够，黄灯亮，罚扣1分，从C桶拿的令牌要归还；
- 如果C桶不够E桶也不够，红灯亮，罚扣12分，并禁止通行，通行卡要归还。

来了之后，司机先拿C桶的；如果不够，就全放下，重新拿E桶的。车有几百米长，就拿几个卡。

那E桶多大，跟C桶一样，容量也是CBS吗？不是。E桶容量叫EBS（Extended Burst Size，超额突发尺寸）。

E桶不能无限大。E桶太大的话，允许通过的车太长、太重，惯性就会太大，车距100米，刹车刹不住，是要追尾的，后果很严重！！



CBS也不能太大，否则允许通过的车太长，同样也会追尾。不过，报文是不会追尾的。对于报文，CBS设置太大，起不到限速作用。 例如，本想限速为100bit/s。假设CBS设置为3600MBytes，某个时间点，令牌桶已充满令牌。这时突然哗哗哗，1小时之内来了各种长度的报文，共3600MBytes，这些报文都能得到令牌，都被转发了。那么这1小时的报文速率为1MByte/s=8000000bit/s，而不是100bit/s（1小时=3600s，1MByte=8000000bit）。

CBS也不能设置太小。例如，本想限速为100bit/s。假设CBS设置为1000Bytes。某个时间点，令牌桶已充满令牌，这时突然哗哗哗，1小时之内来了各种长报文，都比1000Bytes长，那这些报文谁都没有获得足够令牌，因此都被丢弃。那么这1小时，一个报文都没转发，报文速率为0，而不是100bit/s。

CBS不能太小，又不能太大，到底怎么设呢？首先，CBS不能小于MTU；其次，CBS不能小于网络的正常突发量。但网络正常突发量不好确定，因此有人总结了一些经验公式。华为路由器的经验公式：

- CIR <= 100Mbps时，CBS(Bytes) = CIR(kbps) × 1.5(s) ÷ 8
- CIR > 100Mbps时，CBS(Bytes) = 100,000(kbps) × 1.5(s) ÷ 8

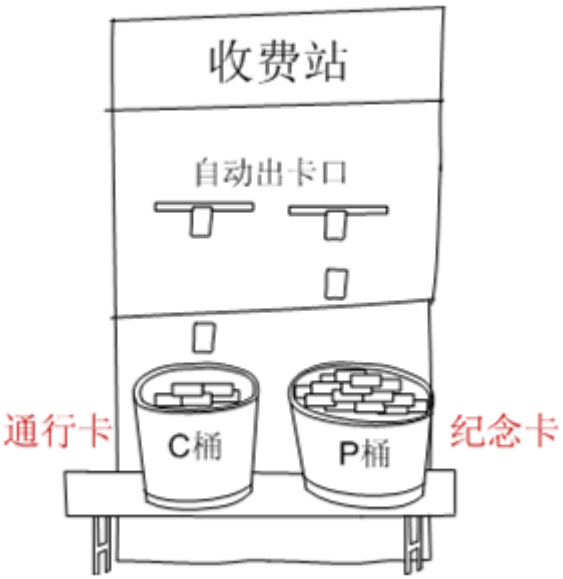
那么，CIR的设置有什么讲究吗？用户购买多大带宽，CIR就设多大。运营商和用户之间签订的SLA（Service Level Agreements）就有CIR、CBS、PIR、PBS等参数。

说明

SLA（Service Level Agreements），服务水平协议，是指用户和服务提供商签署的关于业务流在网络中传递时所应当获得的待遇。

那么，什么是PIR和PBS呢？这两个是双速双桶里的概念。接下来介绍双速双桶吧。

双速双桶就好比是高速入口的那台出卡机有两个出卡口，分别以CIR和PIR（Peak Information Rate，峰值信息速率）的速率出卡，在每个出卡口下都放一个桶。这两个桶分别叫C桶和P桶，C桶的是通行卡，P桶的是纪念卡。



并规定，

- 先拿纪念卡，看是否足够。如果不够，把卡放回P桶，红灯亮，禁止通行，罚款1000元；
- 如果纪念卡足够，可以接着拿通行卡，如果通行卡足够，绿灯亮，通行，把纪念卡和通行卡都拿走；
- 如果通行卡不够，则黄灯亮，回去整改，罚扣1分，纪念卡可以拿走，通行卡放回C桶。

可能有读者会觉得令牌桶算法太复杂，且双速双桶和单速双桶的规则还不一样。其实很简单，概括为：

	双速双桶	单速双桶
	先P桶后C桶	先C桶后E桶
绿灯	两桶都拿	只拿C桶
黄灯	只拿P桶	只拿E桶
红灯	两桶都没拿	两桶都没拿

双速双桶不仅可以测量出是否超速，而且还可以测出超速的程度是否超出法律允许范围。P桶的出卡速率就是法律允许的最高车速。P桶的容量PBS跟EBS是一样的功能，所以有了P桶就不用E桶。

以上为CAR的原理，那么，什么时候要用CAR呢？答案是，一般是在网络入口做限速。就像高速路，通常在入口设收费站，起到限速作用。长假期间，高速入口不收费，结果，高速变成了“龟速”！

那么，应用CAR时，如何选用三种令牌桶？答案如下：

- 只限速，用单速单桶；
- 限速+区分突发量，用单速双桶；
- 限速+区分突发量+区分带宽是否超出峰值，用双速双桶。

流量整形

同CAR一样，流量整形也采用令牌桶算法，但不同的是：流量整形是在队列机制里使用，用于限制某个或某些队列的输出速率，超速了就暂停输出，超速的报文暂时存在缓存里，等空闲了还是会输出，不被丢弃；而CAR与队列无关，CAR是没用到缓存的，超速的报文直接丢弃。

说明

上述的流分类（简单流分类和复杂流分类）、流量限速（CAR和流量整形）、拥塞管理（队列机制）和拥塞避免（丢弃策略）称为QoS的四大组件。下一章将详细介绍这四大组件的处理流程。

5 QoS 处理流程

关于本章

上一章介绍了QoS的基础知识，本章将进一步介绍QoS在路由器转发层面的处理流程。

5.1 QoS 的处理顺序

QoS涉及四大组件：流分类和标记（简单流分类、复杂流分类）、流量限速（CAR和流量整形）、拥塞管理（队列调度）、拥塞避免（丢弃策略），这四大组件在华为设备上是按一定的顺序进行处理的。

有的单板的接口卡内嵌了一个TM芯片，称为eTM（egress Traffic Manager）子卡，有的单板的接口卡没有eTM子卡。带有eTM子卡和没有eTM子卡，对于QoS来说，差别只在于下行的队列调度：带有eTM子卡时，下行队列调度在eTM子卡上进行，没有eTM时，下行队列调度在TM上进行。

图 5-1 无 eTM 子卡时的 QoS 处理流程图

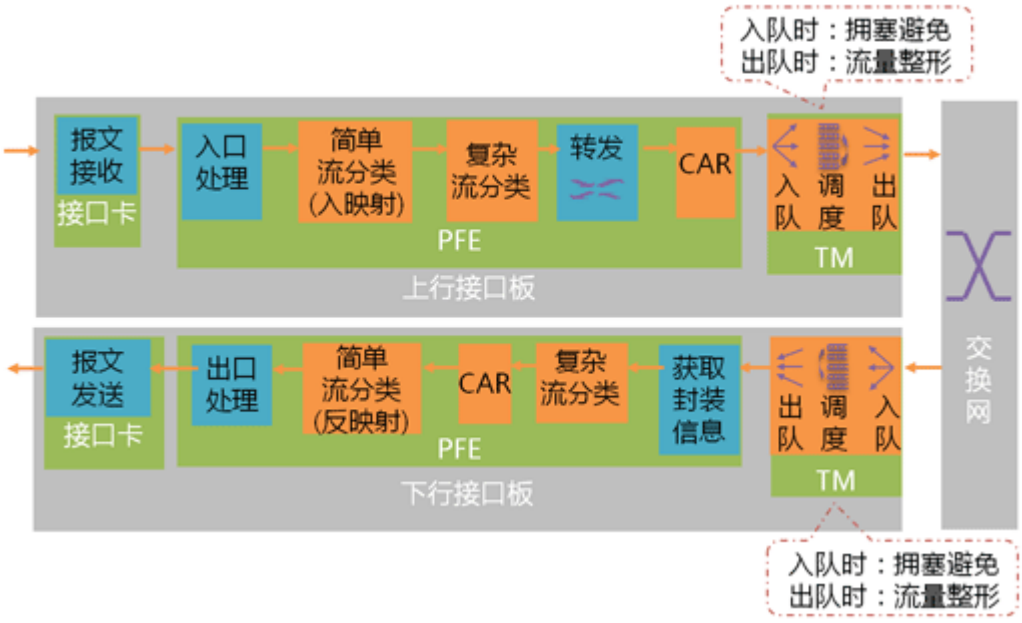
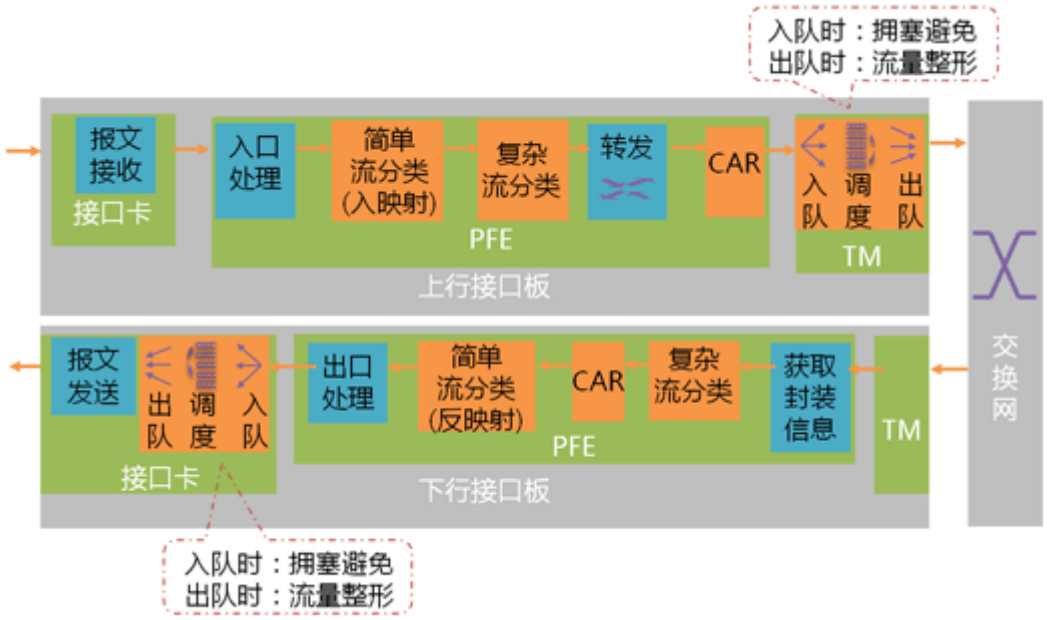
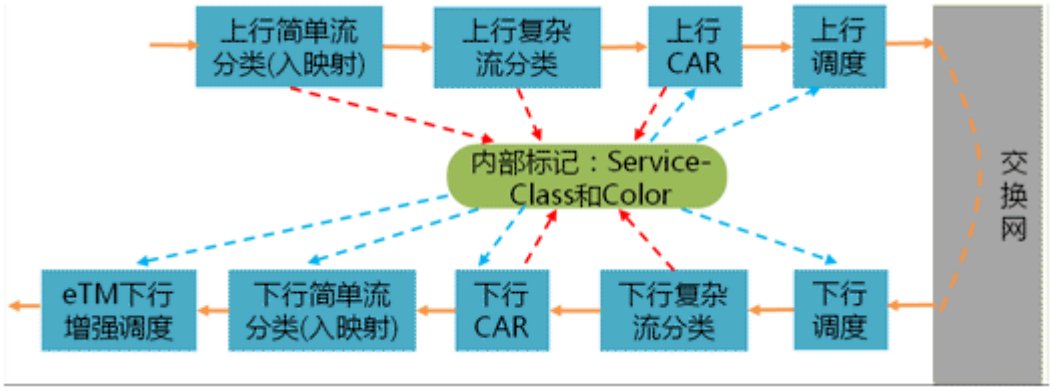


图 5-2 带 eTM 子卡时的 QoS 处理流程图



其中，流分类和标记是实现QoS差分服务的前提和基础；流量监管、流量整形、拥塞管理和拥塞避免是提供差分服务的具体体现。前一章《QoS基础》在介绍简单流分类时介绍过，华为路由器为每个报文设置了两个内部标记：Service-Class和Color，对应服务等级（即调度优先级）和丢弃优先级，设备的后续各种QoS动作都是围绕这两个内部标记进行的。



具体过程：

1. 首先上行PFE初始化报文的内部流标记（Service-class=BE，Color=Green）；
2. 根据入接口配置进行上行简单流分类处理，即提取报文的优先级字段（8021p、DSCP或MPLS Exp）进行流分类，重新设置报文的内部流标记（Service-class & Color）；

3. 根据入接口配置进行上行复杂流分类处理，即提取报文的多个信息字段进行流分类。根据分类结果执行包过滤、重标记、重定向等处理。如果是重标记（remark），则会再次重新设置内部流标记（Service-class & Color）。
4. 查表转发，获得报文的出接口、下一跳信息。
5. 根据报文入接口下配置的CAR，或者是上行复杂流分类中配置的CAR进行上行CAR处理（如果同时配置了接口CAR与复杂流分类CAR，最终按复杂流分类CAR处理）。在CAR处理中，可以根据输入流量的情况执行Pass或者Drop动作，也可以支持Pass+Remark的处理。如果是Pass+Remark处理，则再次重新设置内部流标记（Service-class & Color）。
6. 之后，报文根据Service-class进入上行TM的队列，进行队列调度，根据Color做WRED处理（如果配置了WRED的话）。
7. 报文经过交换网交换到下行。
8. （下行PIC存在eTM的情况下，将跳过此步骤）报文根据Service-class进入下行TM的队列，进行队列调度，根据Color做WRED处理（如果配置了WRED的话）。
9. 报文进入下行包处理引擎PFE，下行PFE获取报文封装信息。
10. 下行PFE根据出接口配置进行下行复杂流分类处理，即提取报文的多个信息字段进行流分类。根据分类结果执行包过滤、重标记、重定向等处理。如果是重标记（remark），则会再次重新设置内部流标记（Service-class & Color）。
11. 根据报文出接口下配置的CAR，或者是下行复杂流分类中配置的CAR进行下行CAR处理（如果同时配置了接口CAR与复杂流分类CAR，最终按复杂流分类CAR处理）。在CAR处理中，可以根据输入流量的情况执行Pass或者Drop动作，也可以支持Pass+Remark的处理。如果是Pass+Remark处理，则再次重新设置内部流标记（Service-class & Color）。
12. 进行下行简单流分类反映射：根据Service-class & Color设置（新封装的报文头）或者改写（已经存在的报文头）出口报文的优先级字段。

13. 经过下行PFE的出口检查和处理后，报文进入下行接口卡。
    - 对于无eTM的接口卡，添加链路层帧间隙、前导码、帧开始界定符和FCS，将报文转发到物理链路上；

- 对于有eTM的接口卡，添加完链路层帧间隙、前导码、帧开始界定符和FCS后，将报文转发到物理链路之前，将进行一轮队列调度：报文根据Service-class进入下行eTM的队列，进行队列调度，根据Color做WRED处理（如果配置了WRED的话）。

5.2 常见 FAQ

报文的优先级字段啥情况会发生变化？

也许有人有这样的困扰：有时不想让报文优先级改变，但实际它却改了；有时我想让它改，它却没改。到底怎么回事？

答：前面提到过，下行简单流分类的反映射操作会根据<service-class, color>来设置新增的优先级字段或修改报文携带的优先级字段，而这个<service-class, color>在整个QoS处理流程中许多环节都会对其进行更改。所以，判断报文的优先级字段改变了没有，需要分析如下几个因素：

1. <service-class, color>经过整个流程后，是否发生了变化。
2. 在入映射规则和反映射规则是否一致，比如，DSCP=12，入映射为<service-class=AF1, color=Yellow>，那么反映射规则中，<service-class=AF1, color=Yellow>是否反映射为DSCP=12。
3. 下行是否做反映射动作。

   特别注意，大部分类型的单板的上、下行复杂流分类的重标记（Remark命令）会直接去修改报文优先级字段，只有LPUF-21/40这类单板是特殊的：它的上行Remark命令不会直接修改报文优先级字段，不过它的下行Remark命令还是会直接改报文优先级字段的。

如何判断下行是否做反映射？

答：设备给每个接口设置了两个“开关”来控制对该接口出方向的报文做“反映射”：

- 开关1：BA。这个开关是在流量的入口配置的，通过内部添加的信息头，会从入口经过交换网传递到出口。
- 开关2：PHB。这个开关是在流量的出口配置的。

只有两个开关同时打开，才会做反映射动作（但是下行为LPUF-41/100单板时，只要PHB开关打开，不管BA开关是什么状态，都做反映射）。默认情况下，BA开关为关闭状态，PHB开关在V600R002及之前版本默认为打开状态，V600R003及后续版本为关闭状态。

- 打开“BA”开关的方法：接口入方向配置“**trust upstream**”、“**remark**”、“**service-class**”、“**diffserv-mode pipe**”或“**diffserv-mode short-pipe**”命令。其中“**diffserv-mode pipe**”和“**diffserv-mode short-pipe**”命令仅用于MPLS场景且仅对MPLS报文生效。
- 关闭"BA"开关的方法：接口入方向配置**service-class** *class-value* **color** *color-value* **no-remark**命令，或者移除上述4条命令的配置。
- 打开“PHB”开关的方法：接口出方向配置“**trust upstream**”或者“**qos phb enable**”命令。
- 关闭"PHB"开关的方法：接口出方向配置"**qos phb disable**"命令或者删除"**trust upstream**"命令。

如果报文同时携带了 DSCP、802.1p 和 MPLS EXP，选哪个做入映射？

答：取决于入接口的配置：

- 入接口没配置**trust upstream**命令，表明不信任报文优先级，报文被映射到默认的<BE, Green>；
- 入接口同时配置**trust upstream**命令和**trust 802.1p**命令，则带VLAN的报文按外层802.1p做映射；没VLAN的映射到<BE, Green>；
- 入接口只配置trust upstream命令，如果是MPLS报文按EXP映射；非MPLS的IP报文按DSCP映射；其他的，如果是设备认识的协议报文，映射到<CS6,Green>；设备不认识的映射到<BE, Green>。

如果报文同时携带了 DSCP、802.1p 和 MPLS EXP，反映射改哪个？

答：取决于下行接口板的类型，和入接口的配置：

- 如果前面描述的BA开关和PHB开关没有同时打开，则报文不会被修改。
- 如果BA开关和PHB开关都打开了，且入接口配置了trust upstream命令和trust 802.1p命令，则MPLS报文修改802.1p和EXP，非MPLS报文只改802.1p。
- 如果BA开关和PHB开关都打开了，但入接口没配置trust 802.1p命令，则存在单板差异，详细请参见《**QoS专题**》的“4 流分类和标记”。

新增的报文头，优先级字段如何设置？

答：存在单板差异，详细请参见《**QoS专题**》的“4 流分类和标记”。

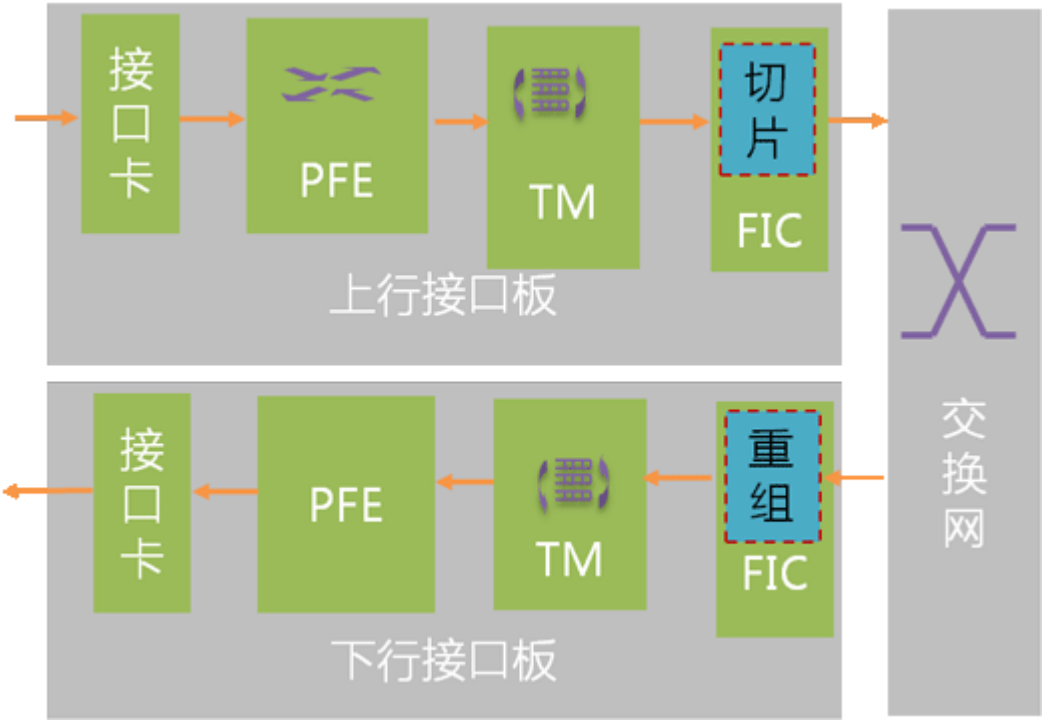
6 转发平面其他处理

关于本章

路由器的转发层面，除了前面几贴介绍的报文收发、解析、转发、交换、封装、QoS处理，还有其他一些处理，本章将介绍其中的几个重要的处理，包括切片与重组、组播复制、广播复制、NAT、包过滤、重定向等等。

6.1 切片与重组

华为高端路由器上通常有多块交换网进行M+N备份。为了实现交换网的负载均衡，交换网采用基于定长信元交换技术：在把报文送往交换网之前，进行切片处理，也就是把报文按一定粒度进行切片，切成固定长度的信元，类似于ATM信元。交换网板基于固定信元长度进行交换。流量到了下行再进行重组。数据包的切片和重组是在接口板的FIC（Fabric Interface Controller，交换网接口控制器）上进行的。



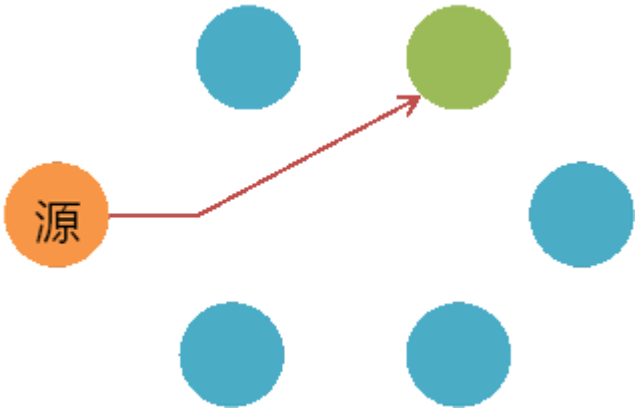
说明

根据送往交换网的数据类型，可将交换网分为基于信元交换和基于包交换。基于包交换不需要切片和重组，但数据包长短不一，有可能导致多个交换网的负载不均衡。 华为高端路由器采用的交换网是基于信元交换，避免了多个交换网的负载不均衡。

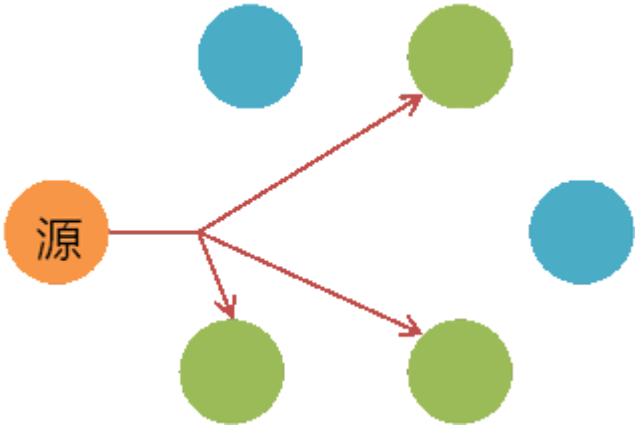
6.2 组播、广播复制

先回忆下单播、组播、广播的通讯模式：

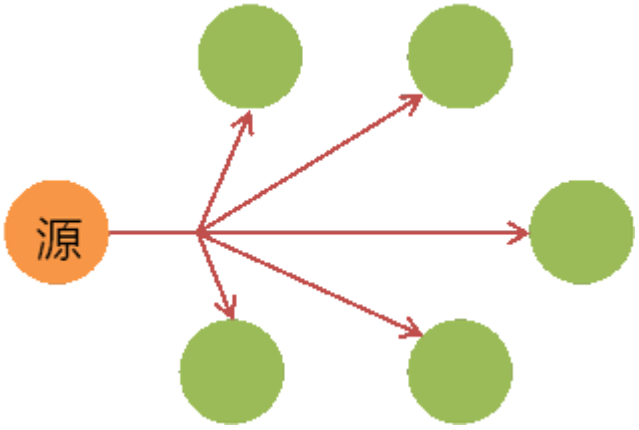
单播：“一对一”的通讯模式，网络对数据只进行转发不进行复制。



组播：“一对一组”的通讯模式，也就是加入了同一个组的主机可以接受到此组内的所有数据，网络只向有需求者复制并转发其所需数据。需要相同数据流的客户端需事先加入相同的“组”，这个组称为“组播组”。

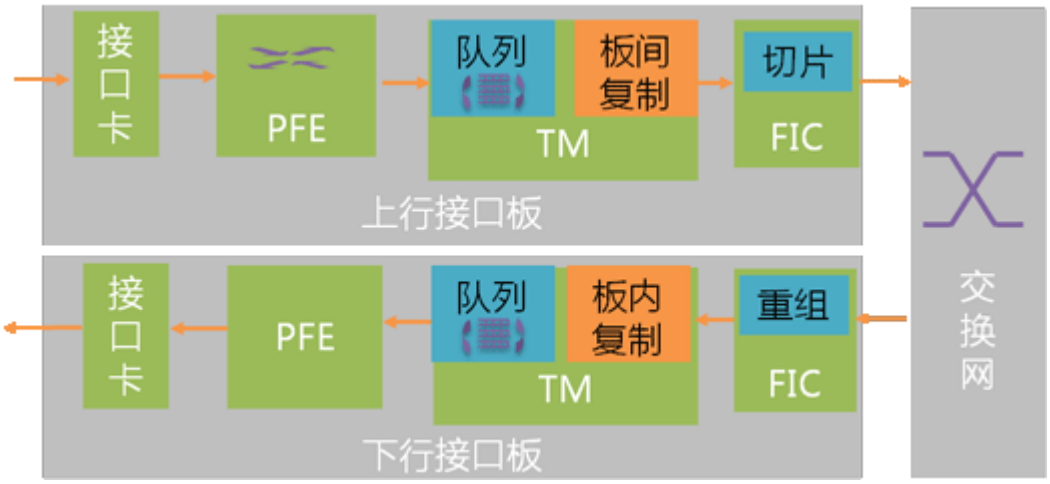


广播：“一对所有”的通讯模式，网络对数据进行无条件复制并转发，所有主机都可以接收到所有信息（不管是否需要）。数据网络中，广播其实是被限制在局域网范围内的，禁止广播数据穿过路由器，防止广播数据影响大面积的主机。



对于组播和广播报文，路由器收到的是一份，但可能需要从多个出接口发送，这些出接口还可能位于多个接口板上。因此对于组播/广播报文，除了查表转发之外，上行需要进行板间复制；下行需要进行同一板内的多个端口的复制。

这些复制操作是在TM上进行的，上行TM进行板间的复制，下行TM进行板内复制（复制到不同端口）。



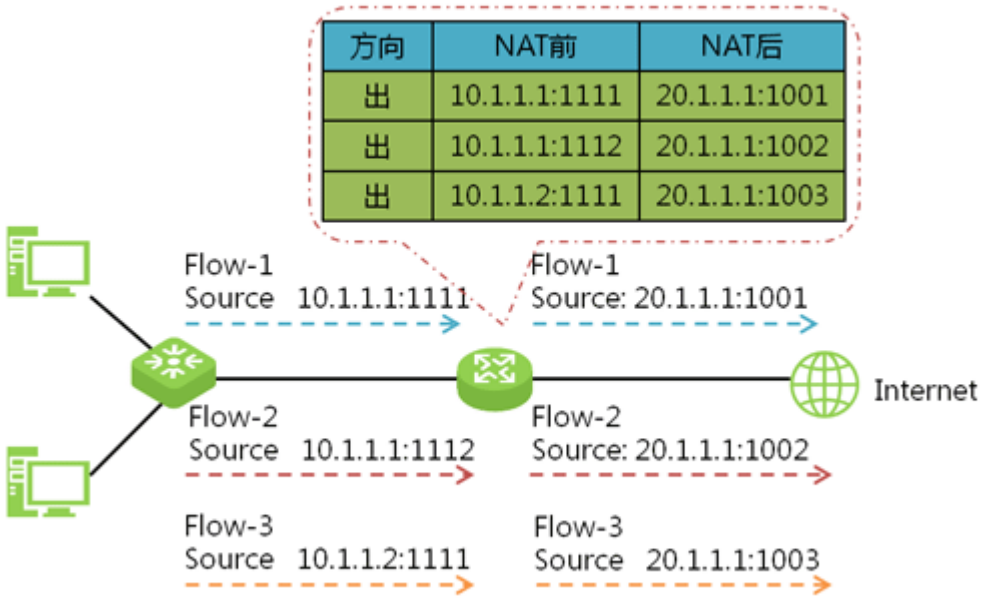
本章只是简单介绍执行组播和广播复制的硬件部件。关于组播和广播的转发流程，请参见**10 IP组播转发流程**。

6.3 NAT

NAT 基本原理

NAT（Network Address Translation，网络地址转换）技术是将IP 数据包头中的IP 地址转换为另一个IP 地址的过程。借助于NAT，局域网可以使用私有地址实现内部互通，如果要访问外部网络，则可以通过NAT设备进行地址转换，将发往公网的数据包的源地址（私有地址）转换成公有IP地址），回程流量再通过NAT设备进行反向转换，这样，一个局域网使用少量公有IP地址（甚至是1个）即可实现局域网内所有主机与Internet的通信需求。

NAT有多种实现方式，华为高端路由器上实现的是NAPT（Network Address Port Translation，网络地址端口转换），可同时映射IP地址和端口号。来自不同内部地址的数据报文的源地址可以映射到同一外部地址，但它们的端口号被转换为该地址的不同端口号。这样，可以更加充分地利用IP地址资源，实现更多内部网络主机对外部网络的同时访问。



如上图所示，三个带有内部地址的数据报文到达NAT设备，其中Flow-1和Flow-2来自同一个内部地址但有不同的源端口号，Flow-1和Flow-3来自不同的内部地址但具有相同的源端口号。通过NAPT映射，3个数据流的源IP地址都被转换到同一个外部地址，但每个数据报都被赋予了不同的源端口号，因而仍保留了报文之间的区别。当回程流量到达时，NAT设备仍能够根据回程流量的目的IP地址和目的端口号来区别该流量应转发到的内部的哪台主机。

NAPT是基于地址池的，事先定义一个NAT地址池，地址池中包含多个公网IP地址。当有用户需要访问外网的时候，数据包到达NAT设备之后，NAT设备从NAT地址池中挑一个IP地址，然后将其映射到这个私有IP，随后进行地址转换动作。一段时间后，如果用户不再有流量发出来，则自动将占用的公有IP放回池中给其他人使用。

关于 NAT 地址池的路由发布

用户访问外网时，回程（从外网发往内网）的数据流的目的IP是地址池的公有IP地址，外网中的设备转发这些回程流量时，需要根据这些公有IP地址进行查转发表转发，也就是说，外网设备需要有到达这些公有IP地址的路由，这意味着，NAT设备需要将NAT地址池的路由发布到外网。然而，地址池的这些公有IP地址是由NAT设备动态分配的，如何发布到这些公有IP地址的路由呢？

答案是，在华为高端路由器上，当创建完NAT公网地址池后，便会生成NAT公网地址池UNR路由。假设NAT公网地址池的IP地址范围是10.0.0.0/22网段，那么这条UNR路由如下图所示：

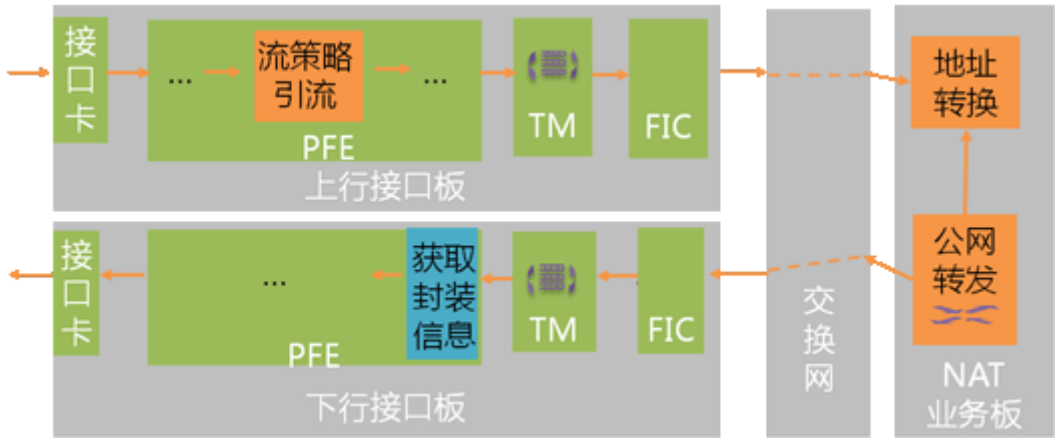
```
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

      100.0.0.0/22  Unr     64   65535       D   127.0.0.1       InLoopBack0
```

将NAT地址池的路由发布到公网的方法很简单，只要在动态路由协议里引入这条UNR路由即可（用**import-route unr**命令）。

NAT 实现流程

在华为高端路由器上，实现NAT转换的关键部件是NAT业务板，业务流程是：上行接口板把流量引入到业务板，业务板负责NAT处理，完成后再交给下行接口板发出。详细实现流程如下：

- **出方向（从私网到公网）转发流程**



同其他业务流的处理流程大部分相同，不同的是：

a. 在上行接口板的包转发引擎PFE（NP芯片）处理时，需要通过流策略（复杂流分类）引流，也就是经过流策略处理后，得到了报文的目的单板为NAT业务板。

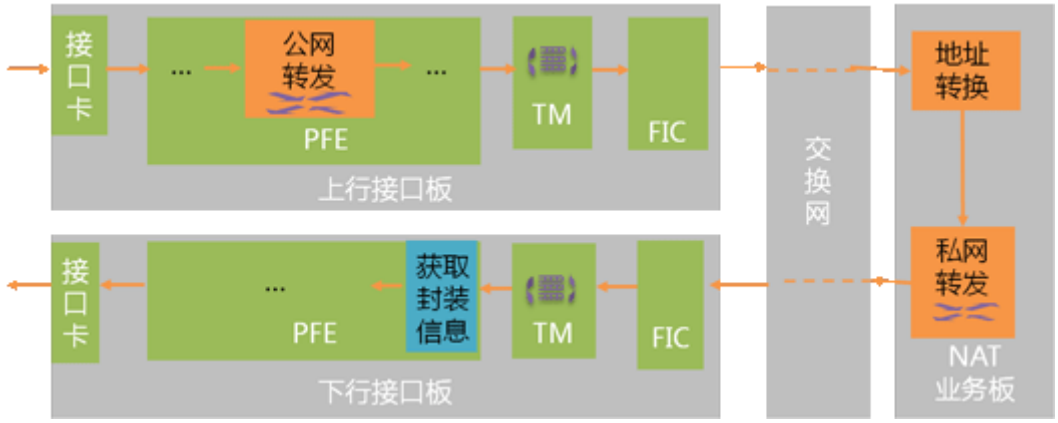
b. 交换网根据目的单板信息将报文发给NAT业务板。

c. NAT业务板首先进行地址转换，将数据包的源IP（私网IP）转换成公网IP，并做端口转换。

d. 之后，NAT业务板查找公网转发表进行转发（NAT业务板上也有转发表，跟接口板上是相同的），得到目的接口板和出接口信息。

e. 交换网将报文交换到下行接口板，之后的处理和同其他业务流，不再赘述。

- **入方向（从公网到私网）转发流程**



同其他业务流的处理流程大部分相同，不同的是：

a. 在上行接口板的包转发引擎PFE（NP芯片）查公网路由表转发时，匹配的是NAT地址池的UNR路由，其目的单板是NAT业务（如果NAT业务板为集中板，则UNR路由信息中带目的单板和目的接口信息； 如果是随板则直接上送CPU处理后得到目的单板和目的接口信息）。

b. 交换网根据目的单板信息将报文发给NAT业务板。
c. NAT业务板首先进行地址转换，将数据包的目的IP（公网IP）转换成私网IP，并做端口转换。
d. NAT业务板查找私网转发表进行转发，得到目的接口板和出接口信息。
e. 交换网将报文交换到下行接口板，之后的处理和同其他业务流，不再赘述。

6.4 包过滤

路由器上的包过滤是基于ACL的，先获取数据包的包头信息，如以太帧头信息、MPLS头信息、IP头信息、TCP/IP端口号等等，然后和设定的ACL规则进行匹配，根据匹配的结果决定对数据包进行转发或者丢弃。

那么，如何将ACL规则和数据包的处理行为（转发或者丢弃）关联起来呢？答案就是使用流策略（即复杂流分类）。

所以，包过滤在转发层面是由上行或下行接口板上执行的，位置如下图。

在本文的4 QoS基础中介绍过，流策略在配置上采用“模板化”方式，分为三部分，流分类、流动作和流策略。那么应用到包过滤时，这三部分定义如下：

- 流分类（Classifier）：设定ACL匹配规则，并应用到if-match语句。
- 流动作（Behavior）：定义针对该类流量实施的流动作为转发（permit）或者丢弃（deny）。
- 流策略（Policy）：将流分类Classifier和流动作Behavior关联，设置完毕后应用到流量的入接口/出接口。

说明

关于ACL的详细介绍请参见《ACL专题》。

6.5 策略路由（重定向）

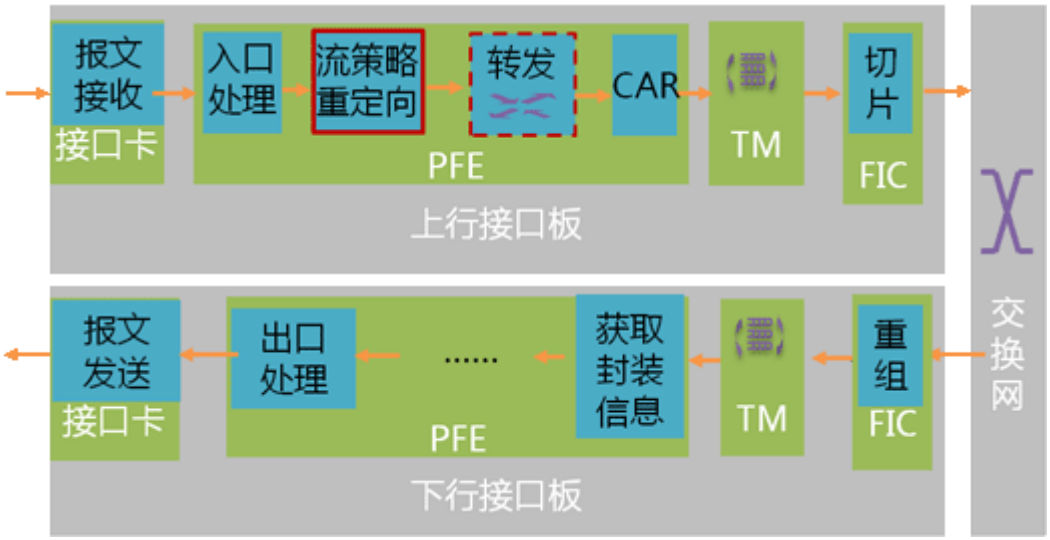
策略路由（Policy-based Routing），也称为路由重定向（Redirect），顾名思义是指基于策略的路由机制。通常，路由器根据报文目的地址查找路由表进行报文转发。策略路由是一种依据用户制定的策略进行路由选择的机制，可应用于安全、负载分担等目的。策略路由不仅能够根据报文的目的地址，而且能够根据报文的其他特征（如源地址、报文长度等）来选择转发路径。

策略路由与路由策略（Routing Policy）不同，请不要混淆。策略路由的操作对象是数据包，在路由表已经产生的情况下，不按照先行路由表进行转发，而是根据需要，依照某种策略改变其转发路径的方法。而路由策略的操作对象是路由信息，在正常的路由协议之上，根据某种规则、通过改变某些参数或者设置某种控制方式来改变路由产生、发布、选择的结果——路由表。

策略路由有强策略和弱策略之分（流行为视图下，**redirect**命令行如果指定了出接口就是强策略，如果未指定出接口就是弱策略）：

- 强策略路由是不查转发表，直接通过指定的下一跳与出接口进行转发。如果出接口UP（当然，如果若是以太接口时，还要求有ARP表项），则转发报文，否则丢弃报文。如果下一跳或出接口不存在，则报文在流策略环节就被丢弃了。
- 弱策略路由是指根据指定的IP地址查转发表，查到则转发；如果查不到，再根据报文的目的IP地址查转发表。注意这里的“查到”的含义，匹配上默认路由也算“查到”。

由上述可知，在转发层面上，策略路由是在复杂流分类（流策略）环节上执行的，而且弱策略还涉及到查表转发环节。



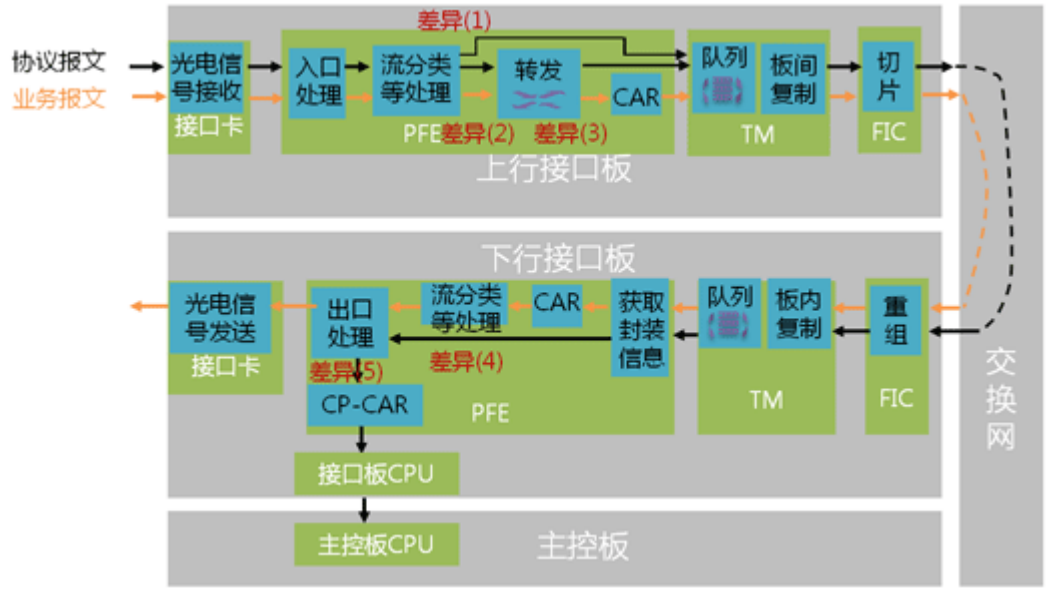
7 协议报文之旅

关于本章

进入路由器的报文，大部分是业务报文，只经过业务板和交换网板处理即可，但还有一小部分特殊的报文，例如路由协议报文、用户上下线报文、异常/错误报文等，业务板需要把这些报文上送到主控板CPU进行处理。本章介绍这些报文的转发流程的不同之处。

7.1 收方向的协议报文之旅

路由器收到的需上送CPU处理的报文，其处理流程和业务报文是几乎相同的，如下图所示。



不同点在于：

- 差异（1）：报文解析识别出的协议报文不再查表转发
- 差异（2）：下一跳地址为127.0.0.1的报文需上送CPU
- 差异（3）：协议报文不再做CAR限速
- 差异（4）：协议报文不再流分类等处理
- 差异（5）：协议报文上送CPU时要经过CP-CAR处理

差异（1）：报文解析识别出的协议报文不再查表转发

在转发引擎PFE（NP或ASIC芯片）上做报文解析时，如果从二层帧头的协议字段就可以直接判断出是需要上送本机CPU处理的协议报文（如ARP、RARP、IS-IS、LLDP、LACP、PPP控制报文等等），或者目的地址为特定的保留组播IP地址（标准中定义，组播地址224.0.0.1～224.0.0.255供路由协议使用）的协议报文，上行不需要查表转发。

前面章节介绍过，报文通过上行查表转发后，获得报文的目的接口板和出接口信息，以便交换网板能根据目的单板信息把数据交换到对应下行单板，而下行单板根据出接口信息发送报文。那么这些通过报文解析就能识别出来的协议报文，上行不查表转发，其目的接口板和出接口信息填什么呢？答案是填成与入接口板相同的板号，出接口为CPU。

差异（2）：下一跳地址为 127.0.0.1 的报文需上送 CPU

在路由器上，如下几类路由的下一跳地址为127.0.0.1，匹配这类路由的报文需要上送CPU处理：

- **接口主机路由和直连子网广播**

每一个配置有IP地址且链路层和IP层协议状态为UP的直连接口，都会生成三条路由，例如，

```
路由表：
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

        10.2.5.0/24  Direct  0    0           D   10.2.5.5        GigabitEthernet1/0/0
        10.2.5.5/32  Direct  0    0           D   127.0.0.1       GigabitEthernet1/0/0
      10.2.5.255/32  Direct  0    0           D   127.0.0.1       GigabitEthernet1/0/0
对应的转发表：
Destination/Mask  Nexthop      Flag  TimeStamp   Interface            TunnelID

10.2.5.0/24       10.2.5.5     U     t[5847]     GigabitEthernet1/0/0     0x0
10.2.5.5/32       127.0.0.1    HU    t[5847]     InLoop0                  0x0
10.2.5.255/32     127.0.0.1    HU    t[5847]     InLoop0                  0x0
```

其中：

第1条是网段路由，表示该路由器的GE1/0/0与10.2.5.0网段直连，到该网段的报文需要从GE1/0/0发出去，因此出接口为GE1/0/0；

第2条是主机路由，目的地址10.2.5.5是GE1/0/0的IP地址。当路由器收目的地址是自己某个接口的IP地址时，认为这是发送给自己的报文，必须上送自己的应用协议栈。在华为路由器上，这类路由的转发表出接口都是环回接口InLoopBack0，表示该类报文要上送CPU处理。

第3条是10.2.5.0/24子网段的广播地址，IP协议规定所有的本网段的三层接口都需要接收该地址的报文，所以广播路由的出接口也为环回接口InLoopBack0，路由器收到该报文，认为是发送给自己的消息，就上送处理。

另外，路由器上有一类特殊的逻辑接口LoopBack接口和VT（Virtual-Template）接口，通常配置为32位掩码的IP地址，对应会生成一条主机路由，如下：

```
路由表：
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

        10.0.0.5/32  Direct  0    0           D   127.0.0.1       LoopBack1
        10.2.3.9/32  Direct  0    0           D   127.0.0.1       Virtual-Template5
对应的转发表：
Destination/Mask   Nexthop       Flag  TimeStamp     Interface     TunnelID

10.0.0.5/32        127.0.0.1     HU    t[142]        InLoop0        0x0
10.2.3.9/32        127.0.0.1     HU    t[28733]      InLoop0        0x0
```

这类接口的主机路由，其下一跳也是127.0.0.1，转发表中出接口也是InLoopBack0，表示收到这类报文时需上送CPU处理。

- **全网广播**

  IP地址为255.255.255.255/32为全网广播，通常在配置主机的启动信息时使用，此时，主机可能还不知道它所在网网络的网络掩码，甚至连它的IP地址也不知道，例如到主机从DHCP或BOOTP服务器获取IP地址时，发送IP地址为255.255.255.255/32的DHCP Request消息报文。这种报文仅允许出现在本地网络中，所以路由器不转发这种报文，收到时上送CPU处理。

```
路由表：
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface
255.255.255.255/32  Direct  0    0           D   127.0.0.1       InLoopBack0
对应的转发表：
Destination/Mask   Nexthop       Flag  TimeStamp     Interface        TunnelID
255.255.255.255/32 127.0.0.1       HU    t[128]        InLoop0        0x0
```

- **UNR路由**

  当路由器作为BRAS（Broadband Remote Access Server，宽带远程接入服务器）/BNG（Broadband Network Gateway，宽带网络业务网关）时，用户通过PPPoE拨号接入BRAS/BNG，起初用户并没有IP地址，需要BRAS/BNG（或向RADIUS服务器申请）给用户分配地址。假设分配给用户的地址为10.111.111.1/32，那么BRAS/BNG应该生成10.111.111.1/32的路由，收到网络返回给

用户的报文时，这个报文应该匹配到这条路由，并上送CPU处理，以便计费等处理，因此这条的下一跳也应为127.0.0.1。

```
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

   10.111.111.1/32  Unr     61   0           D   127.0.0.1       InLoopBack0
```

这种路由比较特殊，不是路由协议学习到的，也不是直连的，也不是静态配置的，而是叫UNR（User Network Route）路由。

另外，为了让网络返回报文给用户，BRAS/BNG需要将这条路由发布出去，使得网络其他设备有到用户的路由，然而，BRAS/BNG可能接入大量用户，如果将每条路由都发布，那需要发布大量路由，不可取。为了避免发布大量路由，BRAS/BNG会根据地址池生成一条UNR路由，其下一跳也是127.0.0.1，出接口为Null0。

```
Destination/Mask    Proto   Pre  Cost      Flags NextHop         Interface

 10.111.111.0/24    Unr     61   0           D   127.0.0.1       NULL0
```

差异（3）：协议报文不再做 CAR 限速

上送CPU的报文在不做上、下行的CAR限速，这是为了避免当流量突发时协议报文因CAR限速被丢弃。

差异（4）：协议报文不再流分类等处理

由于下行就要送往CPU，流分类和标记已经没有作用，跟流分类相关的流策略等功能也是无用的，所有下行不再做跟流分类相关的操作。

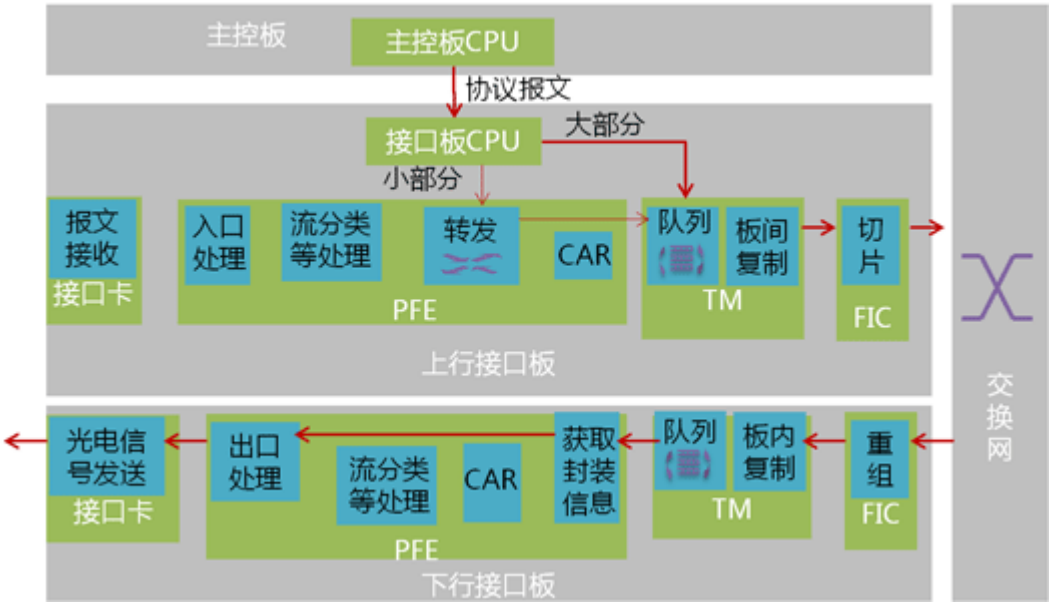
差异（5）：协议报文上送 CPU 时要经过 CP-CAR 处理

为了防止大流量报文上送导致CPU过于繁忙，被确认为需要继续上送到CPU的报文，将进行CP-CAR的过滤，之后再上送。CP-CAR全称是Control Plane CAR（Committed Access Rate），是用来限制转发引擎向接口板CPU发送报文的速度，具体做法跟基于流的CAR机制类似（详细请参见本文第4章“QoS基础”）。CP-CAR根据协议的类型，报文的VLAN或者所属的用户，分成很多个管道，每一个管道使用令牌桶进行速率限制，当上送的流量宽值超过设置的速率时，上送的报文将被会随机丢弃，避免CPU过载。

7.2 发送方向的协议报文之旅

从CPU发送的协议报文，在上行不经过接口卡的处理，直接送到PFE。由于CPU发送的报文，大部分都带有目的单板和出接口信息，CPU已经告诉转发平面报文要发往哪块单板的哪个出接口，因此无需在转发平面查转发表，报文直接入队列。但有一小部分特殊的报文需要再查转发表，例如指定源接口的ping报文（命令行：**ping** *destination-ip* **-si** *interface-name*），由于只指定了源接口，CPU并不清楚源接口对应的IP地址是多少，因此需要由转发平面查表获得。这些查表转发后的报文也不经过CAR处理，直接送到TM。

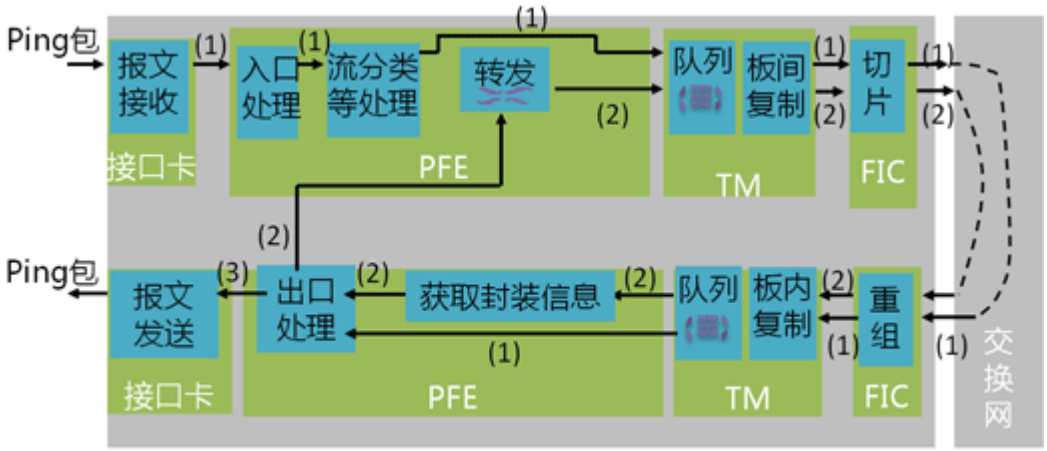
之后，协议报文的后续处理与业务报文相同，最后跟业务报文一起从出接口发送出去。只是在下行不做CAR和流分类等处理，理由和入方向协议报文是相同的，不再赘述。



7.3 快回报文，不上送 CPU

网络管理者或用户通常发起ping包（ICMP请求报文）到网关或者是网络侧地址，来确认链路的连通性。ICMP请求报文通常是上送到CPU进行解析，接着CPU需要构造ICMP回应报文给源端，而且上送前还要做CP-CAR检测，如果有大量的ICMP请求报文，容易造成CPU过载，同时导致Ping流程时延加大。为此，华为高端路由器上实现了Ping快回机制，接收到ICMP请求报文后不上送CPU处理，直接由接口板的包转发引擎PFE构造ICMP响应报文给客户端，大大缩短Ping的时延。

Ping快回的转发流程如下图所示。



如上图所示，Ping包在下行出接口处理时，并不是上送CPU，而是环回到上行进行查表转发。在环回时，PFE把报文的源IP和目的IP对换。

不过，当ping快回的报文大于MTU时，ping快回的报文被分片，同时分片后的报文被当做普通的ping报文上送CPU。

当前大多数版本是默认打开ping快回功能，在实际定位问题时如果用ping模拟业务报文，需要把ping快回功能关闭（slot视图下/系统视图下的命令行**undo icmp-reply fast**）。

与Ping快回类似的还有ARP快回，Web快回，都是不上送CPU，由接口板直接生成回应报文。由于实现原理差不多，此不赘述。

8 IP 单播转发流程

关于本章

本文第1~6章介绍了一个报文在转发层面的处理流程，该流程中最重要的处理就是转发流程。不同业务有不同的转发流程，本文后面几章将分别介绍各类业务的转发流程。

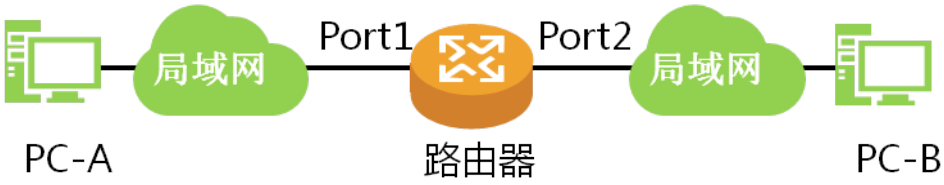
本章介绍IP单播的转发流程，包括IPv4单播和IPv6单播。

8.1 IPv4 单播转发流程

端到端的 IPv4 单播转发过程

以大家熟悉的以太帧为例，先来回顾下IP单播端到端的转发流程。

下图是个最简单的IP转发场景，某局域网的主机A发送报文给另一局域网的主机B，中间经过一台路由器，那么这台路由器就是PC-A的网关。



由主机PC-A向主机PC-B发送IP报文，那么该报文的目的IP地址就是PC-B的IP地址，源IP地址就是主机PC-A的IP地址，目标MAC地址就是其网关路由器Port1的MAC地址，源MAC地址就是PC-A的MAC地址。

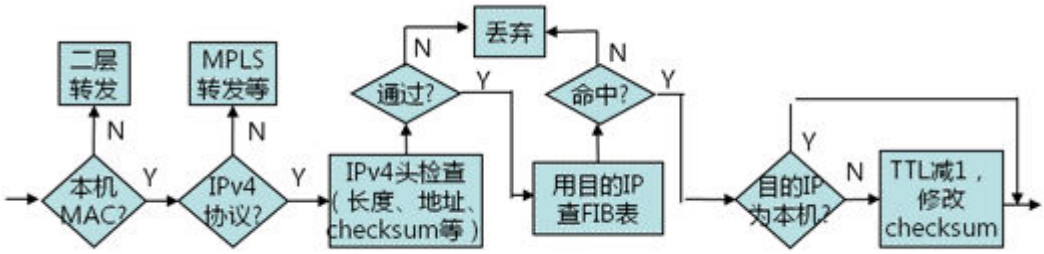
路由器转发过程：

1. 路由器收到这个报文，发现其目的MAC为本机Port1端口的，表明需要本机来进行进一步解析（如果目的MAC不是本机，表明直接进行二层转发，不需要再解析帧的其他内容了）；
2. 路由器进一步解析报文，得知帧所承载的协议类型为IPv4（协议类型值=0x800），即需要进行IPv4转发；
3. 查IP转发表（FIB表），得知该报文并不是发给自己的，而是需要送往出端口Port2，因此，路由器不再继续分析IP头后面的内容；
4. 路由器将目的MAC更换成PC-B的MAC，将源MAC更换成出接口Port2的MAC，并将报文从Port2发送出去。

路由器的 IPv4 转发全流程

IPv4转发全流程如下图所示。需要关注的地方在于“查表转发”和“获取封装信息”两个环节（其他环节在本文的第1~7章中已描述，不再赘述）。

查表转发详细流程：



1. 判断报文的目的MAC是否等于本机MAC，如果不是，则做L2转发；是则继续下一步骤。
2. 判断报文的协议类型是否为IPv4（例如以太帧，eth_type = 0x800），如果不是，则进入其他转发流程；是则继续下一步骤。
3. 检查报文的长度、IP地址、Checksum字段是否正确，如果不正确，则丢弃报文，否则继续下一步骤。
4. 判断目的IP地址是否为单播地址，如果不是单播则其他转发处理，是则进入继续下一步骤。
5. 用目的IP地址查FIB表得到的下一跳IP、出接口等信息（如果是公网的报文，查公网FIB表，如果是VPN报文，则查对应VPN的FIB表）。

   FIB:

   ```
   Destination/Mask  Nexthop     Flag  TimeStamp  Interface             TunnelID
   10.2.5.0/24       10.2.5.5    U     t[5847]    GigabitEthernet1/0/0      0x0
   10.2.5.5/32       127.0.0.1   HU    t[5847]    InLoop0                   0x0
   ```

   - 如果是负载分担，会查到多份这样的信息，于是根据负载分担哈希算法选取其中的一份。关于负载分担的详细介绍请参见《负载分担专题》。
   - 如果是FRR（Fast Reroute）状态，则会根据出接口状态做主备路由选择，如果出接口正常工作，则会选择主路由；否则选择FRR备份路由。
   - 如果出接口为Trunk接口，会再根据Trunk负载分担哈希算法，选择Trunk成员口中的其中一个作为最终的出接口。
6. 如果使能了URPF检查，则用源IP地址查FIB表，如果命中，对于松散的URPF检查只要出接口为真实的外部接口则检查通过（对于出接口为CPU、NULL接口、TE接口、IPv4 Tunnel接口则松散的URPF检查不通过）；对应严格的URPF检查，用报文的入端口信息与源IP地址查FIB表得到的出口信息进行比较，相等则通过，不等则丢弃（对于入接口为VLAN子接口，出接口为入端口信息，同时出接口VLANID也要等于入口的VLANID，则检查才会通过）。

   说明

   URPF（Unicast Reverse Path Forwarding），是一种单播逆向路由查找技术，用来预防伪造源地址攻击的手段。之所以称为逆向，是针对正常的路由查找而言的。一般情况下路由器接收到报文，获取报文的目的地址，针对目的地址查找路由。如果找到了进行正常的转发，否则丢弃该报文。

   URPF的实现原理：通过获取报文的源地址和入接口，以源地址为目的地址，在转发表中查找源地址对应的接口是否与入接口匹配。如果不匹配认为源地址是伪装的，丢弃该报文。通过这种方式，能有效地防范网络中通过修改源地址而进行的恶意攻击行为的发生。

   然而，有的场景（例如负载分担）同一目的地址在路由表中存在有多条路由表项，同一目的IP地址的报文发送的接口不唯一，即非对称路由。此时若应用URPF时，报文会异常丢弃。因此，URPF出现了严格模式和松散模式。严格模式要求接口匹配；而松散模式，不检查接口是否匹配，只要路由表项中存在针对源地址的路由，数据报文就通过URPF检查。

7. 如果目的IP为非本机IP，则报文头的TTL减1，并重新计算并修改Checksum值，继续执行后续的CAR等公共处理。如果目的IP为本机（查表发现下一跳为127.0.0.1），则直接送入上行TM部件。

之后的处理中，交换网根据出接口信息（出接口信息包含了目的单板和目的出接口）信息将报文发送到正确的下行单板。

获取封装信息：

到了下行，下行包转发引擎PFE用下一跳IP或目的IP和VLANID查ARP表项，获取目的MAC信息，根据出接口信息查出接口表项获取端口MAC。因为，路由器需要将报文的目的MAC更换成下一跳设备的MAC，将源MAC更换成自己出接口的MAC。

ARP表：

```
IP ADDRESS      MAC ADDRESS     EXPIRE(M) TYPE   INTERFACE  VPN-INSTANCE
------------------------------------------------------------------------------100.2.150.51
0018-8201-4daa            I -    GE0/0/0
100.2.200.7     0013-d326-a32f  1         D-0    GE0/0/0
192.1.23.1      00e0-fcd5-c877            I -    GE1/0/2
37.1.3.1        00e0-fcd5-c863            I -    GE1/0/3
```

如果查ARP表项没有命中，则启动ARP学习功能，过程如下：

1. 发送ARP请求报文，此报文的目的MAC为广播地址，目的地址为下一跳IP，源IP为自己的IP。
2. 由于是MAC广播报文，在局域网内的设备或主机都能收到，因此下一跳设备也能收到。于是下一跳设备解析报文发现目的IP为自己，便发送ARP回应报文，里面携带了自己的MAC地址。
3. 路由器收到回应报文，得到下一跳的MAC地址，添加到ARP表项中。 通过ARP学习后，重新查ARP表项获取下一跳MAC信息，便可继续后续的处理。

出口检查和封装：

对于目的IP为本机的，在出口处理模块处上送到接口板CPU，再上送给主控板CPU。

对于目的地址非本机的，则在出接口处理模块时，根据需要进行MTU检查。如果报文长度不超过MTU，则将报文发送给接口卡。接口卡用待发送的数据帧内容计算帧检验序列FCS，然后对数据帧加封装帧间隙、前导码、帧开始界定符和FCS，并将数据帧转换成光/电信号，再发送到出接口线路上。

如果报文长度超过MTU，则判断报文头DF位，如果DF位置0，则分片后再发送给接口卡；若DF置1，表示报文源端不允许该报文分片，所以将报文做CP-CAR检查后上送接口板CPU，再上送主控板CPU，以便回应ICMP Too-Big消息给源端。

8.2 IPv6 单播转发流程

IPv6转发流程与IPv4基本相同，不同点在于：

- 所查的表项不同，IPv4查FIBv4表，而IPv6查的是FIBv6表；IPv4查ARP表，而IPv6则是查邻居表。
- IPv6 MTU检查时，超过接口IPv6 MTU时不进行分片，而是上送CPU并返回ICMPv6 Too-big消息给源端（这符合IPv6协议标准）。

IPv6邻居表：

```
[Router] display ipv6 neighbors
------------------------------------------------------------
```

```
IPv6 Address : 2012::2
Link-layer   : 00e0-fcc2-13b6                          State : STALE
Interface    : GE0/0/0                                 Age   : 0
VLAN         : -                                       CEVLAN: -
VPN name     :                                     Is Router: TRUE
Secure FLAG  : UN-SECURE

IPv6 Address : FE80::2E0:FCFF:FEC2:13B6
Link-layer   : 00e0-fcc2-13b6                          State : STALE
Interface    : GE0/0/0                                 Age   : 0
VLAN         : -                                       CEVLAN: -
VPN name     :                                     Is Router: TRUE
Secure FLAG  : UN-SECURE
------------------------------------------------------------------
```

9 二层桥接转发流程

关于本章

本章介绍二层桥接的转发流程，包括二层单播、组播和广播。

9.1 二层桥接转发基础

什么是二层桥接转发？

二层所指的是数据链路层。二层桥接转发，是指数据帧在数据链路层是怎样被转发的。

数据链路层有很多不同的网络类型，Token ring（令牌环网）、Ethernet、FDDI（光纤网络）等等，其中用得最广泛的就是以太网Ethernet，本章要介绍的是以太网的转发原理。

以太网是根据二层帧头信息，切确的说是根据MAC地址，进行转发的。MAC地址对于二层转发而言，是相当的重要。

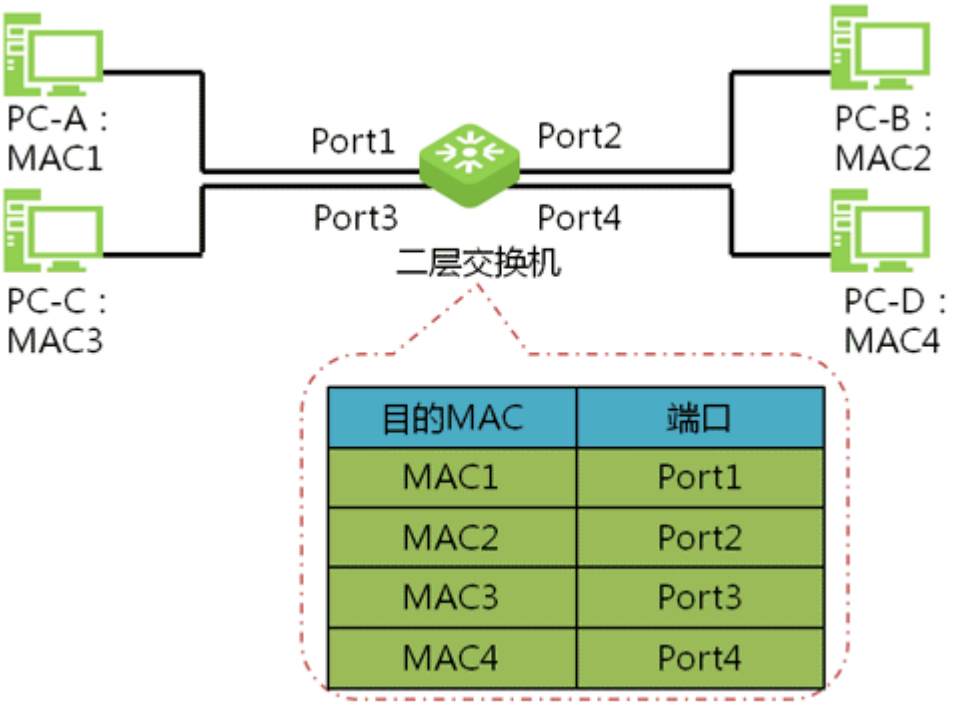
MAC 地址简介

MAC地址是48bit二进制的地址，如：00-e0-fc-00-00-06，MAC地址是全球唯一的，由IEEE统一管理和分配。MAC地址可以分为单播地址、多播地址和广播地址：

- 单播地址：第一字节最低位为0，如：00-e0-fc-00-00-06。
- 多播地址：第一字节最低位为1，如：01-e0-fc-00-00-06。
- 广播地址：48位全1，如：ff-ff-ff-ff-ff-ff。

二层桥接单播转发过程

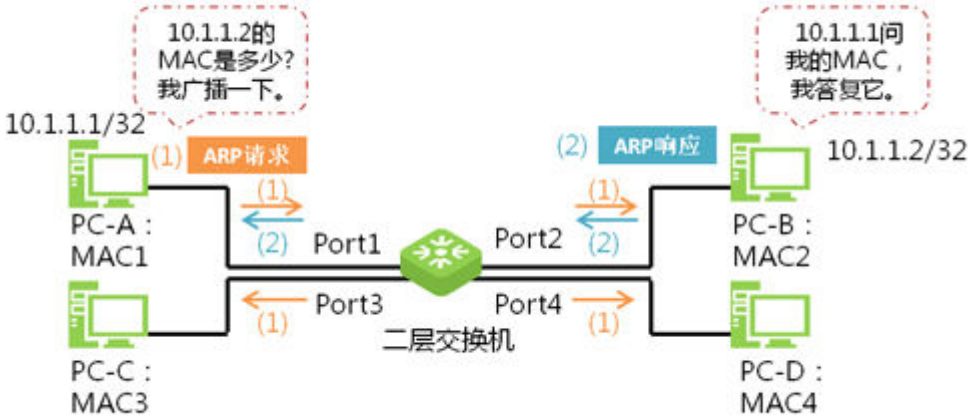
下图是个最简单的二层桥接转发场景，某局域网的主机PC-A发送报文给主机PC-B，中间经过一台二层交换机的交换。由主机PC-A向主机PC-B发送以太帧，那么该以太帧的目的MAC地址就是MAC2，源MAC就是MAC1。



二层交换机转发过程：交换机收到这个以太帧，解析发现其目的MAC为MAC2，查MAC表，发现对应的出端口为Port2，于是将这个以太帧从Port2发送出去。这样，PC-B就收到了这个以太帧。

二层桥接广播帧转发过程

如果上图中，PC-A初始的时候并不知道PC-B的MAC，怎么办呢？这时，PC-A会发ARP请求，这个ARP请求的目的MAC为广播地址，源MAC为自己的MAC，交换机收到这个广播帧，会发给除了Port1以外的所有端口。这样，局域网内所有主机都能收到这个广播帧。



PC-B发现请求的是咨询自己MAC，于是返回ARP响应报文，交换机对ARP响应报文进行单播转发给PC-A。

MAC 地址学习机制

上图中，二层交换机上的MAC地址表是MAC地址和端口的映射表，那么这个映射表是怎么得到的呢？上一章“**IP单播转发流程**”中介绍过，路由器通过ARP机制学习到IP和MAC的映射关系，那么，MAC表是怎么得到的呢？

下面举个例子说明MAC地址的学习过程：

仍然以上图为例，当二层交换机刚上电启动时，其MAC地址表是空的。假设此时PC-A要发数据给PC-B，当交换机收到PC-A发给PC-B的数据帧时：

1. 交换机首先是读取该数据帧的源MAC地址，并且映射该地址和收到数据帧的端口，加入到MAC地址表。
2. 接着，交换机读取数据帧中的目的MAC地址，并且在MAC地址表中查询该MAC地址对应的端口，因为此时交换机的MAC表还没有PC-B的MAC，所以交换机会向所有的端口“洪泛”该数据帧，这样PC-B就能收到这个数据帧了。

上面是交换机学习PC-A的MAC地址的过程。按此方法，当PC-B、PC-C、PC-D都向交换机发送了数据帧后，交换机就学到了所有端口所连接的设备的MAC地址。

MAC 地址表的老化机制

假设上图的PC-D被搬走，或者交换机连接PC-D和链接PC-C的端口互换了如果不及时更正过来，交换机可能会把数据帧发错地方怎么办呢？交换机的处理方式是：对每条MAC地址都设置一个计时器，如果一台主机在指定的老化时间之内没有发送数据帧到

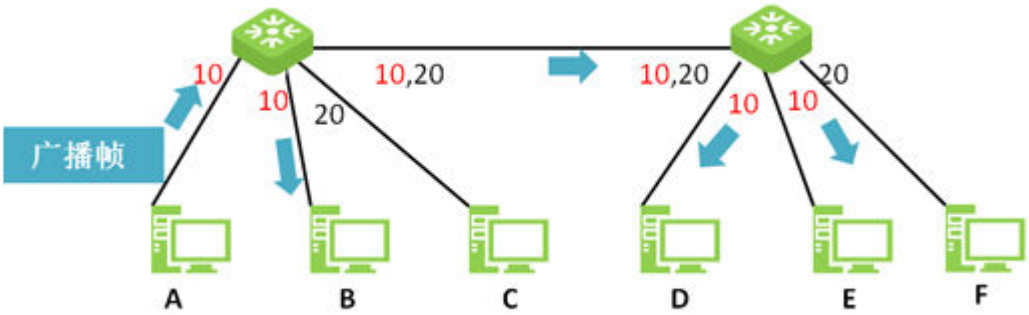
交换机，交换机就会认为它超时，把它的MAC地址从MAC地址表中清除，下次它要发数据帧，交换机再重新学习MAC。

VLAN 基础

上面过程中可知，交换机对广播帧、未知单播都进行广播泛洪，这样，局域网内就存在很多广播帧，消耗很多链路资源，还会占用主机处理广播帧的时间。实际上，广播帧是经常出现的，比如上面的未知单播的泛洪、还有ARP请求；此外，还有DHCP、RIP协议也会频繁发送广播帧。

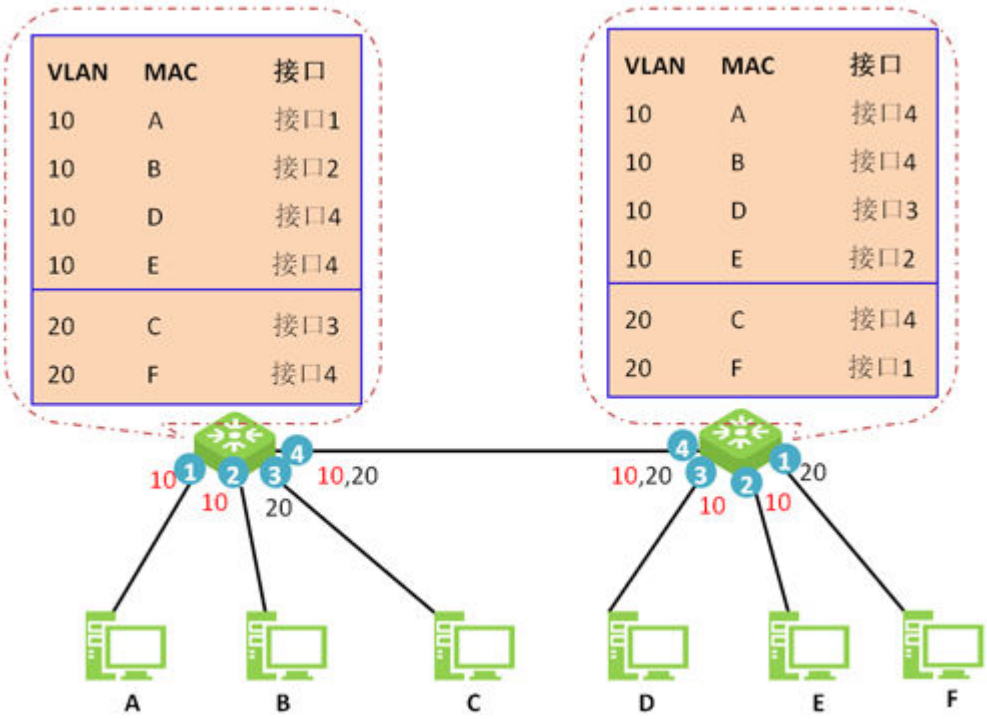
为了减少广播帧，VLAN（Virtual Area Network，虚拟局域网）诞生了。VLAN将一个局域网在逻辑上划分成多个广播域，所有同一个VLAN的主机可以互相通信。那么VLAN是如何实现广播隔离的呢？

首先，交换机每个端口都指定了所属VLAN。交换机和交换机之间的接口可以属于多个VLAN。如下图，交换机收到VLAN10的PC-A发的广播帧，只转发给含有VLAN10的接口，这样，同一个VLAN的广播帧只有VLAN的成员才能收到，不会传输到其他VLAN中去。

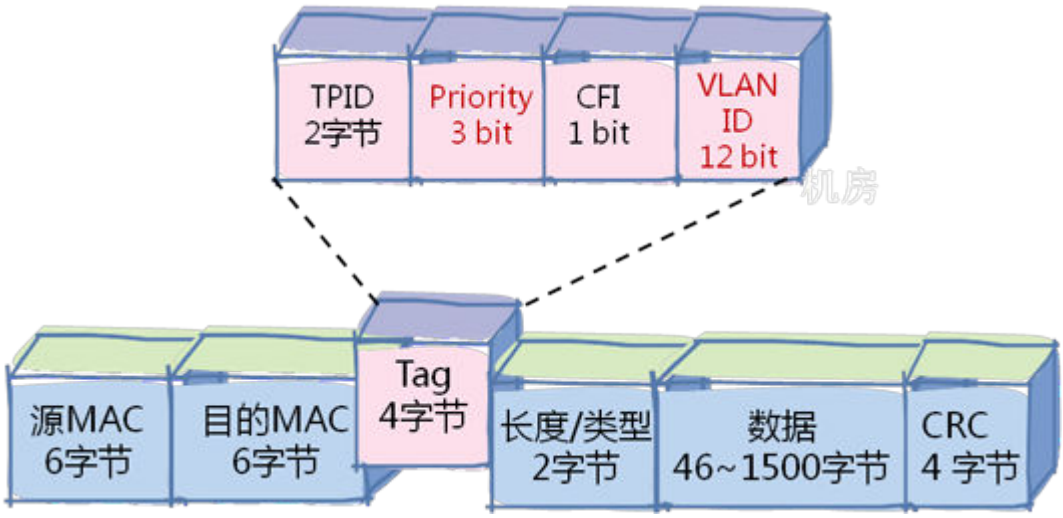


VLAN内如何实现互通？

由于交换机端口配了所属VLAN，交换机的每条转发表项指示了所属VLAN。



交换机收到报文时，根据入接口给报文加上对应VLAN，并根据VLAN和目的MAC转发。打上VLAN的以太帧格式如下：



对端交换机收到报文后，剥除其携带的VLAN，并根据VLAN和目的MAC转发。 注意：VLAN间是无法直接互通的，除非通过路由器中转。

上述例子中，有的交换机端口只允许一个VLAN通过，有的允许多个VLAN通过。它们的处理有什么区别吗？

实际上，VLAN端口可分为三类型：

- Access端口：只属于一个VLAN，用于连接不支持802.1Q封装的设备，如用户计算机。
- Trunk端口：可以属于多个VLAN，允许接收和发送多个VLAN的报文。用于网络设备之间互联。

- Hybrid端口：可以属于多个VLAN，允许接收和发送多个VLAN的报文。可以用于网络设备之间互联，也可以用于连接不支持802.1Q封装的设备。

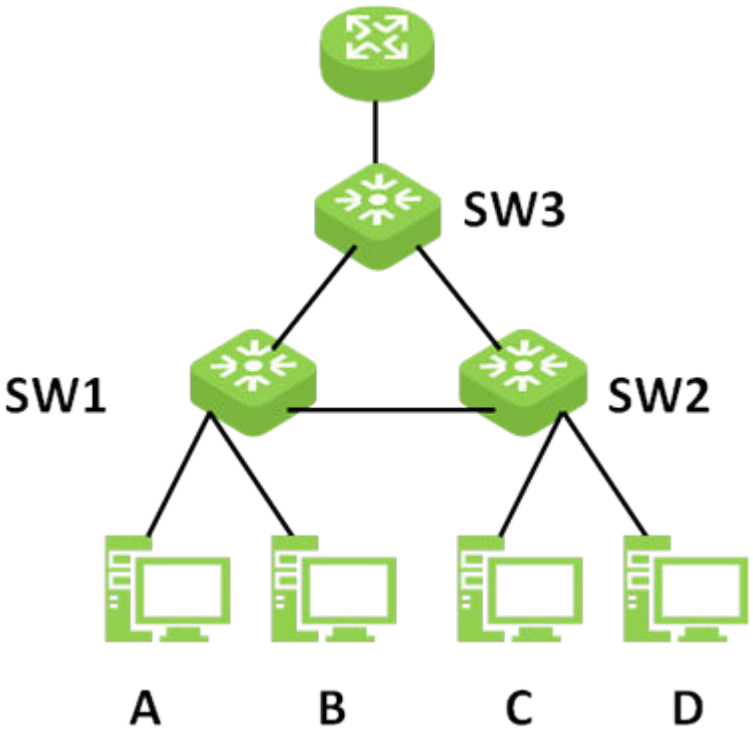
报文处理机制

为了快速高效的处理，交换机内部，报文都是带VLAN转发的，而且，交换机在处理VLAN报文时，在报文入方向和出方向是分别处理的，不同类型的接口处理方式各不相同，而且不同设备处理方式也会不同。在华为高端路由器上的处理方式如下表所示：

端口类型	入方向		出方向
	收到不带VLAN的报文	收到带VLAN的报文	发送报文的方式
Access端口	接收并打上缺省VLAN后转发	报文带的VLAN与端口缺省VLAN相等，则转发，否则丢弃。	剥离VLAN后发送。
Trunk端口	丢弃该报文	报文带的VLAN在允许的VLAN列表中则转发；否则丢弃。	直接发送报文。
Hybrid端口	接收并打上缺省VLAN，如果缺省VLAN在允许的VLAN列表中则转发；否则丢弃。	报文带的VLAN在允许的VLAN列表中则转发；否则丢弃。	报文带的VLAN与端口缺省VLAN相等，则剥离VLAN后转发，否则直接转发。

二层破环技术——生成树协议

以太网络环形组网是一种常见的组网方式，如下图所示。然而这样组网有环路，容易造成广播风暴。



广播风暴如何产生的？举个例子，假设A要与D通讯，但A不知道D的MAC地址，于是发ARP请求。由于ARP请求是MAC广播帧，SW1收到该广播帧，进行泛洪；接着，SW3和SW2也都收到了该广播帧，也进行泛洪，于是SW1和SW2收到了SW3泛洪的广播帧，同时SW1和SW3收到了SW2泛洪的广播帧；如此无限循环，造成广播风暴。

为了检测和消除二层环路，诞生了STP（Spanning-Tree Protocol，生成树协议）及其改进的技术RSTP（Rapid Spanning-Tree Protocol，快速生成树协议）和MSTP（Multiple Spanning-Tree Protocol，多生成树协议）。这些生成树协议都是用于探知链路层拓扑，并对交换机的链路层转发行为进行控制。如果发现网络中存在环路，会在环路上选择一个恰当的位置阻塞链路上的端口——阻止端口转发或接收以太网帧，通过这种方式消除二层网络中可能产生的广播风暴。

本章关注的是数据帧的转发，因此不讨论这三种生成树协议的细节，需要关注的是生成树协议设置的端口状态及其定义的转发行为。

STP定义了5种端口状态：

- Forwarding：端口既转发用户流量也转发BPDU报文（生成树协议报文被称为BPDU报文）。
- Learning：设备会根据收到的用户流量构建MAC地址表，但不转发用户流量。
- Listening：确定端口角色，将选举出根桥、根端口和指定端口，不转发用户流量。
- Blocking：端口仅仅接收并处理BPDU报文，不转发用户流量。
- Disabled：端口不仅不转发BPDU报文，也不转发用户流量，接口状态为Down。

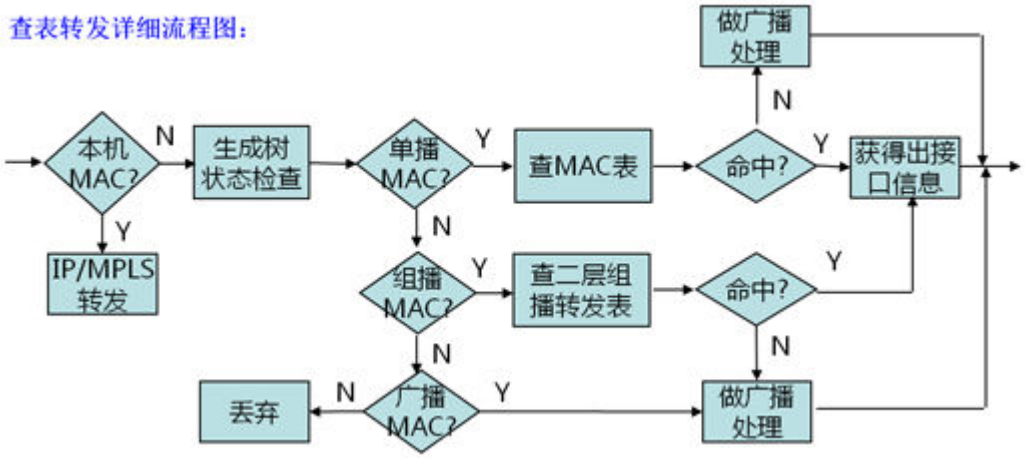
MSTP和RSTP中，将STP的5种端口状态精简为3种：

- Forwarding：端口既转发用户流量也转发BPDU报文。
- Listening：仅接收并处理BPDU报文，不转发用户流量。
- Discarding：不转发BPDU报文，也不转发用户流量。

综上可知，只有Forwarding状态的端口才转发用户流量。下一节介绍用户以太帧的转发流程。

9.2 二层桥接转发流程

华为高端路由器上，如果以太类型的接口配置了**portswitch**命令变成二层接口后，该接口便支持二层桥接转发。其二层桥接转发全流程如下图所示。需要关注的地方在于查表转发和获取封装信息两个环节（其他环节在本文的第1~7章中已描述，不再赘述）。



1. 上行转发引擎PFE（NP或ASIC芯片）解析报文，根据报文VLANID，结合端口类型（Access、Trunk、Hybrid等）进行VLAN合法性检查。不合法的报文将被丢弃。处理规则上文表格已介绍，不再赘述。
2. 判断报文的目的MAC是否为本机MAC，如果是，则做IP或MPLS转发；不是则继续下一步骤。
3. 进行入接口生成树状态检查，如果接口未使能生成树，或者生成树状态为Forwarding时，报文正常转发；如果生成树状态为其他状态，则报文丢弃。
4. 判断目的MAC是单播、组播还是广播：
   - 对于单播：使用Port+VLAN查找对应的MAC表，匹配目的MAC，获取对应的出接口信息（含出接口信息和出方向VLANID，交换网需要根据出接口信息将报文交换到正确的下行接口板）；没有命中则进行广播处理；

- 对于组播：使用Port+VLAN查找对应的L2组播转发表中获取出接口信息和出方向VLANID，如果没有命中进行广播处理。
- 对于广播，则继续后续处理。

对于单播和组播，查表过程中，如果出接口为Trunk接口，会再根据Trunk负载分担哈希算法，选择Trunk成员口中的其中一个作为最终的出接口。

5. 在上行TM芯片处理时，如果是未知单播、未知组播、广播报文，则进行复制到所有目的单板；对于已知组播，复制到组播成员口所在的目的单板；对于单播，则无需复制。
6. 在下行TM，如果是未知单播、未知组播和广播报文，进行板内复制。
7. 如果开启了MAC地址学习功能，则会在下行转发引擎进行MAC地址学习。
8. 在下行获取封装信息时，根据报文VLANID，结合出端口类型（Access、Trunk、Hybrid等）进行对应的处理。处理规则上文表格已介绍，不再赘述。
9. 在出口处理时，会再检查出接口的生成树状态，如果生成树状态为Forwarding/disable时，报文正常转发；如果生成树状态为Learning/Listening，则报文丢弃。

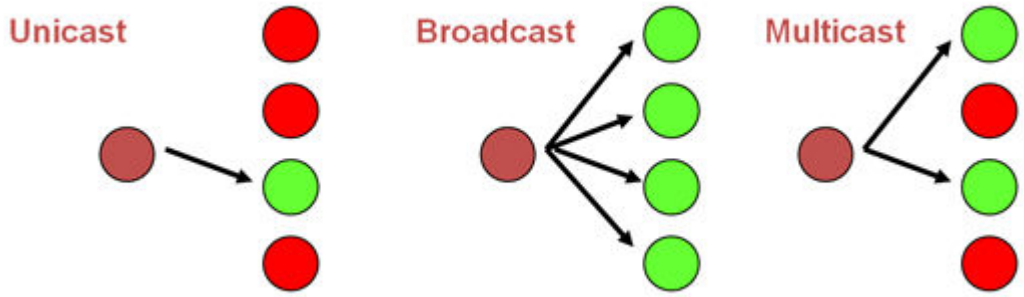
10 IP 组播转发流程

关于本章

10.1 IP 组播快速入门

单播、组播、广播

IP通信有三种方式：

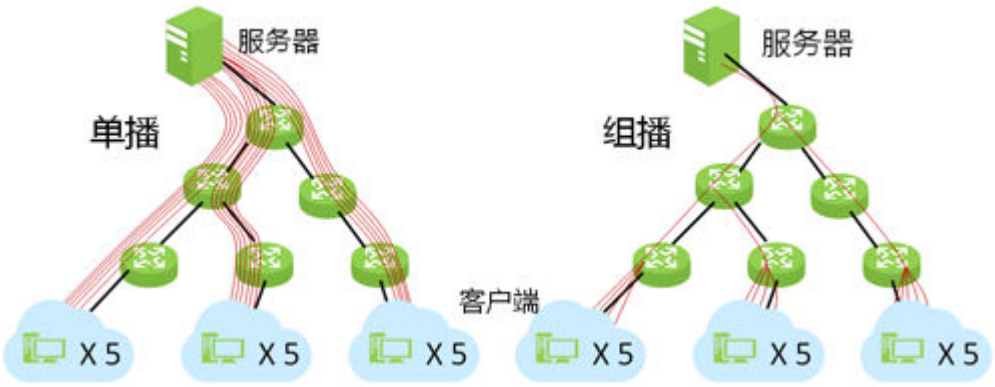


- 第一种是单播（unicast），是在一台源IP主机和一台目的IP主机之间进行。网络上绝大部分的数据都是以单播的形式传输的，例如，电子邮件收发、网页浏览、优酷/土豆等的视频点播，都是采用单播实现的。单播属于一对一（点对点）的通讯方式，同时只有一个发送者和一个接收者，中间的交换机和路由器对数据只进行转发不进行复制。
- 第二种是广播（broadcast），在一台源IP主机和网络中所有其它的IP主机之间进行。广播属于一对所有的通讯方式，无路由过程，中间的交换机和路由器对数据进行无条件的复制和转发，所有主机都可以接收到（不管是否需要）。广播不仅会将信息发送给不需要的主机而浪费带宽，也可能由于路由回环引起严重的广播风暴，所以广播数据被限制在二层交换的局域网范围内，禁止其穿过路由器，防止广播数据影响大面积的主机。但IP网络中，广播也是不可少的，如客户机通过DHCP自动获得IP地址的过程，通过ARP请求获得某IP地址对应的MAC地址的过程，都需要使用广播。
- 第三种是组播（Multicast），在一台源IP主机和网络中多台（一组）IP主机之间进行。组播是一对多（点对多点）的通讯方式，同时有一个发送者和多个接收者，中间的交换机和路由器根据接收者的需要，有选择性地对数据进行复制和转发。视频/音频会议、网络电视、股票行情发布等，便是采用组播形式。

可能有读者会有疑问，网页浏览和网络视频点播，虽然不是所有人都想看，但观看的人数不少，假设有1000个人想看同一个网页/视频，采用单播，服务器就得逐一传送，重复1000次相同工作，那为什么采用单播不是组播呢？

那是因为，不是所有客户都是同一时间想看同一内容，单播能够针对每个客户的及时响应；如果采用组播，用户便有“过了这个村就没那个点“的烦恼。视频/音频会议、股票行情发布，实时性很高，用户在同一时间看同一内容，就很适合采用组播。

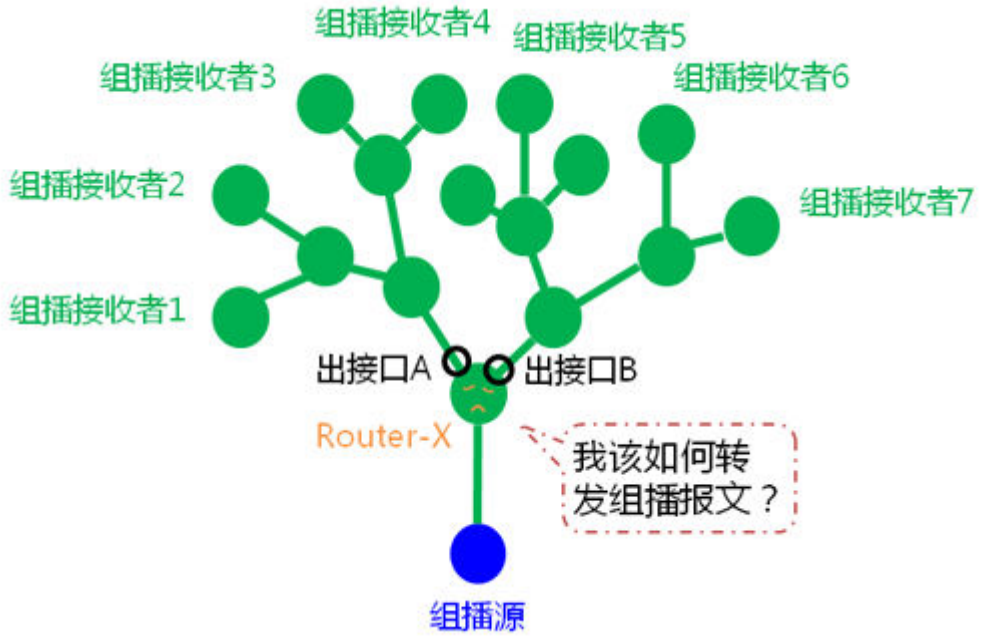
组播对于网络而言还是很有魅力的，比如下图所示场景，一目了然，组播比单播更能节约网络资源，也极大减轻了服务器的负担。



组播分发树

上图右侧所示的网络流量转发路径，像不像一棵倒长的树？这棵树称为“组播分发树”，树根是服务器（称为组播源）；树叶是客户端（称为组播接收者）。只是，组播分发树有个特别之处，从根到叶子都是一样粗，也就是说，网络负载不会随着组播接收者（客户端）数量的增加而增加，这就是组播的魅力所在。

那么，IP网络是如何将报文从组播源沿着组播分发树发送给众多接收者呢？如下图中的Router-X收到组播数据后，该如何转发呢？



按照传统IP的寻址转发机制，首先Router-X要到转发表去匹配目的接收者的地址，找到对应的出接口。上图中，组播的目的地址是组播接收者1、接收者2、……，接收者7；出接口是接口A、接口B。那么，Router-X在转发组播数据时，是拿数据包去逐一匹配目的地址（组播接收者1、接收者2、……，接收者7）吗？这显然效率太低了。如果组播接收者的数量非常巨大，尤其是越靠近组播源的节点，其组播接收者越多。这么大量的组播接收者，如果都在组播转发表项中都列出来，转发时逐一去匹配，是不大现实的。为此，诞生了“组播组”这个词。

组播组

“组播组”用来表示一群组播接收者，即组播接收者的集合。组播组并不代表网络上的具体主机，仅仅代表相应的接收者组成的集合，只用于组播报文从组播源往接收者方向发送，是单向的。“组播组”如同电视频道，组播源如同电视台。电视台向某频道内发送数据；观众想看该频道的节目，就打开电视机切换到该频道即可。

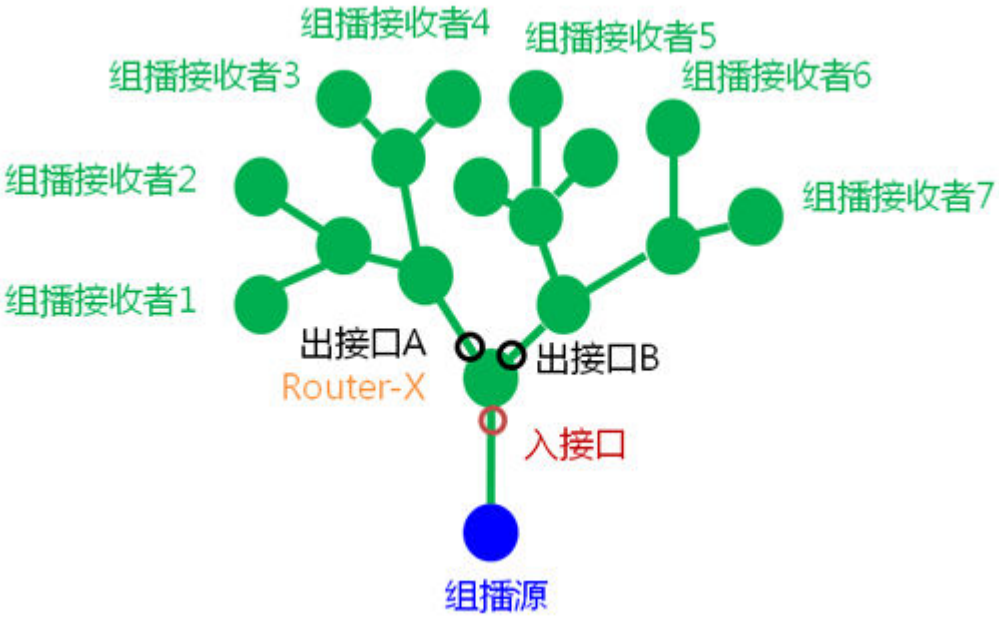
组播组地址

为了让组播数据在网络中能被正确的寻址转发，需要给“组播组”分配地址。组播报文在三层（IP）层转发时，使用的是D类IP地址，即224.0.0.0至239.255.255.255之间的IP地址。协议规定，其中的224.0.0.0至224.0.0.255为保留的组播地址，给本地网络协议使用，比如224.0.0.5和224.0.0.6这两个是OSPF协议使用的组播地址，路由器对收到的目的地址在此范围内的报文，不管报文的TTL值是多少，都不能进行转发。

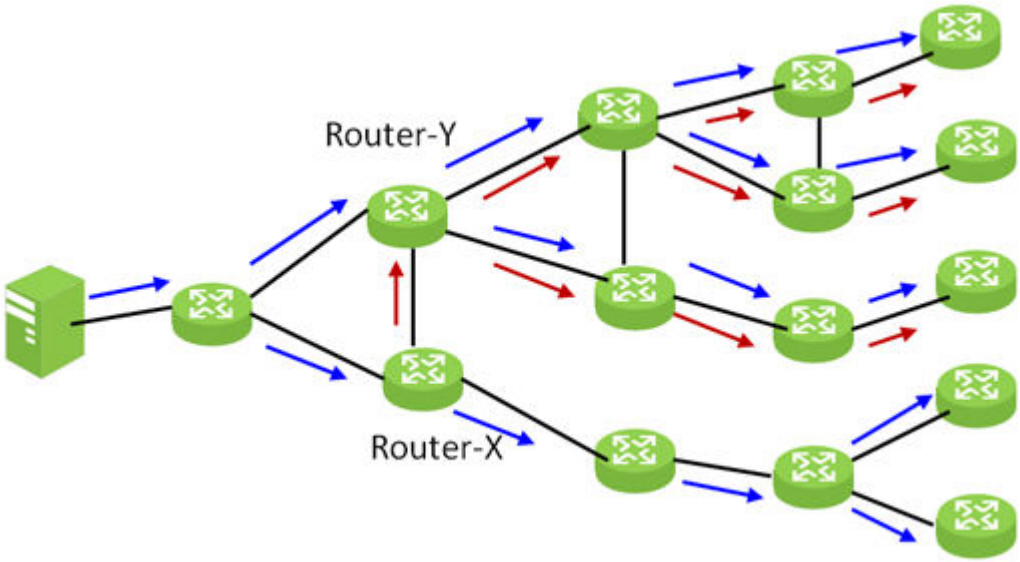
当组播报文在以太二层网络转发时，需要用到组播MAC地址。组播MAC地址同单播MAC地址一样，都是48bit，一般用6字节的十六进制来表示，如XX-XX-XX-XX-XX。IEEE 802.3规定，MAC的第0字节的第0位bit（The first bit）用于表示这个地址是组播/广播地址还是单播地址，如果这一位是0，表示此MAC地址是单播地址，如果这位是1，表示此MAC地址是多播地址或广播地址。其中，广播地址只有一个，即FF-FF-FF-FF-FF-FF。到目前为止，大部分组播MAC地址都是以0x01-00-5E开头，即01-00-5E-XX-XX-XX。

组播转发表四要素

如下图，按照传统IP的寻址转发机制，Router-X收到组播数据，到转发表去匹配目的接收者的地址，找到对应的出接口。根据前面描述，组播转发表的目的接收者的地址是“组播组地址”，出接口列表为{出接口A、出接口B}。如果有匹配的组播组地址，则将组播数据从出接口A和出接口B发送。这样就完美了吗？



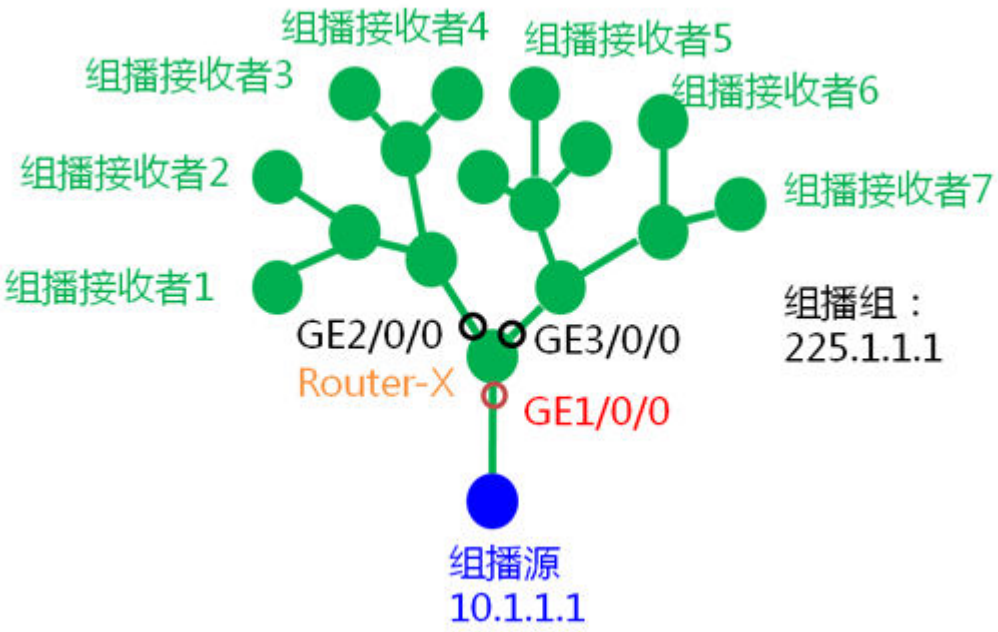
其实不然。组播报文是发送给一组接收者的，如果转发了不该转发的数据，造成的影响很大，比如下图中，正确的组播转发应该是蓝色箭头所示的方向。如果由于某种原因，Router-X把组播数据错发了一份给Router-Y（入红色箭头标识的流量），那么就会多出好多流量，浪费大量网络资源。如果存在路由环路，影响则更大。



为了能确保正确发送组播数据，组播必须严格沿着组播分发树转发，即，沿着远离组播源的方向进行转发。为了做到这点，组播技术引入了RPF（Reverse Path Forwarding，逆向路径转发）检查机制，路由器转发组播数据时，执行RPF检查，确保组播数据流能够沿组播转发树正确的传输，同时可以避免转发路径上环路的产生。RPF检查的过程是：在接收到报文后，在单播转发表中，查找到组播源地址的路由，如果该路由的出接口就是报文的入接口，则检查通过，否则不通过。也就是说，组播转发时，不仅要关心数据要到哪里去，还要关心它从哪里来。

但在实际组播数据转发过程中，如果对每一份接收到的组播数据报文都通过查找单播路由表进行RPF检查，会给路由器带来很大负担。因此，URPF检查应该放在转发表项生成之前进行，也就是路由器在生成路由表的过程中进行URPF检查，得到“组播源”及“到组播源的接口”，并将这两个信息也放入转发表项。这样，在转发组播报文时，匹配组播报文的源地址是否为转发表的“组播源”，报文的目的地址是否为转发表的“组播组地址”，如果匹配上，则判断接收报文的入接口是否就是“到组播源的接口”，如果是，表面数据是安全无误的，可以转发，如果接收报文的入接口不是“到组播源的接口”，则丢弃报文。

所以，组播转发表有四要素：“组播源”、“组播组”、“到组播源的接口”、“出接口列表”，其中组播源和组播组作为匹配对象，是个组合，通常表示为（S,G）。其中，S是Source首字母，表示组播源；G是Group的首字母，表示组播组。如下图的Router-X的组播转发表项为：（10.1.1.1, 255.1.1.1）, GE1/0/0, {GE2/0/0, GE3/0/0}。



组播转发表：

```
Multicast Forwarding Table of VPN-Instance: public net
Total 1 entry, 1 matched
00001. (10.1.1.1, 225.1.1.1), MID: 0, Flags: 0x0:0
     Uptime: 00:08:32, Timeout in: 00:03:26
     Incoming interface: GigabitEthernet1/0/0
     List of 1 outgoing interfaces:
       1: GigabitEthernet2/0/0
       2: GigabitEthernet3/0/0
 Matched 18696 packets(523488 bytes), Wrong If 0 packets
     Forwarded 18696 packets(523488 bytes)
```

那么，这么复杂的组播转发表项是如何生成的呢？这需要路由器解决如下问题：

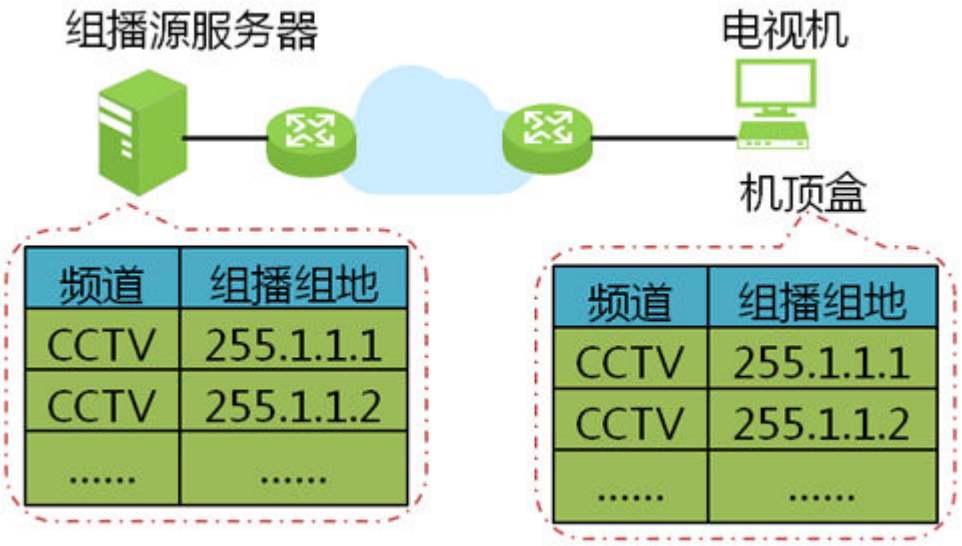
- 有哪些组播组？
- 组播组包含哪些接收者（这决定了出接口列表）？
- 组播组对应的组播源在哪里？

有哪些组播组？

回忆下单播场景下路由器是如何得知有哪些目的IP地址，也就是回忆下单播IP地址的发布过程：在接口上配置IP地址或者由RADIUS服务器动态分配IP地址给主机，然后通过路由协议等将这些IP地址发布到网络中，网络中的路由器通过运行路由协议，收集全网拓扑，然后进行路由选择，生成到这些IP地址的路由表和对应的转发表。报文转发时通过匹配转发表中的目的IP地址来查找对应出接口，从出接口发送报文。

然而，组播报文的“目的地址”是组播组地址，仅代表一个“接收者的集合”，并不代表具体主机，所以组播组地址不可能分配给主机，也不可能配置在某个接口下，那么组播地址配置在哪呢？

实际上，组播组是组播源和组播接收者之间的约定，如同电视频道。在组播源和连接电视机的机顶盒上已有节目单，即“频道”和组播组地址的对应表，如下图所示。



由此可见，组播源和组播接收者都事先配置了组播组地址。

组播组包含哪些接收者？

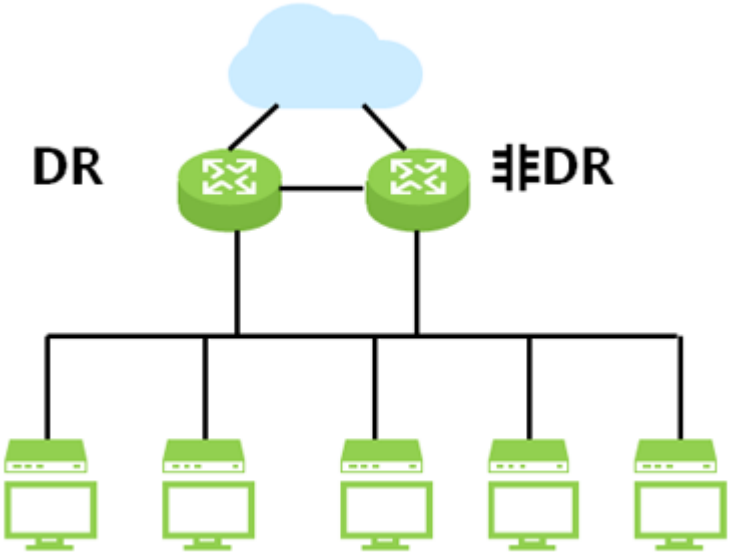
组播组包含哪些接收者？组播组成员是通过互连网组管理协议IGMP（Internet Group Management Protocol）来管理的。IGMP是运行在主机-路由器之间的协议，用于建立直连网段内的组成员关系信息，具体地说，就是哪个接口下有哪个组播组的成员。

下面以IPTV为例，IGMP主要实现过程是：

- 观众打开电视机选择收看某频道的节目时，机顶盒就向路由器发送Report消息，申请加入该频道对应的组播组；
- 当观众不再观看这个频道时，机顶盒就向路由器发送Leave消息，申请离开该频道对应的组播组；
- 路由器发送查询消息并接收主机反馈的Report消息和Leave消息，了解接口连接的网段上有哪些组播组存在接收者，也就是组成员。

机顶盒可以在任意时间、任意位置、不受限制地加入或离开组播组。就像电视频道，观众可以随时控制电视机的开关和频道间的切换。

通常，为了提高可靠性，同一个局域网通常使用两台或多台路由器接入网络，所以IGMP协议还需要选举出一台“代表”，称为接收者DR（Designed Router，指定路由器）。接收者都通过这台DR接收和发送组播数据。



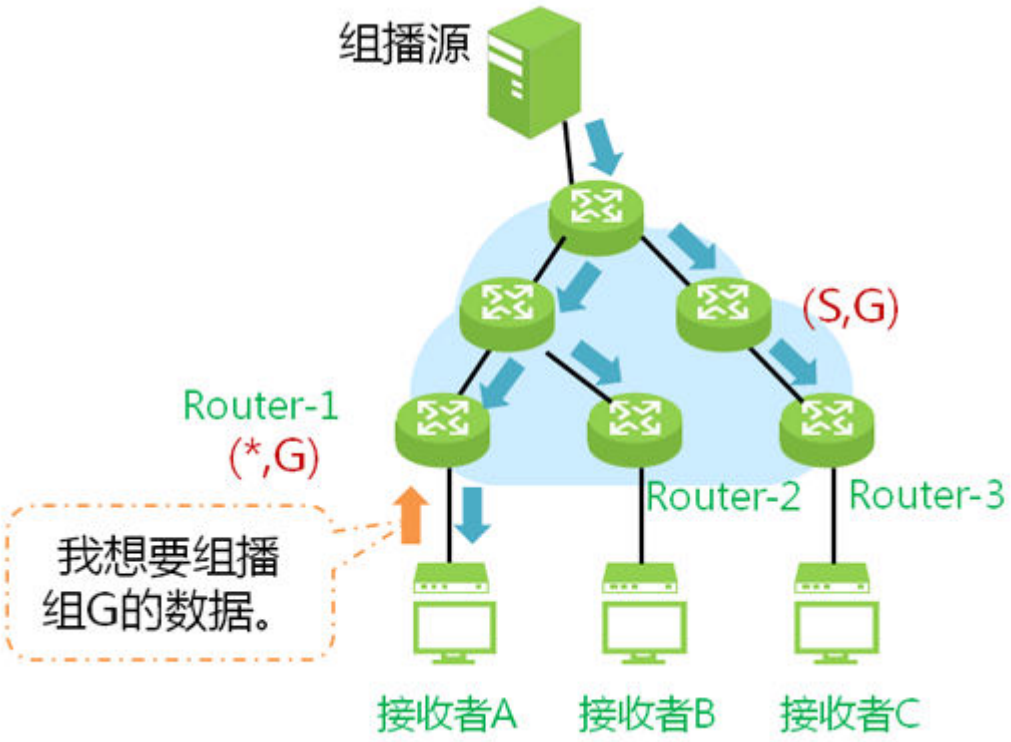
现在，组播组地址有了，组播组的成员有了，剩下的问题是：组播组地址和对应的成员信息如何发布到网络的每台组播路由器？这是通过组播路由协议来实现的。

组播路由协议有很多种，最普遍使用的是PIM（Protocol Independent Multicast，协议无关组播）协议。“协议无关”的意思是：单播路由是什么都没关系！

PIM协议有两种，PIM-DM和PIM-SM，下面简单介绍这两种协议的基本实现原理。

PIM-DM 的实现

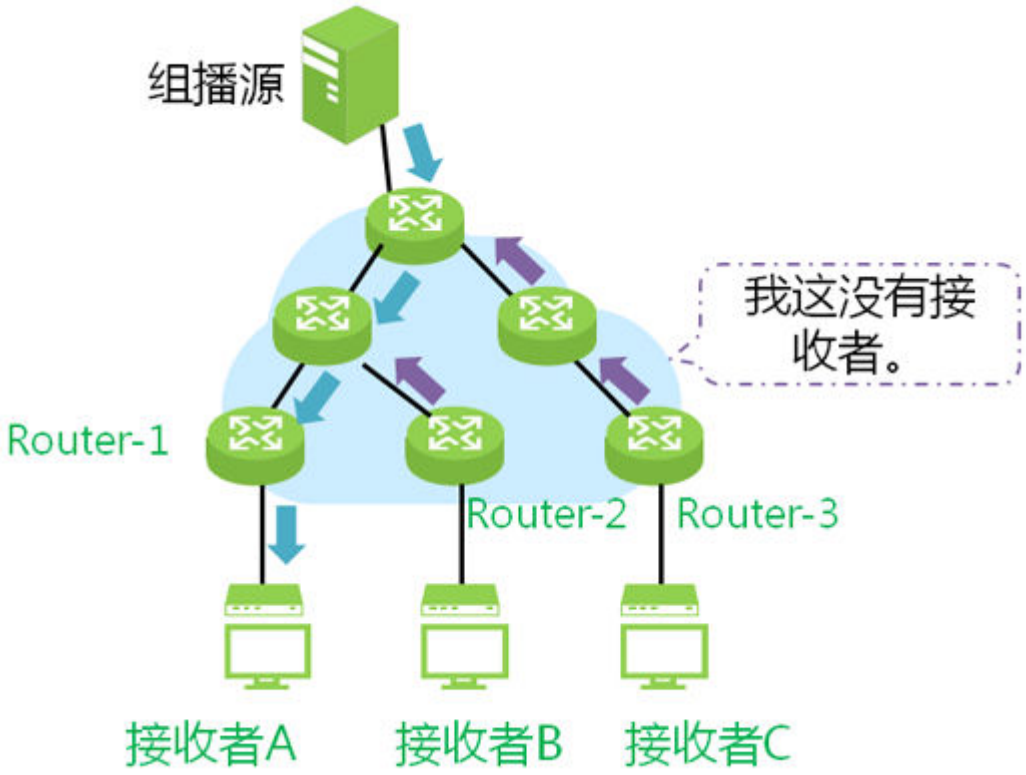
如下图，观众（接收者A）打开电视机选择收看某频道的节目，机顶盒就会通过IGMP协议向接收者DR（Router-1）报告它需要接收组播组G的数据。这样，Router-1就得到了组播组G的地址，但Router-1这时还不知道组播源在哪，因此暂时生成的是（*,G）表项，*表示任意组播源。



当组播源将组播报文发布到网络时，网络中支持组播的路由器都将报文复制转发，这样Router-1、Router-2和Router-3都收到了组播数据。由于接收者A有需求，于是Router-1把报文发送给接收者A；接收者B和接收者C没有接收需求，因此Router-2和Router-3不转发数据给接收者。

组播数据在网络传递过程中，组播路由器可获得组播源的IP地址，到组播源的入接口，以及组播组地址，其出接口列表包含了所有连接下游组播路由器的接口，这样就可以生成组播转发表项（S,G）。而接收者DR路由器可以得知自己有哪些接收者，因此更新出接口列表，把没有接收者的出接口删除，这样就得到了正确的组播转发表项了。

上述过程中，Router-2和Router-3并没有任何接收者，为了节约资源，通知上游路由器把连接自己的接口从出接口列表删除，不要再发组播数据给自己了，该过程称为“剪枝”，如下图所示。这样，网络上的所有路由器都得到了正确的转发表项。



此外，组播源可能通过多台路由器接入网络，所以PIM-DM会选举其中一台路由器作为源DR，组播源通过源DR收发组播数据。

上面过程有个问题：虽然进行了剪枝，但是有的路由器即使没有任何组播接收者，但仍然有（S,G）表项，只是出接口列表为空。于是，更优秀的PIM协议出现了，那就是PIM-SM。

PIM-SM 的实现

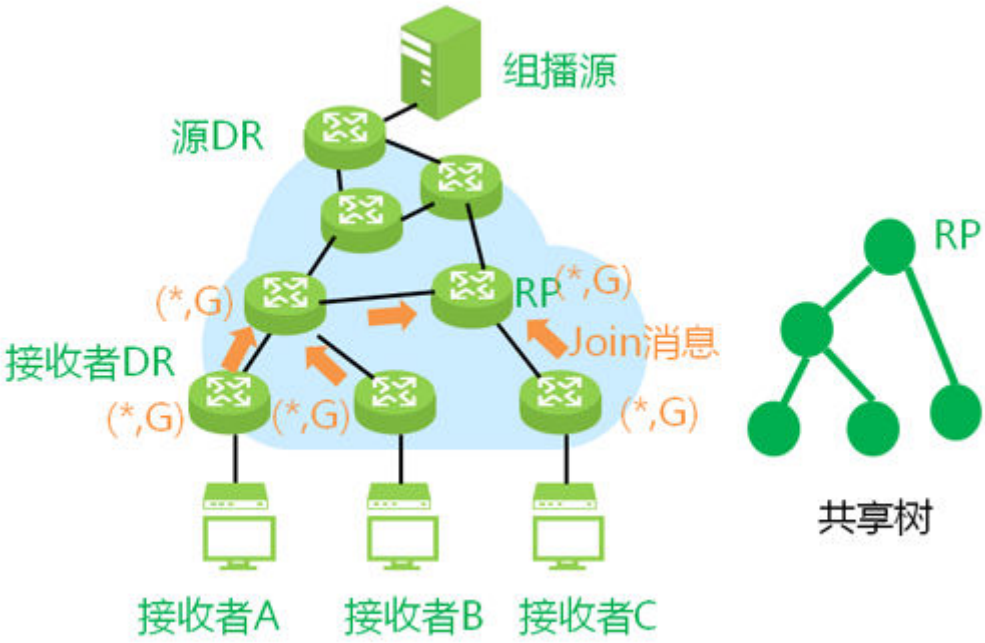
- **选“中介”**

  首先，PIM-SM协议在网络中选择一台组播路由器作为RP(Rendezvous Point)，相当于“中介”。网络中所有路由器都知道RP在哪。此外，组播源可能通过多台路由器接入网络，所以PIM-SM会选举其中一台路由器作为源DR，组播源通过源DR收发组播数据。

- **建“共享树”**

  接着，接收者DR向RP发送加入（Join）消息，于是，从接收者DR到RP及中间的路由器便可得到（*,G）表项，这样就形成了一棵以RP为根的树，这棵树被称为“共享树”。

  同PIM-DM中的实现一样，接收者通过IGMP向接收者DR路由器加入组播组G，该路由器生成（*,G）表项。

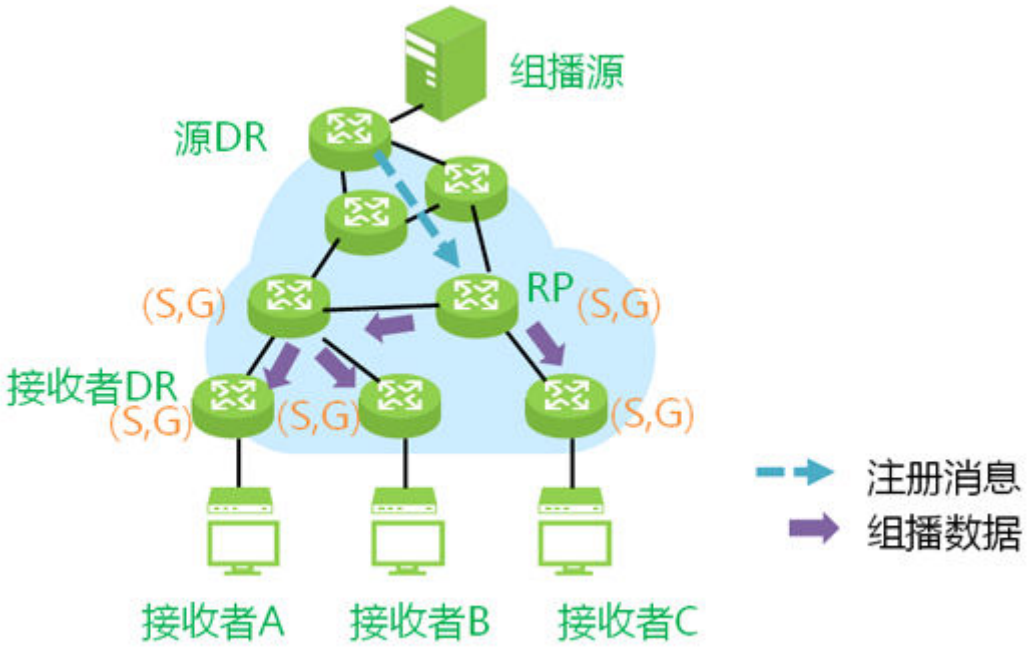


- **组播源“注册报到”**

  当源DR接收到组播源发来的组播数据时，由于没有组播转发表项，因此源DR把收到的播数据包封装在注册（Register）消息中发送给向RP，即向RP注册报到，并在本地创建（S,G）转发表项，出接口列表为空。

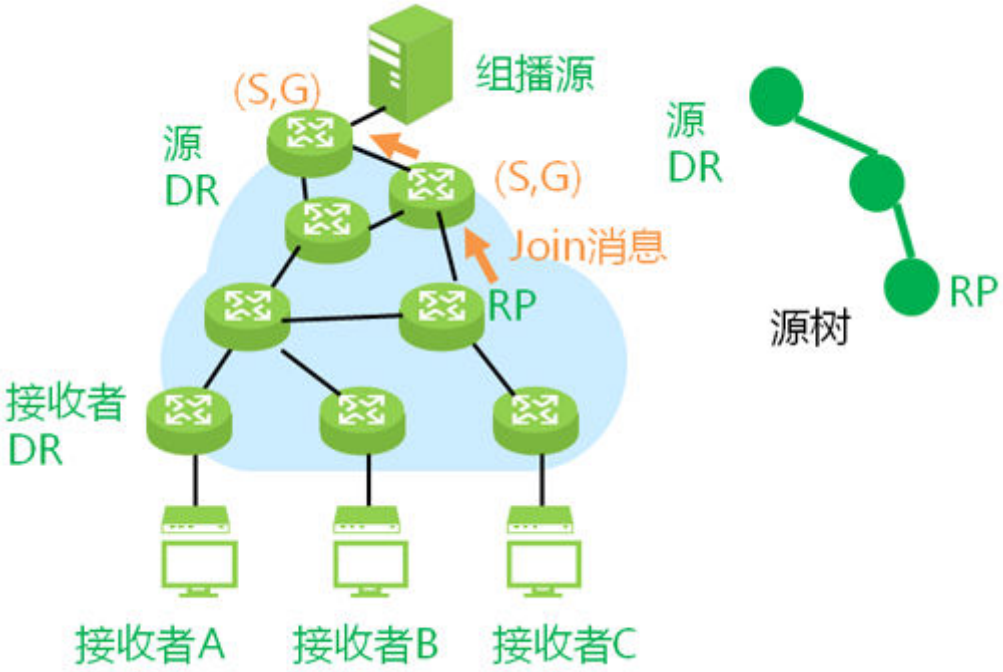
  RP收到Register消息，解封装，并将封装在其中的组播报文沿着共享树转发给接收者，沿途每台路由器便可正确生成(S,G)表项，其出接口列表从（*,G）表项复制。

  如果RP收到Register消息时还没有共享树，则丢弃组播包。



- **建“源树”**

如果有相当多的组播流量要发给RP，源DR要不断的把组播数据封装在Register消息里再发送，RP还得解封装，效率很低。因此，RP准备在自己和源DR之间建立一棵最短路径树SPT（Shortest Path Tree），这样它就可以通过SPT直接接收组播数据。



于是，RP在组播表里建立一个（S，G）表项，入口为收到Join消息的接口，并向源DR发送Join消息，这样，RP上游所有组播路由器就可以得到正确的（S,G）表项，这就形成了以源DR为根的一棵SPT树，这棵树被称为“源树”。

源DR收到RP的Join消息后，且得到了正确的（S,G）表项，此后，源DR就可以通过（S,G）表项转发组播数据。在（S,G）表项形成之前，源DR会一直将收到的组播数据通过Register消息发给RP。

SPT建立后，RP能通过树收到组的流量，RP将（S,G）表项的入口信息改为收到组播包的接口（之前为收到Register消息的接口，这两个接口不一定是相等的）。现在，再从Register消息接收被封装的组播数据就没有意义了，因此RP向源DR发送Register Stop消息，告诉它停止在Register消息中发送组播包。之后，源DR只从SPT发送组播包，不再通过Register消息发送组播包。

说明

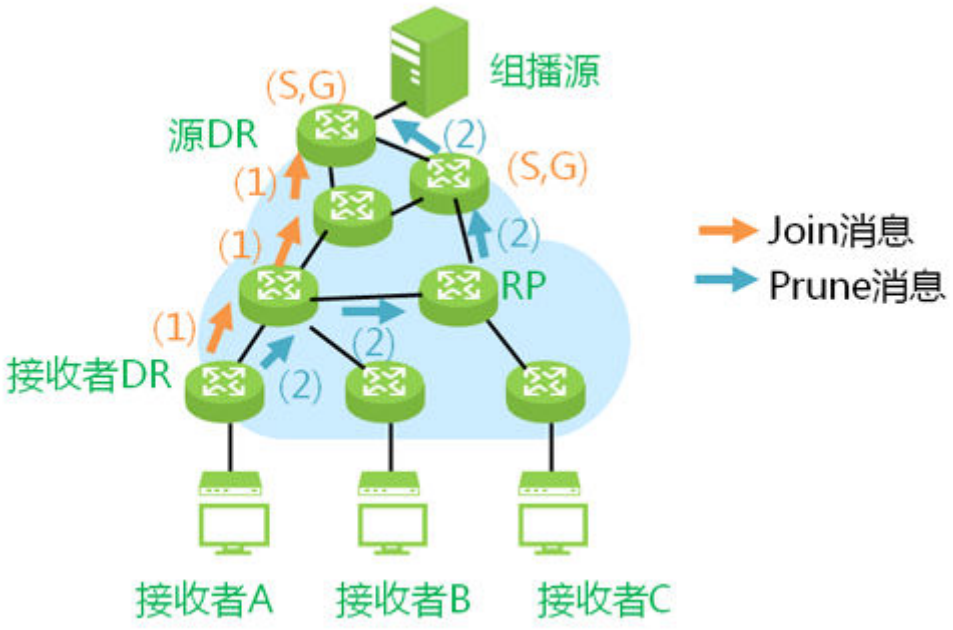
为什么SPT建立后，源DR要继续发Register消息，同时通过SPT发组播数据，即同一份组播消息发两份，直到收到Register Stop消息才停止？

那是因为，Join消息是以组播形式发送的，其源IP地址为本地接口地址，目的地址为224.0.0.13，TTL值为1。RP发的Join消息逐跳向上游发布，到达源DR时，消息的源IP已经不是RP了，源DR无法识别Register消息是RP始发的还是接收者DR始发的（见下文，接收者DR也会发Join消息逐跳转发到源DR），所以源DR无法确定它发的组播报文能否正确被RP接收，只能通过接收到Register Stop消息才能确认。

那源DR为什么需要确认它发的组播报文能正确被RP接收呢？因为，之前源DR发Register消息给RP，RP生成（S,G）表项的入接口为收到Register消息的接口。源DR后续发的组播报文到达RP的接口不一定和Register消息的接口一致，如果不一致，RP进行组播转发时Register Stop消息PF检查失败，组播报文会被丢弃。所以，源DR需要确认它发的组播报文能正确被RP接收才能停止发送Register消息。

- **SPT切换**

上述过程中，组播数据都要通过RP中转，当组播报文速率逐渐较大时，对RP形成巨大的负担。为了解决此问题，PIM-SM允许在组播报文速率较大时，当接收者DR收到组播数据时，沿着源DR的最短路径发送Join消息，生成从接收者DR到源DR的最短路径树SPT（Shortest Path Tree），接着，接收者DR向RP发送剪枝消息，剪掉原来的共享树，接着，RP再向源DR发送剪枝消息，剪掉源树。后续组播数据沿着SPT转发，不再通过RP中转，该过程称为“SPT切换”。

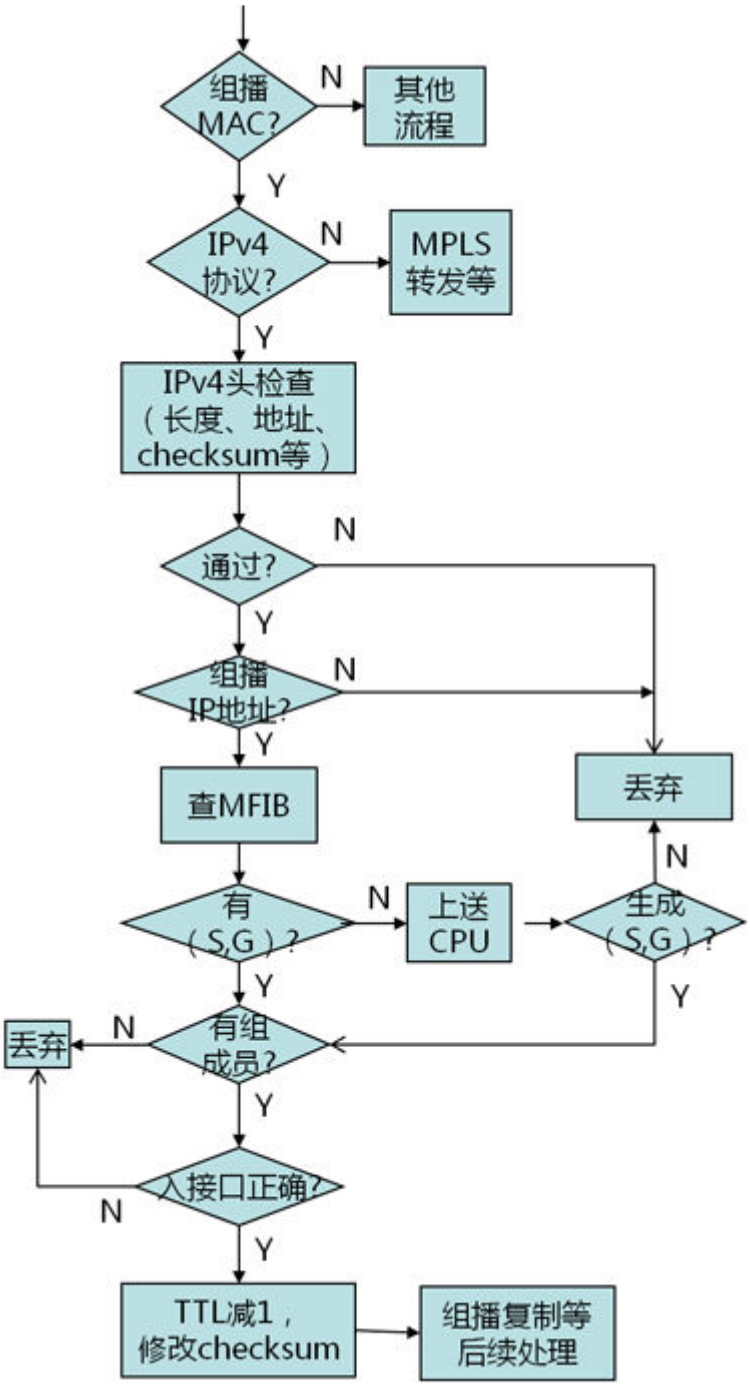


现在，全网的转发表项已经有了，下文将描述一台路由器上的组播转发详细流程。

10.2 IP 组播转发流程

IP组播转发流程如下图所示。需要关注的地方在于查表转发环节（其他环节在本文的前1~7章中已描述，不再赘述）。

查表转发流程：



1. 判断报文的目的MAC是否为组播MAC，如果不是，则做单播转发；是则继续下一步骤。
2. 判断报文的协议类型是否为IP，如果不是，则进入其他转发流程；是则继续下一步骤。
3. 检查报文的长度、IP地址、Checksum字段是否正确，如果不正确，则丢弃报文，否则继续下一步骤。
4. 判断是否为组播IP地址，如果不是组播则丢弃报文，是则进入继续下一步骤。
5. 检查入接口是否使能IP组播，如果不是则丢弃报文，是则继续下一步骤。

6. 在组播FIB（MFIB）表（如果是公网的报文，查公网MFIB表，如果是VPN报文，则查对应VPN的MFIB表）查找是否存在匹配的（S，G）表项：
   - 如果存在匹配的（S，G）表项，并且接收该报文的接口与转发表项的入接口一致，则继续步骤7的处理。特殊地，对于Register状态的出接口，表明本设备为源DR但还没收到RP的Register-Stop消息，这时需要将收到的组播上送CPU（的组播协议模块），将组播报文封装在注册消息发送给RP。
   - 如果存在匹配的（S，G）表项，但入接口不一致，将报文上送CPU处理。CPU进行RPF检查：对照单播路由表，若到组播源的接口与(S,G)表项的入接口一致，则说明（S，G）表项正确，报文来源路径错误，将报文丢弃；否则说明（S，G）表项已过时，于是根据单播路由表更新（S，G）表项中的入接口，并刷新转发表。然后再检查收到报文的接口是否就是更新后的接口，是则继续步骤7的处理，否则将其丢弃。
   - 如果不存在匹配的（S，G）表项，有两种场景：
     - 一种场景是本路由器是源DR，收到组播源发送的第1个组播报文，此时还没有（S,G）表项，需要将报文上送CPU处理，将报文封装成Register消息发给RP，同时CPU下发出接口为空的（S,G）表项，等收到RP的注册报文再添加出接口。如果注册失败，则为了减少上报对CPU的负担，后续的组播数据流就会进行转发，但是因（S,G）表项出接口为空，实际上是把报文丢弃了。
     - 另一种场景是组播组的第1个报文从RP向接收者方向发送时，由于共享树上（除RP外）的路由器只有（*，G）表项，没有(S,G)表项，此时收到的组播报文也上送CPU处理，CPU生成(S,G)表项，其出接口列表从（*，G）表项拷贝。接着CPU对报文进行RPF检查，如果检查失败则丢弃报文，否则继续步骤7的处理。
7. 检查匹配的（S,G）表项是否有对应的组成员，即检查对应的出接口列表是否为空，如果为空，则报文丢弃；否则继续下一步骤。
8. 检查报文的入接口与（S,G）表项的入接口是否一致，如果不一致则丢弃报文，否则继续下一步骤。
9. 进行组播复制等后续处理，如上行TM进行板间组播复制，下行TM进行板内组播复制，详细处理过程在本文的前1~6章中已描述，不再赘述。

11 MPLS 转发流程

关于本章

11.1 MPLS 基础

MPLS 简介及产生背景

90年代中期，基于IP技术的Internet快速普及。但由于当时硬件技术存在限制，基于最长匹配算法的IP技术必须使用软件查找路由，转发性能低下。因此IP技术的转发性能成为当时限制网络发展的瓶颈。为了适应网络的发展，ATM（Asynchronous Transfer Mode）技术应运而生。ATM采用定长标签（即信元），并且只需要维护比路由表规模小得多的标签表，能够提供比IP路由方式高得多的转发性能。然而，ATM协议相对复杂，且ATM网络部署成本高，这使ATM技术很难普及。

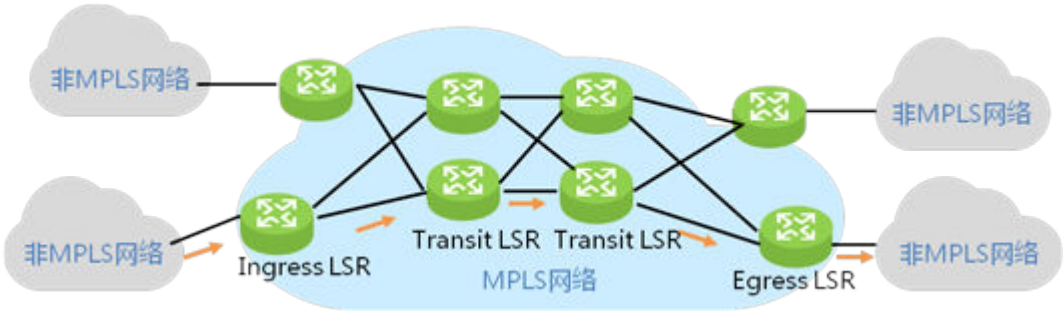
传统的IP技术简单，且部署成本低。如何结合IP与ATM的优点成为当时热门话题。多协议标签交换技术MPLS（Multiprotocol Label Switching）就是在这种背景下产生的。

MPLS最初是为了提高路由器的转发速度而提出的。与传统IP路由方式相比，它在数据转发时，只在网络边缘分析IP报文头，而不用在每一跳都分析IP报文头，节约了处理时间。

随着技术的发展，路由查找速度已经不是阻碍网络发展的瓶颈，这使得MPLS在提高转发速度方面不再具备明显的优势。但是，MPLS支持多层标签和转发平面面向连接的特性，使其在VPN（Virtual Private Network）、流量工程、QoS（Quality of Service）等方面得到广泛应用。

MPLS 网络典型结构

MPLS网络的典型结构如下图所示，其基本组成单元是标签交换路由器LSR（Label Switching Router）。从数据包转发方向看，位于MPLS域边缘、连接其它网络的入口LSR称为Ingress LSR，位于MPLS域的LSR称为Transit LSR，出口LSR称为Egress LSR。



MPLS网络基于标签进行转发。数据包进入MPLS网络时，Ingress LSR分析数据包的内容并且为这些数据包添加合适的标签，后续MPLS网络中的节点都是依据标签来转发数据包的。当该数据包离开MPLS网络时，标签由出口边缘路由器删除。

MPLS 标签

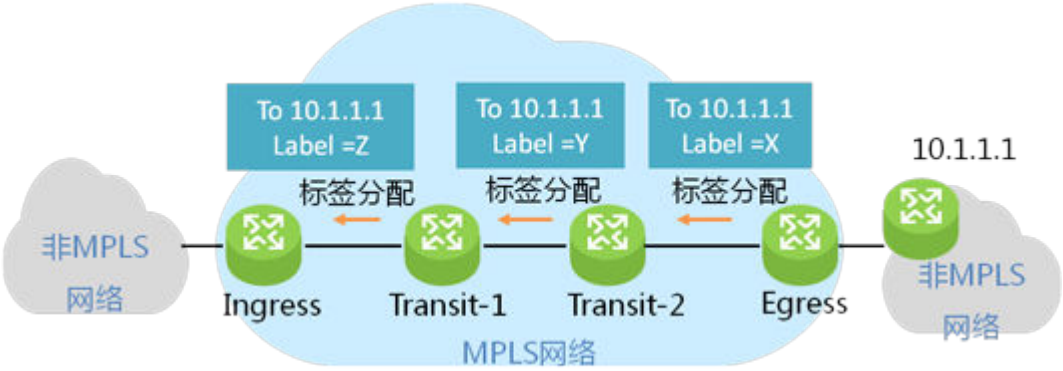
标签长度为4个字节，其结构如下图所示。

- Label：标签值域。
- EXP：用于扩展。现在通常用做CoS（Class of Service），其作用与Ethernet的802.1p类似。
- S：栈底标识。MPLS支持多层标签。S值为1时表明该标签为最底层标签。
- TTL：和IP分组中的TTL（Time To Live）意义相同。

标签分发过程

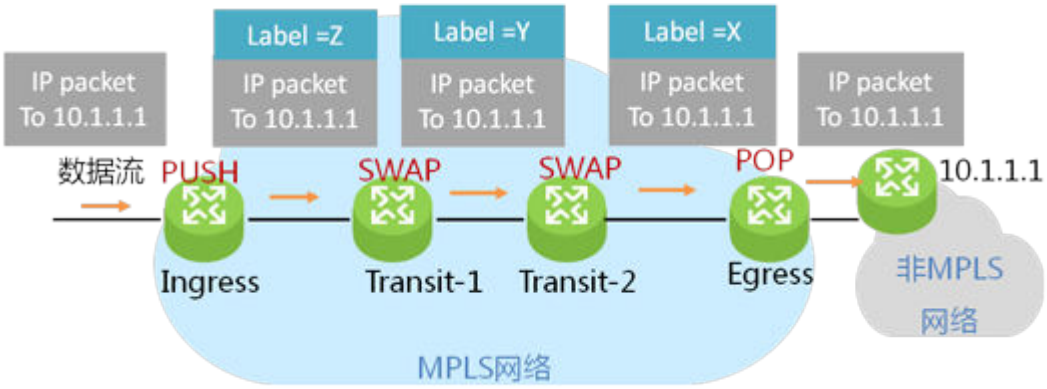
数据包在MPLS网络中经过的路径称为标签交换路径LSP（Label Switched Path）。LSP是一个单向路径，与数据流的方向一致。MPLS转发也是属于“先铺路，后通车”的方式，在MPLS转发之前，要先进行标签分发，建立标签交换路径LSP。

标签由下游分配（假设一条LSP，其数据流方向是A->B->C，那么我们称C为B的下游，B为C的上游，同理B为A的下游，A为B的上游），按从下游到上游的方向分发，如下图所示。



为什么标签是由下游分配呢？答案很简单：因为标签是给下游转发时用的，谁用谁来分配。如果某个LSR自己分配标签给下游转发用，那下游是不知道这个标签代表什么含义呢，除非它们事先进行了协商/约定，这做法还不如下游自己分配来得简单。

报文转发过程



1. Ingress节点收到目的地址为10.1.1.1的IP报文，添加标签Z并转发。
2. Transit-1节点收到该标签报文，进行标签交换，将报文Z标签弹出，换成标签Y。
3. Transit-2节点收到该标签报文，进行标签交换，将报文Y标签弹出，换成标签X。
4. Egress节点收到该报文，将标签X弹出，并进行IP转发，将其发送给目的地10.1.1.1。

标签操作——PUSH、SWAP 和 POP

- 添加标签的操作称为“PUSH”，如上述步骤1。
- 用下一跳分配的标签替换MPLS报文的栈顶标签的操作，称为“SWAP”，如上述步骤2和步骤3。
- 当报文离开MPLS域时，将MPLS报文的标签去掉的操作，称为“POP”，如上述步骤4；另外，在MPLS倒数第二跳节点处去掉栈顶标签（下文马上要介绍的PHP机制）的操作，也称为“POP”。

倒数第二跳弹出(PHP)机制与隐式空标签

先假设一个场景：MPLS数据包到了最后一跳（Egress）节点，Egress查找MPLS转发表，弹出该标签，弹出后发现是个IP包，于是查找IP转发表将报文转发出去。Egress进行了2次查表，这显然降低了转发效率。因此，人们发明了倒数第二跳弹出PHP（Penultimate Hop Popping）机制，在倒数第二跳节点处将标签弹出，最后一跳节点直接进行IP转发或者下一层标签转发，减少最后一跳的负担。

支持PHP的Egress节点分配给倒数第二跳节点的标签只有一种：值为3的标签，称为隐式空标签，这个值不会出现在数据包的标签栈中。当一个LSR发现自己被分配了隐式空标签时，它并不用这个值替代栈顶原来的标签，而是直接执行Pop操作。

MPLS VPN 简介

上文介绍MPLS产生背景时说过，MPLS在VPN得到了广泛应用，那什么是VPN呢？

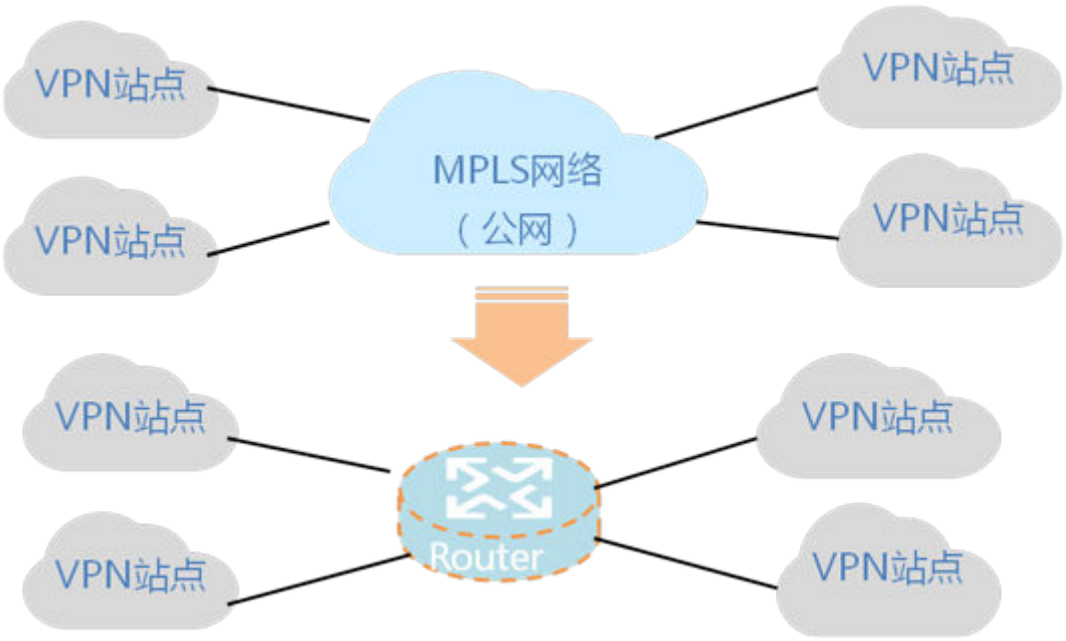
最初，电信运营商是以租赁专线的方式为企业提供二层链路，这条专线所经过的链路是企业独占的。增加一条专线，都要搭建新的物理链路，不仅建设时间长，而且价格昂贵。后来，出现了ATM（Asynchronous Transfer Mode）和帧中继FR（Frame Relay）技术，使得运营商可以在同一个网络上使用虚电路方式为客户提供点到点的专线。这种专线建设时间短、价格低。但虚电路依赖于专用的传输介质：为提供基于ATM的专线服务，运营商需要建立覆盖全部服务范围的ATM网络；为提供基于FR的专线服务，又需要建立覆盖全部服务范围的FR网络。不仅网络建设成本高，而且速率较慢，不能满足当前Internet应用对于速率的要求。

于是，一种新的替代方案产生了：在现有IP网络上模拟传统专网，这种新的方案就是虚拟专用网VPN（Virtual Private Network）。VPN的一个重要本质是使用共享的网络（共享的网络被称为“公网”）提供虚拟的专线业务。于是问题来了，毕竟每个企业都不希望自己在网络上传输的数据随意暴露在大庭广众之下，现在让他们共享同一个网络，那么不同的企业所使用的VPN应该是互相隔离的，因此，这些企业私有报文必须对其他VPN不可见，必须进行透明传输。为此，VPN使用“隧道技术”来传输数据。

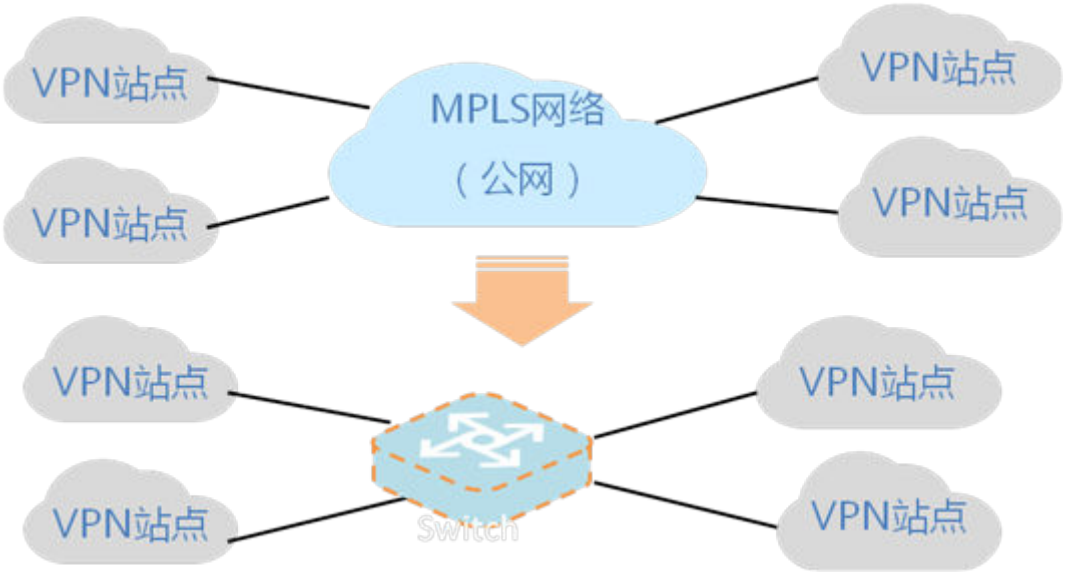
“隧道技术”是指在两个网络节点之间提供一条通道（被称为“隧道”），使数据报能够在这个通路上透明传输。隧道的建立需要使用到隧道协议。隧道协议又很多种，比如GRE（Generic Routing Encapsulation）、L2TP（Layer 2 Tunneling Protocol），还有上文介绍的MPLS。建立好隧道之后，便可以使用隧道来传输报文：通过在隧道的一端给数据加上隧道协议首部字段，在隧道的另一端去掉该数据携带的隧道协议头。隧道是构建VPN不可或缺的部分。在众多类型的隧道中，运营商网络最常用的当属MPLS LSP。使用MPLS LSP作为隧道的VPN就被称为MPLS VPN。

MPLS VPN按实现层次分，可分为MPLS L3VPN、MPLS L2VPN，其中MPLS L2VPN有可分为VPLS和VLL/PWE3。

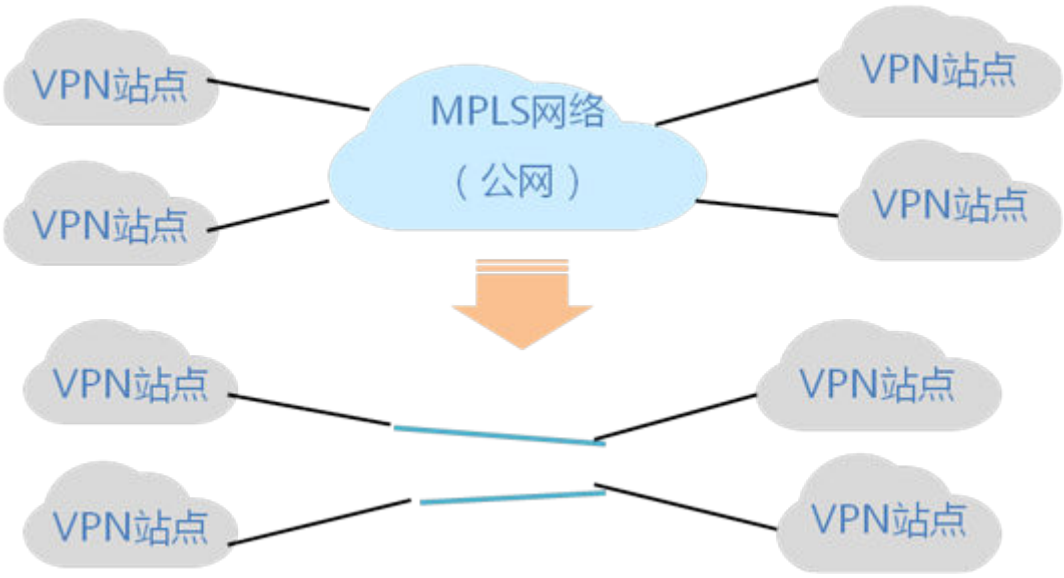
- MPLS L3VPN：从L3VPN的用户角度看，共享网络（公网）就像一台路由器，将VPN各站点连接起来。公网为每个VPN建立专用的路由表和转发表。



- VPLS：VPLS（Virtual Private LAN Service）实现了局域网之间通过虚拟专用网段互连。从VPLS的用户角度看，公网就像一台以太网交换机，将用户各站点连接起来。VPLS又称为E-LAN。



- VLL/PWE3：VLL（Virtual Leased Line）是对传统租用线业务的仿真，使用IP网络模拟租用线，从用户角度看，公网就像一条点到点的链路，将VPN站点连接起来。VLL又称为VPWS（Virtual Private Wire Service）或者E-Line。PWE3（Pseudo Wire Emulation Edge-to-Edge）是VLL的一种扩展。



MPLS 标签的位置

标签封装在链路层和网络层之间，标签在数据包中的位置如下图所示。

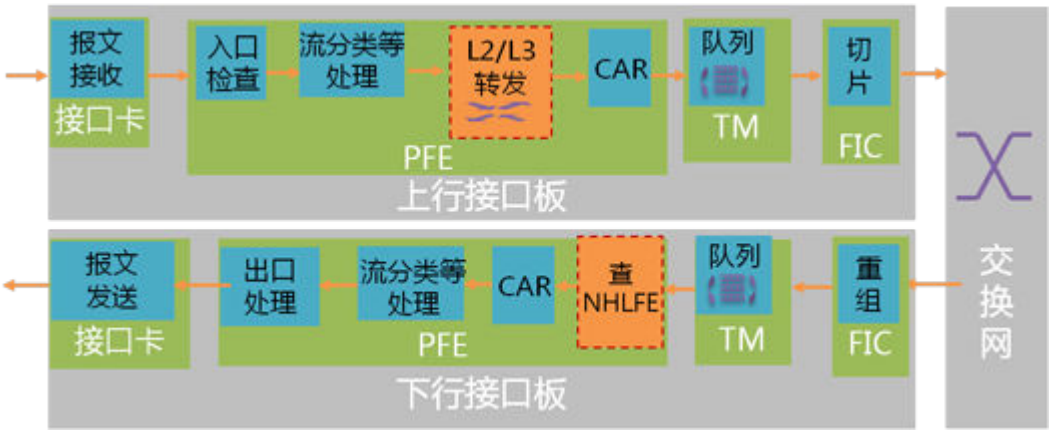
MPLS报文支持同时携带多个标签，靠近物理层的标签称为栈顶标签或外层标签；靠近网络层的标签称为栈底标签，或内层标签。理论上，MPLS标签可以无限嵌套。

在MPLS VPN中，最内层的标签被称为VPN标签或私网标签，外层标签被称为隧道标签或公网标签。

11.2 MPLS 转发流程

Ingress 节点的处理

报文进入MPLS网络时，Ingress节点分析报文的内容并且为这些报文添加合适的标签，后续所有MPLS网络中的节点都是依据标签来转发数据的。当该报文离开MPLS网络时，标签由出口边缘路由器删除。对应于路由器内部处理，Ingress节点的处理是：



1. 上行包转发引擎PFE上解析报文，确定转发类型，此过程和普通的二层转发和三层IP转发流程的上行处理完全一致。如果是VPLS、VLL和PWE3场景，上行做的是二层转发；如果是MPLS L3VPN场景，上行做的是三层IP转发。
   - MPLS L3VPN场景：如果是MPLS L3VPN场景，则上行做的是三层IP转发。三层IP转发时，查找FIB，对应的Tunnel ID如果是0x0，表示做普通IP转发；如果是非0x0，表示MPLS L3VPN转发。

     ```
     Destination/Mask  Nexthop    Flag   TimeStamp   Interface    TunnelID
     1.1.1.1/32        10.0.0.1   DGU    t[347299]   GE1/0/0      0x0
     10.0.0.0/24       10.0.0.1   U      t[257502]   GE1/0/0      0x0
     192.0.0.0/24      10.0.0.1   DGHUT  t[670625]   GE1/0/0      0x2000001
     127.0.0.0/8       127.0.0.1  U      t[102]      InLoop0      0x0
     ```

     VPN报文在MPLS域中是通过LSP隧道中进行透明传输，VPN报文在Ingress节点上的出接口是一条LSP隧道。为了给使用隧道的上层应用（如VPN、路由管理）提供统一的接口，系统自动为每条隧道分配了一个ID，即Tunnel ID。该Tunnel ID只是本地有效。Tunnel ID的格式如下图。

     其中，LSP Token字段用于在MPLS转发表中查找指定的MPLS转发信息。LSP Token只是一个查找MPLS转发信息的索引号。
   - VPLS场景：如果是VPLS场景，上行做的是二层桥接转发。根据目的MAC和VLAN ID查MAC表，可获得出接口信息和LSP Token。

     ```
     MAC Address      VLAN/     PEVLAN CEVLAN Port          Type       LSP/LSR-ID
                      VSI/SI                                          MAC-Tunnel
     ------------------------------------------------------------------------------
     0005-0005-0005   b         -      -      GE3/1/6       static     3/-
     ```

   - VLL/PWE3场景： 如果是VLL或PWE3场景，上行做的是二层VPN转发，查L2VPN表，可获得控制字标识、出接口信息和LSP Token。
2. 进行上行后续的公共处理，并通过交换网板到下行。
3. 到了下行的转发引擎PFE，根据对应的Token查找下一跳标签转发表项NHLFE（Next Hop Label Forwarding Entry），此表用于指导MPLS转发。

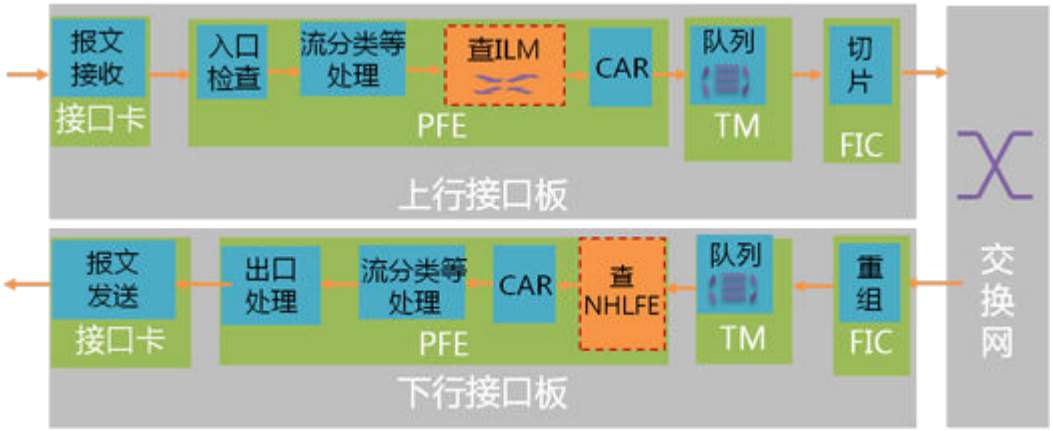
   ```
   NHLFE:
   LSR Type          : Ingress
   Tunnel id         : 0x2000001
   Out interface     : GigabitEthernet1/0/0
   Nexthop           : 10.0.0.1
   Out label         : 4096
   Label operation   : PUSH
   ```

   查看NHLFE表项，可以得到内、外层标签及其标签操作类型，以及出接口和下一跳。

4. 进行后续的下行公共处理。
5. 到了出口处理模块，根据获得的封装信息进行报文封装。在报文中压入2层标签（其中内层标签的S位置1，表示栈底标签，外层标签的S位置0。如果是VLL/PWE3，根据控制字标识决定是否在内层标签和负荷之间添加控制字），并封装链路层信息（如果链路层为以太类型，则Eth-Type值封装为0x8847）。之后将数据送到PIC卡，转换为光/电信号发送出去。

Transit 节点的处理

无论是MPLS L3VPN、VPLS、VLL还是PWE3场景，在Transit节点上的处理都相同。



1. 在上行解析报文发现报文协议类型是MPLS，则用报文携带的最外层（栈顶）的MPLS标签查入标签映射表ILM（Incoming Label Mapping），获得对应的Tunnel ID和出接口信息。出接口信息含目的单板和目的接口。

   如果是负载分担，会查到多份这样的信息，于是根据负载分担哈希算法选取其中的一份。

   如果是FRR（Fast Reroute）状态，则会根据LSP状态和出端口状态做主备路由选择，如果主LSP和对应出端口正常工作，则会选择主LSP；否则选择FRR LSP（即备份LSP）。

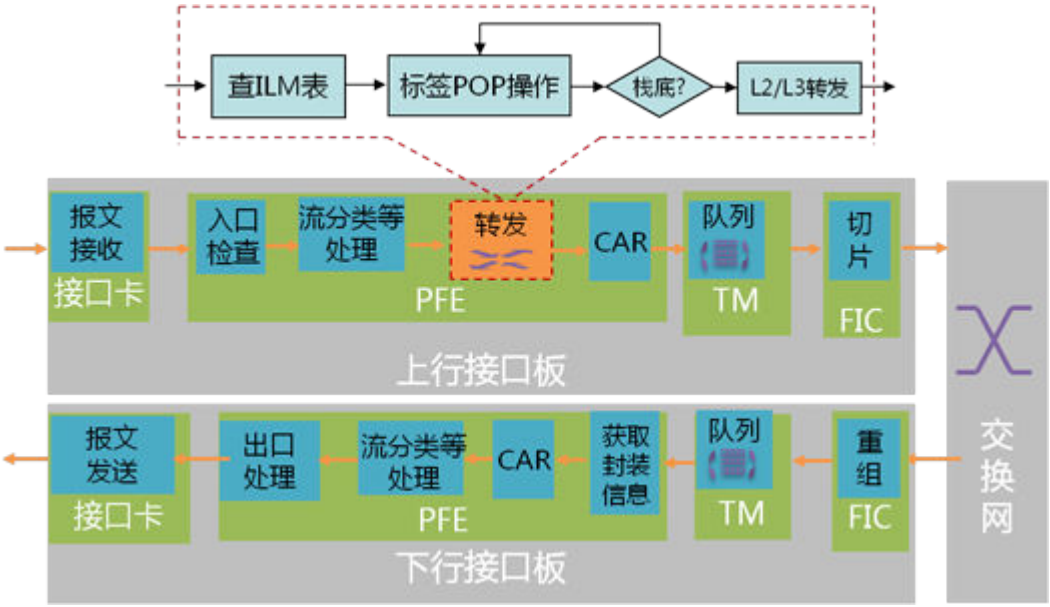
```
ILM:
In Label          : Ingress
Swap label        : --
Load-balance Count: 2
Tunnel id [0]     : 0x2000002
Out interface [0] : GigabitEthernet2/0/0
Nexthop [0]       : 20.2.1.2
Tunnel id [1]     : 0x2000003
Out interface [1] : GigabitEthernet2/0/1
Nexthop [1]       : 20.2.2.2
Has FRR LSP       : No
FRR inner label   : --
FRR tunnel id     : 0
FRR out interface : no
FRR nexthop       : no
```

2. 如果出接口为Trunk接口，会再根据Trunk负载分担哈希算法，选择Trunk成员口中的其中一个作为最终的出接口。
3. 进行后续的公共处理，并经过交换网（交换网根据上行获得的目的单板信息做交换）交换到下行。
4. 到了下行的转发引擎PFE，根据对应的Tunnel ID的LSP Token值查找下一跳标签转发表项NHLFE。得到出接口、下一跳、出标签和标签操作类型。其中，标签操作类型为SWAP或者POP（如果标签值为3则为POP，其他值则为SWAP）。

```
NHLFE:
LSR Type          : Transit
Tunnel id         : 0x2000002
Out interface     : GigabitEthernet2/0/0
Nexthop           : 20.2.1.2
Out label         : 3
Label operation   : SWAP
```

5. 如果出标签值不是3（3表示隐式空标签），则进行SWAP操作，将最外层的入标签值换成出标签值（且TTL值减一），并封装链路层信息（如果链路层为以太类型，则Eth-Type值封装为0x8847）。如果出标签值是3，则弹出最外层标签，将内层标签头的MPLS TTL重新赋值等于外层MPLS TTL减1，然后再封装链路层信息。
6. 进行后续的CAR等公共处理。
7. 在下行流分类时根据QoS策略重新设置MPLS头部的EXP，并在出口做检查，之后将数据送到PIC卡，转换为光/电信号发送出去。

Egress 节点的处理



1. 根据最外层标签查入标签映射表ILM（Incoming Label Mapping），获得对应的标签操作是POP，则进行弹出标签操作。
2. 检查弹出的标签中S字段是否置1。如果S位置0，表示非栈底，需要继续进行标签转发，则重复步骤1操作。如果S位置1，表示栈底标签，则根据负荷做L2转发或L3转发。关于L2转发和L3转发的详细介绍请参见章节“**9 二层桥接转发流程**”和“**8 IP单播转发流程**”。

11.3 MPLS 对 TLL 的处理

RFC3443中定义了两种MPLS对TTL的处理模式：Uniform和Pipe。

- Uniform模式：IP包经过MPLS网络时，在Ingress点，IP TTL减1映射到MPLS TTL字段，此后报文在MPLS网络中TTL减1，IP TTL保持不变。在Egress点将MPLS TTL减1后映射到IP TTL字段。
- Pipe模式：在入节点，IP TTL值减1，MPLS TTL字段为固定值（一般设为255），此后报文在MPLS网络中TTL减1，IP TTL保持不变。在出节点会将IP TTL字段的

值减1。即IP分组经过MPLS网络时，无论经过多少跳，IP TTL只在入节点和出节点分别减1。

在华为高端路由器上，可通过配置，分别设置外层标签（公网隧道）和内层标签（L3VPN）中的TTL模式，即有4种组合：

组合	外层(隧道)TTL模式	内层(L3VPN)TTL模式
组合1	Uniform	Uniform
组合2	Pipe	Uniform
组合3	Uniform	Pipe
组合4	Pipe	Pipe

不同模式的组合，处理方式不同。

Ingress 对 TTL 的处理

- 无论如何，IP TTL先减一；
- 内层 TTL模式为Uniform时，内层MPLS TTL复制IP TTL；
- 内层 TTL模式为Pipe时，内层MPLS TTL=255；
- 外层TTL模式为Uniform时，外层MPLS TTL复制内层MPLS TTL；
- 外层TTL模式为Pipe时，外层MPLS TTL=255。

组合	外层MPLS TTL模式	内层MPLS TTL模式	IP TTL	内层MPLS TTL	外层MPLS TTL
1	Uniform	Uniform	减1	=IP TTL	=内层MPLS TTL
2	Pipe	Uniform	减1	=IP TTL	255
3	Uniform	Pipe	减1	255	=内层MPLS TTL
4	Pipe	Pipe	减1	255	255

Transit 对 TTL 的处理

- 无论如何，IP TTL保持不变；
- 通常情况：外层MPLS TTL减1，内层MPLS TTL不变；
- 倒数第二跳隐式空标签弹出的情况：外层MPLS TTL不减1，内层MPLS TTL复制外层MPLS TTL（此时复制外层MPLS TTL后再减1）。

通常情况：

组合	外层MPLS TTL模式	内层MPLS TTL模式	外层MPLS TTL	内层MPLS TTL	IP TTL
1	Uniform	Uniform	减1	不变	不变
2	Pipe	Uniform	减1	不变	不变
3	Uniform	Pipe	减1	不变	不变
4	Pipe	Pipe	减1	不变	不变

支持PHP的倒数第二跳的处理：

组合	外层MPLS TTL模式	内层MPLS TTL模式	外层MPLS TTL	内层MPLS TTL	IP TTL
1	Uniform	Uniform	标签弹出	=外层MPLS TTL -1	不变
2	Pipe	Uniform	标签弹出	=外层MPLS TTL -1	不变
3	Uniform	Pipe	标签弹出	=外层MPLS TTL -1	不变
4	Pipe	Pipe	标签弹出	=外层MPLS TTL -1	不变

Egress 对 TTL 的处理

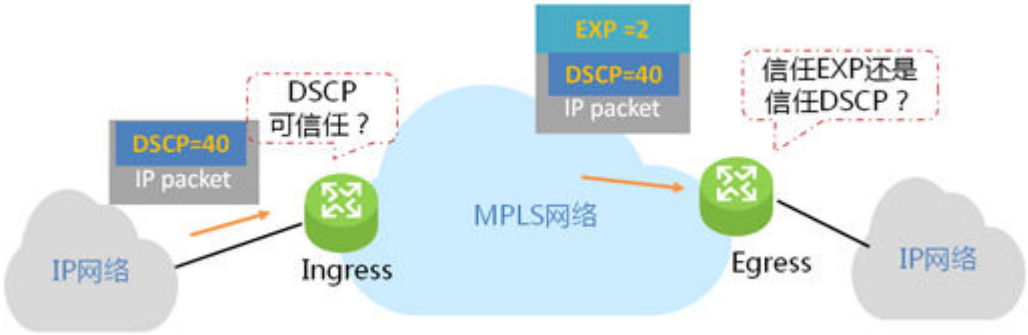
- 外层TTL模式为Uniform时，如果有外层标签，则内层MPLS TTL复制外层MPLS TTL，否则内层MPLS TTL不变（不复制外层的，也不减一）；
- 外层TTL模式为Pipe时，内层MPLS TTL不变（不复制外层的，也不减一）；
- 内层TTL模式为Uniform时，IP TTL复制内层MPLS TTL，并减1；
- 内层TTL模式为Pipe时，IP TTL减1。

组合	外层MPLS TTL模式	内层MPLS TTL模式	外层MPLS TTL	内层MPLS TTL	IP TTL
1	Uniform	Uniform	标签弹出	=外层MPLS TTL（有外层标签时）；或者不变（无外层标签时）	=内层MPLS TTL-1
2	Pipe	Uniform	标签弹出	不变	=内层MPLS TTL-1
3	Uniform	Pipe	标签弹出	=外层MPLS TTL（有外层标签时）；或者不变（无外层标签时）	减1
4	Pipe	Pipe	标题弹出	不变	减1

11.4 MPLS COS 处理模式

QoS DiffServ体系结构允许差分域内的中间节点检查并修改IP Preference、DSCP或Exp值，统称COS（Class of Service）值，这会导致报文的COS值在IP网络和MPLS网络传输过程中都可能发生变化。

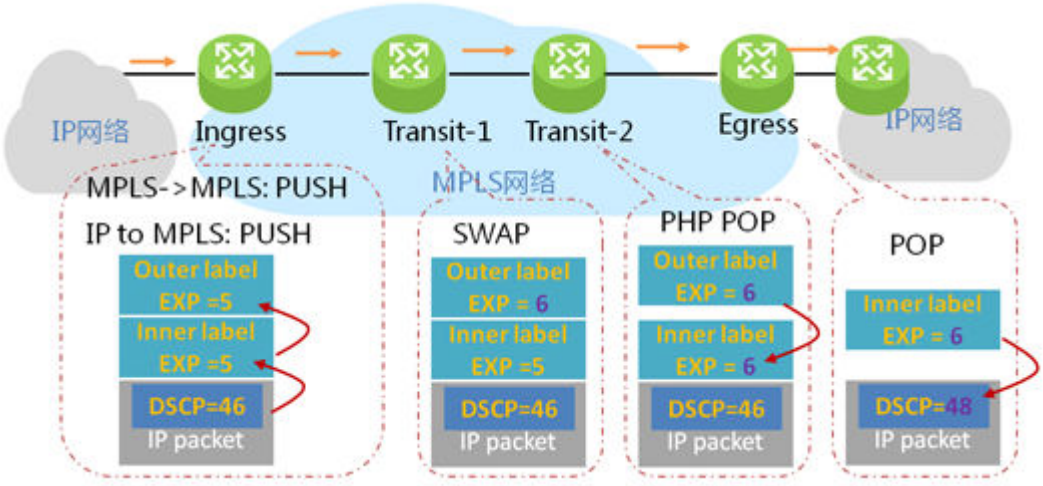
因此，在报文进入MPLS网络或从MPLS网络离开进入IP网络时，运营商需要在MPLS边缘路由器对COS（Class of Service）处理做出选择：是否信任IP/MPLS报文已经携带的COS信息。



RFC3270中定义了三种COS处理模式：Uniform、Pipe和Short Pipe。

Uniform 模式

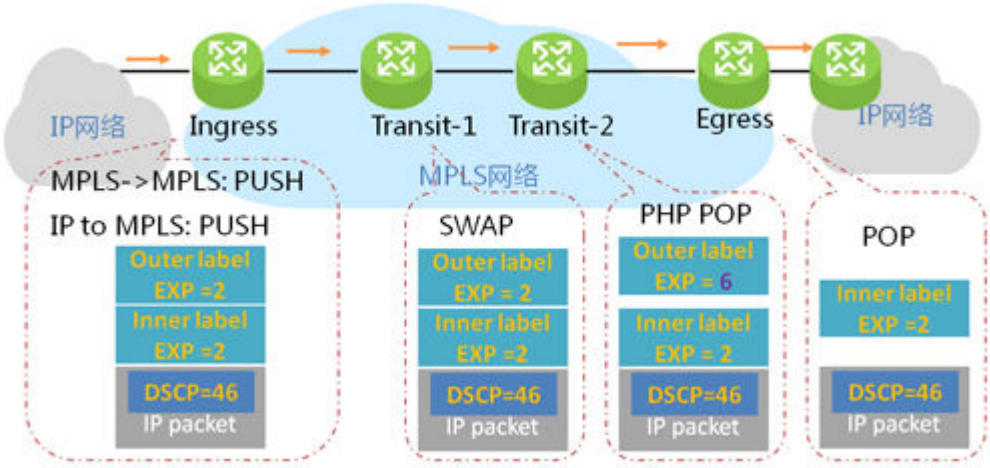
当运营商认为可以完全信任IP网络流量携带过来的COS（IP Preference / DSCP）时，可以采用Uniform模式：MPLS入节点将报文携带上来的COS直接复制到MPLS外层标签的EXP字段中，从而保证在MPLS中给予同样的QoS保证。报文离开MPLS网络时，出口节点再将EXP字段值复制回IP报文的IP Preference/DSCP。



Uniform模式，顾名思义，报文在IP网和MPLS网中的优先级标识是统一的，即报文进入MPLS域和出MPLS域都作优先级映射。但这样做的缺点是，如果报文在MPLS网络中改变了EXP字段的值，会导致报文离开MPLS网络后采用的PHB也发生改变，无法保留用户初始设置的COS值。

Pipe 模式

当运营商认为IP网络流量携带过来的COS不可信任时，可以采用Pipe模式：忽略用户携带的COS，在MPLS域入节点为MPLS外层标签的EXP字段重新赋值，从MPLS入节点到出节点，流量都按照运营商的意愿进行QoS调度，直到流量出MPLS域之后再根据其原来携带的COS值转发。



在Pipe模式中，报文在进入MPLS域时，入节点不拷贝IP Precedence/DSCP到Exp。报文出MPLS域时，出节点也不拷贝Exp到IP Precedence/DSCP。

如果报文在MPLS网络中改变了EXP字段的值，只在MPLS网络中有效。当报文离开MPLS网络后，报文之前携带的COS继续有效。

Short-Pipe 模式

Short Pipe模式是对Pipe模式的改进，报文在MPLS入节点的处理和Pipe模式相同，但在MPLS网络的出节点上，出节点先执行标签弹出操作，再进行QoS调度。即，在MPLS出节点上按照报文之前携带的初始的COS值进行调度，而从报文进入MPLS开始到MPLS倒数第二跳按照运营商的意愿进行QoS调度。

使用Pipe或Short Pipe模式，在运营商内部网络，运营商可以根据自己的自身情况来规划QoS，而不改变客户侧的QoS。

Pipe模式和Short Pipe模式的区别在于：PE上CE侧出方向的QoS调度，Pipe模式运营商用自己的QoS标记，Short Pipe则采用用户的QoS标记。