QOS

目录

概述

在普通的网络中，当用户将数据发向网络设备后，网络设备都是尽最大努力传输数据，直到超出自己的最大负荷为止。当设备达到最大负荷后，如果还有用户发来的数据，那么这些数据将因为网络设备不能提供服务而被丢弃。这样的提供最大化服务的网络被称为尽力而为服务的网络。在尽力而为服务的网络中，所有的数据都被看成是同等重要的，用户的数据有时无法得到保证，所以在某些时候，必须让网络通过放弃传输相对不重要的数据来保证用户的重要数据和传输。因此，就需要在网络中实施 Quality of Service，即 QOS。实施了 QOS 的网络中，可以为特定数据保证带宽，同时也可以限制宽带，可以避免网络拥塞和管理拥塞，甚至可以为数据设置不同的优先级。

QOS 模型

在网络中实施 QOS 时，有三种模型可供参考，这三种模型并不是 QOS 技术，而是用来指导在各种需求下，如何实施 QOS，分为以下三种模型：

Best-Effort Service　尽力而为服务模型

Integrated Service　综合服务模型，简称 Intserv

Differentiated Service　区分服务模型，简称 Diffserv

Best-Effort Service（尽力而为服务模型）

在尽力而为服务模型中，所有网络设备全部都是尽自己最大努力传输数据，所有数据尽管传，不需要得到许可，有多少传多少，任何数据都不能得到保证，延迟也无法预计，所以尽力而为服务模型，其实并没有实施任何 QOS，默认的网络都工作在这种模型下。

Integrated Service（综合服务模型）

在实施了综合服务模型 QOS 的网络中，应用程序在发送数据之前，必须先向网络申请带宽，例如一个视频程序在正常通信下需要 100K 的带宽，那么视频程序在连接之前，必须向网络申请自己需要 100K 的带宽，当网络同意后，视频便可连接，并且将保证能够得到 100K 的带宽，而不会有任何延迟。但是如果某些程序在连接之前没有向网络申请带宽，那么它的流量只能得到尽力而为的服务。由此可见，当某些程序流量需要绝对保证带宽时，可以在综合服务模型的网络中通过申请带宽来保证自己的流量，在申请带宽时，所用到的协议为 Resource Reservation Protocol (RSVP)。在综合服务模型中，重要的数据可以通过申请带宽而得到保证，但是在传送之前必须申请，也需要耗费额外一些时间，在现有的网络中，综合服务模型的 QOS 通常并不被采用。

Differentiated Service（区分服务模型）

在实施了区分服务模型 QOS 的网络中，网络将根据不同数据提供不同服务，因此，所有数据都被分成不同的类别，或者设置为不同的优先级，在网络发生拥塞时，网络总是先保证传输高优先级的数据，从而放弃传输低优先级的数据，但是在网络没有拥塞时，所有数据全部照常传输。在实施区分服务模型的 QOS，就必须先将数据分成不同的类别，或设置成不同的优先级。现在的网络中，实施 QOS 时通常采用区分服务模型。

在网络中，数据从源到目的地，所有的网络设备，包括路由器、交换机、防火墙等,每一台单一的设备对数据包做出的区分服务 QOS 行为称为 per-hop behavior.（PHB 每跳行为），如果数据包从源到目的路径中所有设备都为某类数据执行相同

的区分服务行为，即都执行相同的 QOS 策略,那么这样的 QOS 就被称为 end-to-end QoS(端到端 QOS)。

每一台单一的设备对数据包做出的区分服务 QOS 行为称为 per-hop behavior.（PHB 每跳行为），如果数据包从源到目的路径中所有设备都为某类数据执行相同的区分服务行为，那么就被称为 end-to-end QoS(端到端 QOS)

注：本篇将着重介绍区分服务模型 QOS，之后再介绍综合服务模型 QOS。

QOS 组件

在实施区分服务模型 QOS 时，需要考虑四个 QOS 组件，这些组件相互组合，可以设计出完整的 QOS 策略，而每个组件中，都会有相应的 QOS 技术提供支持，以下是 QOS 四个组件：

分类和标记（classification and marking）

管制和整形（Policing and Shaping）

拥塞管理（Congestion management）

拥塞避免（Congestion avoidance）

分类和标记

要提供区分服务的 QOS，就必须先将数据分为不同的类别，或者将数据设置为不同的优先级。将数据分为不同的类别，称为分类(classification)，分类并不修改原来的数据包。将数据设置为不同的优先级称为标记(marking)，而标记会修改原来的数据包。分类和标记是实施 QOS 的前提，也是基础。

管制和整形

在实施 QOS 策略时，可以将用户的数据限制在特定的带宽，当用户的流量超过

额定带宽时，超过的带宽将不能被传输，只能采取其它方式来处理，如果处理方式为丢弃超出带宽，那么这种行为称为管制(Policing)，如果是将超出的带宽缓存在内存中，等到下一秒再传递，这种行为称为整形（Shaping）。

拥塞管理

当网络发生拥塞后，数据还是要被传递的，正因为接收到的数据远多于自身的传输能力，所以数据被传输时就出现了先后顺序，而依照什么样的方式来传数据，就需要队列的指导，QOS 中的队列定义了数据包被传输的先后顺序。

拥塞避免

当网络发生拥塞后，超出的流量将采取其它方式处理，如果处理方式为管制，那么数据包就会被丢弃，通常情况下，网络设备默认丢弃后到的数据包而传输先到的数据包，这样的丢弃方式称为尾丢弃，但也可以让网络设备在发生拥塞时，先丢低优先级的数据包而传输高优先级的数据包。

并不是所有的 QOS 技术都适合所有网络，边缘路由器和核心路由器操作是不一样的。

比如语音数据，边缘和核心要同时考虑。而通常情况是：

边缘路由器执行：数据包分类和标记

核心路由器执行：拥塞管理，拥塞避免

后面将详细介绍 QOS 四个组件中的各个工具。

MQC（Modular QoS Command-Line）

MQC 就是模块化 QOS 命令行，是配置 QOS 处理数据的一种方式。MQC 可以配

置对特定的数据采取特定的动作，步骤为三步：

定义流量

设置策略

应用策略

定义流量

在使用 MQC 时，只能在命令行下使用，定义流量通过创建 class-map 来匹配特定的数据。

例：

匹配源主机 10.1.1.1 发出的数据

1 创建 ACL 匹配主机 10.1.1.1 发出的数据

Router(config)#access-list 1 permit 10.1.1.1 0.0.0.0

2 创建 class-map，调用 ACL 的数据

Router(config)# class-map match-all ccie

Router(config-cmap)#match access-group 1

说明:class-map 中可以匹配多个数据，当存在多条匹配时，是不是所有条件都需要满足，则靠创建 class-map 时的关键字来判断，关键字 match-all 表示所有条件都要同时满足，默认为 match-all，如果关键字为 match-any，则任一条满足即可。

注：名为 class-default 的 class-map，表示匹配所有数据。

设置策略

当匹配到特定的数据之后，就需要对其设置相应的策略，通过创建 policy-map，然后调用 class-map 匹配到的数据，从而设置相应的策略或动作。

例：

对名为 ccie 的 class-map 所匹配到的数据全部丢弃

Router(config)#policy-map cisco

Router(config-pmap)#class ccie

Router(config-pmap-c)#drop

说明:一个 policy-map 里面可以调用多个 class-map，如果调用 class-default，那么表示之前没有匹配到的流量，全部都会被 class-default 所匹配。

应用策略

当策略设置完毕之后，就需要应用到接口上。

例：

将 policy-map 应用到接口 F0/0 出方向上

Router(config)#interface f0/0

Router(config-if)#service-policy output cisco

多动作 MQC

在配置 MQC 时，可以对匹配到的流量做出多个处理动作

例：

将 class-map ccie 中超出额定带宽的流量设置 IP 优先级为 4DE 为 1，然后再传输

Router(config)#policy-map cisco

Router(config-pmap)#class ccie

Router(config-pmap-c)#police cir percent 10 bc 100 ms

Router(config-pmap-c-police)#conform-action transmit

Router(config-pmap-c-police)#exceed-action set-prec-transmit 4

Router(config-pmap-c-police)#exceed-action set-frde-transmit

说明:当 CIR 设置为百分比时，Bc 则为时间，单位 ms。

令牌桶算法（token bucket algorithm）

在实施 QOS 策略时，可以将用户的数据限制在特定的带宽，当用户的流量超过额定带宽时，超过的带宽将采取其它方式来处理。要衡量流量是否超过额定的带宽，网络设备并不是采用单纯的数字加减法来决定的，也就是说，比如带宽为 100K，而用户发来的流量为 110K，网络设备并不是靠 110K 减去 100K 等于 10K，就认为用户超过流量 10K。网络设备

衡量流量是否超过额定带宽，需要使用令牌桶算法来计算。下面详细介绍令牌桶算法机制：

当网络设备衡量流量是否超过额定带宽时，需要查看令牌桶，而令牌桶中会放置一定数量的令牌，一个令牌允许接口发送或接收 1bit 数据（有时是 1 Byty 数据），当接口通过 1bit 数据后，同时也要从桶中移除一个令牌。当桶里没有令牌的时候，任何流量都被视为超过额定带宽,只有当桶中有令牌时，数据才可以通过接口。令牌桶中的令牌不仅仅可以被移除，同样也可以往里添加，所以为了保证接口随时有数据通过，就必须不停地往桶里加令牌，由此可见，往桶里加令牌的速度，就决定了数据通过接口的速度。因此，我们通过控制往令牌桶里加令牌的速度从而控制用户流量的带宽。而设置的这个用户传输数据的速率被称为承诺信息速率（CIR），通常以秒为单位。比如我们设置用户的带宽为 1000 bit 每秒，只要保证每秒钟往桶里添加 1000 个令牌即可。

例：

将 CIR 设置为 8000 bit/s，那么就必须每秒将 8000 个令牌放入桶中，当接口有数据通过时，就从桶中移除相应的令牌，每通过 1 bit，就从桶中移除 1 个令牌。当桶里没有令牌的时候，任何流量都被视为超出额定带宽，而超出的流量就要采取额外动作。

每秒钟往桶里加的令牌就决定了用户流量的速率，这个速率就是 CIR，但是每秒钟需要往桶里加的令牌总数，并不是一次性加完的，一次性加进的令牌数量被称为 Burst size（Bc），如果 Bc 只是 CIR 的一半，那么很明显每秒钟就需要往桶里加两次令牌，每次加的数量总是 Bc 的数量。

还有就是加令牌的时间，Time interval（Tc），Tc 表示多久该往桶里加一次令牌，而这个时间并不能手工设置，因为这个时间可以靠 CIR 和 Bc 的关系计算得到， Bc/CIR= Tc。

例：

如果 CIR 是 8000,Bc 是 4000，那就是每秒加两次，Tc 就是 4000/8000=0.5，也就是 0.5 秒，即 500 ms。

如果 Bc 设为 2000，那 Tc 就是 2000/8000=0.25，也就是 250 ms。

单速双色

在单速双色的令牌桶算法中，只存在一个令牌桶，并且流量只会出现两种结果，即符合 CIR（conform）和超出 CIR（exceed）。

例：

将 CIR 设置为 8000 bit，每一秒都会往桶里加 8000 个令牌，在一秒钟结束后，没有用完的令牌会被全部清空，由下一秒重新加入。

如

第 1 秒，加入 8000 令牌，用户使用 5000 后，剩余 3000 被清空

第 2 秒，加入 8000 令牌，用户使用 6000 后，剩余 2000 被清空

第 3 秒，加入 8000 令牌，用户使用 8000 后，没有剩余

第 4 秒，加入 8000 令牌，用户使用 7000 后，剩余 1000 被清空

从以上过程可以看出，用户每秒都可以使用 8000 令牌，也就是每秒速度均可达到 8000 bit，而无论上一秒钟是否传过数据，这一秒都可以保持在 8000 bit/s，并且如果每秒流量超过了 8000 后，超过的流量都会采取已经设定的动作。

单速三色

在单速三色的令牌桶算法中，使用两个令牌桶，用户每秒的可用带宽，总是两个桶的令牌之和，第一个桶的令牌机制和单速双色算法没有任何区别，关键在于第二个桶。第二个桶的令牌不能直接加入，只有当一秒钟结束后，第一个桶中存在剩余令牌时，这些剩余令牌就可以从第一个桶中被转移到第二个桶中。但不是第一个桶所有未用令牌都可以放入第二个桶，是有限制的，最大数量被称为 Excess Burst size（Be），由此可见，Be 是不可能超过 CIR 的，因为第一个桶每秒的所有令牌就是 CIR，即使所有令牌全部被移到第二个桶，Be 最多也只能等于 CIR 而不能超过。而 Be 和 Bc 却毫无关系。需要注意的是，在每一秒结束时，如果用户没有将第二个桶的令牌用完，那么第二个桶的令牌也是要全部被清除的，第二个桶中的令牌，都是来自于上一秒第一个桶没用完的令牌。

由于使用了两个桶，所以用户的流量也会出现三种结果：

小于或等于 CIR（也就是符合 CIR） （conform）

大于 CIR 并小于或等于 CIR 与 Be 之和（也就是符合两个桶令牌之和）（exceed）

超过 CIR 与 Be 之和（也就是超过两个桶令牌之和）（violate）

例：

将 CIR 设置为 8000 bit，Be 设置为 2000，每一秒都会往第一个桶里加 8000 个令牌，每一秒结束后，所有第一个桶未使用完的令牌都将放入第二个桶，并且用户每一秒能使用的带宽总是两个桶之和。

如：

	第一个桶令牌数	用户用掉的带宽数	第二个桶令牌数	用户可用带宽总数
第 1 秒	8000	6000	0	8000
第 2 秒	8000	7000	2000	10000
第 3 秒	8000	5000	1000	9000
第 4 秒	8000	9000	2000	10000
第 5 秒	8000	8000	0	8000
第 6 秒	8000	6000	0	8000
第 7 秒	8000	10000	2000	10000

说明：

在第 1 秒时，第一个桶加入 CIR 的数量 8000 个令牌后，第二个桶为空，所以用户可用带宽总数为 8000。用户实际使用了 6000；

在第 2 秒时，第一个桶加入 8000 个令牌后，由于上一秒用户实际使用了 6000，所以第二个桶获得 2000 令牌，此时用户可用带宽为 10000，用户实际用户了 7000；

在第 3 秒时，第一个桶加入 8000 个令牌后，由于上一秒用户实际使用了 7000，所以第二个桶获得 1000 令牌，此时用户可用带宽为 9000，用户实际用户了 5000；

在第 4 秒时，第一个桶加入 8000 个令牌后，由于上一秒用户实际使用了 5000，所以第二个桶获得 2000 令牌，此时用户可用带宽为 10000，用户实际用户了 9000；

在第 5 秒时，第一个桶加入 8000 个令牌后，由于上一秒将第一个桶中令牌用光，所以第二个桶没有获得令牌，此时用户可用带宽为 8000，用户实际用户了 8000；

在第 6 秒时，第一个桶加入 8000 个令牌后，由于上一秒将第一个桶中令牌用光，所以第二个桶没有获得令牌，此时用户可用带宽为 8000，用户实际用户了 6000；

在第 7 秒时，第一个桶加入 8000 个令牌后，由于上一秒用户实际使用了 6000，所

以第二个桶获得 2000 令牌，此时用户可用带宽为 10000，用户实际用户了 10000；

从上面可以看出，第一个桶中的令牌数永远都是 CIR 的数量，而第二个令牌桶只能在上一秒第一个桶存在没有用完的令牌的情况下，才能够获得令牌，但获得令牌的最大数量不能超过 Be。用户的流量也可以出现三种结果：

即小于或等于 CIR,即小于 8000，如 6000

大于 CIE 并小于或等于 CIR+Be，如 9000

大于两个桶之后，如 11000

要使用户在某一秒的速度能够达到CIR+Be,唯一的办法是用户在上一秒钟以低于CIR的速度传输。因此，用户不可能每一秒都以 CIR+Be 的速度传输。

双速三色

在单速三色的令牌桶算法中，用户若想要在某一秒以 CIR+Be 的速度传输，只能在上一秒钟以低于 CIR 的速度传输。因此，用户不可能每一秒都以 CIR+Be 的速度传输。而在双速三色的令牌桶算法中，同样使用两个令牌桶，然而这两个桶是相互独立的，并不会将第一个桶未用的令牌放入第二个桶。第一个桶与以往的算法相同，也就是每秒都有 CIR 的数量，而第二个桶可以直接设置为 CIR+Be 之和，称为 PIR，也就是说第二个桶总是比第一个桶要大，用户的流量总是以第二个桶的大小传输，而不用像单速三色的令牌桶算法中，需要在上一秒钟以低于 CIR 的速度传输。当用户的数据通过接口时，总是先检查第二个桶的最大速率，即 PIR，如果超出则采取动作，如果未超出，再检查是是否符合第一个桶的 CIR，如果超出 CIR，则采取相应动作，如果未超过，则正常传输。

虽然在双速三色的令牌桶算法中，直接设置两个速率，然而，用户可以直接以 CIR+Be 之和的速率进行传输，此外，还可以判断出三种结果。

分类和标记（Classification and Marking）

要实施区分服务的 QOS，就必须先将数据分为不同的类别，或者将数据设置为不同的优先级。将数据分为不同的类别，称为分类(classification)，分类并不修改原来的数据包。将数据设置为不同的优先级称为标记(marking)，而标记会修改原来的数据包。分类和标记是实施 QOS 的前提，也是基础。要正确对数据包进行分类和标记，需要了解数据包的某些特征，最重要的就是数据包的包头：

分类时，可以使用下列任何一种标准来识别特定的流量：

OSI 参考模式第一层特征：

物理接口

子接口

PVC

OSI 参考模式第二层特征：

MAC 地址

802.1Q 或 802.1P 服务类别（COS）

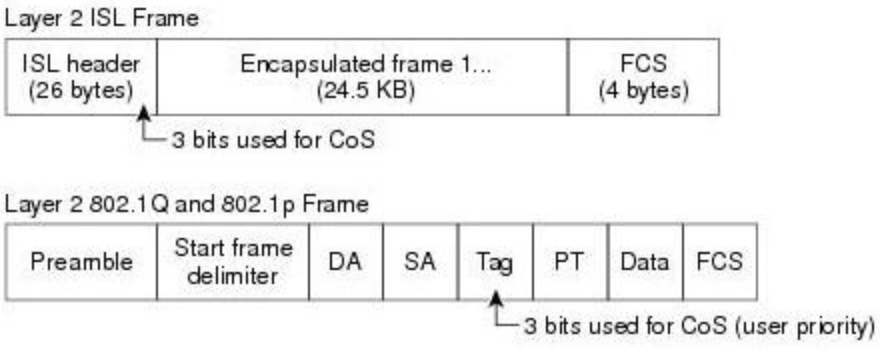
VLAN 标识

帧中继可丢弃指标符（DE）位

802.1Q 或 802.1P 服务类别（COS）

在以太网中，穿越 trunk 的数据将会被额外封装，如封装成 IEEE 802.1Q 或者 ISL 帧，因此，可以利用 trunk 帧头的额外部分来标记，这些字段被称为 class of service

(CoS)，如下图：



Inter-Switch Link (ISL)帧中，预留有 1-byte 的 IEEE 802.1p 字段，其中有 3 bits 可以标记 CoS。

IEEE 802.1Q 帧中，预留有 2-byte 字段，其中同样只有 3 bits 可以标记 CoS，而 IEEE 802.1Q 帧中，native VLAN 是不能被标记的，因为没有额外封装。

CoS 中由于只有 3 bit 可以标记，所以只能标记出 0-7 共 8 类数据，默认标为 0，然而 6 和 7 是被保留的，因此只有 0-5 共 6 类可供用户标记使用。

帧中继可丢弃指标符（DE）位

在帧中继数据包中，有额外的一个字段可以用来指示该数据包的优先级，这个字段被称为可丢弃指标符 Discard eligible (DE)位，默认为 0，设置为 1 表示该数据不重要而优先被丢弃。

注：二层帧头会因为物理传输介质的改变而改变，所以二层标记并不具备端到端的意义。

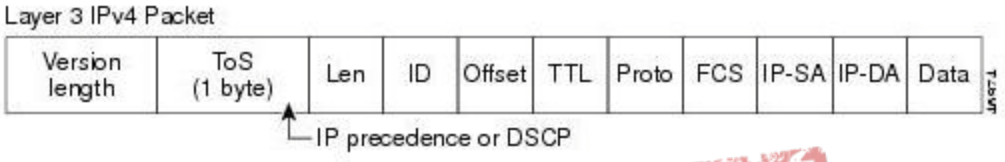
OSI 参考模式第三层特征：

IP Precedence （IP 优先级）

DiffServ 代码点（DSCP）

源 IP 地址或目的 IP 地址

在第三层 IP 数据包头，其中 ToS 字段预留 1-Byte 可供标记使用，如下图：



IP 优先级同 CoS 一样只使用 3 bit 标记，范围是 0-7 共 8 类数据，默认标为 0，然而 6 和 7 是被保留的，只有 0-5 共 6 类可供用户标记使用。

5 建议用于语音

4 由视频会议和流式视频共享

3 建议用于呼叫信令

而 DSCP 则使用 6 bit 标记，范围是 0-63 共 64 类数据。

由于 IP 数据包包头 ToS 字段只有 8 bit 可用，IP 优先级标记时使用 3bit，而 DSCP 标记时使用 6 bit， 要同时标记 IP 优先级和 DSCP 需要使用 9 bit，所以 ToS 中，只能标记 IP 优先级和 DSCP 其中一种。

注： IP 优先级 和 DSCP 被广泛用于标记选项，具有端到端的意义。

IP 优先级 、DSCP、CoS 之间的值可以互相映射。

OSI 参考模式第四层特征：

TCP 或 UDP 端口号

OSI 参考模式第三层特征：

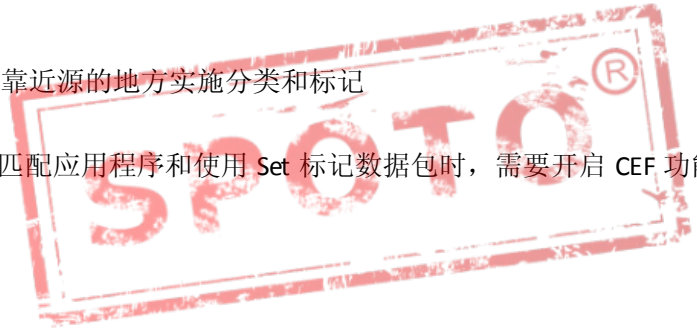
URL

基于网络的应用识别 network-based application recognition (NBAR)，也就是匹配应用程序，通过使用数据包描述语言模块 PDLM 来识别。

注：

建议在尽可能靠近源的地方实施分类和标记

在使用 NBAR 匹配应用程序和使用 Set 标记数据包时，需要开启 CEF 功能



流（Flow）

当数据包的源 IP，目的 IP，协议，端口号，以及会话的 socket 全部相同时，这样的数据被认为是同一个流（flow），同一个流，通常应该得到相同的 QOS 服务。

比如使用某一台主机访问某网站的网页和视频，网页数据和视频数据的源 IP、目的 IP、协议即使都是相同的，但是端口号会不同，即使端口号是相同的，但会话的 socket 也绝不相同，因此，网页的数据因为五个参数都相同而被归为同一个流，而视频的数据也因为五个参数都相同而被归为另一个流，每个流便可设置不同的 QOS 策略。

管制和整形（Policing and Shaping）

在实施 QOS 策略时，可以将用户的数据限制在特定的带宽，当用户的流量超过额定带宽时，超过的带宽将不能被传输，只能采取其它方式来处理，如果处理方式为丢弃超出带宽，那么这种行为称为管制(Policing)，如果是将超出的带宽缓存在内

存中，等到下一秒再传递，这种行为称为整形（Shaping）。

当使用管制时，用户的数据包超过额定带宽后，将会被丢弃，当使用的传输协议为 TCP 时，被丢弃的数据会重传，而传输协议为 UDP 时，用户是无法知道数据被丢弃的，所以管制需要小心使用。

当使用整形时，用户的数据包超过额定带宽后，是不会被丢弃的，而是缓存到内存中等到下一秒再传输，整形使超额的数据能够从接口平滑地输出，但并不表示整形就永远不会丢包。

整形分为三种：

Generic Traffic Shaping (GTS)

Frame Relay Traffic Shaping (FRTS)

Class-Based Shaping

Generic Traffic Shaping (GTS)为通用流量整形，适用于任何接口。

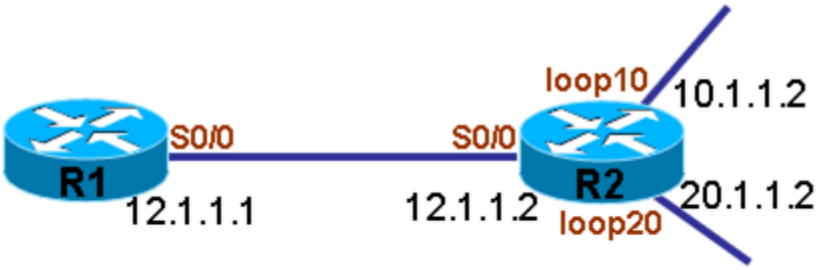
Frame Relay Traffic Shaping (FRTS) 为帧中继流量整形，是专门针对帧中继接口使用的整形技术，通过帧中继专有技术 map-class 来实现。

Class-Based Shaping 是基于类的整形，可以将 GTS 嵌入 FRTS。

注：在整形中，Bc 建议设置为 CIR 的百分之一，即 Bc/CIR=1/100。

配置管制

要对特定的流量进行管制，从而限制其可用带宽，方法为通过 MQC 的形式，先匹配特定的流量，再使用令牌桶算法将流量限制在额定的带宽，对于超额的流量再做其它处理。



说明:以上图为例，将 R1 到目标网络 10.1.1.0/24 的流量管制到 CIR 为 8000 bit，超过的流量则丢弃，R1 到目标网络 20.1.1.0/24 的流量正常通过。

1. 匹配 R1 到目标网络 10.1.1.0/24 的流量

（1）通过 ACL 匹配到网络 10.1.1.0/24 的流量

r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255

（2）class-map 调用 ACL 中的流量

r1(config)#class-map net10

r1(config-cmap)#match access-group 100

2. 管制到目标网络 10.1.1.0/24 的流量

(1)在 policy-map 中调用 class-map 的流量并管制带宽

r1(config)#policy-map band

r1(config-pmap)#class net10

r1(config-pmap-c)# police cir 8000 bc 1000 be 1000 conform-action transmit exceed-action drop

说明:配置中 CIR 为 8000 bit，超过的流量为 drop，关键字 police 则是开启管制

（Policing）。

3．应用策略到接口

（1）将 policy-map 应用到 R1 的 S0/0 出方向

r1(config)#int s0/0

r1(config-if)#service-policy output band

4．测试效果

（1）测试 R1 到目标网络 20.1.1.0/24 的流量

r1#ping 20.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!

Success rate is 100 percent (20/20), round-trip min/avg/max = 404/406/409 ms

r1#

说明:因为并没有对 R1 到目标网络 20.1.1.0/24 的流量进行管制，所以当数据包每个以 800 字节通过时，一切正常，并且速度正常。

（2）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!.!.!.!.!.!.!.!.!.!.

Success rate is 50 percent (10/20), round-trip min/avg/max = 405/405/409 ms

r1#

说明:因为 R1 到目标网络 10.1.1.0/24 的流量被管制为 8000bit 每秒，所以当数据包每个以 800 字节通过时，流量超出额定带宽，从而出现了超额流量被均衡地丢弃。

（3）查看 policy-map 参数

r1#sh policy-map interface s0/0

Serial0/0

Service-policy output: band

Class-map: net10 (match-all)

55 packets, 49020 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: access-group 100

police:

cir 8000 bps, bc 1000 bytes

conformed 20 packets, 16080 bytes; actions:

transmit

exceeded 35 packets, 32940 bytes; actions:

drop

conformed 0 bps, exceed 0 bps

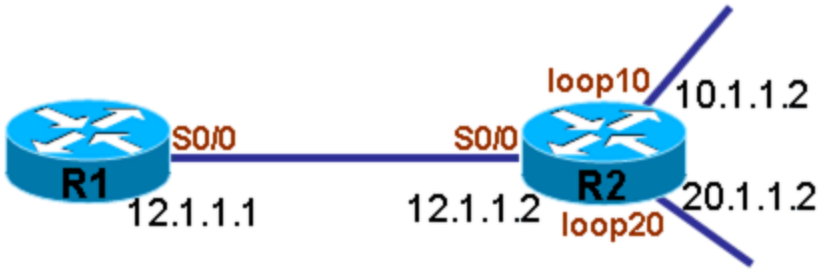
```
  Class-map: class-default (match-any)
    237 packets, 86407 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
r1#
```

说明:从输出中可以看出超额后丢弃的数据包个数。

配置整形

要对特定的流量进行整形，从而限制其可用带宽，方法为先匹配特定的流量，再使用令牌桶算法将流量限制在额定的带宽，但是对于超额的流量不需要做其它处理，因为这些超额的流量是需要被缓存的。下面分别介绍三种整形技术的配置方法：

Generic Traffic Shaping (GTS)



说明:以上图为例，将 R1 到目标网络 10.1.1.0/24 的流量整形到 CIR 为 8000 bit，而超额的流量不需要做其它处理，这些超额的流量默认被缓存；R1 到目标网络 20.1.1.0/24 的流量正常通过。

1. 匹配 R1 到目标网络 10.1.1.0/24 的流量

（1）通过 ACL 匹配到网络 10.1.1.0/24 的流量

```
r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255
```

（2）class-map 调用 ACL 中的流量

```
r1(config)#class-map net10
r1(config-cmap)#match access-group 100
```

2. 整形到目标网络 10.1.1.0/24 的流量

(1)在 policy-map 中调用 class-map 的流量并整形带宽

```
r1(config)#policy-map SSS
r1(config-pmap)#class net10
r1(config-pmap-c)#shape average 8000 1000 0
```


说明:整形的 CIR 为 8000bit,虽然建议 Bc 为 CIR 的 1/100，但 Bc 请不要小于 1000，否则整形不会生效，而 Be 可以为 0。

3. 应用策略到接口

（1）将 policy-map 应用到 R1 的 S0/0 出方向

```
r1(config)#int s0/0
r1(config-if)#service-policy output SSS
```

4. 测试效果

（1）测试 R1 到目标网络 20.1.1.0/24 的流量

```
r1#ping 20.1.1.2 size 800 repeat 20
```

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!

Success rate is 100 percent (20/20), round-trip min/avg/max = 404/406/409 ms

r1#

说明:因为并没有对 R1 到目标网络 20.1.1.0/24 的流量进行整形，所以当数据包每个以 800 字节通过时，一切正常，并且速度正常。

（2）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!

Success rate is 100 percent (20/20), round-trip min/avg/max = 405/406/409 ms

r1#

说明:因为 R1 到目标网络 10.1.1.0/24 的流量被整形为 8000bit 每秒，所以当数据包每个以 800 字节通过时，并没有丢包，但是速度却不能超过 8000bit 每秒，此效果在文本下无法看见，需要自己在实验时查看。

（3）查看 policy-map 参数

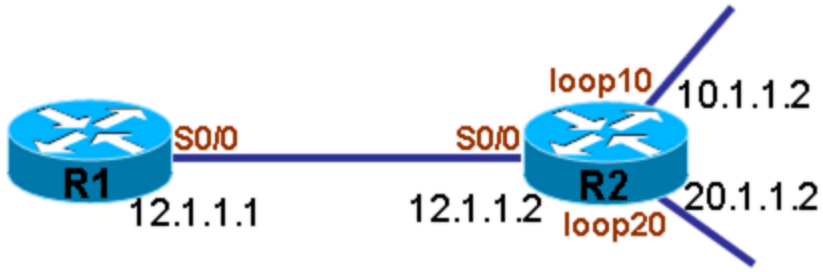
r1#show policy-map interface

Serial0/0

```
  Service-policy output: SSS

    Class-map: net10 (match-all)

      20 packets, 16080 bytes

      5 minute offered rate 1000 bps, drop rate 0 bps

      Match: access-group 100

      Traffic Shaping

          Target/Average   Byte   Sustain   Excess    Interval  Increment

            Rate           Limit  bits/int  bits/int  (ms)      (bytes)

          8000/8000        125    1000      0         125       125

      Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping

      Active Depth                       Delayed   Delayed   Active

      -      0        20        16080     19        15276     no


    Class-map: class-default (match-any)

      29 packets, 16197 bytes

      5 minute offered rate 0 bps, drop rate 0 bps

      Match: any

r1#
```

说明:从输出中可以看出被整形的数据包个数以及其它一些参数。

Frame Relay Traffic Shaping (FRTS)



说明:以上图为例，将 R1 的出口流量整形到 CIR 为 8000 bit，而超额的流量不需要做其它处理，这些超额的流量默认被缓存。

1. 配置 map-class 实现 FRTS

（1）配置 map-class

r1(config)#map-class frame-relay TTT

r1(config-map-class)#frame-relay cir 8000

r1(config-map-class)#frame-relay bc 1000

r1(config-map-class)#frame-relay be 0

说明:FRTS 的工具 map-class 不能针对特定流量整形，而只能对接口所有流量整形。

2. 应用 FRTS

（1）在帧中继接口上开启整形

r1(config)#int s0/0

r1(config-if)#frame-relay traffic-shaping

（2）应用 map-class

r1(config-if)#frame-relay interface-dlci 102

r1(config-fr-dlci)#class TTT

说明:map-class 可以单独应用到某条 PVC 上，也可以应用在整个接口上。

3. 测试效果

（1）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

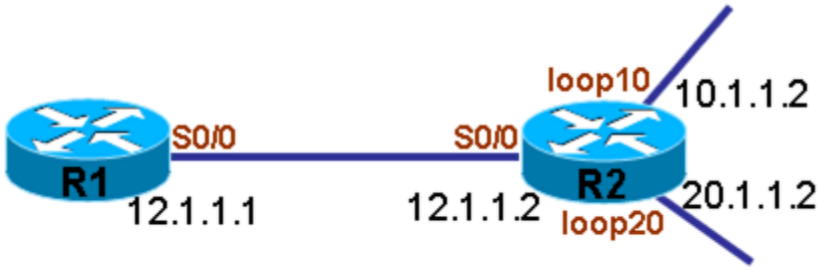
!!!!!!!!!!!!!!!!!!!!

Success rate is 100 percent (20/20), round-trip min/avg/max = 405/406/409 ms

r1#

说明:因为 R1 的出口流量被整形到 CIR 为 8000bit 每秒，所以当数据包每个以 800 字节通过时，并没有丢包，但是速度却不能超过 8000bit 每秒，此效果在文本下无法看见，需要自己在实验时查看。

Class-Based Shaping



说明:以上图为例，将 R1 的出口流量整形到 CIR 为 8000 bit，而超额的流量不需要做其它处理，这些超额的流量默认被缓存。Class-Based Shaping 和 FRTS 一样只能对所有流量整形。class-default

1. 配置 GTS

（1）policy-map 中调用 class-map 的流量并整形带宽

r1(config)#policy-map ccc

r1(config-pmap)#class class-default

r1(config-pmap-c)#shape average 8000 1000 0

说明:Class-Based Shaping 只支持对 class-default 的整形，即对所有流量整形。

2. 配置 Class-Based Shaping

（1）在 map-class 中调用 GTS

r1(config)#map-class frame-relay FFF

r1(config-map-class)#service-policy output ccc

3. 应用 Class-Based Shaping

（1）在帧中继接口上应用 Class-Based Shaping

r1(config)#int s0/0

r1(config-if)#frame-relay class FFF

说明:在应用 Class-Based Shaping 时，接口上必须禁用 frame-relay traffic shaping。

4. 测试效果

（1）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!

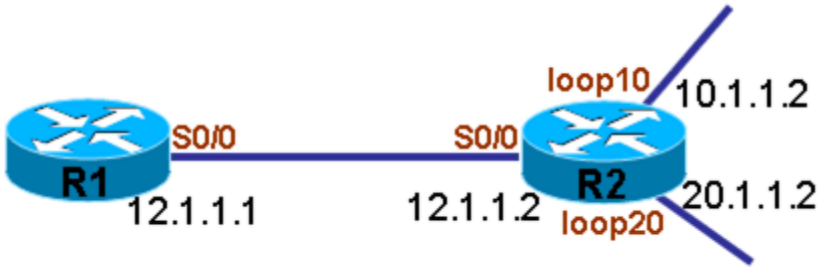
Success rate is 100 percent (20/20), round-trip min/avg/max = 405/406/409 ms

r1#

说明:因为 R1 的出口流量被整形到 CIR 为 8000bit 每秒，所以当数据包每个以 800 字节通过时，并没有丢包，但是速度却不能超过 8000bit 每秒，此效果在文本下无法看见，需要自己在实验时查看。

接口直接开启整形

除了以上的整形方法之外，接口上可以直接配置流理整形，这也是对所有流量生效



说明:以上图为例，将 R1 的出口流量整形到 CIR 为 8000 bit。

1. 开启接口流量整形

（1）在 R1 的 S0/0 上开启整形

r1(config)#int s0/0

r1(config-if)#traffic-shape rate 8000 1000 0

2. 测试效果

（1）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 20

Type escape sequence to abort.

Sending 20, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!

Success rate is 100 percent (20/20), round-trip min/avg/max = 405/406/409 ms

r1#

说明:因为 R1 的出口流量被整形到 CIR 为 8000bit 每秒，所以当数据包每个以 800 字节通过时，并没有丢包，但是速度却不能超过 8000bit 每秒，此效果在文本下无法看见，需要自己在实验时查看。

（2）查看接口整形

r1#sh traffic-shape statistics

I/F	Acc. Queue List Depth	Packets	Bytes	Packets Delayed	Bytes Delayed	Shaping Active
Se0/0	0	20	16080	19	15276	no

r1#

说明:可以看到被整形的数据包个数。

Committed access rate (CAR)承诺访问速率

CAR 是一个 Cisco 工具，有两种功能，可以对数据包进行标记，也可以对流量实现管制。

既然 CAR 可以对数据标记，也就是可以改变数据包的包头信息，CAR 同样可以通过令牌桶算法对流量进行管制，将流量限制在额定的带宽。

在配置 CAR 时，流量可以是进和出任意方向的，因为 CAR 可以标记，所以在配置 CAR 时，必须开启 CEF，并且可以配置在主接口和子接口上，但 Fast EtherChannel，tunnel,，PRI 接口是不能配的，以及不支持 CEF 的接口也不能配置 CAR。

CAR 可以基于接口方向，基于 IP 优先级，以及基于 ACL 和 MAC 地址生效，但是需要注意 MAC 地址是二层地址，经过二层转换之后，MAC 地址信息将会丢失。

一个接口上可以配置多条 CAR。

对于匹配的流量，CAR 的动作有：

continue （就是检测下一条 CAR）

drop （丢弃匹配的包）

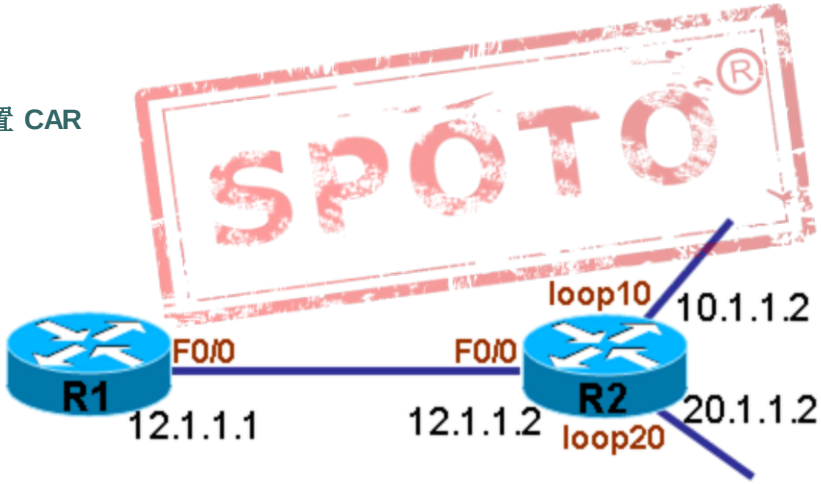
set-prec-continue new-prec （设置新的 IP 优先级，然后检测下一条 CAR）

set-prec-transmit new-prec （设置新的 IPP 然后传输）

transmit （直接传输）

注：以上信息，根据 IOS 不同，会有所增减。

配置 CAR



配置基于接口的 CAR

说明:配置 R1，将所有从接口 F0/0 出去的流量全部管制到 CIR 8000，超过的流量则丢弃

1．在接口 F0/0 上配置 CAR 管制所有流量

（1）在 R1 的接口 F0/0 上配置 CAR

r1(config)#int f0/0

r1(config-if)#rate-limit output 8000 1500 2000 conform-action transmit exceed-action drop

2. 测试效果

（1）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 10

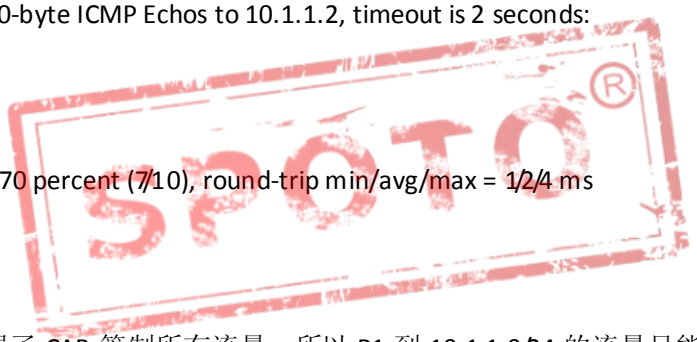
Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!.!!.!!!.

Success rate is 70 percent (7/10), round-trip min/avg/max = 1/2/4 ms

r1#



说明:因为配置了 CAR 管制所有流量，所以 R1 到 10.1.1.0/24 的流量只能以 8000 bit 每秒的速度传输，超过的则被丢弃。

（2）测试 R1 到目标网络 20.1.1.0/24 的流量

r1#ping 20.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!.!!.!!!

Success rate is 80 percent (8/10), round-trip min/avg/max = 1/2/4 ms

r1#

说明:因为配置了 CAR 管制所有流量，所以 R1 到 20.1.1.0/24 的流量只能以 8000 bit 每秒的速度传输，超过的则被丢弃。

（3）查看接口 CAR 信息

```
r1#sh interfaces rate-limit

FastEthernet0/0
  Output
    matches: all traffic
      params:  8000 bps, 1500 limit, 2000 extended limit
      conformed 47 packets, 38258 bytes; action: transmit
      exceeded 13 packets, 10582 bytes; action: drop
      last packet: 17529ms ago, current burst: 1668 bytes
      last cleared 00:01:46 ago, conformed 2000 bps, exceeded 0 bps
r1#
```

说明:可以看到接口上被 CAR 匹配到数据的状况。

配置基于 ACL 的 CAR

说明:配置 R1，将所有去往 10.1.1.0/24 的流量全部管制到 CIR 8000，超过的流量则丢弃

1. 配置 ACL

（1）使用 ACL 匹配去往 10.1.1.0/24 的流量

r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255

2. 配置 CAR

（1）在 R1 的接口 F0/0 上配置 CAR

r1(config)#int f0/0

r1(config-if)#rate-limit output access-group 100 8000 1500 2000 conform-action transmit exceed-action drop

3. 测试效果

（1）测试 R1 到目标网络 20.1.1.0/24 的流量

r1#ping 20.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!!!!!!!!

Success rate is 100 percent (10/10), round-trip min/avg/max = 1/3/4 ms

r1#

说明:因为 CAR 不对去往 20.1.1.0/24 的流量进行管制，所以去往 20.1.1.0/24 的流量正常通过。

（2）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!.!.!!!

Success rate is 80 percent (8/10), round-trip min/avg/max = 1/2/4 ms

r1#

说明:因为 CAR 对去往 10.1.1.0/24 的流量进行管制，所以去往 10.1.1.0/24 的流量只能以 8000bit 每秒的速度通过，超过的流量被丢弃。

（3）查看接口 CAR 信息

r1#sh interfaces rate-limit

FastEthernet0/0

Output

matches: access-group 100

params: 8000 bps, 1500 limit, 2000 extended limit

conformed 8 packets, 6512 bytes; action: transmit

exceeded 2 packets, 1628 bytes; action: drop

last packet: 16483ms ago, current burst: 1672 bytes

last cleared 00:01:25 ago, conformed 0 bps, exceeded 0 bps

r1#

说明:可以看到接口上被 CAR 匹配到数据的状况。

配置基于 DSCP 的 CAR

说明:R1 将去往 10.1.1.0/24 的流量的 DSCP 值设置为 5，让对方路由器 R2 根据 DSCP 值设置 CAR，将 DSCP 为 5 的流量全部丢弃，其它正常通过。

1. 配置 ACL

（1）使用 ACL 匹配去往 10.1.1.0/24 的流量

```
r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255
```

2. 配置 CAR

（1）在 R1 的接口 F0/0 上配置 CAR

```
r1(config)#int f0/0

r1(config-if)#rate-limit output access-group 100 8000 1500 2000 conform-action set-dscp-transmit 5 exceed-action set-dscp-transmit 5
```

说明:CAR 将所有符合与超出的流量的 DSCP 值全部设置成 5 之后再发出去。

3. 在 R2 上配置 CAR

（1）在 R1 的接口 F0/0 上配置 CAR 丢弃 DSCP 值为 5 的流量

```
r2(config)#int f0/0

r2(config-if)#rate-limit input dscp 5 8000 1500 2000 conform-action drop exceed-action drop
```

说明:CAR 将 DSCP 值为 5 的流量，无论是符合还是超出都全部丢弃。

4. 测试效果

（1）测试 R1 到目标网络 20.1.1.0/24 的流量

```
r1#ping 20.1.1.2 size 800 repeat 10
```

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!!!!!!!!

Success rate is 100 percent (10/10), round-trip min/avg/max = 1/3/4 ms

r1#

说明:因为 CAR 只丢弃 DSCP 值为 5 的流量，而去往 20.1.1.0/24 的流量的 DSCP 值并没有被设置成 5，所以正常通过。

（2）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

……

Success rate is 0 percent (0/10)

r1#

说明:因为 CAR 只丢弃 DSCP 值为 5 的流量，而去往 10.1.1.0/24 的流量的 DSCP 值全部被设置成 5，所以全部被丢弃。

（3）查看接口 CAR 信息

r2#sh interfaces rate-limit

FastEthernet0/0

Input

matches: dscp 5

params: 8000 bps, 1500 limit, 2000 extended limit

conformed 10 packets, 8140 bytes; action: drop

exceeded 0 packets, 0 bytes; action: drop

last packet: 169518ms ago, current burst: 0 bytes

last cleared 00:03:33 ago, conformed 0 bps, exceeded 0 bps

r2#

说明:可以看到接口上被 CAR 匹配到数据的状况。

配置基于 MAC 地址的 CAR

说明:配置 R1，将源 MAC 地址为 R2 接口 F0/0 的数据包全部丢弃。

1. 配置匹配 MAC 地址的 ACL

（1）在 R2 上查看接口 F0/0 的 MAC 地址

r2#sh int f0/0

FastEthernet0/0 is up, line protocol is up

Hardware is AmdFE, address is 0013.1a2f.1200 (bia 0013.1a2f.1200)

Internet address is 12.1.1.2/24

说明:R2 上接口 F0/0 的 MAC 地址为 0013.1a2f.1200。

（2）在 R1 上配置匹配 R2 接口 F0/0 的 MAC 地址的 ACL

r1(config)#access-list rate-limit 100 0013.1a2f.1200

2. 配置 CAR

（1）配置 R1，将源 MAC 地址为 R2 接口 F0/0 的数据包全部丢弃

r1(config)#int f0/0

r1(config-if)#rate-limit input access-group rate-limit 100 8000 1500 2000 conform-action drop exceed-action drop

3. 测试效果

（1）测试 R2 去往 R1 的流量

r2#ping 12.1.1.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#

说明:因为 R2 去往 R1 的数据包的源 MAC 地址，即为接口 F0/0 的 MAC 地址，所以流量全被丢弃。

（2）测试 10.1.1.0/24 去往 R1 的流量

r2#ping 12.1.1.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:

.....

Success rate is 0 percent (0/5)

r2#

说明:因为 10.1.1.0/24 去往 R1 的数据包的源 MAC 地址，将被 R2 改为接口 F0/0 的 MAC 地址，所以流量全被丢弃。

（3）查看接口 CAR 信息

r1#sh interfaces rate-limit

FastEthernet0/0

Input

matches: access-group rate-limit 100

params: 8000 bps, 1500 limit, 2000 extended limit

conformed 15 packets, 1710 bytes; action: drop

exceeded 0 packets, 0 bytes; action: drop

last packet: 7973ms ago, current burst: 0 bytes

last cleared 00:02:28 ago, conformed 0 bps, exceeded 0 bps

r1#

说明:可以看到接口上被 CAR 匹配到数据的状况。

拥塞管理（Congestion management）

当网络发生拥塞后，数据还是要被传递的，正因为接收到的数据远多于自身的传输能力，所以数据被传输时就出现了先后顺序，而规定数据按什么样的顺序来传输，这就是拥塞管理。也只有当拥塞发生时，拥塞管理才会生效，对超额的数据流，使用队列算法来决定如何将数据发出，而每种队列技术都指定为解决特定的网络流量问题和获得特定的性能。队列技术需要依懒已经做好的分类和标记，因为队列技术需要根据数据包的不同特征做出不同处理。

一个接口只能使用一个队列技术，需要了解的队列技术有：

FIFO queuing

Priority queuing (PQ)

Custom queuing (CQ)

Weighted fair queuing (WFQ)

Class-based WFQ (CBWFQ)

Low Latency Queuing (LLQ)

IP RTP Priority

下面分别对不同的队列技术进行详细介绍

FIFO Queuing （First In First Out Queuing）

FIFO 队列为先进先出队列，FIFO 队列不对数据包进行分类，当数据包到达接口后，数据包按照到达接口的先后顺序通过接口，数据包没有优先级之分，即使接口发生了拥塞，数据包是先到的就先通过，后到的就后通过。

某些接口是默认开启 FIFO 队列的，不需要手工配置。

速率大于 E1 (2.048 Mbps)的接口全部默认开启 FIFO 队列。

查看接口队列：

```
r1#show interfaces f0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is 0013.1a85.d160 (bia 0013.1a85.d160)
  Internet address is 12.1.1.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:07, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
 Queueing strategy: fifo
  Output queue: 0/40 (size/max)
```


说明:因为接口 F0/0 的速率为 100000 Kbit，大于 E1 (2.048 Mbps)，所以默认队列为 FIFO。

Priority Queuing （PQ）

PQ 被称为优先级队列，是因为 PQ 在发生拥塞时，只传优先级最高的数据，只有当优先级最高的数据全部传完之后，才会传次优先级的数据。PQ 中有 4 个队列，分别是 high，medium，normal，low ，可根据数据包的 IP 优先级或 DSCP 等标识将数据包分配到各个队列中，在发生拥塞时，PQ 先传 high 中的数据，直到全部传完之后，才会传 medium 中的数据，同样只有 medium 中的数据传完之后，才会传 normal 中的数据，最后等前面三个队列的所有数据都传完之后，才轮到 low 中的数据。由此可见，如果高优先级的队列没有传送完毕，低优先级的数据将永远不会传递，造成两级分化，使较低优先级的数据转发困难。

虽然 PQ 只有在高优先级队列数据包全部传完的情况下，才会传下一个队列，但是可以限制每个队列一次性传输的最大数据包个数，当某个队列传输的数据包达到最大数量之后，无论是否还有数据包，都必须传递下一个队列。

配置 PQ

说明:在配置时，可以根据数据包的协议，以及 ACL 等技术将特定流量放入特定队列，除了明确将特定的流量放入某个队列之后，还可以将默认没有匹配的数据统统放入默认的队列。

1．配置 PQ 将特定流量放入特定队列

（1）将源 IP 为 10.1.1.0/24 的数据放入队列 high 中

r1(config)#access-list 10 permit 10.1.1.0 0.0.0.255

r1(config)#priority-list 1 protocol ip high list 10

（2）将源 IP 为 20.1.1.0/24 的数据放入队列 medium 中

r1(config)#access-list 20 permit 20.1.1.0 0.0.0.255

r1(config)#priority-list 1 protocol ip medium list 20

（3）将端口号为 TCP 23 的数据放入队列 normal 中

r1(config)#priority-list 1 protocol ip normal tcp 23

（4）默认其它数据都放入队列 low 中

r1(config)#priority-list 1 default low

2. 限制每个队列的最大数据包个数

（1）限制 high 为 400，medium 为 300，normal 为 200，low 为 100，默认分别为 20,40,60,80。

r1(config)#priority-list 1 queue-limit 400 300 200 100

3. 应用 PQ 到接口

（1）将 PQ 应用到接口 F0/0

r1(config)#int f0/0

r1(config-if)#priority-group 1 ?

4. 查看 PQ

（1）查看接口 F0/0 上的队列情况

r1#show interfaces f0/0

FastEthernet0/0 is up, line protocol is up

  Hardware is AmdFE, address is 0013.1a85.d160 (bia 0013.1a85.d160)

  Internet address is 12.1.1.1/24

  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,

    reliability 255/255, txload 1/255, rxload 1/255

  Encapsulation ARPA, loopback not set

  Keepalive set (10 sec)

  Full-duplex, 100Mb/s, 100BaseTX/FX

ARP type: ARPA, ARP Timeout 04:00:00

Last input 00:00:00, output 00:00:07, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

Queueing strategy: priority-list 1

Output queue (queue priority: size/max/drops):

high: 0/400/0, medium: 0/300/0, normal: 0/200/0, low: 0/100/0

说明:可以看到接口 F0/0 上应用的队列为 PQ。

（2）查看 PQ 参数

r1#sh queueing priority

Current DLCI priority queue configuration:

Current priority queue configuration:

List Queue Args

1 low default

1 high protocol ip list 10

1 medium protocol ip list 20

1 normal protocol ip tcp port telnet

1 high limit 400

1 medium limit 300

1 normal limit 200

1 low limit 100

r1#

Custom queuing (CQ)

CQ 中有 1 到 16 共 16 个队列轮循，每个队列可以限制可传的数据包总数，但实时数据不能得到保证。将数据包分配到 CQ 的各个队列中，当网络发生拥塞时，CQ 先传第 1 个队列中的数据，当传到额定的数据包个数后，就接下去传第 2 个队列中的数据，同样是传到额定的数据包个数后，再传下一个队列，以此类推荐，直到传到第 16 个队列后，再回过去传第一个队列。CQ 除了 1 到 16 个队列外，还有一个 0 号队列，但是 0 号队列是超级优先队列，路由器总是先把 0 号队列中的数据发送完后才处理 1 到 16 号队列中的数据包，所以 0 号队列一般作为系统队列，许多 IOS 不支持手工将指定数据分配到 0 号队列。在配置 1 到 16 号队列时，用户可以配置每个队列同一时间可以占用接口带宽的比例，相当于限速。

配置 CQ

说明:在配置时，可以像配置 PQ 一样，可根据数据包的协议，以及 ACL 等技术将特定流量放入特定队列，除了明确将特定的流量放入某个队列之后，还可以将默认没有匹配的数据统统放入默认的队列。

1. 配置 CQ 将特定流量放入特定队列

（1）将源 IP 为 10.1.1.0/24 的数据放入 1 号队列中

r1(config)#access-list 10 permit 10.1.1.0 0.0.0.255

r1(config)#queue-list 1 protocol ip 1 list 10

（2）将源 IP 为 20.1.1.0/24 的数据放入 2 号队列中

r1(config)#queue-list 1 protocol ip 2 list 20

（3）将端口号为 TCP 23 的数据放入 3 号队列中

r1(config)#queue-list 1 protocol ip 3 tcp 23

（4）默认其它数据都放入 4 号队列中

r1(config)#queue-list 1 default 4

2. 限制队列数据包

（1）限制各个队列每次可传的最大字节数，1 为 100，2 为 200，3 为 300，4 为 400

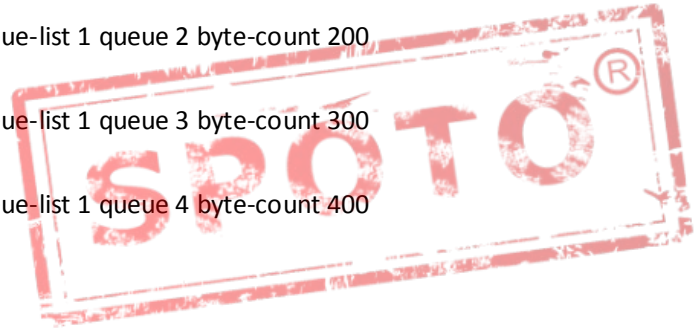
，默认全部为 1500。当传到最大字节数后，将转到传递下一个队列。

r1(config)#queue-list 1 queue 1 byte-count 100

r1(config)#queue-list 1 queue 2 byte-count 200

r1(config)#queue-list 1 queue 3 byte-count 300

r1(config)#queue-list 1 queue 4 byte-count 400



（2）限制各个队列每次可传的最大数据包个数，1 为 10，2 为 20，3 为 30，4 为 40

，默认全部为 20。而这个数据包个数，似乎是带宽限制，此参数不确定。

r1(config)#queue-list 1 queue 1 limit 10

r1(config)#queue-list 1 queue 2 limit 20

r1(config)#queue-list 1 queue 3 limit 30

r1(config)#queue-list 1 queue 4 limit 40

3. 应用 CQ 到接口

（1）将 CQ 应用到接口 F0/0

r1(config)#int f0/0

r1(config-if)#custom-queue-list 1

4. 查看 CQ

（1）查看接口 F0/0 上的队列情况

r1#sh int f0/0

FastEthernet0/0 is up, line protocol is up

Hardware is AmdFE, address is 0013.1a85.d160 (bia 0013.1a85.d160)

Internet address is 12.1.1.1/24

MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation ARPA, loopback not set

Keepalive set (10 sec)

Full-duplex, 100Mb/s, 100BaseTX/FX

ARP type: ARPA, ARP Timeout 04:00:00

Last input 00:00:01, output 00:00:07, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

Queueing strategy: custom-list 1

Output queues: (queue #: size/max/drops)

0: 0/20/0 1: 0/10/0 2: 0/20/0 3: 0/30/0 4: 0/40/0

5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0

10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0

15: 0/20/0 16: 0/20/0

说明:可以看到接口 F0/0 上应用的队列为 CQ。

（2）查看 CQ 参数

r1#show queueing custom

Current custom queue configuration:

List	Queue	Args
1	4	default
1	1	protocol ip list 10
1	2	protocol ip list 20
1	3	protocol ip tcp port telnet
1	1	byte-count 100 limit 10
1	2	byte-count 200
1	3	byte-count 300 limit 30
1	4	byte-count 400 limit 40

r1#

Weighted Fair Queuing (WFQ)

WFQ 是一个基于 Weight 的公平队列，之所以说 WFQ 是公平的，是因为 WFQ 根据数据包的 IP 优先级来分配相应的带宽，优先级高的数据包，分到的带宽就多，优先级低的数据包，分到的带宽就少，并且所有的数据包在任何时刻都可以分到带宽，这就是它的公平之处。WFQ 在根据 IP 优先级给数据包分配带宽时，是基于流（flow）来分配的，也就是说每个流的数据包分配相同的带宽，只有不同的流，才可能分配不同的带宽，如果两个流的 IP 优先级是一样的，那么这两个流分配到的带宽也是一样的。要区分数据包是不是同一个流，需要五个参数完全相同，也就是数据包的源 IP，目的 IP，协议，端口号，以及会话的 socket 全部相同时，这样的数据才被认为是同一个流，既然如此，所以手工是没有办法将两个不同的数据包划分到同一个流的，而计算数据包是不是同一个流，必须由系统自己计算，不需要人工干预。

在配置 WFQ 时，最好已经将不同的数据设置好不同的 IP 优先级，否则所有的流得到的带宽都是一样的，而没办法保证重要的流量。WFQ 根据每个流的 IP 优先级，将接口的可用带宽分配给每个流，计算方法为：

例如现有 4 个流，IP 优先级分别为 0 、1、3、5，WFQ 将所有流的 IP 优先级相加，结果为 0+1+3+5=9，而 9 做为分母；然后每个流的 IP 优先级作为分子，最后得出每个流分配到的带宽为 优先级为 0 得到的带宽为 0/9，优先级为 1 得到的带宽为 1/9，优先级为 3 得到的带宽为 3/9，优先级为 5 得到的带宽为 5/9，这样就可以依靠流的 IP 优先级分配相应的带宽了，但是从上面的算法中可以看出，此算法并不可行，因为优先级为 0 的流得到的带宽为 0/9，而 0/9 就等于 0，也就是优先级为 0 的流得到的带宽为 0，那就是没有带宽，大家知道默认所有的数据包的 IP 优先级恰恰为 0，所以为了防止默认的数据包得不到带宽，最后需要将上面的算法重新调整，结果为 4 个流的 IP 优先级分别为 0 、1、3、5，将原来的优先级全部加 1，分别为 0+1=1 、1+1=2、3+1=4、5+1=6，然后将这些值全部相加，结果为 1+2+4+6=13，将 13 做为分母，然后每个流的 IP 优先级加 1 后的值做为分子，最后得出每个流分配到的带宽为 优先级为 0 得到的带宽为 1/13，优先级为 1 得到的带宽为 2/13，优先级为 3 得到的带宽为 4/13，优先级为 5 得到的带宽为 6/13，这样就有效地防止了默认数据包无法分配到带宽的情况，而且又保证了优级先为 0 的流分到的带宽最少。

注：所有带宽小于或等于 E1 (2.048 Mbps)的接口，默认都启用了 WFQ，即使手工更改带宽后，仍无法改变默认队列机制。

配置 WFQ

1. 查看接口默认 WFQ：

（1）查看接口 S1/0 的默认队列

```
Router#show interfaces s1/0
Serial1/0 is administratively down, line protocol is down
  Hardware is M4T
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  CRC checking enabled
  Last input never, output 00:01:21, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
    Conversations  0/1/256 (active/max active/max total)
    Reserved Conversations 0/0 (allocated/max allocated)
    Available Bandwidth 1158 kilobits/sec
```

说明:因为接口 S1/0 的带宽为 1544 Kbit，也就是小于 E1 (2.048 Mbps)，所以看到接口默认的队列为 weighted fair（WFQ）。

2. 配置 WFQ

（1）在接口 F0/0 上开启 WFQ

```
Router(config)#int f0/0
Router(config-if)#fair-queue
```

3. 查看 WFQ

（1）查看接口 F0/0 的队列情况

```
Router#show interfaces f0/0
FastEthernet0/0 is administratively down, line protocol is down
  Hardware is Gt96k FE, address is c000.0fec.0000 (bia c000.0fec.0000)
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Half-duplex, 10Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: weighted fair
```

Output queue: 0/1000/64/0 (size/max total/threshold/drops)

Conversations 0/0/256 (active/max active/max total)

Reserved Conversations 0/0 (allocated/max allocated)

Available Bandwidth 7500 kilobits/sec

说明:虽然接口 F0/0 的带宽大于 E1 (2.048 Mbps)，但由于手工配置，所以现在的队列为 weighted fair（WFQ）。

Class-based WFQ (CBWFQ)

CBWFQ 的工作原理和 WFQ 是一样的，是基于 WFQ 的，也是对 WFQ 的扩展。因为 CBWFQ 和 WFQ 原理一样，所以不再介绍计算方式。在接口上配置上 WFQ 之后，系统就会将接口所有可用带宽按每个流的 IP 优先级，公平地分给每一个流，并且，接口所有的流都是同时基于接口的全部可用带宽来分配的。CBWFQ 要对 WFQ 进行扩展和优化，就是要为特定的流量划分特定的带宽，让这些特定的流量在分配带宽时，只能从这些划分到的特定带宽中分配，而不是像 WFQ 一样从接口的全部可用带宽中分配。举个例子来解释 WFQ 和 CBWFQ 的区别，例如 A、B、C、D、E 共 5 个人分 100 斤大米，在使用 WFQ 时，是根据 A、B、C、D、E 这五个人的优先级，将 100 斤大米公平地分给五个人的，如果使用 CBWFQ，就可以规定将 80 斤大米按优先级只分配给 A、B、C 三个人，而再将 20 斤大米按优先级分配给 D、E 两个人，可以看出，使用 WFQ 时，竞争大米时，是所有人分配所有的大米，而使用 CBWFQ 时，是不同人群，分配不同数量的大米。

配置 CBWFQ 的方法为 MQC 形式，使用 class-map 匹配指定的流量，然后使用 policy-map 为指定的流量划分特定的带宽，这样之后，指定的流量就只能依靠 IP 优先级从特定的带宽中分配带宽。最后将 CBWFQ 应用到接口，需要注意的是，CBWFQ 只能用在接口的 out 方向。

当划分特定的带宽给某类流量之后，这个带宽是绝对能够为此类流量保证的，而不会被其它流量所抢占，但是如果某类流量超过了被划分的带宽，那么超出的流量将实行尾丢弃。

某个接口的总带宽并不是全部都能被 WFQ 或者 CBWFQ 所使用，默认情况下，一个接口能使用的带宽最多不能超过 75%，所以接口总带宽的 75%才是可用总带宽，

还有保留的 25%是不能使用的，但是可用总带宽是可以随意修改的。当配置 CBWFQ 时，默认没有分配带宽的流量，全部使用保留的 25%。

在配置 CBWFQ 时，接口必须处于默认的队列状态下，并且不支持以太网接口的子接口。

注：一个 CBWFQ，最多可配置 64 类数据流。

配置 CBWFQ

说明：在 R2 上配置 CBWFQ，限制源地址为 10.1.1.0/24 的流量，从带宽 1000 Kbit 中分配带宽，限制源地址为 20.1.1.0/24 的流量，从带宽 2000 Kbit 中分配带宽，而其它流量从所有剩余带宽中分配。

1. 配置 ACL 匹配相应流量

（1）通过 ACL 匹配源地址为 10.1.1.0/24 的流量

r2(config)#access-list 10 permit 10.1.1.0 0.0.0.255

（2）通过 ACL 匹配源地址为 20.1.1.0/24 的流量

r2(config)#access-list 20 permit 20.1.1.0 0.0.0.255

2. 使用 class-map 对流量分类

（1）通过 class-map 匹配源地址为 10.1.1.0/24 的流量

r2(config)#class-map net10

r2(config-cmap)#match access-group 10

（2）通过 class-map 匹配源地址为 20.1.1.0/24 的流量

r2(config)#class-map net20

r2(config-cmap)#match access-group 20

3. 使用 policy-map 为各流量划分带宽

（1）为源地址为 10.1.1.0/24 的流量划分带宽

r2(config)#policy-map CBW

r2(config-pmap)#class net10

r2(config-pmap-c)#bandwidth 1000

说明:命令 bandwidth 就是 CBWFQ 为某些流量划分带宽。

（2）为源地址为 20.1.1.0/24 的流量划分带宽

r2(config)#policy-map CBW

r2(config-pmap)#class net20

r2(config-pmap-c)#bandwidth 2000

（3）其它所有流量从所有剩余可用带宽中分配

说明:当从接口所有可用带宽中划分部分带宽给某类流量之后，那么剩余的可用带宽可以使用百分比的形式分给其它流量，但是需要注意，只有之前的流量也是使用百分比形式划分带宽时，才可以使用剩余可用带宽的方式划分带宽，所以此步只是举例，并不真实，除非将之前的配置改为百分比划分形式。

r2(config)#policy-map CBW

r2(config-pmap)#class class-default

r2(config-pmap-c)#bandwidth remaining percent 100

说明:要让此步配置生效，必须将前面配置改为同样使用百分比分配带宽的形式。

4. 在接口 F0/0 上应用 CBWFQ

（1）改变接口可用带宽总数（此步为可选配置）

r2(config)#int f0/0

r2(config-if)#max-reserved-bandwidth 90

说明:将接口 F0/0 的可用带宽总数从 75%更改到 90%。

（2）在接口 F0/0 上应用 CBWFQ

说明:CBWFQ 只能应用在接口 out 方向

r2(config)#int f0/0

r2(config-if)#service-policy output CBW

5. 查看 CBWFQ

（1）查看接口 F0/0 的队列机制

r2#sh int f0/0

FastEthernet0/0 is administratively down, line protocol is down

Hardware is Gt96k FE, address is c000.0fec.0000 (bia c000.0fec.0000)

MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation ARPA, loopback not set

Keepalive set (10 sec)

Half-duplex, 10Mb/s, 100BaseTX/FX

ARP type: ARPA, ARP Timeout 04:00:00

Last input never, output never, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

Queueing strategy: Class-based queueing

Output queue: 0/1000/64/0 (size/max total/threshold/drops)

Conversations 0/0/256 (active/max active/max total)

Reserved Conversations 2/2 (allocated/max allocated)

Available Bandwidth 4500 kilobits/sec

说明:可以看到接口上的队列为 Class-based queueing （CBWFQ）

Low Latency Queuing（LLQ）

在之前的队列技术中，没有办法保证重要的数据能够一直得到服务，并且一直都有足够的带宽，而且还要不阻断其它流量。比如对延迟和抖动较敏感的语音或视频数据，语音或视频在通信时，需要一直保持着足够的带宽，否则就会受到影响，对于这样的情况，就需要一种队列技术能够为特定的数据流保证特定量的带宽。LLQ 低延迟队列正是为对延迟和抖动较敏感的语音或视频流量设计的，LLQ 为特定的流量划分特定的带宽，划给特定流量的带宽是绝对能够保证的，无论接口有多繁忙，

LLQ 中的流量是能够优先传送的，但是这些流量的带宽却不能超过所分配的带宽，如果超过了，也是没关系的，这些超过的流量只有在拥塞时才会被丢弃。

要将特定的流量分配到 LLQ 中，从而划分绝对保证的带宽，是通过 MQC 的方式来配置的，并且 LLQ 可以结合 CBWFQ，也就是说从接口全部可用带宽中，划出一部分给 LLQ 之后，其它带宽还可以分配给 CBWFQ 中各类数据流。在配置 LLQ 时，可以像 CBWFQ 一样使用具体数字，也可以使用百分比。需要注意，当从接口全部可用带宽中划走一分部给 LLQ 之后，剩下的带宽称为保留带宽（remaining bandwidth），可以将保留带宽以百分比的形式分配给 CBWFQ 中各类数据流。

配置 LLQ

说明:配置 R2，当接口 F0/0 的总带宽为 100 Mbit，且最大可用带宽为 80%，即 80 Mbit 时，使用 LLQ 保证源地址为 10.1.1.0/24 的流量的带宽为 30 Mbit，将全部可用带宽 80 Mbit 中剩下的 50 Mbit，拿出 50%，即 25 Mbit 给源地址为 20.1.1.0/24 的流量。

1. 配置 ACL 匹配相应流量

（1）通过 ACL 匹配源地址为 10.1.1.0/24 的流量

r2(config)#access-list 10 permit 10.1.1.0 0.0.0.255

（2）通过 ACL 匹配源地址为 20.1.1.0/24 的流量

r2(config)#access-list 20 permit 20.1.1.0 0.0.0.255

2. 使用 class-map 对流量分类

（1）通过 class-map 匹配源地址为 10.1.1.0/24 的流量

r2(config)#class-map net10

r2(config-cmap)#match access-group 10

（2）通过 class-map 匹配源地址为 20.1.1.0/24 的流量

r2(config)#class-map net20

r2(config-cmap)#match access-group 20

3. 分配带宽

（1）使用 LLQ 分配 30 Mbit 给源地址为 10.1.1.0/24 的流量

r2(config)#policy-map band

r2(config-pmap)#class net10

r2(config-pmap-c)#priority percent 30

说明:命令 priority 为 LLQ 中的流量分配带宽，percent 后的百分比为接口总带宽的百分比，因为接口总带宽为 100 Mbit，所以 percent 30 就是 30 Mbit。

（2）使用 CBWFQ 将剩余可用带宽的 50%分配给源地址为 20.1.1.0/24 的流量

r2(config)#policy-map band

r2(config-pmap)#class net20

r2(config-pmap-c)#bandwidth remaining percent 50

说明:因为 LLQ 中的流量使用了 30 Mbit，全部可用带宽为 80 Mbit，所以剩余带宽为 50 Mbit，而 remaining percent 则是从分配给 LLQ 后剩余带宽 50 Mbit 中的 50%，即为 25 Mbit。

4. 应用队列到接口

（1）将接口的全部可用带宽改为 80Mbit

r2(config)#int f0/0

r2(config-if)#max-reserved-bandwidth 80

说明:将接口的全部可用带宽从默认的 75%更改到 80%,即 80 Mbit。

（2）应用队列到接口 F0/0

r2(config)#int f0/0

r2(config-if)#service-policy output band

说明:因为 LLQ 结合了 CBWFQ，只要有 CBWFQ，方向只能为 out。

5. 查看队列

（1）查看接口 F0/0 的队列机制

r2#sh int f0/0

FastEthernet0/0 is administratively down, line protocol is down

Hardware is Gt96k FE, address is c000.0e48.0000 (bia c000.0e48.0000)

MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation ARPA, loopback not set

Keepalive set (10 sec)

Half-duplex, 10Mb/s, 100BaseTX/FX

ARP type: ARPA, ARP Timeout 04:00:00

Last input never, output never, output hang never

Last clearing of "show interface" counters never

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

Queueing strategy: Class-based queueing

Output queue: 0/1000/64/0 (size/max total/threshold/drops)

Conversations 0/0/256 (active/max active/max total)

Reserved Conversations 1/1 (allocated/max allocated)

Available Bandwidth 5000 kilobits/sec

说明:看到接口现有队列为 CBWFQ，并且是结合了 LLQ 的。

IP RTP （Real-Time Transport Protocol）

在使用 LLQ 时，可以做到为延迟和抖动较敏感的语音或视频数据绝对保证带宽，但是 LLQ 可以为任何数据流服务，不限制于任何协议。而 IP RTP 尽量只为对延迟要求较高的实时数据提供带宽保证，比如尽量只为语音提供带宽保证。受 RTP 保护的数据流，可以在任何流量之前优先传递，即使是 LLQ 和 RTP 同时出现的情况下，RTP 的流量是优先于 LLQ 传送的。并不是所有的流量都能受到 RTP 的保护，只有 UDP 目标端口号为 16384 至 32767 的数据才能得到保护，并且可以随意定义端口号范围，所以当为语音数据流保证带宽时，并不需要知道准确的语音端口号信息。

当 RTP 在为语音数据流提供带宽保证时，RTP 是不知道有多少条语音会话的，也就是说，所有的语音会话，是共享整个带宽的，比如划分了 100 Kbit 给语音会话，如果此时有 4 条语音会话，那么就是 100 Kbit / 4=25 Kbit，结果就是每个语音会话的带宽为 25 Kbit，而不是 100 Kbit。

RTP 可以和 WFQ 或 CBWFQ 配置于同一个出接口，但 RTP 只支持 Serial interfaces 和 Frame Relay PVC，如果配置于接口下，那么是在整个接口生效，如果配置于 PVC 下，则只在单独的 PVC 下生效，但是如果只需要配置于某条 PVC，需要在工具

map-class 下配置，并将 map-class 应用于 PVC 下。

注：当接口下同时出现 LLQ 和 RTP 时，RTP 的参数优先，例如 LLQ 和 RTP 为同一类数据流带宽保证时，LLQ 的带宽为 80 Kbit，RTP 的带宽为 60 Kbit，那么该流量的带宽则为 60 Kbit。

配置 RTP

1. 在接口下配置 RTP

（1）在路由器 Serial 1/0 接口下配置 RTP，为 UDP 端口号范围为 16384 到 16888 的流量保证带宽 200 Kbit。

Router(config)#int s1/0

Router(config-if)#ip rtp priority 16888 16383 200

说明:如果需要为 VoIP 保证带宽，请将端口号分别设置为 16384 16383。

2. 在 Frame Relay 接口下配置 RTP

（1）在 map-class 下配置 RTP

Router(config)#map-class frame-relay VOIP

Router(config-map-class)#frame-relay cir 100000

Router(config-map-class)#frame-relay bc 1000

Router(config-map-class)#frame-relay be 0

Router(config-map-class)#frame-relay fragment 64

Router(config-map-class)#frame-relay ip rtp priority 16384 16383 100

说明:为 UDP 端口号范围为 16384 到 16383 的流量保证带宽 100 Kbit。配置 RTP 时，先配置 FRTS 和 FRF.12。

（2）将 map-class 应用于 Frame Relay 接口的 PVC 100 下

Router(config)#int s1/0

Router(config-if)#encapsulation frame-relay

Router(config-if)#frame-relay traffic-shaping

Router(config-if)#frame-relay interface-dlci 100

Router(config-fr-dlci)#class VOIP

拥塞避免（Congestion avoidance）

当网络发生拥塞之后，总是有数据是要被丢弃的，在默认情况下，拥塞之后，接口总是先丢弃最后到达的数据包，而将之前已经到达的先转发，这样一来，当网络中有多种流量出现时，就难免会将重要的流量丢弃，或者尽量让某些程序降级带宽而避免拥塞的发生，这些，都是拥塞避免（Congestion avoidance）需要做的事情。拥塞避免需要靠以下技术来完成：

Tail Drop

Weighted Random Early Detection (WRED)

WRED—Explicit Congestion Notification

Frame Relay Discard Dligible (DE)

Tail Drop

尾丢弃（Tail Drop）是接口的默认行为，当接口发生拥塞后，总是将最后到达的数据包丢弃，直到没有拥塞为止，因为最后到的数据就是引起网络拥塞的主要原因。在使用尾丢弃的情况下，是无法保证重要数据流优先传递的，所以尾丢弃不建议使用，但也无法配置。

Weighted Random Early Detection (WRED)

WRED 是基于 weight 的随机早侦测，工作思想和 WFQ 有相同之处，因为 WFQ 在工作时，是依靠流量的优先级来分配相应带宽的，而 WRED 却是依靠流量的优先级来分配相应的丢弃几率的。当网络中有多种数据时，在发生拥塞之后，人们总是希望先将优先级较低的相对不重要的数据丢弃，而优先保证重要数据的传递。WRED 正是迎合了人们的这种期望，在网络发生拥塞之后，总是先保证高优先级的重要数据的传递，而先丢弃普通的数据。

WRED 在网络发生拥塞之后，可根据数据包的 DSCP 或 IP 优先级来丢弃数据包，低优先级的数据总是比高优先级的数据先丢，从而保证重要数据的传递。在默认情况下，是根据数据包的 IP 优先级来决定如何丢弃的。

虽然说 WRED 是丢弃低优先级的数据包而保证高优先级的数据包，但是网络拥塞时，并不是总是先丢低优先级的，这是需要靠公式来计算的。思想为根据各优先级或 DSCP 设置的阀值，如果某优先级或 DSCP 的流量总是触及设定的阀值，那么该流量被丢弃的概率也就越大，所以如果低优先级不经常触及设置的阀值时，也有不被丢的可能。

在数据包被丢弃之后，如果是 TCP 流量，便可以调整窗口大小，从而降低速度，但是除 TCP 之外的其它流量便无能为力了。同时，也只有 TCP 才能够对丢弃的数据包进行重传，所以在使用 WRED 时，需要考虑这些问题。

WRED 在应用时，只能应用于接口下，或者和 WFQ 与 CBWFQ 一起使用，之所以不能和 PQ 一样的队列同时使用，是因为 PQ 或 LLQ 都有自己的保护和丢弃机制，WRED 对数据的操作没有太多意义。

配置 WRED

1. 在接口下配置 WRED

说明:可以选择开启基于 IP 优先级的 WRED 或基于 DSCP 的 WRED，默认基于 IP 优先级。

（1）在接口 F0/0 下开启基于 IP 优先级的 WRED

Router(config)#int f0/0

Router(config-if)#random-detect

说明:默认 WRED 基于 IP 优先级

（2）在接口 F0/1 下开启基于 DSCP 的 WRED

Router(config)#int f0/1

Router(config-if)#random-detect dscp-based

说明:指定 WRED 基于 DSCP 工作。

（3）查看 WRED

Router#sh queueing random-detect

Current random-detect configuration:

FastEthernet0/0

Queueing strategy: random early detection (WRED)

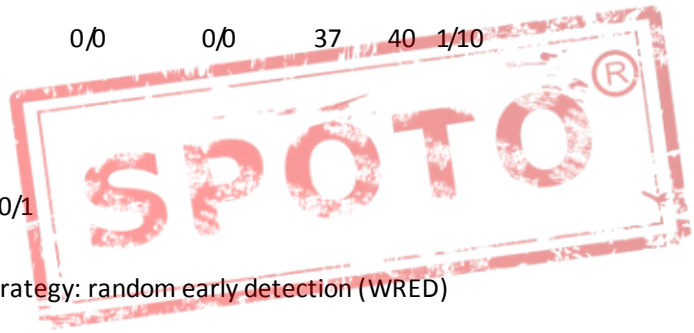
Random-detect not active on the dialer

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0

class	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	thresh	thresh	prob

0	0/0	0/0	20	40	1/10
1	0/0	0/0	22	40	1/10
2	0/0	0/0	24	40	1/10
3	0/0	0/0	26	40	1/10
4	0/0	0/0	28	40	1/10
5	0/0	0/0	31	40	1/10
6	0/0	0/0	33	40	1/10
7	0/0	0/0	35	40	1/10
rsvp	0/0	0/0	37	40	1/10



FastEthernet0/1

Queueing strategy: random early detection (WRED)

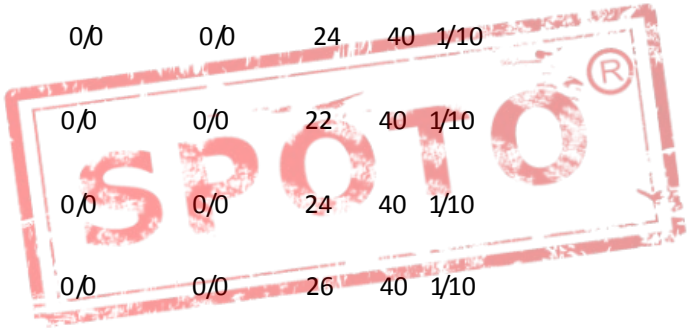
Random-detect not active on the dialer

Exp-weight-constant: 9 (1/512)

Mean queue depth: 0

dscp	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	thresh	thresh	prob
af11	0/0	0/0	33	40	1/10
af12	0/0	0/0	28	40	1/10
af13	0/0	0/0	24	40	1/10

af21	0/0	0/0	33	40	1/10
af22	0/0	0/0	28	40	1/10
af23	0/0	0/0	24	40	1/10
af31	0/0	0/0	33	40	1/10
af32	0/0	0/0	28	40	1/10
af33	0/0	0/0	24	40	1/10
af41	0/0	0/0	33	40	1/10
af42	0/0	0/0	28	40	1/10
af43	0/0	0/0	24	40	1/10
cs1	0/0	0/0	22	40	1/10
cs2	0/0	0/0	24	40	1/10
cs3	0/0	0/0	26	40	1/10
cs4	0/0	0/0	28	40	1/10
cs5	0/0	0/0	31	40	1/10
cs6	0/0	0/0	33	40	1/10
cs7	0/0	0/0	35	40	1/10
ef	0/0	0/0	37	40	1/10
rsvp	0/0	0/0	37	40	1/10
default	0/0	0/0	20	40	1/10



Router#

2．配置 CBWFQ 下的 WRED

（1）配置 WRED 在 CBWFQ 对所有流量生效

Router(config)#policy-map WWW

Router(config-pmap)#class class-default

Router(config-pmap-c)#bandwidth 1000000

Router(config-pmap-c)#random-detect

说明:对所有流量通过命令 bandwidth 配置了 CBWFQ，并通过命令 random-detect 开启了基于 IP 优先级的 WRED。

WRED—Explicit Congestion Notification

在使用 WRED 对过多的流量进行丢弃时，有时会造成不必要的麻烦。比如在使用非 TCP 进行通信时，被丢弃的数据包无法得到重传，然而即使是 TCP，虽然将数据丢弃之后，能够遏制高速流量，但当拥塞消失后，必定还会引起下一次拥塞，所以 WRED 只能治标而不能治本。

引起网络拥塞的原因就是数据源发送了过量的数据包，而 ECN（明确拥塞通告）就是通过发送警告，让数据源知道网络已经发生拥塞，从而可以降低自己发送数据包的速度。但是如果数据源不支持 ECN，那么所发送的流量照丢不误。

配置 ECN 时，必须开启 WRED，并且需要配合 WFQ 或 CBWFQ 使用。

在默认情况下，整形后的帧中继接口在收到 ECN 消息后，会将速度降到原有速度的一半，但是之后的速度也不会低于一半，只会在原有速度和一半之间浮动。这些参数可以调整。

配置 ECN

1. 在 CBWFQ 下配置 ECN

（1）在 CBWFQ 下为所有流量配置 ECN

```
Router(config)#policy-map EEE
Router(config-pmap)#class class-default
Router(config-pmap-c)#bandwidth percent 75
Router(config-pmap-c)#random-detect
Router(config-pmap-c)#random-detect ecn
```

说明:在 CBWFQ 下开启 ECN。

2. 查看配置

（1）查看开启 ECN 的 CBWFQ

```
Router#sh policy-map
  Policy Map EEE
    Class class-default
     Bandwidth 75 (%)
          exponential weight 9
          explicit congestion notification
          class    min-threshold    max-threshold    mark-probablity
          ------------------------------------------------------------
```

0	-	-	1/10
1	-	-	1/10
2	-	-	1/10
3	-	-	1/10
4	-	-	1/10
5	-	-	1/10
6	-	-	1/10
7	-	-	1/10
rsvp	-	-	1/10

3. 更改 Frame Relay 下的 ECN 参数

（1）在收到 ECN 后，带宽的最低限制

Router(config)#map-class frame-relay FFF

Router(config-map-class)#frame-relay cir 100000

Router(config-map-class)#frame-relay bc 1000

Router(config-map-class)#frame-relay be 0

Router(config-map-class)#frame-relay mincir 6000

说明:在收到 ECN 后，带宽不会降到 6000 Bit 以下。

Frame Relay Discard Dligible (DE)

在 Frame Relay 网络中，数据包中标有 Discard Dligible (DE)字段，该字段告诉设备数据包的重要性，如果为 1，表示该数据包并不重要，在网络发生拥塞时可以优先被丢弃，如果为 0，则表示在将为 1 的数据包全部丢光的情况下，才可被丢弃，所有数据包的 DE 字段默认为 0。

可以将特定数据包的 DE 字段设置为 1。

配置 Frame Relay DE

说明:配置 R1，将去往 10.1.1.0/24 的数据包 DE 字段标为 1，在接口拥塞时，可以先丢弃。

1. 配置 ACL 匹配相应流量

（1）通过 ACL 匹配去往 10.1.1.0/24 的流量

```
r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255
```

2. 将指定流量的 DE 设置为 1

(1)设置 ACL 100 中的流是的 DE 为 1

```
r1(config)#frame-relay de-list 1 protocol ip list 100
```

说明:所有被 de-list 匹配到的流量的 DE 都会被设置为 1。

3. 应用策略到 PVC 下

（1）将 de-list 应用到 PVC 下

r1(config)#int s0/0

r1(config-if)#frame-relay de-group 1 102

4．测试结果

（1）在 R1 上向 10.1.1.0/24 发送流量

r1#ping 10.1.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 56/56/57 ms

r1#

（2）查看 PVC 下的结果

r1#sh fram pvc 102

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

input pkts 40 output pkts 40 in bytes 4160

out bytes 4160 dropped pkts 0 in pkts dropped 0

out pkts dropped 0 out bytes dropped 0

in FECN pkts 0 in BECN pkts 0 out FECN pkts 0

out BECN pkts 0 in DE pkts 0 out DE pkts 5

out bcast pkts 0 out bcast bytes 0

5 minute input rate 0 bits/sec, 1 packets/sec

5 minute output rate 0 bits/sec, 0 packets/sec

pvc create time 00:05:34, last time pvc status changed 00:04:54

r1#

说明:可以看到 PVC 下匹配到的 DE 为 1 的数据包数量，当接口拥塞时，此类数据包将优先被丢弃。

链路优化（Link Efficiency Mechanisms）

在使用 QOS 保证重要数据的优先传递时，除了可以使用队列技术之外，还有就是在链路上做一些额外的优化，以获得更高的性能和更明显的效果。链路优化的技术有如下三种：

Link Efficiency Mechanisms

链路层技术，

有：

Multilink PPP (MLP)

Frame Relay Fragmentation

Header Compression

CCIE 考试需要了解以上三种链路优化技术，下面分别来详细介绍。

Multilink PPP (MLP)

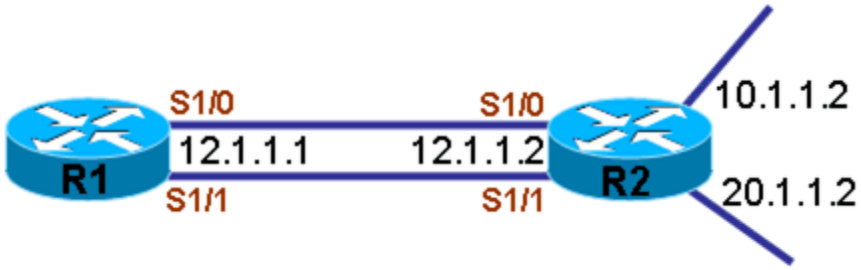
对于低于或等于 768kbps 的链路，由于带宽很低，延迟太大，将对语音或视频的通信带来麻烦。而 Multilink PPP (MLP)技术为了能够提高链路带宽，允许同时将多根低速链路捆绑成逻辑上单一的链路，称为 bundle，从而将数据流分散在多条链路上同时传输，以提高带宽，降低延迟，如下图所示：

上图中，R1 和 R2 之间的两条低速串行链路便可捆绑成一条逻辑链路，在路由器之间传输数据流时，数据流将同时从两根链路上交叉传输，等于使用了两根链路，带宽得到了提高。将串行链路捆绑成 Multilink 时，需要 PPP 封装。

在将多条物理链路捆绑成一条逻辑链路后，还是可以实施各种 QOS 技术的，但是 QOS 必须应用到逻辑链路上，IP 地址也要配置到逻辑链路上。默认情况下，使用 FIFO，但 LLQ，WFQ，CBWFQ 也可以使用，在使用其它 QOS 技术时，应该在配置 Multilink 之前就将 QOS 配置好。

在使用 Multilink 时，有个较为复杂的延迟计算，就是 fragment delay 单位为 ms，是用来控制多条链路交叉传输数据包的，这里不作出计算方式的详细介绍。

配置 Multilink PPP (MLP)



以上图为例，将 R1 与 R2 之间的两条链路 S1/0 和 S1/1 捆绑成一条逻辑链路，并在 R1 上实施 QOS，将 R1 到目标网络 10.1.1.0/24 的流量管制到 CIR 为 8000 bit，超过的流量则丢弃，R1 到目标网络 20.1.1.0/24 的流量正常通过。

1.配置 QOS

（1）通过 ACL 匹配到网络 10.1.1.0/24 的流量

r1(config)#access-list 100 permit ip any 10.1.1.0 0.0.0.255

（2）class-map 调用 ACL 中的流量

r1(config)#class-map net10

r1(config-cmap)#match access-group 100

（3）在 policy-map 中调用 class-map 的流量并管制带宽

r1(config)#policy-map band

r1(config-pmap)#class net10

r1(config-pmap-c)# police cir 8000 bc 1000 be 1000 conform-action transmit exceed-action drop

说明:配置中 CIR 为 8000 bit，超过的流量为 drop。

2.创建 Multilink

(1)在 R1 上创建 Multilink

r1(config)#int multilink 1

r1(config-if)#ip address 12.1.1.1 255.255.255.0

r1(config-if)#service-policy output band

r1(config-if)#ppp multilink fragment delay 10

说明:R1 上的 Multilink 号码为 1，并配置 IP 地址，设置数据包交叉延迟为 10ms。

3.将串口划入 Multilink

(1)将接口 S1/0 划入 Multilink

r1(config)#int s1/0

r1(config-if)#no ip address

r1(config-if)#encapsulation ppp

r1(config-if)#ppp multilink

r1(config-if)#ppp multilink group 1

r1(config-if)#no shutdown

(2)将接口 S1/1 划入 Multilink

r1(config)#int s1/1

r1(config-if)#no ip address

r1(config-if)#encapsulation ppp

r1(config-if)#ppp multilink

r1(config-if)#ppp multilink group 1

r1(config-if)#no shutdown

说明:R1 上的两条链路 S1/0 和 S1/1 封装成 PPP 后，并捆绑成单一逻辑链路。

4.配置 R2

说明:R2 配置与 R1 类似。

5.查看结果

（1）查看单条物理链路带宽

r1#show interfaces s1/0

Serial1/0 is up, line protocol is up

Hardware is M4T

MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation PPP, LCP Open, multilink Open

Link is a member of Multilink bundle Multilink1, crc 16, loopback not

Set

说明:可以看到单条链路的带宽为 1544 Kbit，并且已经工作在 multilink

(2)查看 multilink

r1#sh ppp multilink

Multilink1, bundle name is r2

Endpoint discriminator is r2

Bundle up for 00:03:45, total bandwidth 3088, load 1/255

Receive buffer limit 24000 bytes, frag timeout 1000 ms

0/0 fragments/bytes in reassembly list

0 lost fragments, 0 reordered

0/0 discarded fragments/bytes, 0 lost received

0x1E received sequence, 0x1D sent sequence

Member links: 2 active, 0 inactive (max not set, min not set)

Se1/0, since 00:03:45, 1930 weight, 1496 frag size

Se1/1, since 00:03:30, 1930 weight, 1496 frag size

No inactive multilink interfaces

r1#

说明:可以看到 multilink 中拥有两条链路，因为单一链路带宽为 1544 Kbit，所以 multilink 的带宽为 3088。

（3）查看 multilink IP 信息

r1#sh interfaces multilink 1

Multilink1 is up, line protocol is up

Hardware is multilink group interface

Internet address is 12.1.1.1/24

MTU 1500 bytes, BW 3088 Kbit, DLY 100000 usec,

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation PPP, LCP Open, multilink Open

Open: IPCP, CDPCP, loopback not set

Keepalive set (10 sec)

DTR is pulsed for 2 seconds on reset

说明:可以看到 multilink 拥有配置的 IP 地址。

6.测试 QOS

（1）测试 R1 到目标网络 20.1.1.0/24 的流量

r1#ping 20.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:

!!!!!!!!!!

Success rate is 100 percent (10/10), round-trip min/avg/max = 16/72/236

ms

r1#

说明:因为并没有对 R1 到目标网络 20.1.1.0/24 的流量进行管制，所以当数据包每个以 800 字节通过时，一切正常，并且速度正常。

（2）测试 R1 到目标网络 10.1.1.0/24 的流量

r1#ping 10.1.1.2 size 800 repeat 10

Type escape sequence to abort.

Sending 10, 800-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:

!.!.!.!.!.

Success rate is 50 percent (5/10), round-trip min/avg/max = 156/188/252

ms

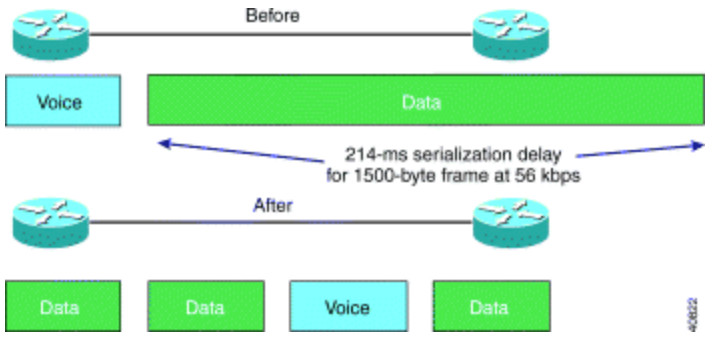
r1#

说明:因为 R1 到目标网络 10.1.1.0/24 的流量被管制为 8000bit 每秒，所以当数据包每个以 800 字节通过时，流量超出额定带宽，从而出现了超额流量被均衡地丢弃，multilink 上的 QOS 已生效。

Frame Relay Fragmentation

当不能接受延迟的敏感数据，如语音在低速的链路上传递时，可能会受到影响。一般情况下，语音需要低于 150 milliseconds (ms)的延迟，而语音在一个接口发送时建议低于 20ms 的延迟，但是当设备在将数据包放入接口进行处理时，这本身就需要花费一定的时间，如果数据包过大，那么将整个数据包放入接口处理可能就要花费过多的时间。

一个大小为 1500-byte 的数据包在进入一个速度为 64 kbps 的接口需要花的时间为 187 ms，很显然这会影响到语音的质量。如果接口上同时有普通数据和语音同时传输时，语音受到的影响则更为严重。因此，如果能够将数据包分割地更小来进行传递，这样就可以减少语音的抖动，如下图：



在帧中继接口上将数据包分割成更小的包以减少延迟，有以下两种技术：

FRF.11 Fragmentation

FRF.12 Fragmentation

FRF.11 已经不建议使用，FRF.12 Fragmentation 将语音和实时数据包分割的更小。FRF.12 在接口上将大的数据包分割成固定大小的更小的数据包再传递，在 FR 接口上，只有数据包大于配置好的大小，才会被分割。

在接口速率超过 768 kbps 时可能不需要 Fragmentation，但是肯定需要 RTP 或 LLQ 来保证语音流量的传递。不同速率的接口，都有一个建议的被分割的值：

Lowest Link Speed in Path	Recommended Fragmentation Size (for 10 ms Serialization)
56 Kbps	70 bytes
64 Kbps	80 bytes
128 Kbps	160 bytes
256 Kbps	320 bytes
512 Kbps	640 bytes
768 Kbps	1000 bytes
1536 Kbps	1600 bytes

配置 FRF.12 Fragmentation

说明:建议对联的两台路由器都配，且参数一致。

1．在路由器上配置 FRF.12 Fragmentation

（1）在 map-class 下配置 FRF.12 Fragmentation

Router(config)#map-class frame-relay FFF

Router(config-map-class)#frame-relay fragment 640

说明:FRF.12 将数据包分割到 640 byte 每个包，默认不跟参数，大小为 53 byte。

(2)将 FRF.12 Fragmentation 用于帧中继接口下

```
Router(config)#int s1/0

Router(config-if)#frame-relay traffic-shaping

Router(config-if)#frame-relay class FFF
```

说明:map-class 用在哪，FRF.12 就在哪生效；traffic-shaping 必开，如果不开，FRF.12 不工作。

2. 查看 FRF.12

（1）查看接口下的 FRF.12

```
r1#sh frame-relay fragment

interface      dlci  frag-type    frag-size  in-frag    out-frag   dropped-frag
Serial0/1      112   end-to-end    80         0          0          0
Serial0/1      113   end-to-end    80         0          0          0
Serial0/1      114   end-to-end    80         0          0          0
Serial0/1      115   end-to-end    80         0          0          0
Serial0/1      116   end-to-end    80         0          0          0

r1#
```


说明:可以看到接口下每条 PVC 的 FRF.12 详细信息。

Header Compression

正常情况下，数据包除了真正的数据之外，还有一部分就是包头信息，为了能够更有效地利用带宽，可以在数据包被发送之前，将 IP 的包头压缩，以缩小多余的头部信息，从而传输更多的数据内容。

将 IP 数据包的包头压缩之后，可以减少过载，加快传输，头部压缩只能对 RTP 和 TCP 数据进行压缩。

配置数据包头部压缩，可以在接口下直接配置，也可能通过 MQC 基于 class 配置。在对数据进行压缩时，只支持 Frame Relay, HDLC 和 PPP 接口，ISDN 也可以。但在 Frame Relay 接口使用 IETF 封装时，不能对 RTP 进行压缩；压缩需要在对联的两台设备之间同时配置。

配置 Header Compression

1．HDLC 接口下配置 RTP 头部压缩

（1）在 HDLC 接口下配置 RTP 头部压缩

```
r1(config)#int s0/0

r1(config-if)#ip rtp header-compression
```

（2）查看压缩

```
r1#sh ip rtp header-compression

RTP/UDP/IP header compression statistics:

  Interface Serial0/0:

    Rcvd:     0 total, 0 compressed, 0 errors, 0 status msgs

              0 dropped, 0 buffer copies, 0 buffer failures

    Sent:     0 total, 0 compressed, 0 status msgs

              0 bytes saved, 0 bytes sent

    Connect: 16 rx slots, 16 tx slots,

              0 long searches, 0 misses 0 collisions, 0 negative cache hits


r1#
```

2. frame-relay 接口下配置 RTP 头部压缩

（1）在 frame-relay 接口下配置 RTP 头部压缩

r1(config)#int s0/0

r1(config-if)#ip rtp header-compression

（2）查看压缩

r1(config)#int s0/0

r1(config-if)#encapsulation frame-relay

r1(config-if)#frame-relay ip rtp header-compression

3 HDLC 接口下配置 TCP 头部压缩

（1）在 HDLC 接口下配置 TCP 头部压缩

r1(config)#int s0/0

r1(config-if)#ip tcp header-compression

4. 配置 class-based 的头部压缩

（1）配置基于 class-based 的 RTP 头部压缩

r1(config)#policy-map COM

r1(config-pmap-c)#compression header ip rtp

说明:基于 class-based 的 TCP 头部压缩配置命令相似，不再重复。

（2）将策略应用于接口

r1(config)#int s0/0

r1(config-if)#service-policy output COM

说明:基于 class-based 的头部压缩只能用于接口 out 方向

返回目录

AutoQoS — VoIP

概述

在手工配置 QOS 为特定的流量提供带宽保证时，需要匹配特定的流量，再根据这些流量做相应的策略。这样的方式比较烦琐，并且对于不了解 QOS 的人来说，要为重要的流量，如语音流量提供带宽保证，可能容易出现人工配置错误。

对于一般流量的 QOS，可能由于需求各不相同，没有一个特定的规范，所以还是需要人工来配置。但是对于如语音或视频这样的特殊流量，因为它们有着固定的 QOS 需求，无论谁来配，只要满足这些要求即可，因此，思科在 IOS 中集成了固定的 QOS，为语音和视频自动配置符合语音的视频特定要求的 QOS，这就是 AutoQoS — VoIP。

在实施 AutoQoS — VoIP 时，有许多要注意的地方：

接口上不能事先配置 QOS，比如配置 service policies

Simple Network Protocol (SNMP) traps（会自动打开）

支持的接口有 Serial interfaces(如封装了 PPP，HDLC，以及 Frame Relay 的接口)，高版本的 IOS 也支持其它类型接口。

Frame Relay 接口在映射了 DLCI 时是不能配置的，且只支持 Frame Relay point-to-point 子接口。

对联的两台设备的接口都应该配置 AutoQoS — VoIP，并且速率应该相同，Frame Relay

接口的 FRF.12 也要相同。

在配置 AutoQoS — VoIP 后，就不能再更改接口带宽了，即使改了，也还是原来的带宽，如果之前更改带宽，也不能随意更改。

在开启 AutoQoS — VoIP 后，会自动为 VoIP 流量标记，而且会使用 LLQ，PQ，RTP 也会实施。AutoQoS — VoIP 会自己在接口上为 VoIP 流量创建相应的 policy maps, class Maps 以及 ACLs。

注：带宽小于或等于 768 kpbs 的被认为是 low-speed links，大于 768 kpbs 的被认为是 high-speed links。

在普通网络实施 AutoQoS — VoIP，只需要一条命令 auto qos voip 即可。

在企业网络里实施 AutoQoS — VoIP 时，需要对流量有个发现和收集的过程，称为 Auto-Discovery，是使用 NBAR 来发现语音流量的。

注：在配置了 AutoQoS — VoIP 之后，若是关掉，那么之前的配置在某些 IOS 下还会保留，需要手工清除配置。

配置 AutoQoS — VoIP

1. 开启 HDLC 接口下的 AutoQoS — VoIP

（1）在 HDLC 接口下开启 AutoQoS — VoIP

```
Router(config)#int s0/0
Router(config-if)#encapsulation hdlc
Router(config-if)#bandwidth 1000
Router(config-if)#auto qos voip
```

说明:将带宽改为 1000Kbit，并开启 AutoQoS — VoIP，改带宽须在开启 AutoQoS — VoIP 之前改好。

（2）查看 AutoQoS — VoIP

Router#show auto qos interface

Serial0/0 -

!

interface Serial0/0

service-policy output AutoQoS-Policy-UnTrust

Router#show auto qos

!

policy-map AutoQoS-Policy-UnTrust

class AutoQoS-VoIP-RTP-UnTrust

priority percent 70

set dscp ef

class AutoQoS-VoIP-Control-UnTrust

bandwidth percent 5

set dscp af31

class AutoQoS-VoIP-Remark

set dscp default

class class-default

fair-queue

```
!
class-map match-any AutoQoS-VoIP-Remark
  match ip dscp ef
  match ip dscp cs3
  match ip dscp af31
!
class-map match-any AutoQoS-VoIP-Control-UnTrust
  match access-group name AutoQoS-VoIP-Control
!
class-map match-any AutoQoS-VoIP-RTP-UnTrust
  match protocol rtp audio
  match access-group name AutoQoS-VoIP-RTCP
!
ip access-list extended AutoQoS-VoIP-RTCP
  permit udp any any range 16384 32767
!
ip access-list extended AutoQoS-VoIP-Control
  permit tcp any any eq 1720
  permit tcp any any range 11000 11999
  permit udp any any eq 2427
  permit tcp any any eq 2428
```

permit tcp any any range 2000 2002

permit udp any any eq 1719

permit udp any any eq 5060

!

rmon event 33333 log trap AutoQoS description "AutoQoS SNMP traps for Voice Drops" owner AutoQoS

rmon alarm 33334 cbQosCMDropBitRate.1249.1251 30 absolute rising-threshold 1 33333 falling-threshold 0 owner AutoQoS

Serial0/0 -

!

interface Serial0/0

service-policy output AutoQoS-Policy-UnTrust

Router#

说明:可以看到接口 S0/0 已开启的 AutoQoS — VoIP，并且详细参数已生效。

2. 开启 Frame Relay 接口下的 AutoQoS — VoIP

(1)在 Frame Relay 子接口下开启 AutoQoS — VoIP

Router(config)#int s0/0

Router(config-if)#encapsulation frame-relay

Router(config-if)#exit

```
Router(config)#int s0/0.1 point-to-point

Router(config-subif)#frame-relay interface-dlci 102

Router(config-fr-dlci)#auto qos voip
```

说明: Frame Relay 只支持 point-to-point 子接口，并且只能在 DLCI 下开启。

（2）查看 AutoQoS — VoIP

```
Router#show auto qos

!

policy-map AutoQoS-Policy-UnTrust

 class AutoQoS-VoIP-RTP-UnTrust

  priority percent 70

  set dscp ef

 class AutoQoS-VoIP-Control-UnTrust

  bandwidth percent 5

  set dscp af31

 class AutoQoS-VoIP-Remark

  set dscp default

 class class-default

  fair-queue

!

class-map match-any AutoQoS-VoIP-Remark

 match ip dscp ef
```

```
 match ip dscp cs3
 match ip dscp af31
!
class-map match-any AutoQoS-VoIP-Control-UnTrust
 match access-group name AutoQoS-VoIP-Control
!
class-map match-any AutoQoS-VoIP-RTP-UnTrust
 match protocol rtp audio
 match access-group name AutoQoS-VoIP-RTCP
!
ip access-list extended AutoQoS-VoIP-RTCP
 permit udp any any range 16384 32767
!
ip access-list extended AutoQoS-VoIP-Control
 permit tcp any any eq 1720
 permit tcp any any range 11000 11999
 permit udp any any eq 2427
 permit tcp any any eq 2428
 permit tcp any any range 2000 2002
 permit udp any any eq 1719
 permit udp any any eq 5060
```

!

rmon event 33333 log trap AutoQoS description "AutoQoS SNMP traps for Voice Drops" owner AutoQoS

rmon alarm 33336 cbQosCMDropBitRate.1399.1401 30 absolute rising-threshold 1 33333 falling-threshold 0 owner AutoQoS

Serial0/0.1: DLCI 102 -

!

interface Serial0/0

frame-relay traffic-shaping

!

interface Serial0/0.1 point-to-point

frame-relay interface-dlci 102

class AutoQoS-FR-Se0/0-102

!

map-class frame-relay AutoQoS-FR-Se0/0-102

frame-relay cir 1544000

frame-relay bc 15440

frame-relay be 0

frame-relay mincir 1544000

service-policy output AutoQoS-Policy-UnTrust

Router#

说明:可以看到接口 S0/0.1 已开启的 AutoQoS — VoIP，并且详细参数已生效。

3．配置企业网络的 AutoQoS — VoIP

（1）在接口 S0/0 下配置 Auto-Discovery

Router(config)#int s0/0

Router(config-if)#auto discovery qos

（2）在接口 S0/0 下配置 AutoQoS

Router(config)#int s0/0

Router(config-if)#auto qos voip

(3)查看 Auto-Discovery

Router#sh auto discovery qos

Serial0/0

AutoQoS Discovery enabled for applications

Discovery up time: 1 minutes, 6 seconds

AutoQoS Class information:

No AutoQoS data discovered

Router#

说明:因为没有语音流量，所有检测到的数据为空。

(4)查看 AutoQoS — VoIP

Router#show auto qos

!

```
policy-map AutoQoS-Policy-UnTrust
 class AutoQoS-VoIP-RTP-UnTrust
  priority percent 70
  set dscp ef
 class AutoQoS-VoIP-Control-UnTrust
  bandwidth percent 5
  set dscp af31
 class AutoQoS-VoIP-Remark
  set dscp default
 class class-default
  fair-queue
!
class-map match-any AutoQoS-VoIP-Remark
 match ip dscp ef
 match ip dscp cs3
 match ip dscp af31
!
class-map match-any AutoQoS-VoIP-Control-UnTrust
 match access-group name AutoQoS-VoIP-Control
!
class-map match-any AutoQoS-VoIP-RTP-UnTrust
```

match protocol rtp audio

match access-group name AutoQoS-VoIP-RTCP

!

ip access-list extended AutoQoS-VoIP-RTCP

permit udp any any range 16384 32767

!

ip access-list extended AutoQoS-VoIP-Control

permit tcp any any eq 1720

permit tcp any any range 11000 11999

permit udp any any eq 2427

permit tcp any any eq 2428

permit tcp any any range 2000 2002

permit udp any any eq 1719

permit udp any any eq 5060

!

rmon event 33333 log trap AutoQoS description "AutoQoS SNMP traps for Voice Drops" owner AutoQoS

rmon alarm 33337 cbQosCMDropBitRate.1493.1495 30 absolute rising-threshold 1 33333 falling-threshold 0 owner AutoQoS

Serial0/0 -

!

```
interface Serial0/0
 service-policy output AutoQoS-Policy-UnTrust
Router#
```

说明:可以看到接口 S0/0 已开启的 AutoQoS — VoIP，并且详细参数已生效。

企业网络其它接口的 AutoQoS — VoIP 配置类似，不再举例。

RSVP

概述

Resource Reservation Protocol (RSVP)被定义为 Signalling 技术，也就是 QOS 中的信号技术，RSVP 发出的信号，也就是一个主机或节点向网络中发出请求自己的数据流需要得到什么样的处理，也就是请求所需要的带宽。Signalling 是一个很有用的带宽请求技术，在端到端的 QOS 中起到非常重要的角色。

端到端的 QOS 要求网络路径中所有的设备（包括路由器和交换机，防火墙等等）都参与相同的 QOS 策略， RSVP 利用三层 IP 技术来穿透整个网络实施 Signalling，能实现这个目的，还有 IP 优先级技术，但是 IP 优先级用于区分服务，而 RSVP 一定会保证带宽。

RSVP 没有自己的路由协议，是根据当前已有的路由来决定路径的，如果路径改变了，那么 RSVP 请求也会重新计算，RSVP 并不能解决所有 QOS 问题，并且 RSVP 自己的申请过程也是要花时间的。

在应用程序申请到带宽后，RSVP 会做好记录的，发送的流量可以超过已申请到的带宽，但是只能保证在已申请到的带宽内，如果设置的空闲带宽足够够，就能提供超额带宽，如果不够了，也就会丢包。

RSVP with LLQ

RSVP 使用 WFQ 提供公平 QOS 服务，给别的流最低的优先级以保证其它流量，但 WFQ 不能保证语音的流量。

由于 LLQ 能为语音提供很好的带宽保证，所以要保证语音，RSVP 必须结合 LLQ，这样就可以将语音放入 LLQ 系统。

在 RSVP with LLQ 时，有以下一些限制：

不支持任何 tunnels

依靠 PQ，如果 LLQ 不支持，那么也不行

必须要支持 RSVP，WFQ 或 LLQ 要支持。

注：所有已经被 RSVP 匹配到的流量，是不会被其它技术所匹配的。RSVP 申请到的带宽，是单向的。

RSVP 在 Frame-relay 下时有以下限制：

不支持 GTS

不支持队列

非语音 RSVP 不支持

组播也不支持。

并且需要有：

RSVP

WFQ on the PVC

LLQ

Frame Relay Forum (FRF).12 on the interface

配置 RSVP 时，需要考虑：

一个用户流需要多少带宽，最多可用多少带宽，默认是可以用尽所有可用带宽

RSVP 所有会话可使用多少带宽，默认 75%，tunnel 是 100%

在 Frame-relay 下时，要为每个 DLCI 分配，命令 ip rsvp bandwidth 需要同时在主接口和子接口下打开。

配置 RSVP

1. 开启 RSVP（接口下）

（1）在接口 F0/0 下开启 RSVP

router(config)#int f0/0

router(config-if)#ip rsvp bandwidth 1000 500

说明:定义 RSVP 可用的总带宽为 1000K，单个流可保护的带宽为 500K。默认 RSVP 可用总带宽为接口的 75%，并且单个流可全用光。

2. 配置 RSVP with LLQ

(1)配置 LLQ 可用的带宽参数

router(config)#ip rsvp pq-profile 200000 8000

说明:LLQ 流最大可用 200KB，Bc 为 8KB。

3. 开启 RSVP with WFQ(全局必配)

(1)在接口下开启 WFQ，并且指定 WFQ 可用的带宽比

router(config)#int f0/0

router(config-if)#fair-queue

router(config-if)#fair-queue 50

说明:WFQ 为必开，可用带宽比为 50%

4. 配置 RSVP path

（1）配置 RSVP path，也就是请求带宽

router(config)#ip rsvp sender 100.1.1.1 10.1.1.2 tcp 80 10000 12.1.1.2 f0/0 200 10

说明:配置为目标地址 100.1.1.1，源地址 10.1.1.2，且目标端口为 TCP 80，源端口为 10000，上一跳为 12.1.1.2，进口为 f0/0 的流量请求保留带宽 200Kbit，Bc 为 10Kbit。

5. 配置 RSVP 保留带宽

（1）配置 RSVP 为某流量保留带宽

router(config)#ip rsvp reservation 100.1.1.1 10.1.1.2 tcp 80 10000 50.1.1.1 f0/1 ff rate 200 10

说明:配置为目标地址 100.1.1.1，源地址 10.1.1.2，且目标端口为 TCP 80，源端口为 10000，下一跳为 12.1.1.2，出口 为 f0/0 的流量保留单个流带宽 200Kbit，Bc 为 10Kbit。

6. 查看 RSVP

（1）查看 RSVP 配置情况

router#sh ip rsvp installed

RSVP: FastEthernet0/0 has no installed reservations

RSVP: FastEthernet0/1

BPS To From Protoc DPort Sport

200K 100.1.1.1 10.1.1.2 TCP 80 10000

router#

说明:可以看到 RSVP 保证带宽的情况。

7. 配置 Frame-relay 下的 RSVP

(1)配置 map-class

router(config)#map-class frame-relay FFF

router(config-map-class)#frame-relay cir 1000000

router(config-map-class)#frame-relay bc 10000

router(config-map-class)#frame-relay mincir 500000

router(config-map-class)#frame-relay fragment 100

router(config-map-class)#frame-relay fair-queue

说明:配置 cir 为 1000Kbit，Bc 为 10Kbit,Mincir 为 500Kbit, FRF.12 为 100Byte，并开启 WFQ。

（2）配置主接口的 RSVP

router(config)#int s0/0

router(config-if)#encapsulation frame-relay

router(config-if)#frame-relay traffic-shaping

router(config-if)#ip rsvp bandwidth 1000 200

说明:在主接口上封装 frame-relay，并开启 FRTS，以及开启 RSVP。

（3）在子接口应用 RSVP

router(config)#int s0/0.1 multipoint

router(config-subif)#ip rsvp bandwidth 1000 200

router(config-subif)#frame-relay interface-dlci 102

router(config-fr-dlci)#class FFF

说明:在 PVC 102 上应用 RSVP，并且子接口同样需要配置 RSVP 参数。

交换机 QOS （Switching QOS）

概述

之前所介绍的 QOS，并不能完全在交换机中实施，在交换机中，有些与路由器不同方式的 QOS，并且不是所有型号的交换机都具有相同的实施方式，在此文档中，由于 CCIE R&S 指定考试型号为 3560，所以仅针对 3560 为基础进行介绍。

在交换机上，流量进入和出去，是需要分开考虑的，在整台交换机上，流量进入时，有两个队列，进入交换机的所有流量，都根据这两个队列来处理，然后再将流量发送到内部环，最终从交换机上的接口被发出。

在流量进入交换机时，应该被分配到哪一个队列，是靠 SRR 来分配的。两个进的队列中，其中有一个是 PQ，默认为队列 2，也就是此队列中的流量可以被优先处理。

不仅流量在进入交换机时，需要依靠队列来决定先后顺序，而且在离开交换机时，也需要靠队列来决定。而出去的队列，是基于每个交换机接口的，交换机上每一个接口在出去时都有 4 个队列，流量属于哪一个队列，依靠数据包的 CoS 值。在出口的 4 个队列中，也有一个队列是被优先处理的，默认为队列 1，也就是只有队列 1 的流量被转发完了，才能转发其它队列的流量。

由上可见，所有流量在进入接口时，总带宽是可能超过交换机内部环的，所以在出去时，很有可能会被出口队列所编排顺序。无论在流量进入交换机或离开交换机，都有可能超过交换机的负载，对于超载的流量，交换机是需要将其丢弃的，默认为尾丢弃（Tail Drop），而这里的尾丢弃不同于普通的尾丢弃，因为这里的尾丢弃是有一定算法的，被称为 Weighted Tail Drop (WTD)。

WTD 在丢弃算法中，控制了每个队列的最大容量和丢弃阀值，也就是说队列中的流量是否该被丢弃，完全根据队列的最大容量和丢弃阀值来决定的。

WTD 在每个队列中根据数据包的 CoS 值将其分配到不同的阀值，每个队列拥有 3 个阀值，范围为 1%到 100%，其中前面两个阀值是可以任意配置的，而第三个则不可以，因为第三个阀值固定为 100%。这样一来，当某个 CoS 值的流量超过相应队列总容量的相应阀值之后，也就意味着这该流是要被丢弃的。

SRR Shaping and Sharing

在出口上，4 个队列依靠 Shaping 和 Sharing 的方式来分配接口总带宽。要注意，进口只有 Sharing 的模式。

队列都有自己的 weight 值，当队列为 Sharing 模式时，就是所有 Sharing 模式的队列根据各自的 weight 值来分配相应的接口带宽，比如 4 个 Sharing 模式的队列 weight 值分别为 10,20,30,40，那么将所有 weight 值相加 10+20+30+40=100，weight 值为 10 的队列分到和带宽为 10/100，weight 值为 20 的队列分到和带宽为 20/100，

weight 值为 30 的队列分到和带宽为 30/100，weight 值为 40 的队列分到和带宽为 40/100；当某个队列并没有流量传递时，那么这些空闲的带宽也同样按照各队列的 weight 值分配给每个队列使用。

而在队列为 Shaping 模式时，该队列依靠自己的 weight 值获得相应的带宽，如值为 3，则获得接口总带宽的 1/3，为 5 则获得接口总带宽的 1/5，并且该队列永远不能超过这个值，即使接口非常空闲也不能超过。无论是 Shaping 模式还是 Sharing 模式，分配到的带宽是保证的，但 Sharing 模式的队列在接口空闲时却可以完全利用。

当 Shaping 和 Sharing 同时存在于接口时，接口总带宽减去 Shaping 模式分配到的带宽后，剩余的带宽由 Sharing 模式分配。

前提配置

1．开启 QOS

（1）全局开启多层交换 QOS 功能

switch(config)#mls qos

说明:配置任何 QOS 之前，必须开启 QOS 功能。

（2）查看多层交换 QOS 功能

switch#sh mls qos

QoS is enabled

QoS ip packet dscp rewrite is enabled

switch#

说明:QOS 功能已经开启。

2. 给接口配置 CoS

（1）信任数据原来的 COS 值（应在 trunk 上）：

```
switch(config)#int f0/1
switch(config-if)#mls qos trust cos
```

（2）查看接口 CoS 值

```
switch#sh mls qos interface f0/1
FastEthernet0/1
trust state: trust cos
trust mode: trust cos
trust enabled flag: ena
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
Trust device: none
qos mode: port-based
switch
```

说明:接口信任数据原来的 COS 值，默认接口 COS 值为 0。

（3）给接口配置新的 COS 值：

```
switch(config-if)#mls qos cos 5
```

（4）查看接口 CoS 值

```
switch#sh mls qos interface f0/1

FastEthernet0/1

trust state: trust cos

trust mode: trust cos

trust enabled flag: ena

COS override: dis

default COS: 5

DSCP Mutation Map: Default DSCP Mutation Map

Trust device: none

qos mode: port-based

switch#
```

说明:接口默认的 COS 值为 5。

（5）去除数据包原有的 COS 值

```
 switch(config)#int f0/1

switch(config-if)#mls qos cos override
```

3. 配置映射：

说明:配置数据包各个参数之间的映射

（1）配置 CoS-to-DSCP map

```
switch(config)#mls qos map cos-dscp 5 10 15 20 25 30 35 40
```

（2）查看 CoS-to-DSCP map：

```
switch#sh mls qos maps cos-dscp
  Cos-dscp map:
    cos:  0  1  2  3  4  5  6  7
   ------------------------------
    dscp:  5 10 15 20 25 30 35 40

switch#
```

（3）配置 ip-prec-to-dscp map

```
switch(config)#mls qos map ip-prec-dscp 11 12 13 14 15 16 17 18
```

(4)查看 ip-prec-to-dscp map

```
switch#sh mls qos maps ip-prec-dscp
  IpPrecedence-dscp map:
   ipprec:  0  1  2  3  4  5  6  7
   ------------------------------
    dscp:  11 12 13 14 15 16 17 18

switch#
```

配置进口队列

1. 根据 COS 值将数据包放入进口队列和 WTD 阀值（全局配置）

（1）将 COS 值为 1、2、3 数据包放入队列 1 和阀值 1 中，将 COS 值为 4、5 数据包放入队列 2 和阀值 2 中

switch(config)#mls qos srr-queue input cos-map queue 1 1 2 3 1

switch(config)#mls qos srr-queue input cos-map queue 2 4 5 2

说明:默认为

Default CoS Input Queue Threshold Map	
CoS Value	Queue ID-Threshold ID
0-4	1-1
5	2-1
6, 7	1-1

2. 在进口配置 WTD 的两个阀值（全局配置）

(1)配置队列 1 的 WTD 前两阀值分别为 50%和 70%

switch(config)#mls qos srr-queue input threshold 1 50 70

说明:默认 3 个 WTD 阀值都为 100%，值就是可用缓存的百分比，

3. 调整两个进口队列使用缓存的百分比（全局配置）

（1）配置两个进口队列使用缓存的百分比分别为 50%

switch(config)#mls qos srr-queue input buffers 50 50

说明:默认分别为 90 和 10

percentage1 percentage2, the range is 0 to 100.

(2)查看进口队列使用缓存的百分比

switch#sh mls qos input-queue

Queue : 1 2

---

buffers : 50 50

bandwidth : 4 4

priority : 0 10

threshold1: 100 100

threshold2: 100 100

switch#

4．配置进口队列的 PQ：

可以配置进 Q 中的一个为 PQ（全局配置）

（1）配置队列 1 为 PQ，并且保证 20%的带宽

switch(config)#mls qos srr-queue input priority-queue 1 bandwidth 20

说明:配置队列 1 为 PQ，默认队列 2 为 PQ，总带宽的 20%可以保证，剩下的再由两个队列根据 weight 分配。

（2）查看接口队列

switch#sh mls qos input-queue

Queue : 1 2

---

buffers : 50 50

bandwidth : 4 4

priority : 20 0

threshold1: 100 100

threshold2: 100 100

switch#

说明:队列 2 为 PQ，并且保证带宽 20%

5．控制进口队列的可用带宽（全局配置）

(1)关闭 PQ 功能，并且分配两个队列的可用带宽比为 25%和 75%

switch(config)#mls qos srr-queue input priority-queue 1 bandwidth 0

switch(config)#mls qos srr-queue input bandwidth 25 75

说明:默认分别 4 、4

（2）查看进口队列

```
switch#sh mls qos input-queue
Queue     :       1       2
----------------------------------------
buffers   :      50      50
bandwidth :      25      75
priority  :       0       0
threshold1:     100     100
threshold2:     100     100
switch#
```


说明:两个队列的可用带宽比为 25%和 75%

配置出口队列

说明:出口有 expedite queue，也就是 PQ，配置这些队列的方式为配置 queue-set，总共为 2 个 queue-set，再将接口放入相应 queue-set，即得到相应 queue-set 的配置参数。

如果 expedite queue 打开了，默认队列 1 将优于 SRR shaped 和 shared 分配的 weight

如果 expedite queue 关了，默认队列 1 将被 shaped 模式取代

如果 expedite queue 关了，并且没有配置 SRR，则默认队列 1 在 shared 模式。

1. 为队列分配内存

（1）为 4 个队列分配可用的内存比例

switch(config)#mls qos queue-set output 1 buffers 20 20 20 40

说明:四个值加起来必须是 100%，默认分配为 25、25、25、25

2. 为队列配置 WTD 阀值和队列最小内存占用率于最大内存占用率(全局配置)

（1）配置 queue-set 1,将队列 1 的前两个 WTD 阀值分别设置为 50%和 70%，最小内存占用率于最大内存占用率分别为 50%和 80%

switch(config)#mls qos queue-set output 1 threshold 1 50 70 60 80

说明:默认全部为 100%

3. 在出口队列中根据 COS 值将数据包放入相应队列和相应 WTD 阀值(全局配置)

（1）配置 Cos 4 5 到队列 1 和 WTD 阀值 2

switch(config)#mls qos srr-queue output cos-map queue 1 threshold 2 4 5

说明:默认为

Default CoS Output Queue Threshold Map	
CoS Value	Queue ID-Threshold ID

0, 1	2-1
2, 3	3-1
4	4-1
5	1-1
6, 7	4-1

（2）查看 COS 值到相应队列和相应 WTD 阀值

```
switch#sh mls qos maps cos-output-q

  Cos-outputq-threshold map:

        cos:  0   1   2   3   4   5   6   7

        ------------------------------------

queue-threshold: 2-1 2-1 3-1 3-1 1-2 1-2 4-1 4-1

switch#
```


4. 将接口放入相应 queue-set（接口配置）

（1）将接口 f0/1 放入 queue-set 2

```
switch(config)#int f0/1

switch(config-if)#queue-set 2
```

说明:默认在 queue-set 1

（2）查看接口所在 queue-set

```
switch#sh mls qos interface queueing

FastEthernet0/1

Egress Priority Queue : disabled

Shaped queue weights (absolute) :  25 0 0 0

Shared queue weights  :  25 25 25 25

The port bandwidth limit : 100  (Operational Bandwidth:100.0)

The port is mapped to qset : 2
```

5. 接口下开 PQ：

（1）将接口 f0/1 开启 PQ

```
switch(config)#int f0/1

switch(config-if)#priority-queue out
```

说明： 默认 PQ 是关闭，且默认 PQ 是队列 1，在传数据时，只有所有 PQ 中的数据全部传完，才能传其它队列的数据。

（2）查看接口 PQ

说明:只能使用命令 show running-config

配置 SRR 中的 shared 和 shaped

1. 在接口为 4 个队列分配 weights 值（使用 shared 分配）

(1)分配 weights 值

switch(config)#int f0/1

switch(config-if)#srr-queue bandwidth share 20 20 20 20

说明:默认 4 个队列的 weights 值分别为 25、25、 25、 25

（2）查看接口队列的 weights 值：

switch#sh mls qos interface f0/1 queueing

FastEthernet0/1

Egress Priority Queue : enabled

Shaped queue weights (absolute) : 25 0 0 0

Shared queue weights : 20 20 20 20

The port bandwidth limit : 100 (Operational Bandwidth:100.0)

The port is mapped to qset : 2

switch#

2．配置接口的 shaped 模式

(1)将接口 f0/1 的队列 1 模式设置为 shaped，且使用接口带宽的 1/4

switch(config)#int f0/1

switch(config-if)#srr-queue bandwidth shape 4 0 0 0

说明:队列 1 为 shaped 模式，且使用接口带宽的 1/4，即 25%，值为 0 的队列表示在 shared 模式。

（2）查看接口的队列模式

switch#sh mls qos interface f0/1 queueing

FastEthernet0/1

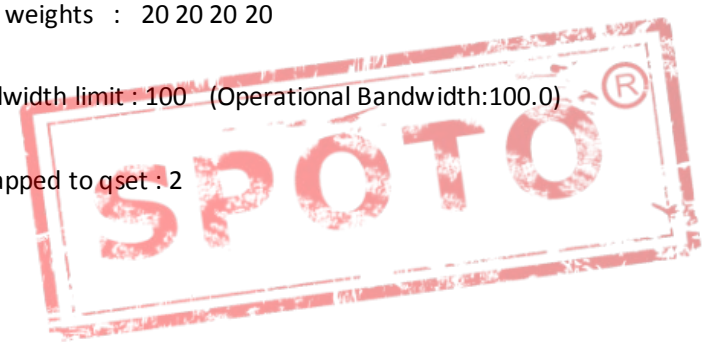
Egress Priority Queue : enabled

Shaped queue weights (absolute) : 4 0 0 0

Shared queue weights : 20 20 20 20

The port bandwidth limit : 100 (Operational Bandwidth:100.0)

The port is mapped to qset : 2



switch#

3. 配置接口的 shared 模式

配置接口 4 个队列的 weight 值分别为 1、2、3、4

(1)switch(config)#int f0/1

switch(config-if)#srr-queue bandwidth share 1 2 3 4

说明:接口 4 个队列的 weight 值分别为 1、2、3、4，所以每个队列可用带宽分别为 1/(1+2+3+4), 2/(1+2+3+4), 3/(1+2+3+4), and 4/(1+2+3+4)

(2)查看接口的模式

switch#sh mls qos interface f0/1 queueing

FastEthernet0/1

Egress Priority Queue : enabled

Shaped queue weights (absolute) : 4 0 0 0

Shared queue weights : 1 2 3 4

The port bandwidth limit : 100 (Operational Bandwidth:100.0)

The port is mapped to qset : 2

switch#

说明:因为队列 1 为 shaped 模式，占用带宽 25%，所以队列 2、队列 3、队列 4 的合用带宽为 75%，结果分别为接口总带宽的 2/(2+3+4)×75、3/(2+3+4)×75、4/(2+3+4)×75

4. 限制出口带宽

说明:比如用户只付了相应带宽的钱，所以要限制接口的可用带宽

（1）限制接口的可用带宽为总带宽的 80%

switch(config)#int f0/1

switch(config-if)#srr-queue bandwidth limit 80

说明:范围是 10 to 90.，默认是 100%。

(2)查看接口可用带宽

switch#sh mls qos interface f0/1 queueing

FastEthernet0/1

Egress Priority Queue : enabled

Shaped queue weights (absolute) : 4 0 0 0

Shared queue weights : 1 2 3 4

The port bandwidth limit : 80 (Operational Bandwidth:80.0)

The port is mapped to qset : 2

switch#

说明:接口可用带宽为 80%

配置 auto-QOS

1. 接口配置 Auto-QOS

（1）在接口 f0/1 开启 Auto-QOS

switch(config)#int f0/1

switch(config-if)#auto qos voip cisco-phone

或

switch(config-if)#auto qos voip trust

说明:开启了接口的 Auto-QOS，并自动为 cisco-phone 配置 QOS 策略，接口使用 CDP 发现 cisco-phone，

(2)查看接口 Auto-QOS

switch#show auto qos interface

FastEthernet0/1

auto qos voip cisco-phone

switch#

说明:接口已经启用 cisco-phone 的 Auto-QOS。