QOS 随身手册

资料来源：红头发文档、freebeme文档、华为 3com文档、www.cisco.com www.net130.com特此表示感谢

作者：流星之命 各大论坛ID：kkblue

联系方式 QQ：32427292 msn：bluebluenight@hotmail.com

技术讨论群：cisco黑暗精英群组

总群号码：2597707（已经人满）

分群号码：10996079（恭请加入）

胡言乱语部分：我要感谢CCTV 、channel V、我的唱片公司（cisco黑暗精英群组）和发行公司（www.net130.com），感谢我的朋友（以下排名不分先后）红头发、南极星、Vdsl、freebeme、lavender、microdot、cisconet、师傅吕品以及所有支持和帮助我的男女老少，你们都是活雷锋。最后感谢老婆把电脑让给我用，我才能写出东西在圈子里招摇拉风

真正的光明绝不是永没有黑暗的时间，只是永不被黑暗所掩蔽罢了。真正的英雄绝不是永没有卑下的情操，只是永不被卑下的情操所屈服罢了。

在这个温暖陌生的年代……

第一章　整体回顾

1.1 QOS 概念

英文解释 Quality of Service（服务质量保证），具体做什么的不解释了，个人认为主要应用于在带宽还不充裕的情况下可以平衡一下各种服务流量占用的矛盾，如果带宽足够的话，QOS 是没有任何使用价值的。

1.2 QOS 的服务模型

网络应用是端到端的通讯结构，比如两个不同网络的主机进行通讯，中间可能跨越各种 router 和核心 switch，那么想整体的实现所谓的 QOS，就必须全局考虑，QOS 的服务模型的概念就是采用通过什么模式全局实现服务质量保证，一共分成三种。

Best-Effort service 尽力而为服务模型

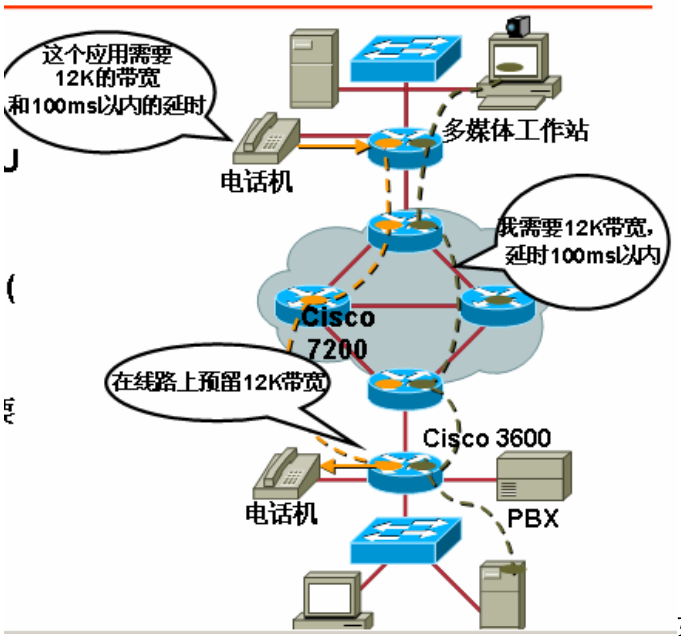
Integrated service 综合服务模型 简称 Intserv

Differentiated service 区分服务模型 简称 Diffserv

1.2.1 Best-Effort 服务模型

Best-Effort 是一个单一的服务模型，也是最简单的服务模型，应用程序可以在任何时候，发出任意数量的报文，而且不需要事先获得批准，也不需要通知网络，对 Best-Effort 服务，网络尽最大的可能性来发送报文，但对时延、可靠性等性能不提供任何保证 Best-Effort 服务是现在 Internet 的缺省服务模型，它适用于绝大多数网络应用，如 FTP、 E-Mail 等。其实 best-effort 并非是什么 QOS，就是互联网的简单数据传输方式而已，有什么传什么，阻塞也就阻塞了，丢且也就丢弃了。只不过 QOS 指一种全局网络通讯的质量保证机制，不得不把这种低智商的服务模式提及一下

1.2.2 Intserv 服务模型



如图所示（摘自 cisco.com.cn）

Intserv：集成服务模型，它可以满足多种 QoS 需求。这种服务模型在发送报文前，需要向网络申请特定的服务。应用程序首先通知网络它自己的流量参数和需要的特定服务质量

请求：包括带宽、时延等。应用程序一般在收到网络的确认信息，即确认网络已经为这个应用程序的报文预留了资源后，才开始发送报文，同时应用程序发出的报文应该控制在流量参数描述的范围以内。

网络在收到应用程序的资源请求后，执行资源分配检查 Admission control 即基于应用程序的资源申请和网络现有的资源情况，判断是否为应用程序分配资源，一旦网络确认为应用程序的报文分配了资源，则只要应用程序的报文控制在流量参数描述的范围内，网络将承诺满足应用程序的 QoS 需求。而网络将为每个流 flow 由两端的 IP 地址、端口号、协议号确定、维护一个状态，并基于这个状态执行报文的分类、流量监管、policing、排队及其调度来实现对应用程序的承诺。

在 IntServ 服务模型中，负责传送 QoS 请求的信令是 RSVP（Resource Reservation Protocol）资源预留协议，它通知路由器应用程序的 QoS 需求。RSVP 是在应用程序开始发送报文之前来为该应用申请网络资源的。 Intserv 实际上是一种对服务的预定机制，通过申请来获取相应得服务，这里面主要依靠的就是 RSVP——资源预留协议。做一个简单介绍

RSVP 是第一个标准 QoS 信令协议，它用来动态地建立端到端的 QoS，它允许应用程序动态地申请网络带宽等。RSVP 协议不是一个路由协议，相反，它按照路由协议规定的报文流的路径为报文申请预留资源，在路由发生变化后，它会按照新路由进行调整，并在新的路径上申请预留资源。RSVP 只是在网络节点之间传递 QoS 请求，它本身不完成这些 QoS 的要求实现，而是通过其他技术来完成这些要求的实现。**（RSVP 只是一种用来预定的协议）**

RSVP 的处理是接收方发出资源请求，按照报文发送的反向路径发送资源请求，所以它可以满足非常大的多播组，多播组的成员也可以动态变化，RSVP 协议是针对多播设计的 单播可以看作是多播的一个特例。

由于 RSVP 在 Internet 上还没有得到广泛的推广，在主机不支持 RSVP 的情况下，我们可以通过配置 RSVP 代理，即代替不支持 RSVP 的主机发送 RSVP 报文来获得这种服务，对报文流路径上不支持 RSVP 的路由器，它只需要简单的转发 RSVP 报文 所以对 RSVP 协议不会有太大影响，但这些节点不会对报文提供所要求的 QoS 。**（这是 RSVP 的一个缺点）**

RSVP 信令在网络节点之间传送资源请求，而网络节点在收到这些请求后，需要为这些请求分配资源，这就是资源预留。网络节点比较资源请求和网络现有的资源，确定是否接受请求，在资源不够的情况下，这个请求可以被拒绝，可以对每个资源请求设置不同的优先级。这样，优先级较高的资源请求可以在网络资源不够的情况下，抢占较低优先级的预留资源，来优先满足高优先级的资源请求。

资源预留判断是否接受资源请求，并承诺对接受了的资源请求提供请求的服务，但资源预留本身不实现承诺的服务，需要通过队列等其他技术来实现。

上面的描述都是大白话，相信大家看完之后都能对 Intserv 这种服务模型和 RSVP 有所了解，就是两个关键字——预定，你理解成预约也行。Intserv 有它的好处，但是也有严重缺点，首先就是 RSVP 协议数据太多，而且不断刷新，并且这种给单一数据流的路径进行带宽预留的解决思路在浩瀚的 Internet 上实现简直是不可能的，而且 RSVP 的部署，厂商之间设备的互联，业务管理方面 等存在着种种问题，所以这么模型在 1994 年推出之后就没有获得任何规模的商业应用。

1.2.3 DiffServ 服务模型

DiffServ 是一个多服务模型，它可以满足不同的 QoS 需求，与 IntServ 不同，它不需要使用 RSVP 即应用程序在发出报文前，不需要通知路由器为其预留资源，对 DiffServ 服务模型，网络不需要为每个流维护状态 ，它根据每个报文指定的 QoS 来提供特定的服务 可以用不同的方法来指定报文的 QoS，如 IP 报文的优先级位 IP Precedence) ，报文的源地址和

目的地址等，网络通过这些信息来进行报文的分类、流量整形、流量监管和队列调度。

DiffServ 一般用来为一些重要的应用提供端到端的 QoS 它通过下列技术来实现

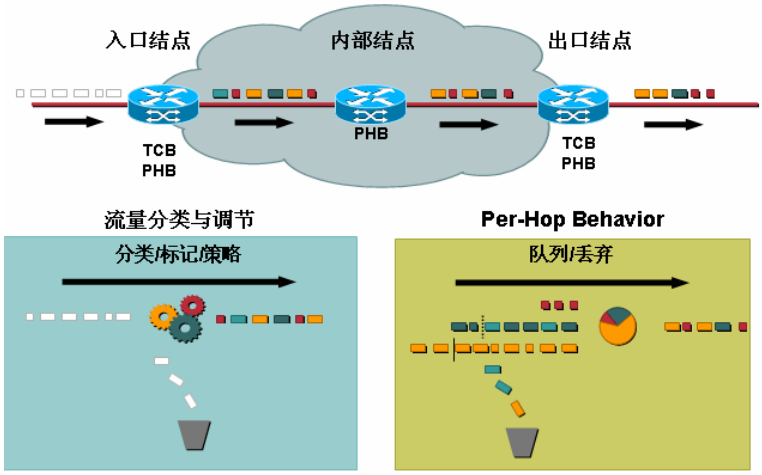
CAR：它根据报文的 ToS 或 CoS 值（对于 IP 报文是指 IP 优先级或者 DSCP 等等）IP 报文的五元组（指源地址目的地址协议端口号）等信息进行报文分类，完成报文的标记和流量监管。

队列技术：WRED、PQ、CQ、WFQ、CBWFQ 等队列技术对拥塞的报文进行缓存和调度，实现拥塞管理。

第二章 QOS 在 Packet 或者 Frame 的中的分类

2.1 基础知识

通常在配置 DiffServ 时，边界路由器通过报文的源地址和目的地址等对报文进行分类，对不同的报文设置不同的 CoS 值，而其他路由器只需要用 CoS 值来进行报文的分类。

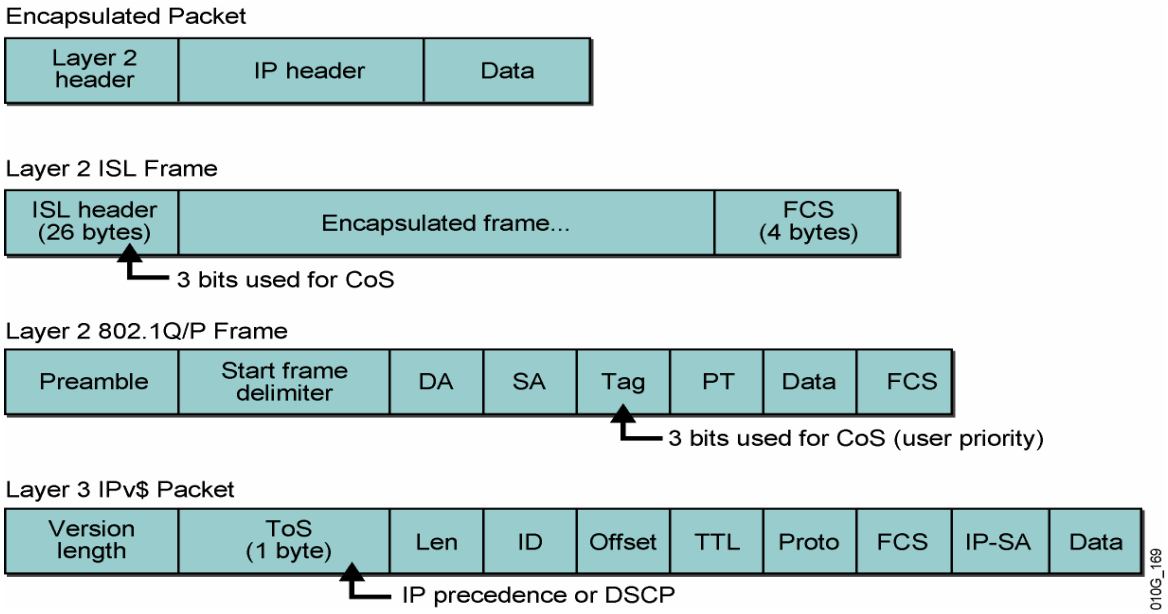


Diffserv 服务模型

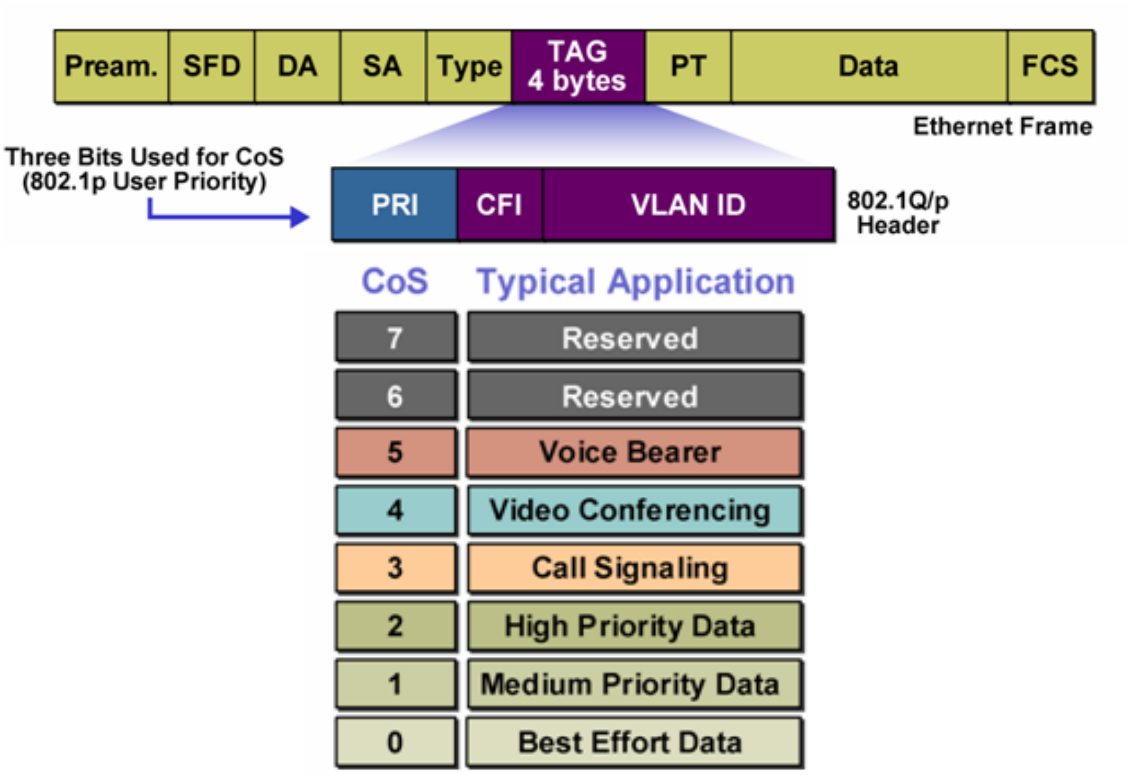
个人理解：如果把流量想象成福利彩票里面的乒乓球的话，这种服务模型实际上是把这些乒乓球标号，如果带宽完全够用，那么这些乒乓球肯定会随意的传递，但是如果出现阻塞时，或者带宽较低又需要保证某些流量的带宽质量时，就可以针对于不同标号的乒乓球执行相应的策略（比如配置队列）。

2.2 DSCP

这种标记在 frame 中和 packet 中本身就存在，frame 中存在 cos 字段，packet 中有 tos 字段，只不过在正常的传输模型中（例如 best-effort）并不使用，而在 Diffserv 中，数据包的标记作用就体现出来了，并且在 packet 中，Diffserv 提出了一个新的标记，就是 DSCP（Diffesentiated Services Code Point ）区分服务代码点



如上图所示，正常的以太网 frame 中是不存在标记的，但是 ISL 和 dot1q 的 frame 中有三个 bit 定义服务级别，一共有 6 个服务级别可以使用



需要关注的是，Frame 中的 cos 只使用 0-5，6-7 并不使用。

而针对于数据包，却有两种标识服务类型的方法（如下图所示）分别是 IP precedence（ip 优先级）和 DSCP（区分服务代码点）

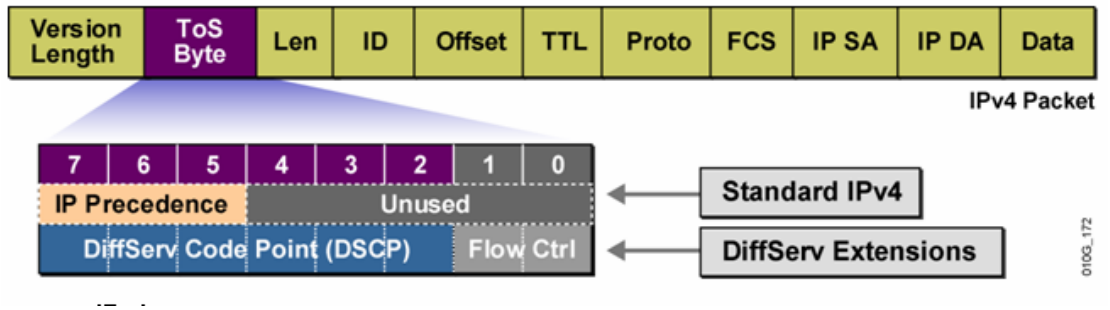


Table 3 IP Precedence Values

Number	Name
0	routine
1	priority
2	immediate
3	flash
4	flash-override
5	critical
6	internet
7	network

TOS 整个字段一共 8 位，图中上面部分是 ip precedence 的标识方法，使用前三位，也就是 P0、P1、P2 一共三位，共 8 个 bit 位，也就是 8 个优先级，分别是 0-7，其中 6 和 7 一般保留，常用的是 0-5,图中下面部分介绍的事 IP 优先级的含义，提供这个表格的意义在于进行更改数据包优先级等配置时，，我们既可以使用数字，也可以使用名称。

在来看看 DSCP，它使用 tos 中的前 6 个字节，即 DS1-DS5，（如上图所示）定义了 0-63 一共 64 个优先级。

分类是没有范围限制的，也就是说我们可以对数据进行灵活的分类，比如说把某一个源 ip 到目的 ip 的流量优先级进行更改 IP 优先级或者 DSCP 的操作。也可以定义去更改某一个特定的流量（扩展控制列表）的优先级。

通常于网络边界处对报文进行分类时，同时标记 IP 优先级或 DSCP。这样，在网络的内部就可以简单的使用 IP 优先级或 DSCP 作为分类的标准，而队列技术如 WFQ 等 CBWFQ 就可以使用这个优先级来对报文进行不同的处理。

什么是标识，实际上就是通过更改这些优先级字段将这些数据分出种类来，即便是上面的图标中有所谓的什么 0-7 优先级，好似 7 要比 0 就会优先级大一些，但是一定要清楚这只是区分，执行的策略要靠后面的队列机制来解决，实际上之所以这样定义优先级我认为只是为了制定一个共同遵守的类别优先标准，没有实际的意义，真正的操作是在配置上针对于不同优先级采用的措施——例如在队列里面使用什么标识的数据包属于什么队列等等。

但是在进行配置时，由于存在二三层优先级的对应关系，所以下面这个表格请各位惠存，以便不时之需

DCSP-to-Transmit queue map	DSCP 0-15 Queue 1 DSCP 16-31 Queue 2 DSCP 32-47 Queue 3 DSCP 48-63 Queue 4

上表是 DSCP 与 WRR 中应用的队列匹配。

CoS to DSCP map (DSCP set from CoS values)	CoS 0 = DSCP 0 CoS 1 = DSCP 8 CoS 2 = DSCP 16 CoS 3 = DSCP 24 CoS 4 = DSCP 32 CoS 5 = DSCP 40 CoS 6 = DSCP 48 CoS 7 = DSCP 56
DSCP to CoS map (CoS set from DSCP values)	DSCP 0-7 = CoS 0 DSCP 8-15 = CoS 1 DSCP 16-23 = CoS 2 DSCP 24-31 = CoS 3 DSCP 32-39 = CoS 4 DSCP 40-47 = CoS 5 DSCP 48-55 = CoS 6 DSCP 56-63 = CoS 7

上表是 COS 和 DSCP 彼此匹配时的对应关系。

OK，在实际的网络通讯中，packet 也好，frame 也好，实际上都是没有什么优先级的，也就是说如果我们不做任何设置，优先级等于 0，那么要想分类，就可以去更改某些数据包的优先级，这也是我们去执行 QOS 策略的基础，就像刚才我说的，先要给乒乓球标号。

第三章 标记（Marking）

3.1 理论知识

标记也就是着色的工作，也就是修改IP优先级或者DSCP，但是由于IP优先级和DSCP都是占用TOS字段，后者相当于前者的扩展,所以不能同时设置这两种值,如果同时设置了这两种值,那么只有IP DSCP 的值生效。标记是后续很多QOS策略应用的基本，使用的是policy map。以下文字会引用部分红头发文档，尊重版权。

3.2 配置方法

1、先定义class map，其实这是一个匹配列表，我觉得就是一种超级增强型的ACL，可以定义规定的特定的流量，而后面的policy map就可以对class map里面的东东进行更改操作

kkblue(config)#class-map [match-all|match-any] {*map-name*}

创建一个class map，可以match-all（匹配所有条件，这是默认配置）也可以match-any（至少要有一个条件匹配），后面的{*map-name*}就是你要写的class-map的名字。

2、定义匹配语句:

kkblue(config-cmap)#{*condition*}

以下是常用的condition配置选项

match access-group {*ACL*}　　匹配IP ACL （主要就是对应数据包了）

match protocol {*protocol*}　　匹配协议（这个在NBAR—基于网络应用中使用，使用时可以敲？察看一下）

match input-interface {*interface*}　　匹配进站接口

match qos-group {*Group ID*}　　匹配组ID（不知道干啥的）

match destination-address {mac *mac-address*}　匹配目标MAC 地址

match source-address {mac *mac-address*}　　匹配源MAC 地址

match ip {dscp *dscp*}　　匹配IP DSCP 值

match ip {precedence *precedence*}　　匹配IP 优先级

match class-map {*map-name*}　　匹配class map（class map嵌套）

match vlan {*vlan-id*}　　匹配VLAN

3、设置policy map:（创建一个policy map）

kkblue(config)#policy-map {*policy-name*}（后面填写的就是policy-map的名称）

4、调用class map:

kkblue(config-pmap)#class {*class-map*}

5、设置标记:

kkblue(config-pmap-c)#{*action*}

一些用于标记的动作选项:

set ip {precedence *precedence*}　　设置IP 优先级

set ip {dscp *dscp*}　　设置IP DSCP 值

set qos-group {*Group ID*}　　设置组ID

set cos {*cos*}　　设置CoS 值

priority {*kbps*|percent *percent*} [*Bc*]　　定义优先级流量的保留的带宽以及突发流量

bandwidth {*kbps*|percent *percent*}　　定义保留的带宽

police {*CIR Bc Be*} conform-action {*action*} exceed-action {*action*} [violate-action {*action*}]　　使用令牌桶算法进行限速

random-detect　　启用WRED
queue-limit {*packets*}　　定义队列中数据包的最大个数
service-policy {*policy-map*}　　使用别的策略进行嵌套,做为match语句匹配的标准
shape {average|peak} {*CIR* [*Bc*] [*Be*]}　　定义CIR,Bc 以及Be 进行整形

总结一下，实际上她们之间的存在一个模块调用关系，class map是一个匹配模块，里面我们配置中所使用种种match指定的流量，而policy map是执行模块，将class map模块中的东东进行更改，当然，可以class map可随意被任何一个policy map调用，policy做各种更改不受限制，但是个人觉得在实际的工程里面还是以简洁为主，就别耍花枪了

那么接下来就是要执行这种标记的工作了，在加载到接口上执行，

kkblue（config-if）service-policy　[input|output] *policy-name*

想在哪个接口上执行，就加载到哪个接口，input 指的是进站流量，output 指的是出站流量，后面填写的就是 policy map 的名称了。

3.3 察看命令

kkblue#show policy-map [*policy-name*]
2.查看接口的policy map 信息:
kkblue#show policy-map interface [*interface*]

3.4　配置案例

3.4.1　Case one

把来自192.168.10.0/24 的出站telnet 流量的IP优先级设置为5,其他的出站流量的IP优先级设置为1:

```
access-list 133 permit tcp 192.168.10.0 0.0.0.255 any eq telnet
（ACL133定义特定流量）
class-map match-all telnet
 （telnet是class map的名称，而不是什么协议噢）
match access-group 133                    （调用ACL133）
policy-map kkblue                         (kkblue是policy map的名称)
class telnet                              (policy调用名字叫做telnet的class map)
    set ip precedence 5                   （把优先级修改成5）
class class-default                       （class-default指的就是其他的数据）
    set ip precedence 1                   （优先级次修改为1）
interface Serial1
ip address 10.0.0.1 255.255.255.252
service-policy output kkblue              （在接口上执行）
```

这就是一个简单的分类标记工作，实际工程中可能会命令会多一点，复杂一点，但是道理都是一样的。

3.5 附加知识

关于class－map嵌套:有两点理由在创建class map的时候去调用一个已有的class map:
1、管理方便,在已有的基础上增加一个修改进行平滑的过度。
2、允许用户在同一个class map里分别使用匹配所有(match-all)和匹配任何(match-any)。比如4个匹配标准：A、B、C和D。现在想让class map 匹配A，或匹配B，或同时匹配C和D，

就可以使用class map的嵌套：创建一个新的class map，定义为匹配所有(match-all)新标准为匹配E即同时匹配C和D；然后定义另一个匹配任何(match-any)的class map，去匹配A，或B，或E(即同时匹配C 和D)。

3.5.1 Case one 使用嵌套的class map 做为匹配标准并进行限速:

```
class-map match-any parent      (parent是class map名称)
match access-group 1
match class-map child            (调用child这个class map)
class-map match-all child        (child是class map名称)
match input-interface Serial0
match destination-address mac 1.1.1
!
policy-map kkblue                (kkblue是policy map的名称)
class parent                      (调用parent这个class map)
police 8000 2000 4000 conform-action transmit exceed-action set-qos-transmit 25
violate-action drop               (其余的是策略，可以先不看)
!
interface Serial1
ip address 172.16.0.1 255.255.255.252
service-policy input kkblue       (接口上加载kkblue)
```

点评：实际上parent使用的是match any，要么符合access－group ，要么符合class map child。Child中的要求match any。

3.5.2 Case two 使用嵌套的policy map 进行策略:

```
!
class-map match-all kkblue          (class map的名称是kkblue)
match access-group 1
!
policy-map parent                    (policy map名称是parent)
class class-default
police 8000 2000 4000 conform-action transmit exceed-action set-qos-transmit 25
violate-action drop                  (执行的策略先不去考虑，后面有介绍)
service-policy child                 (policy map调用了child这个policy map)
policy-map child                     (创建child这个class map)
class kkblue                          (加载映射class map)
shape average 10000000              (策略，先不研究)
!
interface Serial1
ip address 172.16.0.1 255.255.255.252
service-policy input parent          (加载接口上)
!
access-list 1 permit 192.168.0.0 0.0.0.255
```

嵌套，就是套着用，体现出灵活

第四章 NBAR应用

4.1 理论知识

我们来看在做了分类之后的一个常规应用，就是NBAR。基于网络的应用程序识别(NBAR)可以对使用动态分配TCP/UDP 端口号的应用程序和HTTP 流量等进行分类。在使用NBAR的时候要先启用CEF特性。而且可以使用数据包描述语言模块(PDLM)从路由器的闪存里加载，用于在不使用新的Cisco IOS 软件，或重启路由器的情况下对新的协议或应用程序进行识别。

4.2 注意事项

NBAR不能在以下几种逻辑接口上使用：

1、快速以太网信道

2、使用了隧道或加密技术的接口

3、SVI

4、拨号接口

5、多链路PPP(MLP)

4.3 使用限制

1、不支持多于24个的并发URL，HOST或MINE的匹配类型

2、不支持超过400字节的URL 匹配

3、不支持非IP流量

4、不支持组播或其他非CEF的交换模式

5、不支持被分片的数据包

6、不支持源自或去往运行NBAR 的路由器的IP流

4.4 配置步骤

1、启用CEF 特性：

```
kkblue(config)#ip cef
```

2、把流量分类,定义class map：

```
kkblue(config)#class-map [match-all|match-any] {map-name}
```

3、定义NBAR 要匹配的协议：

```
kkblue(config-cmap)#match protocol {protocol}
```

4、设置policy map：

```
kkblue(config)#policy-map {policy-name}
```

5、调用class map：

```
kkblue(config-pmap)#class {class-map}
```

6、设置策略：

```
kkblue(config-pmap-c)#{action}
```

7、把策略应用在接口上：

```
kkblue(config-if)#service-policy {input|output} {policy-map}
```

4.5 察看命令

1、查看流量分类信息： kkblue#show class-map [*map-name*]

2、查看policy map： kkblue#show policy-map [*policy-name*]

3、查看接口的policy map 信息： kkblue#show policy-map interface [*interface*]
4、显示NBAR所使用的PDLM： kkblue#show ip nbar pdlm
5、显示NBAR使用的协议到端口号的映射信息：kkblue#show ip nbar port-map
和刚才的配置大同小异，还是看看红头发大虾给我们提供的案例吧，非常经典。

4.6 配置案例

4.6.1 Case one

使用 NBAR 识别 BitTorrent 程序流量

1、加载bittorrent.pdlm 到路由器闪存里（事先要把pdlm复制到flash中，哪有pdlm？baidu搜搜，cisco网站找找，实在不行朝红头发要，他写的配置他肯定有）

```
kkblue(config)#ip nbar pdlm flash://bittorrent.pdlm
```

2、定义class map,识别BitTorrent程序流量,并将进站的BitTorrent程序流量丢弃

```
ip cef
!
class-map bittorrent     （bittorrent是名称）
match protocol bittorrent     （这个才是协议咧）
!
policy-map drop-bittorrent     （policy-map的名称）
class bittorrent
drop                              （执行策略：丢就一个字，）
!
interface Serial0
ip address 192.168.0.1 255.255.255.0
service-policy input drop-bittorrent     （加载到接口上）
!
```

4.6.2 Case two

用NBAR 对进站的HTTP 流量下载进行限速，其中凡是下载的图象格式包括jpg,jpeg 和gif的，速率限制为100kbps。配置如下：

```
ip cef
!
class-map match-any HTTP
match procotol http url "*.jpeg|*.jpg"  (匹配url中带有jpeg和jpg的连接)
match procotol http url "*.gif"          (匹配url中有gif的连接)
!
policy-map kkblue
class HTTP
police 100000 conform-action transmit exceed-action drop （限制速率为100k，如果超过该速率会采用尾部丢弃）
!
interface Serial0
ip address 10.0.0.1 255.255.255.252
```

service-policy input kkblue　　（在进站执行策略）

4.6.3 Case Three

使用NBAR 来防止红色代码(Code Red)和尼姆达(Nimda)蠕虫病毒，配置如下

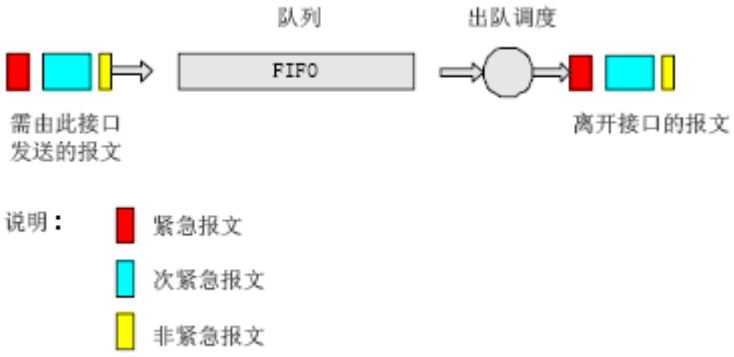
```
ip cef
!
class-map match-all DENY-ATTACK
match protocol http url "*.ida*"
match protocol http url "*cmd.exe*"
match protocol http url "*root.exe*"
match protocol http url "*readme.eml*"
```
（把URL中包含这些名字的连接都挑出来，然后统一执行策略，丢！！！）
```
!
policy-map kkblue
class DENY-ATTACK
drop
interface Serial0
ip address 10.0.0.1 255.255.255.252
service-policy input kkblue
```

第五章　拥塞管理

5.1前言

终于到了最大块头的东东上了，就是拥塞管理技术，QOS的策略也就在这里体现出来，队列技术的原理就是使报文在路由器中按一定的策略暂时缓存到队列中然后再按一定的调度策略把报文从队列中取出在接口上发送出去，常见的队列技术有很多，一个一个牵出来看看，但是注意哦，这些队列技术在接口中只能使用一个

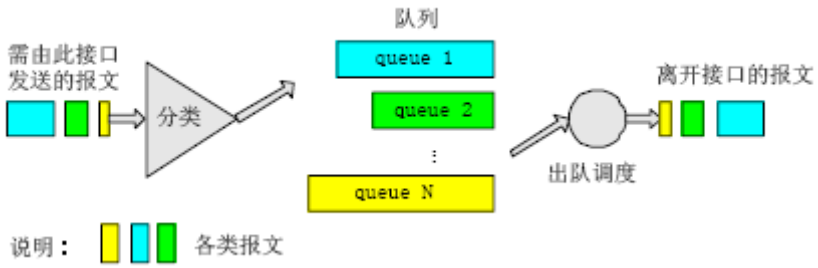
5.2 先进先出队列（First In First Out Queuing, FIFO）



如图所示FIFO队列，不对报文进行分类，当报文进入接口的速度大于接口能发送的速度时，FIFO按报文到达接口的先后顺序让报文进入队列，同时FIFO在队列的出口让报文按进队的顺序出队先进的报文将先出队后进的报文将后出队。注意：**当没有使用其他的队列机制时，除了传输速率小于2.048Mbps 的串行接口以外的所有接口，默认都使用这种队列机制，比如以太网口等。**

5.3　加权公平队列（Weighted Fair Queuing, WFQ）

5.3.1 基本原理



如图所示， WFQ对报文按流进行分类（对于IP网络相同源IP地址、目的IP地址、源端口号、目的端口号、协议号、IP优先级的报文属于同一个流）每一个流被分配到一个队列，该过程称为散列，采用HASH算法来自动完成。尽量将不同的流分入不同的队列，WFQ的队列数目N可以配置，在出队的时候WFQ按流的IP优先级来分配每个流应占有出口的带宽优先级的数值越小所得的带宽越少，优先级的数值越大所得的带宽越多，这样就保证了相同优先级业务之间的公平，体现了不同优先级业务之间的权值。例如接口中当前有8个流它们的优先级分别为0、1、2、3、4、5、6、7、则带宽的总配额将是所有流的优先级 + 1 之和即1 + 2 + 3

+ 4 + 5 + 6 + 7 + 8 = 36。

每个流所占带宽比例（为自己的优先级数 + 1）/ （所有（流的优先级 + 1)之和）即每个流可得的带宽比例分别为1/36、2/36、3/36、4/36、5/36、6/36、7 /36、8/36。

又比如当前共4个流，3个流的优先级为4，1个流的优先级为5，则带宽的总配额将是

(4 + 1) * 3 + (5 + 1) = 21

那么3个优先级为4的流获得的带宽比例均为5/21，优先级为5的流获得的带宽比例为6/21。由此可见WFQ在保证公平的基础上对不同优先级的业务体现权值，而权值依赖于IP报文头中所携带的IP优先级。**注意： WFQ是传输速率低于2.048Mbps的串行接口默认的队列机制**。但是WFQ存在一些限制：第一个是WFQ不支持隧道或采用了加密技术的接口，因为这些技术要修改数据包中WFQ 用于分类的信息。第二个WFQ提供的带宽控制的精确度不如CBWFQ和CQ等队列机制。

5.3.2 配置命令

接口下启用WFQ: kkblue(config-if)#fair-queue

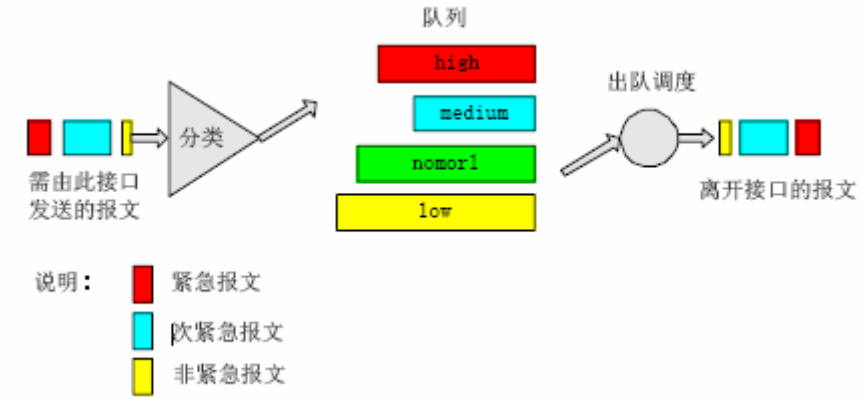
显示公平队列的配置状态: kkblue#show queueing fair

显示接口的队列信息: kkblue#show queue [*interface*]

上面介绍的两个队列其实不需要什么配置，可见QOS原本在接口上就有体现，只不过是最基本的操作（说白了就不是什么QOS，尽力转发而已），下面来点难度

5.4 优先级队列（Priority Queuing, PQ）

5.4.1 基本原理



如图所示， PQ对报文进行分类，对于IP网络可以根据IP报文的优先级/DSCP等条件进行分类，将所有报文分成最多至4类，分别属于PQ的4个队列中的一个，然后按报文的类别将报文送入相应的队列。PQ的4个队列分别为高优先队列、中优先队列、正常优先队列和低优先队列、它们的优先级依次降低。在报文出队的时候，PQ首先让高优先队列中的报文出队并发送，直到高优先队列中的报文发送完，然后发送中优先队列中的报文，同样直到发送完，然后是正常优先队列和低优先队列，这样分类使属于较高优先级队列的报文将会得到优先发送，而较低优先级的报文将会在发生拥塞时被较高优先级的报文抢先，使得高优先级业务如VoIP 的报文能够得到优先处理较低优先级业务，如E-Mail 的报文在网络处理完关键业务后的空闲中得到处理既保证了高优先级业务的优先又充分利用了网络资源。

PQ使用的限制和缺点：第一，由于PQ是静态配置的，因此它不能适应网络结构的改变。第二，由于数据包要经过处理器卡的分类，因此PQ对数据包转发的速度要比FIFO慢。第三，

PQ 不支持隧道接口。另外PQ一个非常显著的缺点就是如果高优先级的队列没有发送完成，低优先级的数据将永远不会发送，造成两级分化，使较低优先级的数据转发困难。

5.4.2 配置命令

1、定义优先级列表,可以基于协议或基于进站接口：

基于协议：kkblue(config)#priority-list *list-number* protocol *protocol-name* {high|medium|normal|low} *queue-keyword keyword-value*

priority-list号码为1-16，关注一下红色标记部分，其实我们可以添加一些扩充选项来准确定位流量比如

```
fragment              (IP packets with non-zero fragment offset)
gt/lt <size>          based on packet size (including L2 frame)
list <acl>            ACL classification
tcp/udp <port>        TCP or UDP port number
```

前面是配置参数，后面是解释，常用的是最后三个参数。

基于进站接口：kkblue(config)# priority-list list-number interface interface-type interface-number {high | medium | normal | low}

2、定义默认的优先级队列,未分类的流量默认被分配进该队列,优先级默认为normal:

kkblue(config)#priority-list {*list*} default {high|medium|normal|low}（可以不写）

3、定义每个队列中数据包的最大个数,由高到低,默认为20,40,60 和80.可以更改:

kkblue(config)# priority-list {*list*} queue-limit {*high-limit medium-limit normal-limit low-limit*}

4、把优先级列表应用在接口上:

kkblue(config-if)#priority-group {*list*}

5.4.3 察看命令

1、显示接口队列信息： kkblue#show queue [*interface*]

2、显示PQ 列表信息: kkblue#show queueing priority

5.4.5 配置案例

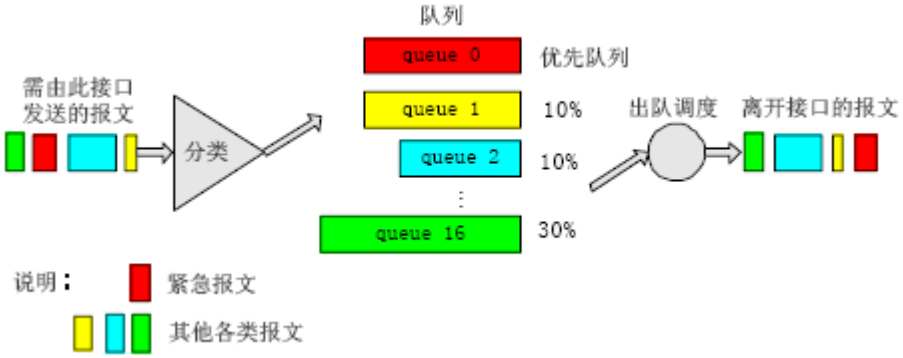
要求sna流量高优先级，www 的ip 流量低优先级别，其他流量中等优先级别，给sna流量设置队列深度为50消息

```
interface serial 0      //在接口下应用队列
priority-group 1

priority-list 1 protocol sna high      //sna 是高优先级
priority-list 1 protocol ip low tcp 80    //www 是低优先级
priority-list 1 protocol ip medium       //其他 ip 流量是中等优先级
priority-list 1 queue-limit 50 40 60 80    //高优先级队列，队列深度 50
```

这个案例里面就没有写为分类的数据优先级是否改动，配置优先级队列时 protocol 后面有哪些协议，各位敲个？自己看看吧。PQ 虽好，但是缺点也不少，下面看看 CQ

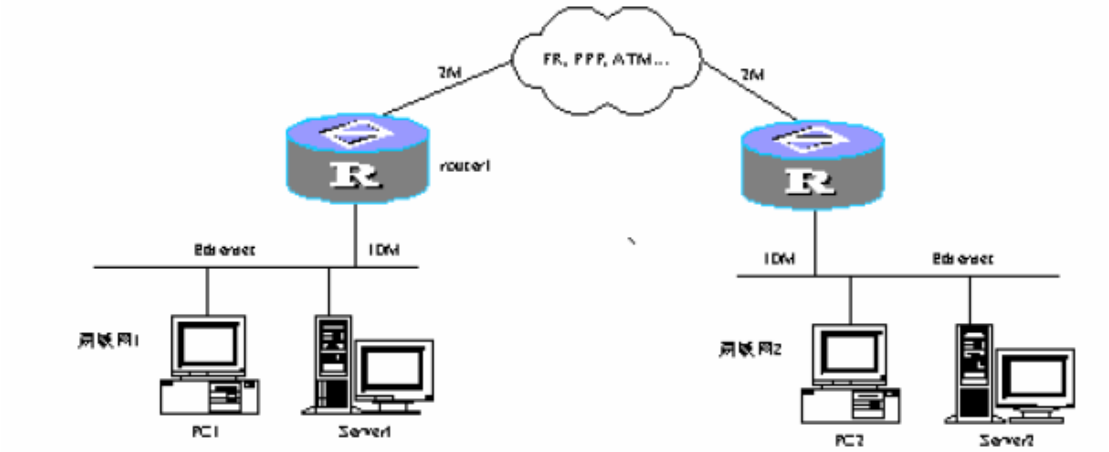
5.5 自定义队列（Custom Queuing, CQ）

5.5.1 基本原理



如图所示 CQ对报文进行分类，报文分成最多16类，分别属于CQ的16个队列中的一个，然后按报文的类别将报文送入相应的队列，实际上队列的号码是0-16一共17个，但是0号队列是超级优先队列，路由器总是先把0号队列中的报文发送完然后才处理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，通常把实时性要求高的交互式协议和链路层协议报文放到0号队列中。1到16号队列可以按用户的定义分配它们能占用接口带宽的比例，在报文出队的时候，CQ按定义的带宽比例分别从1到16号队列中取一定量的报文在接口上发送出去。

可以将CQ和PQ做个比较，PQ赋予较高优先级的报文绝对的优先权，这样虽然可以保证关键业务的优先，但在较高优先级的报文的速度总是大于接口的速度时，将会使较低优先级的报文始终得不到发送的机会。采用CQ则可以避免这种情况的发生，CQ可以把报文分类然后按类别将报文分配到CQ的一个队列中去，而对每个队列又可以规定队列中的报文所占接口带宽的比例，这样就可以让不同业务的报文获得合理的带宽，从而既保证关键业务能获得较多的带宽，又不至于使非关键业务得不到带宽。但是由于CQ轮循调度，各个队列它对高优先级尤其是实时业务的时延保证不如PQ。



在如图所示的网络图中，假设局域网1的服务器向局域网2的服务器发送关键业务的数据，局域网1的PC向局域网2的PC发送非关键业务的数据，如果对路由器1的串口1配置CQ进行拥塞管理，同时配置服务器间的数据流的进入队列1，队列1中的报文占有60%的带宽，例如每次出队6000个字节的报文，PC间的数据流进入队列2，队列2中的报文占有20%的带宽，例如每次出队2000个字节的报文，则CQ对这两种不同业务的报文将做区别，对待报文的发送采用轮询调度的方式，首先让队列1中的报文出队，并发送直到此队列中的报文被发送的字节

数不少于6000字节，然后才开始发送队列2中的报文，直到此队列中的报文被发送的字节数不少于2000字节，然后是其他队列，如果路由器1的串口1的物理带宽是2M，则局域网1的服务器向局域网2的服务器发送关键业务的数据所能占的带宽将至少为1.2M（2*0.6），局域网1的PC向局域网2的PC发送非关键业务的数据所能占的带宽将至少为0.4M（2*0.2），当路由器1的串口1中除了上述两个数据流外没有其他数据要发送时，这两种数据流将按比例分享接口的剩余空闲带宽即局域网1的服务器向局域网2的服务器发送关键业务的数据所能占的带宽将为1.5M [2 * 0.6 / (0.2 + 0.6)]。局域网1的PC向局域网2的PC发送非关键业务的数据所能占的带宽为0.5M [2 * 0.2 / (0.2 + 0.6)]，当局域网1的服务器向局域网2的服务器不发送关键业务的数据时并且除了局域网1的PC向局域网2，的PC发送非关键业务的数据外没有其他的数据流，则局域网1的PC向局域网2的PC发送非关键业务的数据所能占的带宽将可以为2M。

为了能够为每个队列分配一定的带宽，必须为每个队列定义一定字节数的数据包。自定义队列中的数据包也按照队列号顺序被转发，当队列为空或超出本次队列允许发送的数据包时，接下来会轮到下一个队列。但是假如定义的字节数为100字节，而某个数据包的大小为1024字节，那么该队列每次将转发的数据包的大小即为1024字节，而不是100字节。

假如有3 个队列，每个队列中的数据包大小分别为500 字节，300字节和200字节如果想让这3个队列平均的占用带宽，为这3个队列定义的字节数分别为200字节，200字节和200字节，但是实际上生效的带宽占用比为5/3/2，因此如果把队列中数据包的字节数定义的过小的话，将导致带宽分配的不尽如人意。但是如果把队列中数据包的字节数定义的过大，那么将导致下一个队列中的数据包被转发的等待时间过长。

CQ使用中存在一些限制：1、由于CQ是静态配置的，因此它不能适应网络结构的改变。2、由于数据包要经过处理器卡的分类，因此CQ对数据包转发的速度要比FIFO慢。

5.5.2 配置命令

1、定义CQ列表：

kkblue(config-if)#custom-queue-list {*list*} （后面是列表的号码）

2、定义队列中数据包的字节数或最大个数：

定义最大个数

kkblue(config)# queue-list *list-number* queue *queue-number* limit *limit-number*

（queue的号码是队列的号码，limit的后面接的是队列里面的数量，默认为20个，范围是0到32767）

定义队列大小（我觉得就是定义队列的带宽）

kkblue(config)#queue-list {*list*} queue {*queue-number*} byte-count *bytes*

默认为1500字节

3、把数据包分配进特定的CQ 中，可以基于协议或基于进站接口：

基于协议

kkblue(config)#queue-list {*list*} protocol *protocol* {*queue-number*} queue-keyword *keyword-value*

同样的道理，也可以加入相应的参数，参看刚才的PQ配置的参数

基于接口

kkblue(config)#queue-list {*list*} interface *interface* {*queue-number*}

4、定义默认的CQ 队列，未分类的流量默认被分配进该队列：

kkblue(config)#queue-list {*list*} default {*queue-number*}

5.5.3 察看命令

1、显示接口队列信息：　　kkblue#show queue [*interface*]
2、显示CQ列表信息：　　kkblue#show queueing custom

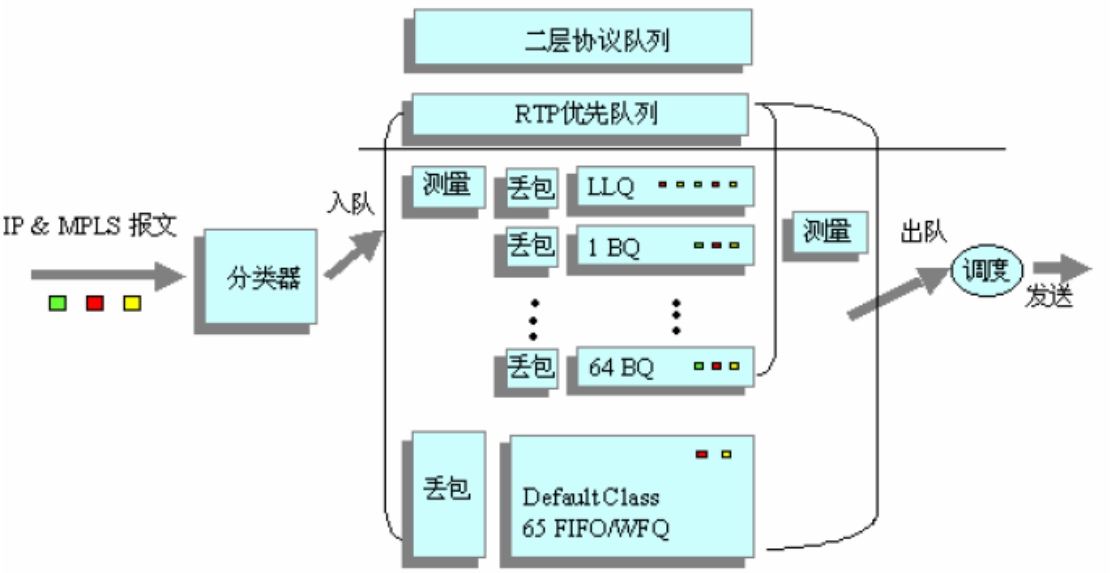
5.5.4 配置案例

```
interface serial 0
custom-queue-list 1        //在接口应用队列

queue-list 1 protocol sna 1     //在 queue-list 1 中定义 sna,ip,ipx 队列。
queue-list 1 protocol ip 2
queue-list 1 protocol ipx 3

queue-list 1 queue 1 byte-count 15000        //queue 1,sna被设为3*(1500+3500)=15000
queue-list 1 queue 2 byte-conut 3500         //queue 2,ipx,被设为 3500 字节
queue-list 1 queue 3 byte-conut 1500      //其实这条不写也成，默认就是 1500
```

5.6　基于类的加权公平队列

（Class Based Weighted Fair Queuing, CBWFQ）

5.6.1 基本原理



CBWFQ，首先根据IP优先级、DSCP或者输入接口的IP数据流等规则来对报文进行分类，让不同类别的报文进入不同的队列，对于不匹配任何类别的报文报文被送入系统定义的缺省类。

图中所示LLQ（Low Latency Queueing，低延迟队列，后面有）是一个具有较高优先级

的队列，它的优先级仅次于二层协议队列（如同CQ中的0号队列），与RTP优先队列（RTP优先队列的参见后文介绍）一个或多个类的报文可以被设定进入LLQ，队列不同类别的报文可设定占用不同的带宽，在调度出队的时候，若LLQ中有报文则总是优先，发送LLQ中的报文直到LLQ中没有报文时或者超过为LLQ配置的最大预留带宽时，才调度发送其他队列中的报文。

进入LLQ的报文在接口没有发生拥塞的时候，此时所有队列中都没有报文，所有属于LLQ的报文都可以被发送在接口，发生拥塞的时候队列中有报文时，进入LLQ的报文被限速超出规定流量的报文将被丢弃，这样在接口不发生拥塞的情况下可以使属于LLQ的报文能获得空闲的带宽在，接口拥塞的情况下又可以保证属于LLQ的报文不会占用超出规定的带宽，保护了其他报文的应得带宽，另外由于只要LLQ中有报文系统就会发送LLQ中的报文，所以LLQ中的报文被发送的延迟最多是接口发送一个最大长度报文的时间，无论是延时还是延时抖动LLQ都可以将之降低为最低限度，这为对延时敏感的应用如VoIP业务提供了良好的服务质量保证

上图中1到64的队列为各类报文的队列每类报文占一个队列，我们称它们为BQ Bandwidth Queueing。在系统调度报文出队的时候，按用户为各类报文设定的带宽将报文出队发送这种队列技术，应用了先进的队列调度算法可以实现各个类的队列的公平调度属于1到N1号BQ队列的报文可以被确保得到用户设定的带宽，当接口中某些类别的报文没有时，BQ队列的报文还可以公平地得到空闲的带宽，大大提高了线路的利用率，同时在接口拥塞的时候仍然能保证各类报文得到用户设定的最小带宽。

当报文不匹配用户设定的所有类别时，报文被送入系统定义的缺省类，虽然允许为缺省类配置带宽使其作为BQ类进行基于类的队列调度，但是更多的情况是为缺省类配置WFQ，使所有进入缺省类的报文进行基于流的队列调度。

CBWFQ最多允许配置64个BQ类，缺省类的WFQ的队列个数N可以由用户设定。

对于缺省类的WFQ和BQ，当队列的长度达到队列的最大长度时，缺省采用尾丢弃的策略。但用户还可以选择用加权随机预检测（Weighted Random Early Detection, WRED）的丢弃策略（加权随机预检测的丢弃策略请参见后面加权随机预检测WRED的描述）。

对于LLQ，由于在接口拥塞的时候流量限制开始起作用，所以用户不必设置队列的长度，由于优先队列中的报文一般是语音报文Voice over IP, VoIP 采用的是UDP报文所以没有必要采用WRED的丢弃策略。

综上所述CBWFQ有一个低时延队列 - LLQ 用来支撑EF（加速转发）类业务，被绝对优先发送，另外有64个BQ用来支撑AF（确保转发）类业务，可以保证每一个队列的带宽及可控的时延，还有一个WFQ对应BE（尽力而为传输）业务使用接口剩余带宽进行发送。CBWFQ可根据报文的输入接口ACL、IP优先级/DSCP等规则对报文进行分类，进入相应队列规则可以是通手工配置，对于进入LLQ和BQ的报文要进行测量，考虑到链路层控制报文的发送链路层封装开销及物理层开销，建议RTP优先队列LLQ与BQ占用接口的总带宽不要超过接口带宽的75%(默认就是只占用75％带宽）LLQ只采用尾丢弃，BQ可采用尾丢弃或者WRED（基于IP优先级DSCP）WFQ可采用尾丢弃和WRED。CBWFQ可为不同的业务定义不同的调度策略，如带宽时延等。由于涉及到复杂的流分类，对于高速接口GE以上启用CBWFQ特性系统资源存在一定的开销。

RSVP也可以和CBWFQ 协同工作.当一个接口同时配置了CBWFQ 和RSVP,它们之间的工作是独立的.并且当CBWFQ 不存在的时候RSVP 还是会继续工作.CBWFQ使用存在的一些限制:第一、目前流量和整形不支持CBWFQ。第二、CBWFQ 不支持以太网子接口

5.6.2 配置步骤和命令

（其实个人觉得就是一个policy map的应用，顺便我们回顾一下刚才的命令）

一、定义分类的策略，即class map

二、设置策略，即定义policy map

三、把policy map 应用在相关接口上

定义class map 步骤如下：

1、定义class map

kkblue(config)#class-map [match-all|match-any] {*map-name*}　　和刚才一样

2、定义匹配语句

kkblue(config-cmap)#{*condition*}

一些匹配条件选项:

match access-group {*ACL*}	匹配IP ACL
match protocol {*protocol*}	匹配协议
match input-interface {*interface*}	匹配进站接口
match qos-group {*Group ID*}	匹配组ID
match destination-address {mac *mac-address*}	匹配目标MAC 地址
match source-address {mac *mac-address*}	匹配源MAC 地址
match ip {dscp *dscp*}	匹配IP DSCP 值
match ip {precedence *precedence*}	匹配IP 优先级
match class-map {*map-name*}	匹配class map
match vlan {*vlan-id*}	匹配VLAN

定义分类的策略,即policy map 的步骤如下:

1.设置policy map:

kkblue(config)#policy-map {*policy-name*}

2.调用class map 或默认的class map(所有未分类的流量默认都属于该分类,否则未分类的流量将以尽力而为的方式被处理):

kkblue(config-pmap)#class {*class-map*|class-default}

3.设置策略:

kkblue(config-pmap-c)#bandwidth {*kbps*|percent *percent*}

4.定义尾丢弃机制允许的队列中数据包个数的上限,默认值为64:

kkblue(config-pmap-c)#queue-limit {*packets*}

其他配置参数

kkblue(config-pmap-c)#**random-detect**	用于WRED
kkblue(config-pmap-c)#**shape**	令牌桶参数
kkblue(config-pmap-c)#**police**	(car限速）
kkblue(config-pmap-c)#**priority**	优先级，低延迟队列（LLQ).

在出站接口上应用policy map:

kkblue(config-if)#service-policy output {*policy-name*}

其他命令

更改用于RSVP 和CBWFQ 等队列机制保留的最大带宽值,默认为75%:

kkblue(config-if)#max-reserved-bandwidth {*percent*}

5.6.3 察看命令

1、查看policy map 信息:　kkblue#show policy-map [*policy-name*]

2、查看接口的policy map 信息:　kkblue#show policy-map interface [*interface*]

3.显示接口的队列信息:　kkblue#show queue [*interface*]

5.6.4 配置案例

5.6.4.1 Case one

限制源自192.168.10.0/24 的流量的带宽为1000kbps:

```
!
class-map match-all kkblue      class map名称kkblue
match access-group 1
!
policy-map blue                  policy-map名称blue
class kkblue                     匹配class map kkblue
bandwidth 1000                   限制带宽1000k
queue-limit 30                   限制队列数据包上限30个包
class class-default              其他的放置到默认队列
!
interface Serial1
ip address 172.16.10.1 255.255.255.252
service-policy output blue
!
access-list 1 permit 192.168.10.0 0.0.0.255
```

5.6.4.2 Case two 综合性配置案例

```
class-map match-any kkblue1        定义一个kkblue2的class map，匹配任意一个条目
match protocol sqlnet              匹配sqlnet协议
match protocol ipsec               匹配ipsec协议
match access-group 100             匹配access-group 100
match ip precedence 4 5            匹配优先级是4和5的ip数据包
!
class-map match-all kkblue2        定义一个kkblue2的class map，要求匹配所有
match access-group 101             匹配两个访问控制列表101和102
match access-group 102
!
class-map kkblue3                  定义一个kkblue2的class map，要求匹配所有
match access-group 103             匹配一个访问控制列表103
policy-map kiss                    创建kiss这个policy map，下面的就不介绍了
class kkblue1
bandwidth 6000
class kkblue2
bandwidth 3000
class kkblue3
bandwidth 700
class class-default
bandwidth 200
!
interface ethernet 1/1
```

service-policy output kiss

看了这两个配置，好像刚才的那些原理都变得清晰了，因为命令一点也不难，多数使用的是bandwidth命令，其实个人觉得就是使用分类给各种应用合理的带宽。

5.7 LLQ Low Latency Queuing 低延迟队列

5.7.1 理论知识

低延迟队列(LLQ)把优先级队列的特性加入到了CBWFQ中，这点和IP RTP 优先级特性类似。如果没有LLQ，对于一些实时的数据流量，比如语音数据流量，CBWFQ 对于每个定义好的分类的操作是基于WFQ的。采用了LLQ 之后，该分类的操作将优先于别的分类。LLQ 减少了语音会话的抖动。LLQ 和IP RTP 优先级特性的区别在于，它不受UDP 端口号的限制。

5.7.2 配置命令

kkblue(config-pmap-c)#priority {*bandwidth*}

刚才在CBWFQ中使用bandwidth命令是用于定义普通队列的，但是如果改用priority，我们配置的就是低延时队列，凌驾于CBWFQ的上面

kkblue（config-if）#max-reserved-bandwidth *percent* 为LLQ和IP RTP设置占用带宽的百分比

5.7.3 察看命令

1、显示接口队列信息： kkblue#show queue [*interface*]

2、调试优先级队列： kkblue#debug priority

5.7.4 配置案例 使用LLQ给视频流量提供保留带宽

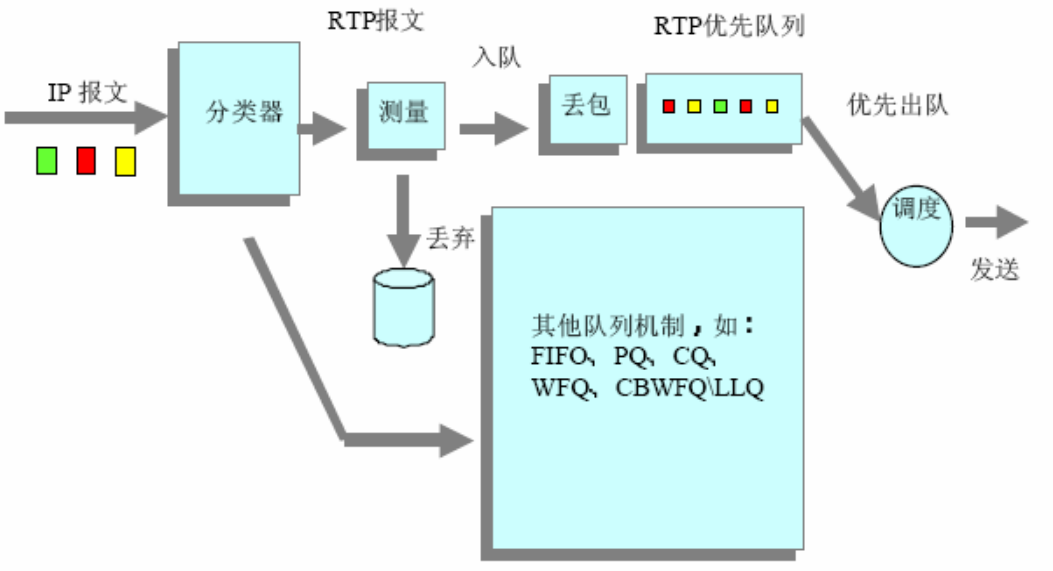
```
class-map match-all kkblue1
match access-group 100
class-map match-all kkblue2
match access-group 101
!
policy-map kiss1                    //定义LLQ策略
class kkblue1
priority 1518               //使用priority定义为Video预留的带宽
set ip precedence 5         （更改优先级）
class class-default
fair-queue
policy-map kiss2
class kkblue2
priority 1518
set ip precedence 5
class class-default
fair-queue
!
interface FastEthernet1/0/0
no ip address
```

```
no ip route-cache distributed
full-duplex
!
interface FastEthernet1/0/0.1
description HQ1 LAN
encapsulation dot1Q 1
ip address 129.2.80.21 255.255.0.0
service-policy output kiss1 //在接口下应用策略
!
interface Serial1/0/0
description Router 1 to ROUTER 2
ip address 206.141.24.1 255.255.255.0
ip route-cache policy
no ip route-cache distributed
service-policy output kiss2
!
access-list 100 permit ip host 141.2.20.20 host 129.2.20.34 precedence critical
//通过ACL匹配
access-list 101 permit ip host 129.2.20.34 host 141.2.20.20 precedence critical
!
```

解释一下，命令看着很多，其实一点都不难，实际上在配置之前肯定事先已经将视频流量的数据包优先级更改成5，后续的工作就是给ACL100和101定义的数据包加入class map，然后使用policy map中的priority命令来保证1518k的带宽，并且把优先级set到5，交给下一跳路由器，不是很难，但是在不同链路上可以有很多应用，大家可以去cisco.com下载一下LLQ的配置文档。

5.8 RTP优先队列（Real Time Protocol Priority Queueing）

5.8.1 基本原理



如图所示：RTP优先队列是一种解决实时业务包括语音与视频业务服务质量的简单队列技术。其原理就是将承载语音或视频的RTP报文送入高优先级队列使其得到优先发送，保证延时和抖动降低为最低限度，从而保证了语音或视频这种对时延敏感业务的服务质量，RTP优先队列将RTP报文送入一个具有较高优先级的队列，RTP报文是端口号在一定范围内为偶数的UDP报文，端口号的范围可以配置一般为16384~32767，RTP优先队列可以同前面所述的任何一种队列包括FIFO、PQ、CQ、WFQ与CBWFQ 结合使用。它的优先级是最高的。

一般语音数据包的体积较小，如果有体积较大的数据包要从该接口被转发出去，该接口应配置链路分片和交叉（LFI）特性.体积较大的数据包被分片为体积较小的数据包。该特性防止语音数据包要等待到体积较大的数据包被转发完毕之后才能被转发。这样语音数据包可以和被分片的数据包交叉被转发出去。从而减少了语音数据包转发消耗的时间。

由于对进入RTP优先队列的报文进行了限速，超出规定流量的报文将被丢弃这样，在接口拥塞的情况下可以保证属于RTP优先队列的报文不会占用超出规定的带宽，保护了其他报文的应得带宽解决了PQ的高优先级队列的流量可能饿死低优先级流量的问题。

5.8.2 配置命令

kkblue(config-if)#ip rtp priority {*starting-rtp-port-number port-number-range*} {*bandwidth*}　　（添加起始port和port范围，后面的bandwidth填写的是预留的带宽）

kkblue（config-if）#max-reserved-bandwidth *percent* 为LLQ和IP RTP设置占用带宽的百分比

5.8.3 察看命令

1、显示接口队列信息:　　kkblue#show queue [*interface*]

2、调试优先级队列:　　kkblue#debug priority

5.8.4 配置案例

5.8.4.1 Case one：RTP同CBWFQ结合使用

定义class map

```
kkblue(config)# class-map kkblue
kkblue(config-cmap)# match access-group 101
kkblue(config-cmap)# exit
```

定义和使用policy map

```
kkblue(config)# policy-map kiss
kkblue(config-pmap)# class kkblue
kkblue(config-pmap-c)# bandwidth 3000
kkblue(config-pmap-c)# queue-limit 30
kkblue(config-pmap-c)# random-detect
kkblue(config-pmap-c)# random-detect precedence 0 32 256 100
kkblue(config-pmap-c)# exit
kkblue(config)# interface Serial1
kkblue(config-if)# service-policy output kkblue
```

设置RTP

```
kkblue(config-if)# ip rtp priority 16384 16383 40
```

解释一下，RTP是直接应用到接口上的，所以优先级超级超级高，另外看看端口范围，起始端口是16384，范围是从16384加上后面的range=16383，应该是从16384--32767这个范

围，存在一个加法公式。

5.8.5 附加知识 CRTP

5.8.5.1 理论知识

实时传输协议(RTP)是一种用于传输实时数据流量的协议。RTP包含数据部分和包头部分。数据部分用于支持实时传输应用程序的各种属性。包头部分的体积比较大，最少包括了12 字节的RTP包头,20字节的IP 包头(IPH)和8字节的UDP包头,因此IP/UDP/RTP包头总长度至少为40 字节。根据IP/UDP/RTP包头长度的不同,RTP数据包的长度为20字节到160字节不等。如果不对IP/UDP/RTP包头进行压缩的话,对于RTP数据包的传输是很没效率的。因此后来出现了压缩IP/UDP/RTP包头的压缩式实时传输协议(CRTP)。

CRTP是一种逐跳的压缩机制。CRTP可以把IP/UDP/RTP包头从40字节压缩为2到5字节。当广域网接口带宽不高，并且RTP数据流量过大的话，应该考虑使用CRTP；但是对于高于T1线路速率的接口，无需使用CRTP。CRTP支持ISDN，支持使用PPP，HDLC或FR封装的接口.但是FR的封装格式只能使用Cisco特有的格式。

5.8.5.2 基本配置命令

1、启用CRTP：如果不指定关键字passive，将对所有IP/UDP/RTP包头进行压缩；如果指定passive关键字，当进站RTP数据包的IP/UDP/RTP包头被压缩，才相应的压缩出站的RTP数据包的IP/UDP/RTP包头：

kkblue(config-if)#ip rtp header-compression [passive]

2、更改IP/UDP/RTP包头压缩的连接数,默认为16条.可选:

kkblue(config-if)#ip rtp compression-connections {*number*}

RTP头压缩，默认16条，最大500条

kkblue(config-if)#ip tcp compression-connections {*number*}

TCP头压缩，默认16条，最大128条

5.8.5.3 Frame relay中的CRTP配置

1、在物理接口上启用CRTP，那么该物理接口相关的子接口将继承CRTP的配置信息。如果不指定关键字passive，将对所有IP/UDP/RTP包头进行压缩；如果指定passive关键字，当进站RTP数据包的IP/UDP/RTP包头被压缩，才相应的压缩出站的RTP数据包的IP/UDP/RTP包头：

kkblue(config-if)#frame-relay ip rtp header-compression [passive]

2、更改IP/UDP/RTP包头压缩的连接数，默认为16条，可调整:

kkblue(config-if)#frame-relay ip rtp compression-connections {*number*}

3、只针对特定的PVC启用CRTP。如果指定关键字active，将对所有IP/UDP/RTP包头进行压缩；如果指定passive关键字，当进站RTP数据包的IP/UDP/RTP包头被压缩，才相应的压缩出站的RTP数据包的IP/UDP/RTP包头。还可以指定最大的IP/UDP/RTP包头压缩的连接数,默认为16条，可调整:

kkblue(config-if)#frame-relay map ip {*ip-address*} {*dlci*} [broadcast] rtp header-compression [active|passive] [connections *number*]

4、在特定PVC上同时启用CRTP 和TCP 头部压缩。

kkblue(config-if)#frame-relay map ip {*ip-address*} {*dlci*} [broadcast] compress

5.8.5.4 察看命令

1、显示IP/UDP/RTP包头的压缩统计信息：

kkblue#show ip rtp header-compression [*interface*] [detail]

2、显示FR 的IP/UDP/RTP包头的压缩统计信息：

kkblue#show frame-relay ip rtp header-compression [*interface*]

5.9 加权轮询队列（WRR Weighted Round-Robin）

WRR是应用在多层交换机上面的QOS技术，本人研究的不是很深，所以摘抄了一部分内容，让大家有所了解

5.9.1 Cisco Catalyst 3550 交换机 QoS 时序及队列

3550 交换机有两种不同类型的端口：千兆端口和非千兆端口（10/100M 端口）每个 3550 的端口上都有 4 个不同的输出队列。这些队列中的一个可以被配置为优先级队列。余下的几个端口被配置为非绝对的优先级队列，并使用 Weighted Round Robin (WRR)。 所有的端口上，数据包根据各自的服务类别（CoS）被分配为四中可能的类别之一。

其中千兆端口还能支持每个队列的管理机制。每个队列可以使用 Weighted Random Early Discard (WRED)或者双线程的 tail drop 。队列大小可调（每个队列均分配相应的缓冲区）。非千兆端口不支持任何队列管理机制，例如 WRED 或者双线程 tail drop 10/100M 端口支持 FIFO 队列。 每个端口队列的大小都不可改变。但是你可以为每个队列分配最小的保留带宽。

5.9.2 WRR的操作和应用

5.9.2.1 CoS 到队列映射

本节讨论 3550 如何决定将每个数据包放置到队列中去。数据包队列取决于服务类别（CoS）。通过使用 CoS 到队列的接口映射命令，每个八种可能的 Cos 数值将被映射到相应的四个队列。下面是该命令的示例：

```
(config-if)# wrr-queue cos-map queue-id cos1... cos8
```

下面是一个例子：

```
kkblue(config-if)# wrr-queue cos-map 1 0 1
kkblue(config-if)# wrr-queue cos-map 2 2 3
kkblue(config-if)# wrr-queue cos-map 3 4 5
kkblue(config-if)# wrr-queue cos-map 4 6 7
```

该示例将 CoS 0 和 1 映射到 Q1，CoS 2 和 3 映射到 Q2，CoS 4 和 5 映射到 Q3，CoS 6 和 7 映射到 Q4。

每个端口的 CoS 到队列的映射情况可以通过使用下面的命令来进行验证：

```
kkblue# sh mls qos int gig 0/1 queueing
GigabitEthernet0/1
...Cos-queue map:
cos-qid
0 - 1
1 - 1
2 - 2
3 - 2
4 - 3
5 - 3
6 - 4
7 - 4.
```

当然，DSCP 也可以有默认的映射，如下图

DCSP-to-Transmit queue map	DSCP 0-15 Queue 1 DSCP 16-31 Queue 2 DSCP 32-47 Queue 3 DSCP 48-63 Queue 4

5.9.2.2 绝对的优先级队列

绝对的优先级队列在初始状态下通常是空的。这就意味着一旦有数据包进入队列，该包将马上被转发。当 WRR 队列中所有的数据包都被转发后，优先级队列根据需要关闭并清空。绝对的优先级队列被特别设计来处理对延迟/抖动比较敏感的数据流，例如语音。绝对的优先级队列将导致其他队列严重滞后。在其他三个 WRR 中的数据包在绝对的优先级队列中数据传输完成之前，将不会被转发。注意：要避免其他队列的严重滞后，要特别注意放到优先级队列中的流量。

该队列通常用于语音数据流，而此类型应用并不占用很高的带宽。但是若有人将一些占用带宽较多的应用（例如数据转移或备份）放到绝对的优先级队列。这将引起其他流量的严重滞后。要避免该问题，特殊的数据流应被放置在分类/准入，并在网络中标记该数据流。 例如，你可能需要采取一下预防措施：

1、在非可信的源端口使用非可信的端口 QoS 状态；

2、在使用 Cisco IP 电话端口可靠的边界特性时，确信 IP 电话配置于其它应用是可信的

3、修正进入绝对优先级队列的数据流。在千兆端口上修正数据流的流量限制为 100M。

在 3550 上，可以配置一个队列为优先队列，（总是 Q4），在端口模式下使用如下命令：

```
kkblue(config-if)# priority-queue out （加载这个命令之后，Q4 就会成为优先队列）
```

如果某个端口没有配置优先队列，则 Q4 被当做标准的 WRR 队列（下节将详细描述）

你可以通过输入和下面一样的 IOS 命令来验证某端口是否被配置为绝对优先级队列，

```
kkblue#sh mls qos interface gig 0/1 queueing
GigabitEthernet0/1
Egress expedite queue: ena
```

5.9.3 Catalyst 3550 上的 WRR(Weighted Round Robin)

在 3550 上，WRR 是一个对输出时间序列进行管理的机制。WRR 在三个或四个队列（如果没有绝对优先级队列）之间工作。使用 WRR 模式的队列在循环方式下是置空的，可以为每个队列配置相应的权值。

例如，配置了不同的权值，不同的队列将提供不同的服务，如下所示：

```
Serving WRR Q1 : 10% of time
Serving WRR Q2 : 20% of time
Serving WRR Q3 : 60% of time
Serving WRR Q4 : 10% of time
```

对每个队列，你可以在端口模式使用以下命令来配置四个权值（各自相对于一个队列）：

```
kkblue(config-f)#wrr-queue bandwidth weight1 weight2 weight3 weight4
```

示例如下：

```
kkblue(config)# interface gigabitethernet0/1
kkblue(config-if)# wrr-queue bandwidth 1 2 3 4
```

注意：权值是相对的，下面是计算方式

Q1 = weight 1 /(weight1 + weight2 + weight3 + weight4) = 1/(1+2+3+4) = 1/10

Q2 = 2/10

Q3 = 3/10

Q4 = 4/10

WRR 可通过以下两种方式执行：

1.、WRR per bandwidth: 每个权值描述了可以用于发送的特别带宽。权 Q1 允许使用大约 10%的带宽， Q2 将获得大约 20%的带宽，以此类推。此方案目前仅在 Catalyst 6000 系列交换机上实现。

2、 WRR per packet: 该算法在 3550 交换机上实现。 这表示每个权值表示了某个数量的数据包将被发送，而不管包的大小如何。

3550 上实现 WRR per packet 表现为如下形式：

Q1 传输 1/10 的数据包

Q2 传输 2/10 的数据包

Q3 传输 3/10 的数据包

Q4 传输 4/10 的数据包

如果被传送的包是同样大小则是最理想的情况。在 4 个队列中你依然能够获得理想的共享带宽。然而，如果队列间的平均包大小有差异，则会在拥塞事件发生时对传输产生巨大的影响。

例如，假设当前交换机只有两个数据流， 同时假设处于以下的情形：

一个千兆口的队列 2（Q2）以 Cos 3 类别方式每秒传输少量的交互应用数据流（80 字节/帧）

一个千兆口的队列 1（Q1）以 Cos 0 类别方式每秒传输大型文件数据流（1518 字节/帧）

两个队列都将以传输 1 Gbps 的速率传输数据。两个数据流需要共享同一个输出的千兆口。假设我们已经为 Q1 和 Q2 设置了同样的权值，WRR 应用到每个数据包，并且每个队列内传输的数据量不同于两个队列之间的数据量。每个队列都转发了同样数量的数据包，然而交换机实际上发送了下面数量的数据：

77700 包/秒由 Q2 输出 = (77700 x 8 x 64) bits/sec (大约 52 Mbps)

77700 包/秒由 Q1 输出= (77700 x 8 x 1500) bits/sec (大约 948 Mbps)

注意：

如果你想要每个队列都公平的接入网络，需要考虑每个数据包的平均值。每个数据包都被假设放置在同一个队列，因而权值得到改善。

例如：如果你想要为四个队列赋予相同的接入（每个队列各自分配到 1/4 的带宽），流量表现为如下形式：

Q1: 最佳的互联网数据流量。假定数据流的平均包大小为 256 字节。

Q2 ： 文件备份形成的文件传输，主要由 1500 字节构成的数据包。

Q3 ： 视频流，每个包被分成 192 字节。

Q4 ： 交互应用，主要由 64 字节构成的数据包。

这就产生了以下的情形：

Q 1 消耗 4 倍于 Q 4 的带宽

Q 2 消耗 24 倍于 Q 4 的带宽

Q 3 消耗 3 倍于 Q 4 的带宽

若要以同样的带宽接入网络，采用如下的配置：

Q1 权值设为 6

Q2 权值设为 1

Q3 权值设为 8

Q4 权值设为 24

如果分配了以上的权值，则在拥塞事件发生时，四个队列将分享到同样的带宽。

如果设置了绝对优先级队列，WR 权值将在其余三个队列中重新分配。下面是一个设置了绝对优先级，而 Q4 没有进行配置的情况下，队列 1、2、3。

Q1 = 1 / (1+2+3) = 1/6 数据包输出

Q2 = 2/6 数据包输出

Q3 = 3/6 数据包输出

队列的权值可以通过 IOS show 命令进行验证：

```
kkblue#sh mls qos interface gig 0/1 queueing
GigabitEthernet0/1
QoS is disabled. Only one queue is used
When QoS is enabled, following settings will be applied
Egress expedite queue: dis
wrr bandwidth weights:
qid-weights
1 - 25
2 - 25
3 - 25
4 - 25
```

如果启用了快速优先级队列，Q4 的权值仅在快速队列失效时使用。

看下面的示例：

```
kkblue#sh mls qos interface gig 0/1 queueing
GigabitEthernet0/1
Egress expedite queue: ena
wrr bandwidth weights:
qid-weights
1 - 25
2 - 25
3 - 25
4 - 25
```

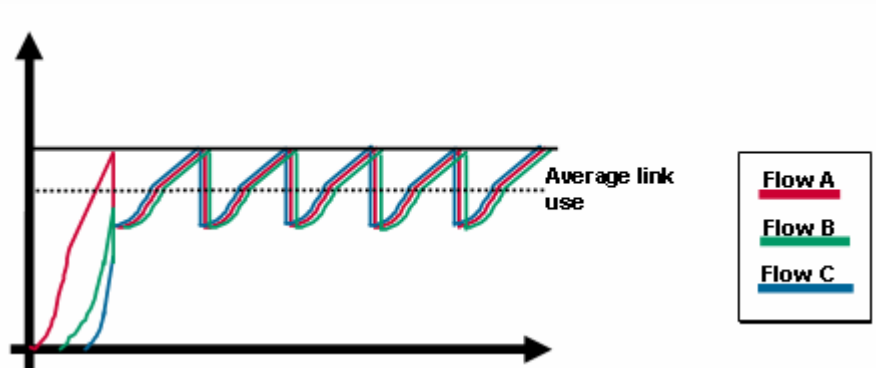
其实并不是很难的

基本上所有的队列就都介绍完了，那么接下来就要讨论一下，出现拥塞，有了QOS中的队列，我们可以进行拥塞管理，但是你说要是数据还是源源不断的流怎么办，接口缓存有没有天那么大，所有的阻塞数据都能容得下，怎么办？这就需要拥塞避免，也就是说如果还是拥塞，接口就要丢弃数据包，那么改怎么丢？？看看下文

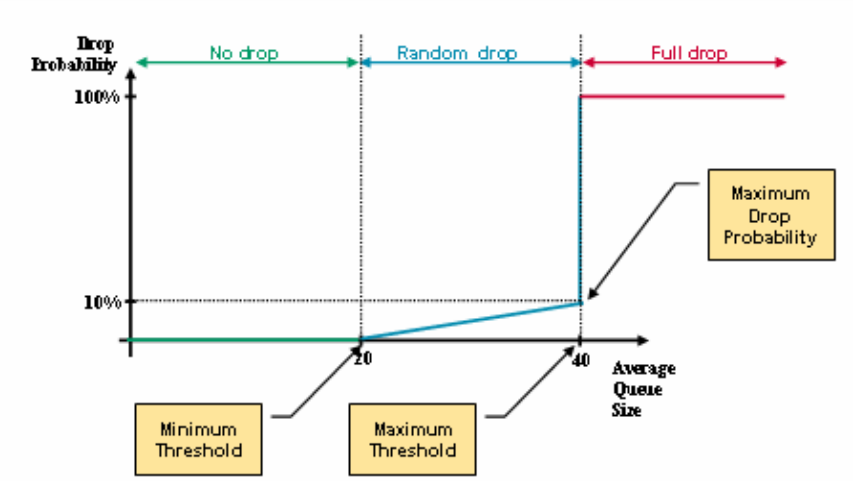
第六章　拥塞避免

6.1　理论知识

由于内存资源的有限按照传统的处理方法，当队列的长度达到规定的最大长度时，所有到来的报文都被丢弃，对于TCP报文如果大量的报文被丢弃将造成TCP超时，从而引发TCP的慢启动和拥塞避免机制。使TCP减少报文的发送。当队列同时丢弃多个TCP连接的报文时将造成多个TCP连接同时进入慢启动和拥塞避免。称之为TCP全局同步。这样多个TCP连接发向队列的报文将同时减少，使得发向队列的报文的量不及线路发送的速度，减少了线路带宽的利用，并且发向队列的报文的流量总是忽大忽小，使线路的上的流量总在极少和饱满之间波动如下图所示

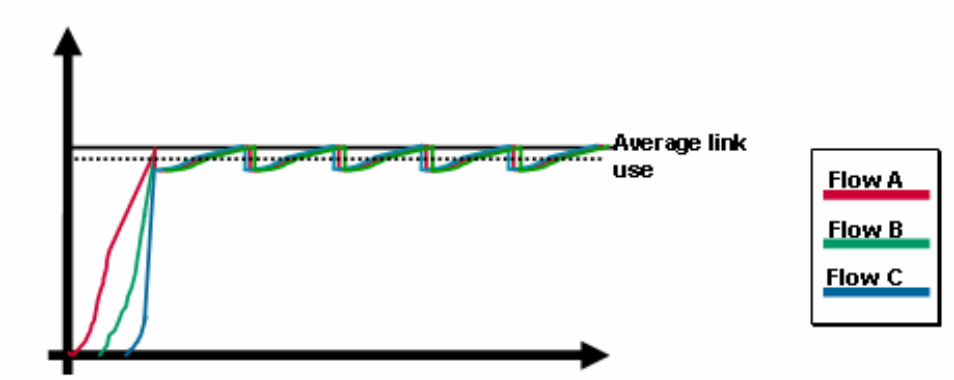


为了避免这种情况的发生引入了RED技术（Random Early Detection）随机预检测，如图

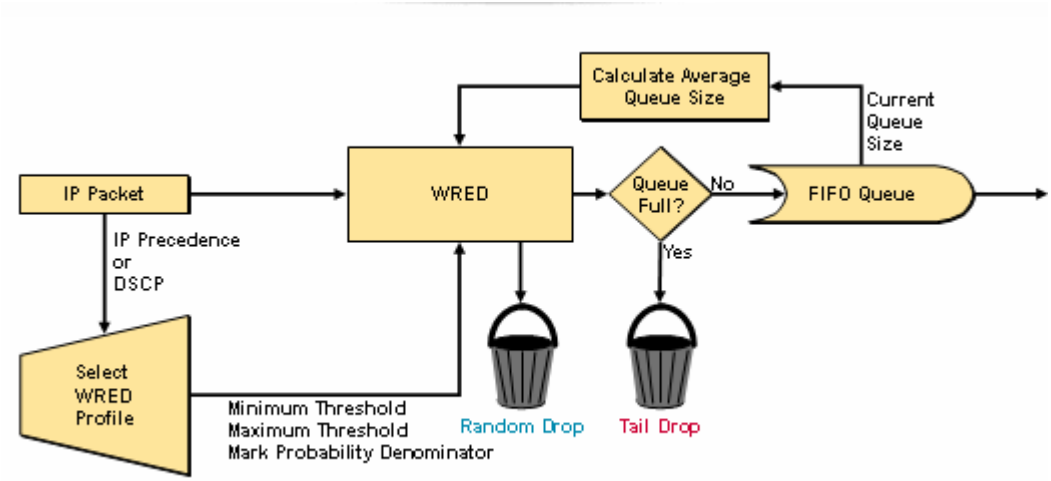


采用RED时用户可以设定队列的阈值threshold，当队列的长度小于低阈值时，不丢弃报文；当队列的长度在低阈值和高阈值之间时，WRED开始随机丢弃报文，队列的长度越长，丢弃的概率越高；当队列的长度大于高阈值时，丢弃所有的报文。

由于RED随机地丢弃报文，将避免使多个TCP连接同时降低发送速度，从而避免了TCP的全局同步现象，当某个TCP连接的报文被丢弃开始减速发送的时候，其他的TCP连接仍然有较高的发送速度，这样无论什么时候总有TCP连接在进行较快的发送，提高了线路带宽的利用率。



如上图所示，图线平稳了很多，但是RED只是随机进行丢弃，所以灵活性要差一点，现在所采用的基本上都是WRED（Weighted Random Early Detection）。原理都是一样的。只不过WRED与RED的区别在于前者引入IP优先权DSCP值来区别丢弃策略，可以为不同IP优先级DSCP设定不同的队列长度、队列阈值、丢弃概率，从而对不同优先级的报文提供不同的丢弃特性。这是WRED的重要特点。下图介绍了WRED和队列之间的关系



在设置时如果直接采用队列的长度与用户设定的阈值比较并进行丢弃(这是设置队列门限的绝对长度)，将会对突发性的数据流造成不公正的待遇，不利于数据流的传输，所以在与设定的阈值比较并进行丢弃时采用队列的平均长度(这是设置队列门限与平均长度比较的相对值)队列的平均长度既反映了队列的变化趋势又对队列长度的突发变化不敏感避免了对突发性的数据流造成不公正的待遇。另外还要注意WRED不能配置在使用了基于路由交换处理器(RSP)的CQ、PQ和WFQ队列机制的接口上。

6.2 配置命令

WRED可以在接口上进行配置，也可以在policy上进行配置，可以针对于precedence进行RED，也可以针对于DSCP值进行RED，当然，两者之间只能选择一个。

基于DSCP

1.使用IP DSCP 来配置WRED:

```
kkblue(config-if)#random-detect dscp-based
```

2.设置丢弃数据包的最小值,最大值和丢弃数据包的轮循间隔:

```
kkblue(config-if)#random-detect dscp {dscp} {min max mark}
```

基于IP precedence的配置

1、启用WRED:

kkblue(config-if)#random-detect

2、设置WRED 丢弃数据包的最小值,最大值和丢弃数据包的轮循间隔:

kkblue(config-if)#random-detect precedence {*precedence*|rsvp} {*min max mark*}

6.4 察看命令

1、显示接口队列信息： kkblue#show queue [*interface*]

2、显示WRED信息： kkblue#show queueing random-detect

6.5 配置案例

6.5.1 Case one：基于接口的配置

```
interface Serial 0/1/0
   ip address 200.200.14.250 255.255.255.252
   random-detect
   random-detect precedence 0 10 25 10
   random-detect precedence 1 20 35 10
   random-detect precedence 2 15 25 10
   random-detect precedence 3 25 35 10
   random-detect precedence 4 1   2  1
   random-detect precedence 5 35 40 10
   random-detect precedence 6 30 40 10
   random-detect precedence 7 30 40 10
```

6.5.2 Case two：基于policy map的配置

```
Switch(config)#class-map c1
Switch(config-cmap)#match access-group 101
Switch(config)#policy-map p1
Switch(config-pmap)#class c1
Switch(config-pmap-c)#bandwidth 48
Switch(config-pmap-c)#random-detect dscp-based
Switch(config-pmap-c)#random-detect dscp 8 24 40
Switch(config-pmap-c)#interface Serial/0
Switch(config-if)#service-policy output p1
```

6.6 基于流的WRED

6.6.1 基本理论

当WRED和WFQ配合使用时还可以实现基于流的WRED，这是因为在进行分类的时候，不同的流有自己的队列，对于流量小的流，由于其队列长度总是比较小，所以丢弃的概率将比较小，而流量大的流将会有较大的队列长度，从而丢弃较多的报文，保护了流量较小的流的利益。即使WRED和其他的队列机制配合使用，对于流量小的流，由于其报文的个数较少，所以从统计概率来说被丢弃的概率也会较小，也可以保护流量较小的流的利益。

6.6.2 基本配置

在配置基于流的WRED 之前，必须先启用WRED。步骤如下：

1、启用基于流的WRED：

kkblue(config-if)#random-detect flow

2、设置平均深度因素(average depth factor)的值,值必须为2的幂,默认值为4.可选：

kkblue(config-if)#random-detect flow average-depth-factor {*scaling-factor*}

这个参数是改变一个乘法的比例因数.从而改变对列的大小，其实就是改变队列的长度，我们可以不去研究

3、设置基于流的WRED 的数据流数目,默认值为256.可选：

kkblue(config-if)#random-detect flow count {*number*}

6.6.3 察看命令

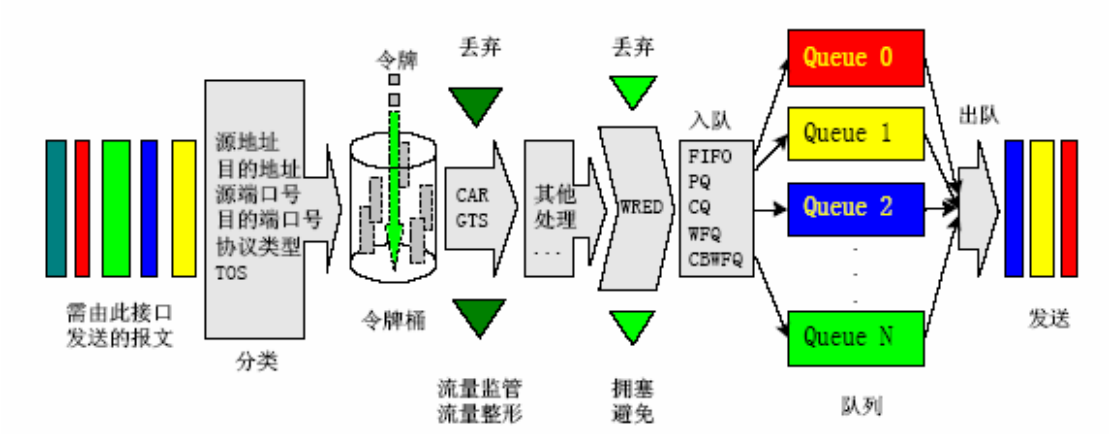
1、显示接口队列信息： kkblue#show queue [*interface*]

2、显示WRED 信息： kkblue#show queueing random-detect

第七章 流量策略

7.1 理论知识

OK，现在我们了解了WRED的操作，但是还有很多问题需要解决，刚才讲到的仅仅是QOS的中间过程，也就是出现拥塞时候的排队技术和处理，其实为什么不能一开始就减少拥塞产生的几率呢，接下来我们来看一个整体的QOS操作模型



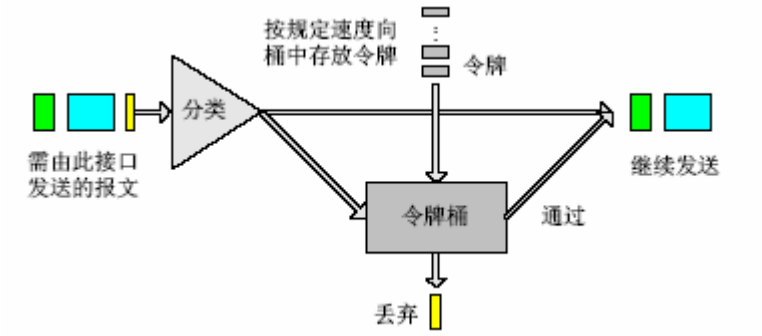
我们可以看到，在WRED和队列技术之前，还存在着一个流量管理和流量整形技术，通过这种技术来管理流量，在结合后面的WRED和各种队列技术，才能整体实现了QOS。

流量策略（traffic policing ）的典型作用是限制进入某一网络的某一连接的流量与突发（当然，限制发出也是可以的），在报文满足一定的条件时，如某个连接的报文流量过大流量，监管就可以对该报文采取不同的处理动作例如丢弃报文或重新设置报文的优先级等，通常的用法是使用CAR来限制某类报文的流量例如限制HTTP报文不能占用超过50%的网络带宽。

7.2 承诺访问速率CAR（Committed Access Rate）

7.2.1 理论知识

对于ISP来说对用户送入网络中的流量进行控制是十分必要的。对于企业网，对某些应用的流量进行控制也是一个有力的控制网络状况的工具，网络管理者可以使用约定访问速率CAR来对流量进行控制。CAR利用令牌桶（Token Bucket, TB）进行流量控制。如下图：



上图所显示的是利用CAR进行流量控制的基本处理过程，首先根据预先设置的匹配规则来对报文进行分类，如果是没有规定流量特性的报文就直接继续发送，并不需要经过令牌桶的处理；如果是需要进行流量控制的报文，则会进入令牌桶中进行处理，如果令牌桶中有足够的令牌可以用来发送报文，则允许报文通过，报文可以被继续发送下去；如果令牌桶中的令牌不满足报文的发送条件则报文被丢弃，这样就可以对某类报文的流量进行控制。

令牌桶按用户设定的速度向桶中放置令牌，并且用户可以设置令牌桶的容量，当桶中令牌的量超出桶的容量的时候，令牌的量不再增加；当报文被令牌桶处理的时候，如果令牌桶中有足够的令牌可以用来发送报文，则报文可以通过可以被继续发送下去，同时令牌桶中的令牌量按报文的长度做相应的减少，当令牌桶中的令牌少到报文不能再发送时，报文被丢弃。令牌桶是一个控制数据流量的很好的工具，当令牌桶中充满令牌的时候，桶中所有的令牌代表的报文都可以被发送，这样可以允许数据的突发性传输，当令牌桶中没有令牌的时候报文将不能被发送，只有等到桶中生成了新的令牌报文才可以发送，这就可以限制报文的流量只能是小于等于令牌生成的速度，达到限制流量的目的。

在实际应用中CAR不仅可以用来进行流量控制，还可以进行报文的标记（mark）或重新标记（re-mark），具体来讲就是CAR可以设置IP报文的优先级或修改IP报文的优先级，达到标记报文的目的。例如当报文符合流量特性的时候可以设置报文的优先级为5，当报文不符合流量特性的时候可以丢弃，也可以设置报文的优先级为1并继续进行发送，这样后续的处理可以尽量保证不丢弃优先级为5的报文。在网络不拥塞的情况下也发送优先级为1的报文。当网络拥塞时首先丢弃优先级为1的报文。然后才丢弃优先级为5的报文。

CAR可以为不同类别的报文设置不同的流量特性和标记特性，即首先对报文进行分类，然后不同类别的报文有不同的流量特性和标记特性，此外CAR的策略还可以进行串联处理。例如可以对所有的报文限制一个总的流量，然后在总的流量中再限制部分报文的流量符合某个流量特性。

CAR通常使用在网络边界路由器的接口上，用来限制进入或离开该网络的流量速率。每个接口可以配置多个CAR策略，当数据包进入使用了多个策略的接口时，路由器将检查每个策略，直到数据包和某个策略相匹配；如果没有找到匹配的策略，默认操作是转发该数据包。.

CAR的使用限制：第一、CAR只能对IP 流量限速。第二、CAR不支持快速以太网信道(Fast EtherChannel)第三、CAR不支持隧道接口第四、CAR不支持ISDN PRI 接口。

7.2 基本配置

7.2.1 基础配置

kkblue(config-if)#rate-limit {input|output} {*CIR Bc Be*} conform-action {*action*}

exceed-action {*action*}

output|input指输出或者输入的流量。CIR配置的是承诺接入速率，它的值的范围是在8000-2000000000 bit每秒。Bc是普通突发，它的值应在1000-512000000byte，Be是最大突发，其值范围为2000-1024000000bytes.conform-action后面规定的是遵从条件时候的动作，exceed-action 后面规定的是超出时的动作，遵从的条件说得就是当要发的数据小于正常突发(bc)的时候。最大条件说得是要发的数据大于普通突发，小于最大突发(be)的时候。违章条件是说得是要发的数据大于最大突发(be)的时候。

那么我们来看看action都有什么操作

continue	继续执行下一条CAR 语句
drop	丢弃该数据包
set-prec-continue {*precedence*}	设置IP 优先级并继续执行下一条CAR 语句
set-prec-trasnmit {*precedence*}	设置IP 优先级并转发该数据包
set-dscp-continue {*dscp*}	设置IP DSCP 值并继续执行下一条CAR 语句
set-dscp-trasnmit {*dscp*}	设置IP DSCP 值并转发该数据包
set-qos-continue {*group ID*}	设置QoS 组ID 并继续执行下一条CAR 语句
set-qos-transmit {*group ID*}	设置QoS 组ID 并发送该数据包
transmit	转发该数据包

基本上就是这些，但是我们发现好像这样做是管理整个接口的流量，于是乎得寸进尺，能不能去对某一个流量进行CAR管理呢？或者针对IP优先级或者根据DSCP进行管理，实际上都是可以的。

7.2.2 扩展的配置

针对于DSCP值进行CAR

kkblue(config-if)#rate-limit {input|output} [dscp *dscp*] {*CIRBc Be*} conform-action {*action*} exceed-action {*action*}

针对于ACL进行CAR

kkblue(config-if)#rate-limit {input|output}access-group {*ACL*} {*CIRBc Be*} conform-action {*action*} exceed-action {*action*}

针对于限速ACL进行CAR

kkblue(config-if)#rate-limit {input|output} access-group rate-limit {*ACL*} {*CIR Bc Be*} conform-action {*action*} exceed-action {*action*}

限速ACL是一种特殊的ACL，其实也没啥特殊的，就是一个调用关系

kkblue(config)#access-list rate-limit {*ACL*} {*precedence*|*mac-address*}

ACL说得是限速ACL的号码，可以匹配优先级，也可以匹配源mac地址

7.2.3 察看命令

1、查看限速ACL：　　kkblue#show access-lists rate-limit [*ACL*]

2、查看接口的限速信息：　　kkblue#show interfaces [*interface*] rate-limit

7.3 配置案例

8.3.1 Case one ： 基于接口进行配置

```
interface Hssi0/0/0
 description 45Mbps to R1
 rate-limit input 15000000 2812500 2812500 conform-action transmit exceed-action
drop
```

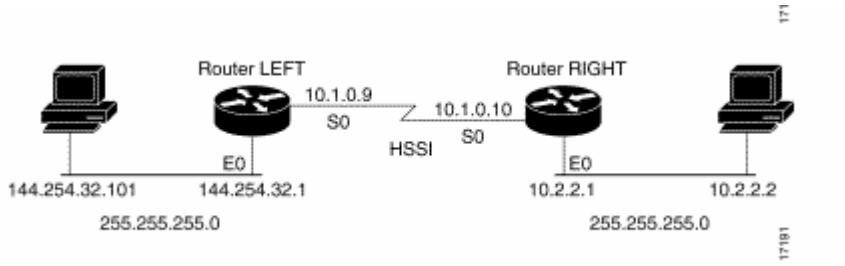
```
 ip address 200.200.14.250 255.255.255.252
 rate-limit output 15000000 2812500 2812500 conform-action transmit exceed-action
drop
```

解释一下，hssi 高速串口是 45M 的带宽，但是 ISP 的接入承诺信息速率为 15M，并且限定普通突发大小为 2812500，最大突发大小也是 2812500 超过这个值的话，丢就一个字

7.3.2 Case two:对IP DSCP值为1的出站流量进行限速（基于DSCP值）

```
interface Serial1
ip address 10.0.0.1 255.255.255.252
rate-limit output dscp 1 20000000 24000 32000 conform-action transmit exceed-action
drop
```

7.3.3 Case three ： 基于 ACL 进行配置



案例要求：1、所有的 www 流量都得发出，而且 web 中遵从第一个速率策略的流量设置 ip 优先级为 5,不遵从的就把 ip precedence 设为 0(尽力而为的传输)。2、ftp 流量遵从第二个速率策略的 ip precedence 设置为 5,如果 ftp 超出速率策略就扔包。3、其他剩余流量限制到 8m，普通突发大小为 16000byte,最大突发大小为 24000byte；遵从策略的流量设 ip precedence 为 5,超出的流量扔包。

```
interface hssi0/0/0
desription 45mbps to r2
rate-limit out put accees-group 101 200000000 24000 32000 conform-action set
prec-transmit 5 exceed-action set-prec-transmit 0
rate-limit output access-group 102 10000000 24000 32000 conform-action
set-prec-tranmit 5 exceed-action drop
rate-limit output 8000000 16000 24000 conform-action set-prec-transmit 5
exceed-action drop
ip address 10.1.0.9 255.255.255.0
!
access-li 101 per tcp any any eq www
access-li 102 per tcp any any eq ftp
```

7.3.4 Case four对匹配192.168.0.0/24 的出站流量进行限速

```
interface Serial1
ip address 10.0.0.1 255.255.255.252
rate-limit output access-group 1 20000000 24000 32000 conform-action transmit
```

```
exceed-action drop
!
access-list 1 permit 192.168.0.0 0.0.0.255  (基于ACL)
```

7.3.5 Case five ： 对IP 优先级为3 的出站流量进行限速:

```
interface Serial1
ip address 10.0.0.1 255.255.255.252
rate-limit output access-group rate-limit 1 20000000 24000 32000 conform-action
transmit
exceed-action drop
!
access-list rate-limit 1 3   (基于限速ACL)
```

7.3.6 Case six 对于源mac地址是00e0.34b0.7777的流量进行限速

```
interface Fddi2/1/0
 rate-limit input access-group rate-limit 100 80000000 64000 80000
conform-action   transmit exceed-action drop
 ip address 200.200.6.1 255.255.255.0
!
access-list rate-limit 100 00e0.34b0.7777    (基于限速 ACL)
```

7.4 policy map中的CAR操作

刚才我们介绍的流量策略都是针对于接口的设置，当然，和WRED一样，这种策略同样可以在在policy map上面实现，是不过命令有一点点差别而已，我们可以把它理解成CAR针对于队列的应用。

7.4.1配置和察看命令

kkblue(config-pmap-c)#police {*CIR Bc Be*} conform-action {*action*}
exceed-action {*action*}
[violate-action {*action*}]

看看，无非是把rate-limit改成了police而已，后面增加了一个violate-action，违规操作，也就是超过了Be流量之后的操作

Action的操作命令

continue	继续执行下一条CAR 语句
drop	丢弃该数据包
set-prec-continue {*precedence*}	设置IP 优先级并继续执行下一条CAR 语句
set-prec-trasnmit {*precedence*}	设置IP 优先级并转发该数据包
set-dscp-continue {*dscp*}	设置IP DSCP 值并继续执行下一条CAR 语句
set-dscp-trasnmit {*dscp*}	设置IP DSCP 值并转发该数据包
set-qos-continue {*group ID*}	设置QoS 组ID 并继续执行下一条CAR 语句
set-qos-transmit {*group ID*}	设置QoS 组ID 并发送该数据包
transmit	转发该数据包

察看命令

1.查看policy map: kkblue#show policy-map [*policy-name*]

2.查看接口的policy map 信息:　kkblue#show policy-map interface [*interface*]

7.4.2　配置案例

7.4.2.1 Case one

限制来自192.168.0.0/24的进站数据包的平均速率为8000bps，突发流量(Bc)为2000 字节，额外突发流量(Be)为4000 字节。对突发流量和额外突发流量分别采取转发和设置QoS组ID为25的策略;对违反突发流量和额外突发流量的数据流量采取丢弃的策略:

```
!
class-map match-all kkblue
match access-group 1
!
policy-map kiss
class kkblue
police 8000 2000 4000 conform-action transmit exceed-action set-qos-transmit 25
violate-action drop
!
interface Serial1
ip address 172.16.0.1 255.255.255.252
service-policy input kiss
!
access-list 1 permit 192.168.0.0 0.0.0.255
```

7.4.2.2 Case two

```
7200-uut(config)# class-map acgroup2
7200-uut(config-cmap)# match access-group 2
7200-uut(config-cmap)# exit
7200-uut(config)# policy-map police
7200-uut(config-pmap)# class acgroup2
7200-uut(config-pmap-c)# police 8000 2000 4000 conform-action transmit
exceed-action
set-qos-transmit 4 violate-action drop
7200-uut(config-pmap-c)# exit
7200-uut(config-pmap)# exit
7200-uut(config)# interface fastethernet 0/0
7200-uut(config-if)# service-policy input police
```

第八章　流量整形（traffic shaping）

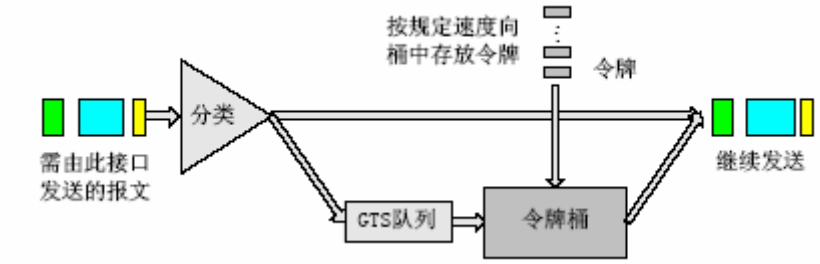
8．1　理论知识

流量整形（traffic shaping）典型作用是限制流出某一网络的某一连接的流量与突发，使这类报文以比较均匀的速度向外发送流量整形通常使用缓冲区和令牌桶来完成，当报文的发送速度过快时，首先在缓冲区进行缓存，在令牌桶的控制下再均匀地发送这些被缓冲的报文。

流量整形采用的技术叫做Generic Traffic Shaping（通用流量整形，简称GTS），可以对不规则或不符合预定流量特性的流量进行整形，以利于网络上下游之间的带宽匹配

GTS与CAR一样均采用了令牌桶技术来控制流量，GTS与CAR的主要区别在于：利用CAR进行报文流量控制时对不符合流量特性的报文进行丢弃，而GTS对于不符合流量特性的报文则是进行缓冲减少了报文的丢弃，同时满足报文的流量特性。

GTS的基本处理过程如下图所示，其中用于缓存报文的队列称为GTS队列



GTS可以对接口上指定的报文流或所有报文进行整形当报文到来的时候，首先对报文进行分类如果报文不需要进行GTS处理，就继续发送不经过令牌桶的处理；如果报文需要进行GTS处理，则与令牌桶中的令牌进行比较，令牌桶按用户设定的速度向桶中放置令牌，如果令牌桶中有足够的令牌可以用来发送报文，则报文直接被继续发送下去，同时令牌桶中的令牌量按报文的长度做相应的减少，当令牌桶中的令牌少到报文不能再发送时，报文将被缓存入GTS队列中。当GTS队列中有报文的时候。GTS按一定的周期从队列中取出报文进行发送。每次发送都会与令牌桶中的令牌数作比较。直到令牌桶中的令牌数减少到队列中的报文不能再发送或是队列中的报文全部发送完毕为止/

例如路由器A和路由器B相连为了减少报文的丢失，可以在路由器A的出口对报文进行GTS处理，对于超出GTS流量特性的报文将在路由器A中缓冲，当可以继续发送下一批报文时，GTS再从缓冲队列中取出报文进行发送，这样发往路由器B的报文将都符合路由器B的流量规定，从而减少报文在路由器B上的丢弃，相反如果不在路由器A的出口做GTS处理，则所有超出路由器2的CAR流量特性的报文将被路由器B丢弃。

8.2 配置命令

配置流量整形GTS有两种方式，一种是基本的GTS

启用GTS：kkblue(config-if)#traffic-shape rate {*CIR* [*Bc* [*Be*]]}

一种是基于ACL 的GTS，这是可选配置：

kkblue(config-if)#traffic-shape group {*ACL*} {*CIR* [*Bc* [*Be*]]}

ACL怎么做我就不需要搞了吧?

察看命令

1. 查看GTS 的配置信息：　kkblue#show traffic-shape [*interface*]

2. 查看GTS 的统计信息：　kkblue#show traffic-shape statistics [*interface*]

8.3 配置案例

```
access-list 101 permit udp any any
interface Ethernet0
traffic-shape group 101 1000000 125000 125000  对于udp的数据，限制其流量稳定在5M
!
interface Ethernet1
traffic-shape rate 5000000 625000 625000   整体接口流量保持在5M
```

8.4 GTS在FR上的实现

1.启用GTS:

kkblue(config-if)#traffic-shape rate {*CIR* [*Bc* [*Be*]]}

2.当接口收到向后显性拥塞通知(BECN)时,估算流量速率的最低值:

kkblue(config-if)#traffic-shape adaptive {*CIR*}

3.以向前显性拥塞通知(FECN)做为BECN 的响应.可选:

kkblue(config-if)#traffic-shape fecn-adapt

8.4.1 配置案例

限制接口流量传输速率的上限为128kbps,下限为64kbps,并以FECN 做为BECN 的响应:

```
!
interface Serial1.1 point-to-point
ip address 172.16.0.1 255.255.255.252
traffic-shape rate 128000 7936 1000
traffic-shape adaptive 64000
traffic-shape fecn-adapt
```

8.5 GTS在policy map上的实现

刚才介绍的是基本应用，和CAR一样，GTS也可以应用在policy map上，基于分类的流量整形可以启用在支持GTS 的任何接口上。基于分类的流量整形可以打破普通GTS仅仅以ACL分类的限制。它还可以定义平均值和峰值的流量整形。并且可以在配置GTS 的时候采用CBWFQ。

8.5.1 配置命令

配置基于分类的流量整形的步骤如下：

1、定义平均值和峰值的CIR,Bc 和Be:

kkblue(config-pmap-c)#shape {average|peak} {*CIR* [*Bc*] [*Be*]} average指的是平均值，peak说得是峰值

2.定义缓冲区上限,默认值为1000.可选:

kkblue(config-pmap-c)#shape max-buffers {*number-of-buffers*}

3、在策略上应用CBWFQ。可选:

kkblue(config-if)#service-policy output {*policy-name*}

一些辅助性的命令：

1、查看基于分类的流量整形的配置信息:

kkblue#show traffic-shape [*interface*]

2、查看基于分类的流量整形的统计信息:

kkblue#show traffic-shape statistics [*interface*]

3、查看policy map:

kkblue#show policy-map [*policy-name*]

4、查看接口的policy map 信息:

kkblue#show policy-map interface [*interface*]

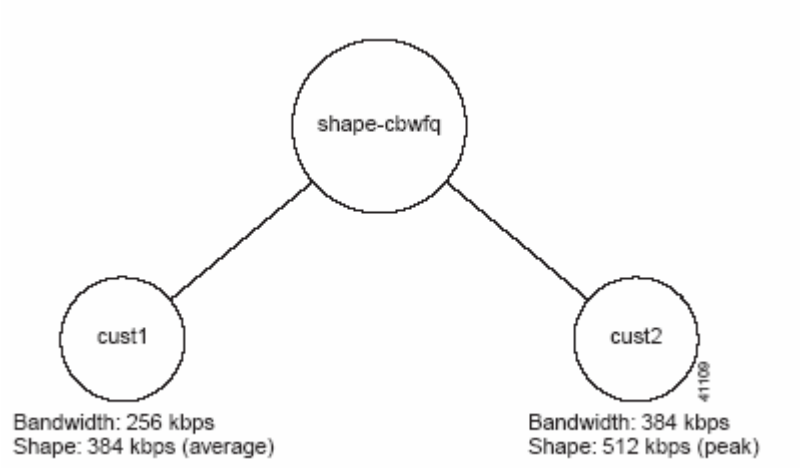
8.6 综合案例

8.6.1 Case one ： 简单的分类GTS

```
kkblue(config)# policy-map shape
kkblue(config-pmap)# class c1
kkblue(config-pmap-c)# shape average 38400 15440
kkblue(config-pmap-c)# configure terminal
kkblue(config)# interface Serial 3/3
kkblue(config-if)# service out shape
```

流量整形平均速率稳定在38400字节，突发流量15440字节

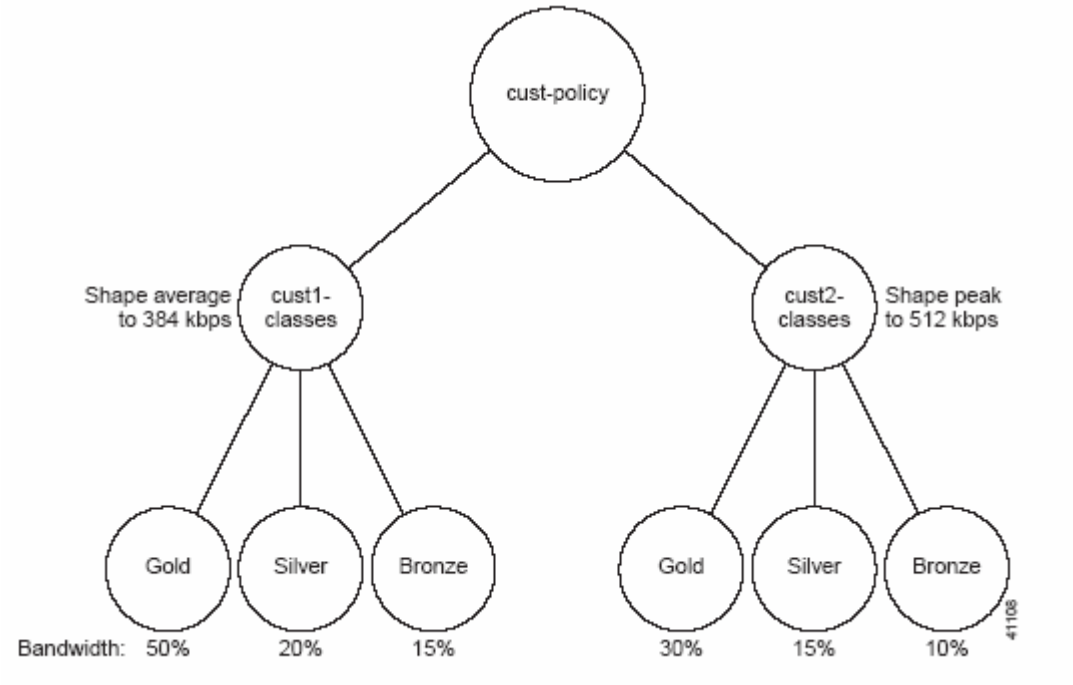
8.6.2 Case two ： GTS和CBWFQ关联使用




```
kkblue(config)# policy-map shape-cbwfq
kkblue(config-pmap)# class cust1
kkblue(config-pmap-c)# shape average 384000
kkblue(config-pmap-c)# bandwidth 256
kkblue(config-pmap)# class cust2
kkblue(config-pmap-c)# shape peak 512000
kkblue(config-pmap-c)# bandwidth 384
kkblue(config-pmap-c)# configure terminal
kkblue(config)# interface Serial 3/3
kkblue(config-if)# service out shape-cbwfq
```

上面的案例中，cust1和cust2是两个队列，当出现拥塞时，使用CBWFQ使其带宽分别限制带宽为256k和384k，但是如果链路的带宽足够用，可以通过GTS把cust1的平均带宽整形为384k，同时也把cust2的最大带宽限定在384k，来保证他们两个队列无休止的占用流量。

8.6.3 Case three ： 流量整形和policy map的复杂结合




```
cust1-classes Configuration
kkblue(config)# policy-map cust1-classes
kkblue(config-pmap)# class gold
kkblue(config-pmap-c)# bandwidth percent 50
kkblue(config-pmap)# class silver
kkblue(config-pmap-c)# bandwidth percent 20
kkblue(config-pmap)# class bronze
kkblue(config-pmap-c)# bandwidth percent 15
cust2-classes Configuration
kkblue(config)# policy-map cust2-classes
kkblue(config-pmap)# class gold
kkblue(config-pmap-c)# bandwidth percent 30
kkblue(config-pmap)# class silver
kkblue(config-pmap-c)# bandwidth percent 15
kkblue(config-pmap)# class bronze
kkblue(config-pmap-c)# bandwidth percent 10
Customer Policy and QoS Features Configuration
kkblue(config)# policy-map cust-policy
kkblue(config-pmap)# class cust1
kkblue(config-pmap-c)# shape average 384000
kkblue(config-pmap-c)# service-policy cust1-classes
kkblue(config-pmap)# class cust2
kkblue(config-pmap-c)# shape peak 512000
kkblue(config-pmap-c)# service-policy cust2-classes
kkblue(config-pmap-c)# interface Serial 3/2
```

kkblue(config-if)# service out cust-policy

其实这就是一个policy map的调用。首先需要注意的是，创建了两个子policy map（请允许我在这里使用子policy map的说法，儿子的意思），里面针对于不同的队列做了带宽的调整（什么黄金啊，白银啊，怎么看怎么像圣斗士），然后创建了一个母policy map，名称是cust-policy，对这两个子policy做了流量整形，就这么回事。

第九章 桢中继流量整形 Frame Relay Traffic Shaping

9.1 理论知识

FR中的FECN和BECN用于暗示网络上发生了拥塞，当收到带有BECN标记的数据包时，FR 流量整形(FRTS)将动态的对流量进行整形。**注意：**FRTS只能使用在FR的PVC和SVC上。

9.2 配置命令

配置FRTS 的步骤如下：

1、启用FRTS：

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

2、全局定义map class。当定义了map class 之后，所有VC将继承该map class的FRTS参数

kkblue(config)#map-class frame-relay {*name*}

3、基于接口的定义map class。当基于接口的定义了map class之后，所有该接口的子接口的VC将继承该map class的FRTS参数。可选：

kkblue(config-if)#frame-relay class {*name*}

4、定义该map class的速率的平均值和峰值：

kkblue(config-map-class)#frame-relay traffic-rate {*average* [*peak*]}

5、定义CIR,Bc和Be，如果不指定方向，则定义为双向。可选：

kkblue(config-map-class)#frame-relay {cir [in|out] *CIR*|bc [in|out] *Bc*|be [in|out] *Be*}

6、定义CIR 的最低值。可选：

kkblue(config-map-class)#frame-relay mincir [in|out] {*min-CIR*}

7、定义以BECN 做为拥塞通知符。可选：

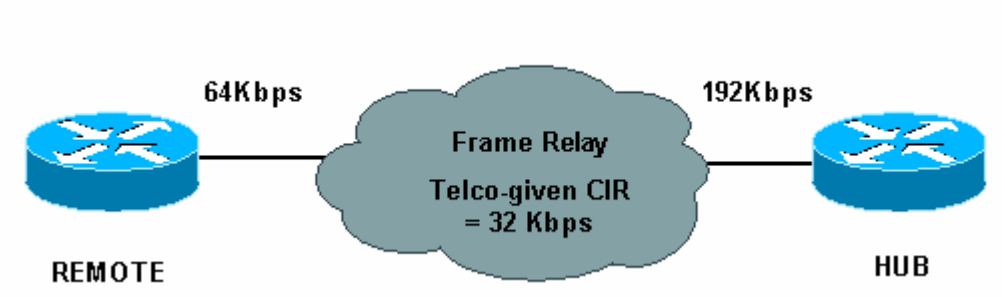
kkblue(config-map-class)#frame-relay adaptive-shaping becn

8、定义CQ 列表和PQ 列表。可选：

kkblue(config-map-class)#frame-relay {custom-queue-list|priority-group} {*list*}

9.3 配置案例

9.3.1 Case one



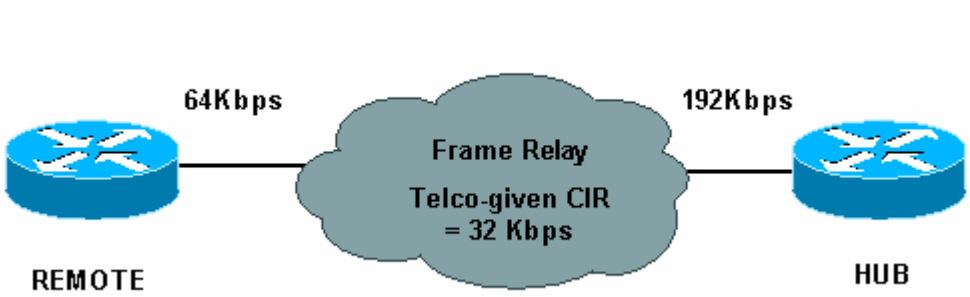
- HUB - access rate = 192 Kbps, guaranteed rate = 32Kbps
- REMOTE - access rate = 64Kbps, guaranteed rate = 32Kbps

HUB 配置

interface Serial0/0

```
 no ip address
 encapsulation frame-relay
 no fair-queue
 frame-relay traffic-shaping

!--- Apply traffic shaping to main interface
interface Serial0/0.1 point-to-point
 ip address 10.1.1.1 255.255.255.0
 frame-relay interface-dlci 16
 frame-relay class cisco

!--- Apply map class to the DLCI / subinterface
!
!

!--- Configure map class parameters
map-class frame-relay cisco
 frame-relay cir 64000
 frame-relay mincir 32000
 frame-relay adaptive-shaping becn
 frame-relay bc 8000
 frame-relay be 16000
!

Remote配置
interface Serial0/0
no ip address
encapsulation frame-relay
no fair-queue
frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
ip address 10.1.1.2 255.255.255.0
frame-relay interface-dlci 16
frame-relay class cisco
!
map-class frame-relay cisco
frame-relay cir 64000
frame-relay mincir 32000
frame-relay adaptive-shaping becn
frame-relay bc 8000
```

9.3.2 Case two ： FR，GTS 和 CBWFQ 结合案例



HUB - physical rate = 192 Kbps, guaranteed rate = 32 Kbps

REMOTE - physical rate = 64 Kbps, guaranteed rate = 32 Kbps

Hub 配置

```
!
class-map match-all YYY
  match access-group 101
!
!
policy-map ZZZ
  class YYY
   bandwidth percent 50
REMOTE
interface Serial0/0
  no ip address
  encapsulation frame-relay
  no fair-queue
  frame-relay traffic-shaping

 interface Serial0/0.1 point-to-point
  ip address 10.1.1.1 255.255.255.0
  frame-relay interface-dlci 16
  frame-relay class XXX
 !
 map-class frame-relay XXX
  frame-relay cir 64000
  frame-relay mincir 32000
  frame-relay adaptive-shaping becn
  frame-relay bc 8000
  service-policy output ZZZ
```

REMOTE 配置

```
!
access-list 101 permit ip host 10.0.0.1 host 11.0.0.1
interface Serial0/0
```

```
no ip address
encapsulation frame-relay
no fair-queue
frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
ip address 10.1.1.2 255.255.255.0
frame-relay interface-dlci 16
frame-relay class XXX
!
map-class frame-relay XXX
frame-relay cir 64000
frame-relay mincir 32000
frame-relay adaptive-shaping becn
frame-relay bc 8000
```

第十章 RSVP

10.1 基本理论

RSVP是第一个在大型网络中动态建立端到端QoS服务模型的工业标准的信令协议。RSVP运行在IP之上，它可以让应用程序在网络上预留带宽。主机和路由器使用RSVP沿着数据流传输的线路进行传输相应的QoS请求信息，比如带宽和延迟。RSVP本身不实行路由决策，相反带宽预留的请求由下层的路由协议执行，因此RSVP 能够适应网络拓扑的变化。RSVP的操作对于不支持RSVP的路由器是透明的。RSVP和现有的一些队列机制协同工作，而不是替代现有的队列机制，并且RSVP 支持组播，RSVP 目前通常为组播应用程序，比如视频会议,进行资源的预留。

主机使用RSVP请求特定的QoS服务来为它的应用程序预留带宽。只要带宽足够，应用程序能够以超过请求预留带宽的速率进行数据的传输；如果带宽不足，那么这些超过请求预留带宽的部分将被丢弃。网络资源预留的要求对于数据流量和实时传输流量是不同的，前者对资源预留的要求很小；后者反之。

资源预留和队列机制的结合使用两个关键点：

1、端到端的RSVP数据流：数据流从单一或多个源地址向单一或多个目标地址进行单向传输。

2、路由器到路由器的WFQ会话：穿越特定接口的单一传输层会话或网络层数据流，WFQ会话通过源地址和目标地址，端口号或协议号等属性进行区分。

10.2 基本配置

配置RSVP步骤如下：

1、启用RSVP,默认带宽预留上限为接口带宽的75%。可以指定RSVP数据流带宽总量，也可以指定每个RSVP数据流的带宽：

```
kkblue(config-if)#ip rsvp bandwidth [interface-kbps [single-flow-kbps]]
```

2、指定只接收符合特定条件的邻居路由器的RSVP请求。可选：

```
kkblue(config)#ip rsvp neighbor {ACL}
```

3、对于符合RSVP所定义的带宽和超出RSVP所定义的带宽的数据包分配IP优先级。可选:

```
kkblue(config-if)#ip rsvp precedence {[conform precedence] [exceed precedence]}
```

10.3 察看命令

1、允许远程管理工作站监视RSVP相关的信息：

```
kkblue(config)#snmp-server enable traps rsvp
```

2、显示接口的RSVP信息：

```
kkblue#show ip rsvp interface [interface]
```

3、显示接口的RSVP过滤和带宽信息：

```
kkblue#show ip rsvp installed [interface]
```

4、显示当前的RSVP邻居信息：

```
kkblue#show ip rsvp neighbor [interface]
```

5、显示RSVP发送方,接收方以及请求信息：

```
kkblue#show ip rsvp {sender|reservation|request} [interface]
```

FIN By kkblue @ 狼藉斋