第 25 章 QOS

网络带宽的发展永远跟不上需求，因此当网络出现堵塞时如何保证网络的正常工作呢？QOS（服务质量）是一个解决方法，QOS 的基本思想就是把数据分类，放在不同的队列中。根据不同类数据的要求保证它的优先传输或者为它保证一定的带宽。QOS 是在网络发生堵塞才起作用的措施，因此 QOS 并不能代替带宽的升级。本章将介绍简单的 QOS 配置，实际上 Cisco 路由器现在推荐是模块化的 QOS 配置（MQC，Module QOS Config）。大量的 QOS 知识无法在本书中一一介绍。

25.1 QOS 简介

25.1.1 QOS

QOS 有三种模型：尽最大努力服务、综合服务、区分服务。尽最大努力服务实际上就是没有服务，先到的数据先转发。综合服务的典型就是预留资源，在通信之前所有的路由器先协商好，为该数据流预先保留带宽出来。区分服务是比较现实的模型，该服务包含了一系列分类工具和排队机制，为某些数据流提供比其他数据流优先级更高的服务。下面我们介绍典型的区分服务。

25.1.2 优先级队列

优先级队列（PQ，Priority Queue）中，有高、中、普通、低优先级四个队列。数据包根据事先的定义放在不同的队列中，路由器按照高、中、普通、低顺序服务，只有高优先级的队列为空后才为中优先级的队列服务，依次类推。这样能保证高优先级数据包一定是优先服务，然而如果高优先级队列长期不空，则低优先级的队列永远不会被服务。我们可以为每个队列设置一个长度，队列满后，数据包将被丢弃。

25.1.3 自定义队列

自定义队列（CQ，Custom Queue）和 PQ 不一样，在 CQ 中有 16 个队列。数据包根据事先的定义放在不同的队列中，路由器将为第一个队列服务一定包数量或者字节数的数据包后，就转为为第二个队列服务。我们可以定义不同队列中的深度，这样可以保证某个队列被服务的数据包数量较多，但不至于使得某个队列永远不会被服务。CQ 中的队列 0 比较特殊，只有队列 0 为空了，才能为其他队列服务。

25.1.4 加权公平队列

加权公平队列（WFQ，Weight Fair Queue）是低速链路（2.048M 以下）上的默认设置。WFQ 将数据包区分为不同的流，例如在 IP 中利用 IP 地址和端口号可以区分不同的 TCP 流或者 UDP 流。WFQ 为不同的流根据权重分配不同的带宽，权因子是 IP 数据包中的优先级字段。例如有 3 个流，两个流的优先级为 0，第三个为 5，总权为（1＋1＋6）=8，则前两个流每个得到带宽的 1/8，第三个流得到 6/8。

25.1.5 基于类的加权公平队列

基于类的加权公平队列（CBWFQ，Class Based Weight Fair Queue）允许用户自定义类别，并对这些类别的带宽进行控制。这在实际中很有用，例如我们可以控制我们的网络访问 Internet 时的 web 流量的带宽。可以根据数据包的协议类型、ACL、IP 优先级或者输入接口

等条件事先定义好流量的类型，为不同类别的流量配置最大带宽、占用接口带宽的百分比等。CBWFQ 可以和 NBAR、WRED 等一起使用。

25.1.6 低延迟队列

低延迟队列（LLQ，Low Latency Queue）的配置和 CBWFQ 很类似。有的数据包，例如 VOIP 的数据包，对数据的延迟非常敏感。LLQ 允许用户自定义数据类别，并优先让这些类别的数据传输，在这些数据没有传输完之前不会传输其他类别的数据。

25.1.7 加权随机早期检测

加权随机早期检测（WRED，Weight Random Early Detect）是 RED 的 Cisco 实现。当多个 TCP 连接在传输数据时，全部连接都按照最大能力传输数据，很快造成队列满，队列满后的全部数据被丢失；这时所有的发送者立即同时以最小能力传输数据，带宽开始空闲。接着全部发送者开始慢慢加大速度，于是又同时达到最大速率，又出现堵塞，如此反复。这样网络时空时堵，带宽的利用率不高。RED 则随机地丢弃 TCP 的数据包，保证链路的整体利用率。WRED 是对 RED 的改进，数据包根据 IP 优先级分成不同队列，每个队列有最小阀值、最大阀值，当平均长度小于最小阀值时，数据包不会被丢弃；随着平均队列的长度增加，丢弃的概率也增加；当平均长度大于最大阀值时，数据包按照设定的比例丢弃数据包。

25.1.7 CAR

承诺访问速率（CAR，Commited Access Rate）是一种流量策略的分类和标记的方法，它基于 IP 优先级、DSCP 值、MAC 地址或者访问控制列表来限制 IP 流量的速率。标记则可以改变 IP 优先级或者 DSCP。

CAR 使用令牌桶的机制，检查令牌桶中是否有足够的令牌。如果一个接口有可用的令牌，令牌可以从令牌桶中挪走，数据包被转发，当这个时间间隔过去后，令牌会重新添加到令牌桶中。如果接口没有可用的令牌，那么 CAR 可以定义对数据包采取的行为。CAR 使用 3 种速率定义来定义流量的速率：

- Normal rate（正常的速率）：令牌被添加到令牌桶中的平均速率，就是数据包的平均传输速率。
- Normal burst（正常的突发）：正常的突发时在时间间隔内允许正常流量速率的流量。
- Excess burst（过量突发）：超过正常突发的流量。当配置过量突发时，会借令牌并且将它添加到令牌桶中来允许某种程度的流量突发。当被借的令牌已经使用后在这个接口上收到的任何超出的流量会被扔掉。流量突发只会发生在短时间内，直到令牌桶中没有令牌存在才停止传输。

通常建议正常的流量速率配置为等于在一段时间内的平均流量速率。正常的突发速率应当等于正常速率的 1.5 倍。过量速率是正常突发速率的 2 倍。

25.1.8 基于网络的应用识别

基于网络的应用识别（NBAR，Network Based Application Recognition）实际上一个分类引擎，它查看数据包，对数据包包含的信息进行分析。NBAR 使得路由器不仅要做转发数据的工作，还要对数据包进行检查，这样会大大增加负载。NBAR 可以检查应用层的内容，例如可以检查 URL 是否有“.java”字样。NBAR 可以和许多 QOS 配合使用。

25.2 实验 1：PQ

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 PQ 的工作原理

（2） 掌握 PQ 的配置

2. 实验拓扑

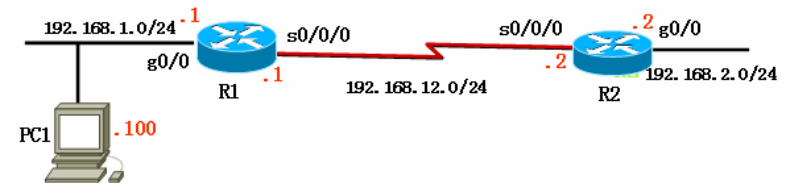

图 25-1 实验 1--实验 8 拓扑图

3. 实验步骤

（1） 步骤 1：配置 IP 地址、配置路由协议

（2） 步骤 2：配置 PQ

```
R1(config)#priority-list 1 protocol ip high tcp telnet
//创建 1 个优先级队列，标号为 1。把 telnet 流量放在高优先级队列中
R1(config)#priority-list 1 protocol ip high list 101
//以上把 ACL 101 定义的流量也放在高优先级队列中
R1(config)#priority-list 1 protocol ip medium gt 1000
//以上把数据包大小大于 1000 字节的流量放在中优先级队列中
R1(config)#priority-list 1 interface GigabitEthernet0/0 normal
//以上把从 g0/0 接口接收到流量放在普通优先级队列中
R1(config)#priority-list 1 default low
//以上把其他的流量放在低优先级队列中
R1(config)#access-list 101 permit ip host 10.1.1.1 any
//以上定义 ACL 101
R1(config)#priority-list 1 queue-limit 20 30 40 50
//以上定义优先级队列高、中、普通、低队列中的长度，如果队列超过这些长度，数据包将被丢弃。

R1(config)#int s0/0/0
R1(config-if)#priority-group 1
//以上把定义好的优先级队列应用在 s0/0/0 接口上
```

4. 实验调试

（1） 检查接口上的队列

```
R1#show interfaces s0/0/0
Serial0/0/0 is up, line protocol is up
  Hardware is GT96K Serial
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
```

```
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:04, output 00:00:03, 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
```

//接口上的队列策略是优先级队列，标号为 1

（此处省略）

（2） 查看队列的配置

```
R1#show queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:
List   Queue  Args
1      low    default
1      high   protocol ip          tcp port telnet
1      high   protocol ip          list 101
1      medium protocol ip          gt 1000
1      normal interface GigabitEthernet0/0
```

（3） 测试队列是否生效

先从 PC1 ping R2 上的 192.168.2.2，然后：

```
R1#debug priority
```

从 PC1 ping R2 的 g0/0 接口，R1 上有信息，如下：

```
*Feb 28 02:59:57.299: PQ: Serial0/0/0 output (Pk size/Q 24/0)
*Feb 28 03:00:07.299: PQ: Serial0/0/0 output (Pk size/Q 24/0)
*Feb 28 03:00:08.679: PQ: Serial0/0/0: ip (defaulting) -> low
*Feb 28 03:00:08.679: PQ: Serial0/0/0 output (Pk size/Q 56/3)
*Feb 28 03:00:14.755: PQ: Serial0/0/0: cdp (defaulting) -> low
*Feb 28 03:00:14.755: PQ: Serial0/0/0 output (Pk size/Q 326/3)
*Feb 28 03:00:17.299: PQ: Serial0/0/0 output (Pk size/Q 24/0)
```

25.3 实验 2：CQ

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 CQ 的工作原理

（2） 掌握 CQ 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

（1） 步骤 1：配置 IP 地址、配置路由协议

（2） 步骤 2：配置 CQ

```
R1(config)#queue-list 1 protocol ip 1 tcp telnet
```

//创建 1 个自定义队列，标号为 1。把 telnet 流量放在队列 1 中

```
R1(config)#queue-list 1 protocol ip 2 list 101
```

//以上把 ACL 101 定义的流量放在队列 2 中

```
R1(config)#queue-list 1 protocol ip 3 gt 1000
```

//以上把数据包大小大于 1000 字节的流量放在队列 3 中

```
R1(config)#queue-list 1 interface GigabitEthernet0/0 5
```

//以上把从 g0/0 接口接收到流量放在普通优先级队列 5 中

```
R1(config)#queue-list 1 default 6
```

//以上把其他的流量放在队列 6 中

```
R1(config)#access-list 101 permit ip host 10.1.1.1 any
```

//以上定义 ACL 101

```
R1(config)#queue-list 1 queue 1 limit 40
```

//以上定义队列 1 的深度为 40，也就是说路由器将为队列 1 服务 40 个数据包后，转向队列 2 的服务

```
R1(config)#queue-list 1 queue 2 limit 35
R1(config)#queue-list 1 queue 3 limit 30
R1(config)#queue-list 1 queue 5 limit 25

R1(config)#int s0/0/0
R1(config-if)#custom-queue-list 1
```

//以上把定义好的自定义队列应用在 s0/0/0 接口上

4. 实验调试

（1） 检查接口上的队列

```
R1#show interfaces s0/0/0
Serial0/0/0 is up, line protocol is up
  Hardware is GT96K Serial
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:05, output 00:00:04, 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: 0/40/0 2: 0/35/0 3: 0/30/0 4: 0/20/0
     5: 0/25/0 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
```

//接口上的队列策略是自定义队列，标号为 1 ，可以看到每个队列的深度

（此处省略）

（2） 查看队列配置情况

```
R1#show queueing priority
Current custom queue configuration:
List   Queue  Args
1      6      default
1      1      protocol ip          tcp port telnet
1      2      protocol ip          list 101
1      3      protocol ip          gt 1000
1      5      interface GigabitEthernet0/0
1      1      limit 40
1      2      limit 35
1      3      limit 30
1      5      limit 25
```

（3） 测试队列是否生效

```
R1#debug custom-queue
```

25.4 实验 3：WFQ

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 WFQ 的工作原理

（2） 掌握 WFQ 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

（1） 步骤 1：配置 IP 地址、配置路由协议

（2） 步骤 2：配置 WFQ

```
R1(config)#int s0/0/0
R1(config-if)# fair-queue 512 1024 10
```

//以上是在接口上启用 WFQ，实际上在 E1 速（2.048M）或者更低速率的链路上，WFQ 是默认启用的。512 是丢弃值，当队列达到 512 数据包时，数据将被丢弃；1024 是最大的会话数；10 是 RSVP 可预留队列。

4. 实验调试

```
R1#show interfaces s0/0/0
Serial0/0/0 is up, line protocol is up
  Hardware is GT96K Serial
  Internet address is 192.168.12.1/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

```
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:09, output 00:00:08, 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: 0/1000/512/0 (size/max total/threshold/drops)
   Conversations  0/0/1024 (active/max active/max total)
   Reserved Conversations 0/0 (allocated/max allocated)
   Available Bandwidth 96 kilobits/sec
```

25.5 实验 4：CBWFQ

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 CBWFQ 的工作原理

（2） 掌握 CBWFQ 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

（1） 步骤 1：配置 IP 地址、配置路由协议

（2） 步骤 2：定义 class-map

```
R1(config)#class-map match-any CLASS-MAP1
```

//以上定义了一个 class-map，名为 CLASS-MAP1，class-map 命令参见下文解释。

```
R1(config-cmap)#match protocol http
R1(config-cmap)#match protocol ftp
```

//以上定义只要是 http 或者 ftp 流量就属于 CLASS-MAP1

```
R1(config)#class-map match-all CLASS-MAP2
R1(config-cmap)#match protocol telnet
```

//以上定义只要是 telnet 流量就属于 CLASS-MAP1。系统有一个默认的 class-map，名为 class-default，凡是没有定义的流量就属于这个 class-map。

【技术要点】class-map 命令格式为：“**class-map [match-all | match-any] name**”：

- match-all：指明下面的条件必须全部满足，才可以执行，此为默认值；
- match-any：表示匹配任何一个条件就可以执行。

在 class-map 模式下，可以设置各种匹配条件，例如：

- 匹配一种协议类型：**match protocol** ***protocol-name*** 。协议类型包括 EGP，ICMP，EIGRP，DNS，HTTP，Telnet 等上百种具体协议。
- 匹配访问列表：**match access-group** { ***number*** | **name** ***acl_name*** } 。可以匹配基于号码的 list 和基于 Name 的 Access list。
- 匹配 CoS (class of Servie)：**match cos** ***cos-value*** 。匹配 IP 包中的 CoS 值。

- 匹配 IP 优先级（IP Precedence）：**match ip precedence** *precedence-value* 。匹配 IP 包中的 IP 优先级值。
- 匹配 DSCP 值（Differentiated Services Code Point）：**match ip dscp** *dscp_value* 。匹配 IP 包中的 DSCP 值。
- 匹配入接口：**match input-interface** *type number*。匹配 IP 包的进入接口。

（3） 步骤 3：定义 Policy-map

R1(config)#**policy-map MY-POLICY**

//以上是定义 policy-map。

（4） 步骤 4：配置带宽

R1(config-pmap)#**class CLASS-MAP1**

R1(config-pmap-c)#**bandwidth 60**

R1(config-pmap)#**class CLASS-MAP2**

R1(config-pmap-c)#**bandwidth 10**

//以上配置 CLASS-MAP1 流量的带宽为 60K，CLASS-MAP2 流量的带宽为 10K。该接口的总带宽为 128K。该格式为：“**bandwidth** { *bandwidth_value* | **percent** *percent_value* } 。

- 可以指定具体带宽：单位为 K。
- 也可以指明百分比：percent 关键字指定接口可用带宽百分比，可以 0--100 取值，默认情况下接口可用最大带宽为物理带宽的 75％（其余 25％留给系统自己用），所以 percent 值是这 75%的 percent，而不是物理带宽的 percent，我们可以在接口下使用“**max-reserved-bandwidth precent**”命令更改最大可用带宽。

（5） 步骤 5：将 policy-map 应用到接口上

R1(config)#**int s0/0/0**

R1(config-if)#**service-policy output MY-POLICY**

//以上把我们定义的策略应用在接口的 output 方向上，CBWFQ 只能在 output 方向。这样我们就在接口上限制了 http、ftp 和 telnet 流量的带宽。

4. 实验调试

（1） 检查 class-map 和 policy-map

R1#**show interfaces s0/0/0**

R1#**show class-map**

```
 Class Map match-all CLASS-MAP2 (id 2)
   Match protocol telnet
 Class Map match-any CLASS-MAP1 (id 1)
   Match protocol http
   Match protocol telnet
 Class Map match-any class-default (id 0)
   Match any
```

R1#**show policy-map**

```
  Policy Map MY-POLICY
    Class CLASS-MAP1
      Bandwidth 60 (kbps) Max Threshold 64 (packets)
    Class CLASS-MAP2
      Bandwidth 10 (kbps) Max Threshold 64 (packets)
```

（2） 检查策略在接口上的运用情况

```
R1#show policy-map interface s0/0/0
```

25.6 实验 5：LLQ

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 LLQ 的工作原理

（2） 掌握 LLQ 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

在实验 4 的基础上继续本实验。

（1） 步骤 1：定义 class-map3，把 IP 优先级为 critical 的 IP 流量包含进来

```
R1(config)#class-map match-any CLASS-MAP3
R1(config-cmap)#match ip precedence critical
```

（2） 步骤 2：配置 LLQ

```
R1(config)#policy-map MY-POLICY
R1(config-pmap)#class CLASS-MAP3
R1(config-pmap-c)#priority 15
```

//LLQ 的配置和 CQWFQ 配置很类似，不过使用了 **priority** 命令，我们这里限制它的带宽为 15k，超过这个带宽的数据包将被丢弃。这样 CLASS-MAP3 的流量将优先被发送，然后才发送 CLASS-MAP1 和 CLASS-MAP2 等流量。

4. 实验调试

（1） 检查 policy-map

```
R1#show policy-map
  Policy Map MY-POLICY1
    Class CLASS-MAP1
  Policy Map MY-POLICY
    Class CLASS-MAP1
      Bandwidth 60 (kbps) Max Threshold 64 (packets)
    Class CLASS-MAP2
      Bandwidth 10 (kbps) Max Threshold 64 (packets)
    Class CLASS-MAP3
      Strict Priority
      Bandwidth 15 (kbps) Burst 375 (Bytes)
```

（2） 检查策略在接口上的应用情况

```
R1#show policy-map interface s0/0/0
R1#show policy-map interface s0/0/0
 Serial0/0/0
```

（此处省略）

```
  Class-map: CLASS-MAP3 (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: ip precedence 5
    Queueing
      Strict Priority
      Output Queue: Conversation 40
      Bandwidth 15 (kbps) Burst 375 (Bytes)
      (pkts matched/bytes matched) 0/0
      (total drops/bytes drops) 0/0
```

（此处省略）

25.7 实验 6：WRED

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 WRED 的工作原理

（2） 掌握 WRED 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

（1） 步骤 1：配置 IP 地址、路由协议

（2） 步骤 2：配置 WRED

```
R1(config)#int s0/0/0
R1(config-if)#random-detect
//以上在接口上启用 WRED
R1(config-if)#random-detect precedence 0 18 42 12
```

//以上配置 IP 优先级为 0 的队列，最低阀值为 18，平均队列长度小于 18 时，数据包不会被丢弃；当平均队列长度大于 18 时，开始丢弃数据包，平均队列长度越大，丢弃的数据包越多；最大阀值为 42，平均队列长度小于 42 时，数据包按照 1/12 的比例丢弃。

4. 实验调试

```
R1#show queueing random-detect
Current random-detect configuration:
  Serial0/0/0
    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**	**18**	**42**	**1/12**
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

//以上显示 WRED 的配置情况，默认时不同 IP 优先级的队列的最低有所不同，我们更改了 IP 优先级为 0 的队列。

25.8 实验 7：CAR

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 CAR 的工作原理

（2） 掌握 CAR 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

（1） 步骤 1：配置 IP 地址、路由协议

（2） 步骤 2：配置 WRED

```
R1(config)#int s0/0/0
R1(config-if)#rate-limit output access-group 101 64000 12000 16000 conform-action set-prec-transmit 3 exceed-action set-prec-transmit 0
```

//以上在接口上启用 CAR，对于符合 ACL 101 的流量，平均速率为 64000 位/秒，正常突发量为 12000 字节/秒，过量突发量为 12000 字节/秒。

```
R1(config-if)#rate-limit output access-group 102 16000 8000 9000 conform-action set-prec-transmit 2 exceed-action drop
R1(config-if)#rate-limit output 48000 8000 10000 conform-action set-prec-transmit 0 exceed-action drop
R1(config)#access-list 101 permit tcp any any eq www
R1(config)#access-list 102 permit tcp any any eq smtp
```

【技术要点】rate-limit 的命令格式为：

rate-limit { output | input } { CIR BC BE } conform-action { action } exceed-action { action }

- CIR 单位是 bit/s；而 BC 和 BE 的单位是 byte/s。
- conform-action 的条件是指当要发的数据小于正常突发(bc)的时候

- exceed-action 是指要发的数据大于普通突发，小于最大突发(be)的时候。
- action 的选项共有如下这些：

  continue：继续执行下一条 CAR 语句

  drop：丢弃数据包

  tranmsit：转发数据包

  set-prec-continue { precedence }：设置 IP 优先级并继续执行下一条 CAR 语句

  set-prec-transmit { precedence }：设置 IP 优先级并转发数据包

  set-dscp-continue { dscp }：设置 dscp 值并继续执行下一条 CAR 语句

  set-dscp-transmit { dscp }：设置 dscp 值并转发数据包

4. 实验调试

```
R1#show interfaces rate-limit
Serial1/1
  Output
    matches: access-group 101
      params:  64000 bps, 12000 limit, 16000 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 3
      exceeded 0 packets, 0 bytes; action: set-prec-transmit 0
      last packet: 9703244ms ago, current burst: 0 bytes
      last cleared 00:03:49 ago, conformed 0 bps, exceeded 0 bps
    matches: access-group 102
      params:  16000 bps, 8000 limit, 9000 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 2
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 9703256ms ago, current burst: 0 bytes
      last cleared 00:03:41 ago, conformed 0 bps, exceeded 0 bps
    matches: all traffic
      params:  48000 bps, 8000 limit, 10000 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 0
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 9703272ms ago, current burst: 0 bytes
      last cleared 00:03:33 ago, conformed 0 bps, exceeded 0 bps
```

25.9 实验 8：NBAR

1. 实验目的

通过本实验，读者可以掌握如下技能：

（1） 理解 NBAR 的工作原理

（2） 掌握 NBAR 的配置

2. 实验拓扑

如图 25-1。

3. 实验步骤

NABR 的配置和 CBWFQ 没什么差别，因为 NBAR 实际上只是一个分类技术。我们这里将利用 NBAR 来禁止 BT 和 edonkey 下载。如下：

```
R1(config)#class-map match-any BT
R1(config-cmap)#match protocol bittorrent
R1(config-cmap)#match protocol edonkey
//定义流量，匹配 bittorrent 和 edonkey
R1(config)#policy-map DENY-BT
R1(config-pmap)#class BT
R1(config-pmap-c)#drop
//定义策略，匹配 bittorrent 和 edonkey 的流量被丢弃
R1(config)#int s0/0/0
R1(config-if)#service-policy output DNEY-BT
```

【提示】在旧的 IOS 中，class-map 模式下不能使用“**match protocol bittorrent**”等命令，要先从 Cisco 网站下载 **bittorrent.pdlm** 等文件，上传到路由器上的 FLASH 中，并使用命令“**ip nbar pdlm flash: bittorrent.pdlm**”后，才能在 class-map 模式下，使用“**match protocol bittorrent**”命令。

【提示】NBAR 需要路由器启用 CEF,默认时 CEF 是开启的,如果没有开启,可以使用“**ip cef**”命令。

25.10 本章小结

本章介绍了 QOS 的目的和基本工作原理，QOS 的各种概念显得杂乱无章。QOS 有各种拥塞避免技术：FIFO、PQ、CQ、WFQ 和 CBWFQ。它们的共同特点就是把数据流进行分类，放入不同的队列中，不同的队列有不同的处理方式。本章一一介绍以上这些技术的配置。CAR 和 NBAR 是高级的 QOS 应用，可以用来限速，甚至禁止 BT 下载等。表 25-1 是本章的命令汇总。

表 25-1 本章命令汇总

命令	作用
priority-list 1 protocol ip high tcp telnet	创建优先级队列，标号为 1。把 telnet 流量放在高优先级队列中
priority-list 1 queue-limit 20 30 40 50	定义优先级队列高、中、普通、低队列中的长度
priority-group 1	把定义好的优先级队列应用接口上
show queueing priority	查看优先级队列情况
debug priority	调试优先级队列
queue-list 1 protocol ip 1 tcp telnet	创建自定义队列，标号为 1。把 telnet 流量放在队列 1 中
queue-list 1 queue 1 limit 40	定义队列 1 的深度为 40，
custom-queue-list 1	把定义好的自定义队列应用接口上
fair-queue 512 1024 10	在接口上启用 WFQ，512 是丢弃值，1024 是最大的会话数，10 是 RSVP 可预留队列
class-map match-any CLASS-MAP1	定义 class-map，名为 CLASS-MAP1
match protocol http	匹配 http 协议

bandwidth 10	配置 CLASS-MAP 流量的带宽为 60K
service-policy output MY-POLICY	把定义好的策略应用在接口的 output 方向上
show class-map	显示 class-map 信息
show policy-map	显示 policy-map 信息
show policy-map interface s0/0/0	显示接口 s0/0/0 上的 policy-map 配置
priority 15	配置 LLQ，带宽为 15k
random-detect	在接口上启用 WRED
random-detect precedence 0 18 42 12	配置 WRED，对于 IP 优先级为 0 的队列，最低阀值为 18，最大阀值为 42，按照 1/12 的最大比例丢弃数据包
show queueing random-detect	显示 WRED 的配置情况
rate-limit output access-group 101 64000 12000 16000 conform-action set-prec-transmit 3 exceed-action set-prec-transmit 0	在接口上启用 CAR，限制符合 ACL 101 的流量
show interfaces rate-limit	显示各接口上 CAR 的情况
drop	丢弃数据包