# linux 下流量控制工具 TC 详细说明及应用实例

## 目录


一、TC 的安装.....1
二、TC 原理介绍.....1
三、TC 规则.....2
**3.1、流量控制方式**.....2
**3.2、流量控制处理对象**.....2
**3.3、操作原理**.....3
**3.4、命名规则**.....4
**3.5、单位**.....4
四、TC 命令.....5
五、具体操作.....5
**5.1、基本实现步骤**.....6
**5.2、环境模拟实例**.....6
5.2.1. 建立队列.....6
5.2.2. 建立分类.....6
5.2.3. 建立过滤器.....7
5.2.4.建立路由.....7
5.2.5. 监视.....8
5.2.6. 维护.....10
六、dms 小组应用场景一个实例.....11
**参考资料**.....12

## 一、TC 的安装

TC 是 linux 自带的模块，一般情况下不需要另行安装，可以用 man tc 查看 tc 相关命令细节，tc 要求内核 2.4.18 以上。

> **注意**，64 位机器上，先执行下面命令：
>
> ln -s /usr/lib64/tc/ /usr/lib/tc

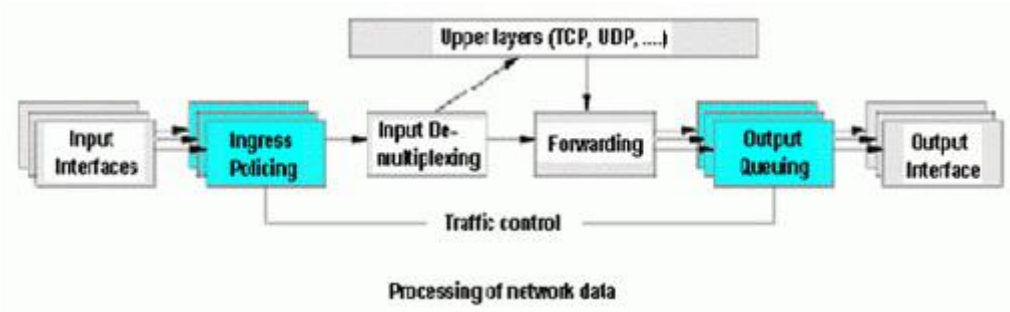
## 二、TC 原理介绍

Linux 操作系统中的流量控制器 TC（Traffic Control）用于 Linux 内核的流量控制，它利用队列规定建立处理数据包的队列，并定义队列中的数据包被发送的方式， 从而实现对流量的控制。TC 模块实现流量控制功能使用的队列规定分为两类，一类是无类队列规定， 另一类是分类队列规定。 无类队列规定相对简单，而分类队列规定则引出了分类和过滤器等概念，使其流量控制功能增强。

无类队列规定是对进入网络设备(网卡) 的数据流不加区分统一对待的队列规定。使用无类队列规定形成的队列能够 接受数据包以及重新编排、延迟或丢弃数据包。这类队列规 定形成的队列可以对整个网络设备( 网卡) 的流量进行整形， 但 不能细分各种情况… 。常用的无类队列规定主要有 pfifo _fast (先进现出) 、TBF ( 令牌桶过滤器) 、SFQ(随机公平队列) 、ID (前 向随机丢包)等等。这类队列规定使用的流量整形手段主要 是排序、 限速和丢包。

分类队列规定是对进入网络设备的数据包根据不同的需求以分类的方式区分对待的队列规定。 数据包进入一个分类的队列后， 它就需要被送到某一个类中， 也就是说需要对数据包做分类处理。对数据包进行分类的工具是过滤器，过滤器会返回一个决定，队列规定就根据这个决定把数据包送入相应的类进行排队。每个子类都可 以再次使用它们的过滤器进行进一步的分类。直到不需要进一步分类时， 数据包才进入该类包含的队列排队。 除了能够包含其它队列规定之外， 绝大多数分类的队列规定还能够对流量进行整形。 这对于需要同时进行调度( 如使用 S F Q ) 和流量控制的场合非常有用。

Linux 流量控制的基本原理如下图所示。




接收包从输入接口（Input Interface）进来后，经过流量限制（Ingress Policing）丢弃不符合规定的数据包，由输入多路分配器（Input De-Multiplexing）进行判断选择：如果接收包的目的是本主机，那么将该包送给上层处理；否则需要进行转发，将接收包交到转发块（Forwarding Block）处理。转发块同时也接收本主机上层（TCP、UDP 等）产生的包。转发块通过查看路由表，决定所处理包的下一跳。然后，对包进行排列以便将它们传送到输出接口（Output Interface）。一般我们只能限制网卡***发送***的数据包，不能限制网卡接收的数据包，所以我们可以通过改变发送次序来控制传输速率。Linux 流量控制主要是在输出接口排列时进行处理和实现的。

## 三、TC 规则

### 3.1、流量控制方式

流量控制包括以下几种方式：

**SHAPING(限制)**

当流量被限制，它的传输速率就被控制在某个值以下。限制值可以大大小于有效带宽，这样可以平滑突发数据流量，使网络更为稳定。shaping（限制）只适用于向外的流量。

**SCHEDULING(调度)**

通过调度数据包的传输，可以在带宽范围内，按照优先级分配带宽。SCHEDULING(调度)也只适于向外的流量。

**POLICING(策略)**

SHAPING 用于处理向外的流量，而 POLICIING(策略)用于处理接收到的数据。

**DROPPING(丢弃)**

如果流量超过某个设定的带宽，就丢弃数据包，不管是向内还是向外。

### 3.2、流量控制处理对象

流量的处理由三种对象控制，它们是：**qdisc(排队规则)、class(类别)和 filter(过滤器)。**

QDisc(排队规则)是 queueing discipline 的简写，它是理解流量控制(traffic control)的基础。**无论何时，内核如果需要通过某个网络接口发送数据包，它都需要按照为这个接口配置的 qdisc(排队规则)把数据包加入队列。**然后，内核会尽可能多地从 qdisc 里面取出数据包，把它们交给网络适配器驱动模块。最简单的 QDisc 是 pfifo 它不对进入的数据包做任何的处理，数据包采用先入先出的方式通过队列。不过，它会保存网络接口一时无法处理的数据包。

QDISC 的分为 CLASSLESS QDisc 和 CLASSFUL QDISC 类别如下：

**（1）、CLASSLESS QDisc(不可分类 QDisc)**

1>无类别 QDISC 包括：

**[p|b]fifo**，使用最简单的 qdisc，纯粹的先进先出。只有一个参数：limit，用来设置队列的长度,pfifo 是以数据包的个数为单位；bfifo 是以字节数为单位。

**pfifo_fast**，在编译内核时，如果打开了高级路由器(Advanced Router)编译选项，pfifo_fast 就是系统的标准 QDISC。它的队列包括三个波段(band)。在每个波段里面，使用先进先出规则。而三个波段(band)的优先级也不相同，band 0 的优先级最高，band 2 的最低。如果 band 里面有数据包，系统就不会处理 band 1 里面的数据包，band 1 和 band 2 之间也是一样。数据包是按照服务类型(Type of Service,TOS)被分配多三个波段(band)里面的。

**red**，red 是 Random Early Detection(随机早期探测)的简写。如果使用这种 QDISC，当带宽的占用接近于规定的带宽时，系统会随机地丢弃一些数据包。它非常适合高带宽应用。

**sfq**，sfq 是 Stochastic Fairness Queueing 的简写。它按照会话(session--对应于每个 TCP 连接或者 UDP 流)为流量进行排序，然后循环发送每个会话的数据包。

**tbf**，tbf 是 Token Bucket Filter 的简写，适合于把流速降低到某个值。

2>无类别 QDisc 的配置

如果没有可分类 QDisc，不可分类 QDisc 只能附属于设备的根。它们的用法如下：

```
tc qdisc add dev DEV root QDISC QDISC-PARAMETERS
```

要删除一个不可分类 QDisc，需要使用如下命令：

```
tc qdisc del dev DEV root
```

一个网络接口上如果没有设置 QDisc，pfifo_fast 就作为缺省的 QDisc。

**（2）、CLASSFUL QDISC(分类 QDisc)**

可分类 QDISC 包括：

**CBQ**，CBQ 是 Class Based Queueing(基于类别排队)的缩写。它实现了一个丰富的连接共享类别结构，既有限制(shaping)带宽的能力，也具有带宽优先级管理的能力。带宽限制是通过计算连接的空闲时间完成的。空闲时间的计算标准是数据包离队事件的频率和下层连接(数据链路层)的带宽。

**HTB**，HTB 是 Hierarchy Token Bucket 的缩写。通过在实践基础上的改进，它实现了一个丰富的连接共享类别体系。使用 HTB 可以很容易地保证每个类别的带宽，虽然它也允许特定的类可以突破带宽上限，占用别的类的带宽。HTB 可以通过 TBF(Token Bucket Filter)实现带宽限制，也能够划分类别的优先级。

**PRIO**，PRIO QDisc 不能限制带宽，因为属于不同类别的数据包是顺序离队的。使用 PRIO QDisc 可以很容易对流量进行优先级管理，只有属于高优先级类别的数据包全部发送完毕，才会发送属于低优先级类别的数据包。为了方便管理，需要使用 iptables 或者 ipchains 处理数据包的服务类型(Type Of Service,ToS)。

### 3.3、操作原理

类(Class)组成一个树，每个类都只有一个父类，而一个类可以有多个子类。某些 QDisc(例如：CBQ 和 HTB)允许在运行时动态添加类，而其它的 QDisc(例如：PRIO)不允许动态建立类。允许动态添加类的 QDisc 可以有零个或者多个子类，由它们为数据包排队。此外，每个类都有一个叶子 QDisc，默认情况下，这个叶子 QDisc 使用 pfifo 的方式排队，我们也可以使用其它类型的 QDisc 代替这个默认的 QDisc。而且，这个叶子叶子 QDisc 有可以分类，不过每个子类只能有一个叶子 QDisc。当一个数据包进入一个分类 QDisc，它会被归入某个子类。我们可以使用以下三种方式为数据包归类，不过不是所有的 QDisc 都能够使用这三种方式。

如果过滤器附属于一个类，相关的指令就会对它们进行查询。过滤器能够匹配数据包头所有的域，也可以匹配由 ipchains 或者 iptables 做的标记。

树的每个节点都可以有自己的过滤器，但是高层的过滤器也可以直接用于其子类。如果数据包没有被成功归类，就会被排到这个类的叶子 QDisc 的队中。相关细节在各个 QDisc 的手册页中。

### 3.4、命名规则

所有的 QDisc、类和过滤器都有 ID。ID 可以手工设置，也可以有内核自动分配。ID 由一个主序列号和一个从序列号组成，两个数字用一个冒号分开。

QDISC，一个 QDisc 会被分配一个主序列号，叫做句柄(handle)，然后把从序列号作为类的命名空间。句柄采用象 10:一样的表达方式。习惯上，需要为有子类的 QDisc 显式地分配一个句柄。

类(CLASS)，在同一个 QDisc 里面的类分享这个 QDisc 的主序列号，但是每个类都有自己的从序列号，叫做类识别符(classid)。类识别符只与父 QDisc 有关，和父类无关。类的命名习惯和 QDisc 的相同。

过滤器(FILTER)，过滤器的 ID 有三部分，只有在对过滤器进行散列组织才会用到。详情请参考 tc-filters 手册页。

### 3.5、单位

tc 命令的所有参数都可以使用浮点数，可能会涉及到以下计数单位。

1）、带宽或者流速单位：

| | |
|---|---|
| kbps | 千字节 / 秒 |
| mbps | 兆字节 / 秒 |
| kbit | KBits / 秒 |
| mbit | MBits / 秒 |
| bps 或者一个无单位数字 | 字节数 / 秒 |

2）、数据的数量单位：

| | |
|---|---|
| kb 或者 k | 千字节 |
| mb 或者 m | 兆字节 |
| mbit | 兆 bit |
| kbit | 千 bit |
| b 或者一个无单位数字 | 字节数 |

3）、时间的计量单位：

| | |
|---|---|
| s、sec 或者 secs | 秒 |
| ms、msec 或者 msecs | 分钟 |
| us、usec、usecs 或者一个无单位数字 | 微秒 |

## 四、TC 命令

tc 可以使用以下命令对 QDisc、类和过滤器进行操作：

add，在一个节点里加入一个 QDisc、类或者过滤器。添加时，需要传递一个祖先作为参数，传递参数时既可以使用 ID 也可以直接传递设备的根。如果要建立一个 QDisc 或者过滤器，可以使用句柄(handle)来命名；如果要建立一个类，可以使用类识别符(classid)来命名。

remove，删除有某个句柄(handle)指定的 QDisc，根 QDisc(root)也可以删除。被删除 QDisc 上的所有子类以及附属于各个类的过滤器都会被自动删除。

change，以替代的方式修改某些条目。除了句柄(handle)和祖先不能修改以外，change 命令的语法和 add 命令相同。换句话说，change 命令不能一定节点的位置。

replace，对一个现有节点进行近于原子操作的删除 / 添加。如果节点不存在，这个命令就会建立节点。

link，只适用于 DQisc，替代一个现有的节点。

例：

```
tc qdisc [ add | change | replace | link ] dev DEV [ parent qdisc-id | root ] [ handle qdisc-id ] qdisc [ qdisc specific parameters ]

tc class [ add | change | replace ] dev DEV parent qdisc-id [ classid class-id ] qdisc [ qdisc specific parameters ]

tc filter [ add | change | replace ] dev DEV [ parent qdisc-id | root ] protocol protocol prio priority filtertype [ filtertype specific parameters ] flowid flow-id

tc [-s | -d ] qdisc show [ dev DEV ]

tc [-s | -d ] class show dev DEV tc filter show dev DEV
```

## 五、具体操作

Linux 流量控制主要分为建立队列、建立分类和建立过滤器三个方面。

### 5.1、基本实现步骤

（1） 针对网络物理设备（如以太网卡 eth0）绑定一个队列 QDisc；

（2） 在该队列上建立分类 class；

（3） 为每一分类建立一个基于路由的过滤器 filter；

（4） 最后与过滤器相配合，建立特定的路由表。

### 5.2、环境模拟实例

流量控制器上的以太网卡(eth0) 的 IP 地址为 192.168.1.66，在其上建立一个 CBQ 队列。假设包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，可接收冲突的发送最长包数目为 20 字节。

假如有三种类型的流量需要控制：

1) 是发往主机 1 的，其 IP 地址为 192.168.1.24。其流量带宽控制在 8Mbit，优先级为 2；

2) 是发往主机 2 的，其 IP 地址为 192.168.1.30。其流量带宽控制在 1Mbit，优先级为 1；

3) 是发往子网 1 的，其子网号为 192.168.1.0，子网掩码为 255.255.255.0。流量带宽控制在 1Mbit，优先级为 6。

#### 5.2.1. 建立队列

一般情况下，针对一个网卡只需建立一个队列。

将一个 cbq 队列绑定到网络物理设备 eth0 上，其编号为 1:0；网络物理设备 eth0 的实际带宽为 10 Mbit，包的平均大小为 1000 字节；包间隔发送单元的大小为 8 字节，最小传输包大小为 64 字节。

```
·tc qdisc add dev eth0 root handle 1: cbq bandwidth 10Mbit avpkt 1000 cell 8 mpu 64
```

#### 5.2.2. 建立分类

分类建立在队列之上。

一般情况下，针对一个队列需建立一个根分类，然后再在其上建立子分类。对于分类，按其分类的编号顺序起作用，编号小的优先；一旦符合某个分类匹配规则，通过该分类发送数据包，则其后的分类不再起作用。

1） 创建根分类 1:1；分配带宽为 10Mbit，优先级别为 8。

```
·tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 10Mbit rate 10Mbit
maxburst 20 allot 1514 prio 8 avpkt 1000 cell 8 weight 1Mbit
```

该队列的最大可用带宽为 10Mbit，实际分配的带宽为 10Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 8，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 1Mbit。

2）创建分类 1:2，其父分类为 1:1，分配带宽为 8Mbit，优先级别为 2。

```
·tc class add dev eth0 parent 1:1 classid 1:2 cbq bandwidth 10Mbit rate 8Mbit maxburst 20 allot 1514 prio 2 avpkt 1000 cell 8 weight 800Kbit split 1:0 bounded
```

该队列的最大可用带宽为 10Mbit，实际分配的带宽为 8Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 1，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 800Kbit，分类的分离点为 1:0，且不可借用未使用带宽。

3）创建分类 1:3，其父分类为 1:1，分配带宽为 1Mbit，优先级别为 1。

```
·tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 1 avpkt 1000 cell 8 weight 100Kbit split 1:0
```

该队列的最大可用带宽为 10Mbit，实际分配的带宽为 1Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 2，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 100Kbit，分类的分离点为 1:0。

4）创建分类 1:4，其父分类为 1:1，分配带宽为 1Mbit，优先级别为 6。

```
·tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 6 avpkt 1000 cell 8 weight 100Kbit split 1:0
```

该队列的最大可用带宽为 10Mbit，实际分配的带宽为 1Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 6，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 100Kbit，分类的分离点为 1:0。

### 5.2.3. 建立过滤器

过滤器主要服务于分类。

一般只需针对根分类提供一个过滤器，然后为每个子分类提供路由映射。

1） 应用路由分类器到 cbq 队列的根，父分类编号为 1:0；过滤协议为 ip，优先级别为 100，过滤器为基于路由表。

```
·tc filter add dev eth0 parent 1:0 protocol ip prio 100 route
```

2） 建立路由映射分类 1:2, 1:3, 1:4

```
·tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 2 flowid 1:2

·tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 3 flowid 1:3

·tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 4 flowid 1:4
```

### 5.2.4.建立路由

该路由是与前面所建立的路由映射一一对应。

1） 发往主机 192.168.1.24 的数据包通过分类 2 转发(分类 2 的速率 8Mbit)

```
·ip route add 192.168.1.24 dev eth0 via 192.168.1.66 realm 2
```

2） 发往主机 192.168.1.30 的数据包通过分类 3 转发(分类 3 的速率 1Mbit)

```
·ip route add 192.168.1.30 dev eth0 via 192.168.1.66 realm 3
```

3）发往子网 192.168.1.0/24 的数据包通过分类 4 转发(分类 4 的速率 1Mbit)

```
·ip route add 192.168.1.0/24 dev eth0 via 192.168.1.66 realm 4
```

注：一般对于流量控制器所直接连接的网段建议使用 IP 主机地址流量控制限制，不要使用子网流量控制限制。如一定需要对直连子网使用子网流量控制限制，则在建立该子网的路由映射前，需将原先由系统建立的路由删除，才可完成相应步骤。

### 5.2.5. 监视

主要包括对现有队列、分类、过滤器和路由的状况进行监视。

1）显示**队列**的状况

简单显示指定设备(这里为 eth0)的队列状况

```
·tc qdisc ls dev eth0
```

```
qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
```

详细显示指定设备(这里为 eth0)的队列状况

```
·tc -s qdisc ls dev eth0
```

```
qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
     Sent 7646731 bytes 13232 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 31 undertime 0
```

这里主要显示了通过该队列发送了 13232 个数据包，数据流量为 7646731 个字节，丢弃的包数目为 0，超过速率限制的包数目为 0。

2）显示**分类**的状况

简单显示指定设备(这里为 eth0)的分类状况

·tc class ls dev eth0

```
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
  class cbq 1:1 parent 1: rate 10Mbit prio no-transmit #no-transmit 表示优先级为 8
                class cbq 1:2 parent 1:1 rate 8Mbit prio 2
                class cbq 1:3 parent 1:1 rate 1Mbit prio 1
 class cbq 1:4 parent 1:1 rate 1Mbit prio 6
```

详细显示指定设备(这里为 eth0)的分类状况

·tc -s class ls dev eth0

```
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
            Sent 17725304 bytes 32088 pkts (dropped 0, overlimits 0)
                 borrowed 0 overactions 0 avgidle 31 undertime 0
              class cbq 1:1 parent 1: rate 10Mbit prio no-transmit
            Sent 16627774 bytes 28884 pkts (dropped 0, overlimits 0)
              borrowed 16163 overactions 0 avgidle 587 undertime 0
                 class cbq 1:2 parent 1:1 rate 8Mbit prio 2
             Sent 628829 bytes 3130 pkts (dropped 0, overlimits 0)
               borrowed 0 overactions 0 avgidle 4137 undertime 0
                 class cbq 1:3 parent 1:1 rate 1Mbit prio 1
               Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
              borrowed 0 overactions 0 avgidle 159654 undertime 0
                 class cbq 1:4 parent 1:1 rate 1Mbit prio 6
            Sent 5552879 bytes 8076 pkts (dropped 0, overlimits 0)
 borrowed 3797 overactions 0 avgidle 159557 undertime 0
```

这里主要显示了通过不同分类发送的数据包，数据流量，丢弃的包数目，超过速率限制的包数目等等。其中根分类(class cbq 1:0)的状况应与队列的状况类似。

例如，分类 class cbq 1:4 发送了 8076 个数据包，数据流量为 5552879 个字节，丢弃的包数目为 0，超过速率限制的包数目为 0。

显示**过滤器**的状况

·tc -s filter ls dev eth0

```
filter parent 1: protocol ip pref 100 route
filter parent 1: protocol ip pref 100 route fh 0xffff0002 flowid 1:2 to 2
 filter parent 1: protocol ip pref 100 route fh 0xffff0003 flowid 1:3 to 3
 filter parent 1: protocol ip pref 100 route fh 0xffff0004 flowid 1:4 to 4
```

这里 flowid 1:2 代表分类 class cbq 1:2，to 2 代表通过路由 2 发送。

显示现有**路由**的状况

·ip route

```
192.168.1.66 dev eth0 scope link
192.168.1.24 via 192.168.1.66 dev eth0 realm 2
202.102.24.216 dev ppp0 proto kernel scope link src 202.102.76.5
192.168.1.30 via 192.168.1.66 dev eth0 realm 3
192.168.1.0/24 via 192.168.1.66 dev eth0 realm 4
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.66
172.16.1.0/24 via 192.168.1.66 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 202.102.24.216 dev ppp0
default via 192.168.1.254 dev eth0
```

如上所示，结尾包含有 realm 的显示行是起作用的路由过滤器。

### 5.2.6. 维护

主要包括对队列、分类、过滤器和路由的增添、修改和删除。

增添动作一般依照"队列->分类->过滤器->路由"的顺序进行；修改动作则没有什么要求；删除则依照"路由->过滤器->分类->队列"的顺序进行。

**1）队列的维护**

一般对于一台流量控制器来说，出厂时针对每个以太网卡均已配置好一个队列了，通常情况下对队列无需进行增添、修改和删除动作了。

2）分类的维护

增添，增添动作通过 tc class add 命令实现，如前面所示。

修改，修改动作通过 tc class change 命令实现，如下所示：

```
·tc class change dev eth0 parent 1:1 classid 1:2 cbq bandwidth 10Mbit rate 7Mbit maxburst
20 allot 1514 prio 2 avpkt 1000 cell 8 weight 700Kbit split 1:0 bounded
```

对于 bounded 命令应慎用，一旦添加后就进行修改，只可通过删除后再添加来实现。

删除，删除动作只在该分类没有工作前才可进行，一旦通过该分类发送过数据，则无法删除它了。因此，需要通过 shell 文件方式来修改，通过重新启动来完成删除动作。

3）过滤器的维护

增添，增添动作通过 tc filter add 命令实现，如前面所示。

修改，修改动作通过 tc filter change 命令实现，如下所示：

```
·tc filter change dev eth0 parent 1:0 protocol ip prio 100 route to 10 flowid 1:8
```

删除，删除动作通过 tc filter del 命令实现，如下所示：

```
·tc filter del dev eth0 parent 1:0 protocol ip prio 100 route to 10
```

4）与过滤器一一映射路由的维护

增添，增添动作通过 ip route add 命令实现，如前面所示。

修改，修改动作通过 ip route change 命令实现，如下所示：

```
·ip route change 192.168.1.30 dev eth0 via 192.168.1.66 realm 8
```

删除，删除动作通过 ip route del 命令实现，如下所示：

```
·ip route del 192.168.1.30 dev eth0 via 192.168.1.66 realm 8

·ip route del 192.168.1.0/24 dev eth0 via 192.168.1.66 realm 4
```

## 六、dms 小组应用场景一个实例

前面的内容大多是查找的一些资料，下面将介绍一下 dms 曾经用到的一个脚本实例，脚本如下：

```
#!/usr/bin/env bash
#脚本功能：对目的端口为“32123”的数据包进行随机延迟和丢包（注：tc 默认只对出流量有效）
randloss=`expr $RANDOM % 30`
randdelay=`expr $RANDOM % 20`

tc qdisc add dev eth1 root handle 1: prio
tc qdisc add dev eth1 parent 1:1 handle 10: netem delay `expr $randdelay`ms distribution
normal loss $randloss%
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dport 32123 0xffff flowid 1:1
```

**说明：**

`tc qdisc add dev eth1 root handle 1: prio` 用以建立一个 root（注，这里的 root 不是指 linux 下 root 用户，仅仅单纯指“根”这一概念）优先级队列，句柄为 1（**对于 tc 队列的构成及原理请详见参考资料[1]，上面讲得非常透彻**），此队列所作用的网卡接口为 eth1

`tc qdisc add dev eth1 parent 1:1 handle 10: netem delay `expr $randdelay`ms distribution normal loss $randloss%` 用以在根队列下建立一个子队列，子队列的句柄为 10（当然也可以指定为其他，一般为了方便查看，根队列为 1 位数，子队列为两位……），这个子队列的作用是调用 netem 模块让通过的网络流量

延迟，并按一定的丢包率丢包。**（注：netem 是 linux 中的另一个模块，并不是内嵌于 tc 工具中，tc 仅仅是调用这模块，所以 man tc 是看不到相关于 netem 的信息的，关于 netem 的使用，详见参考资料[5]）**

需要再说明一点是，为什么这里要建一个子队列。可以看到，我们的目标是针对相关的端口进行流量控制，但子队列现在的功能是对所有的流量都起作用，因此，需要挂载一个过滤器(filter)，对流量进行一个过滤，但在 tc 中，filter 是不能直接挂在根队列上的。

tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dport 32123 0xffff flowid 1:1
在子队列上挂载一个 filter，所有通过子队列 1:1 的流量，都会被过滤，只有满足 filter 条件的流量才会从该子队列中通过，这个条件为“目的地址为 32123，IP 地址无限制”（注，tc 默认只对出流量有效，因此在构建规则时要仔细考虑进这个因素）。

如果不想 tc 中应用的规则再起作用，执行下以脚本删除它：

```
tc qdisc del dev eth1 root
```

**参考资料**


[1]高旸、聂永峰，一种基于 Linux TC 的流量控制管理架构，计算机工程与设计，2006.10

[2]网络文章，tc，学习

http://blog.csdn.net/tqyou85/archive/2008/11/06/3226773.aspx

[3] Linux 下 TC 使用说明，百度文库

[4]网络文章，linux tc 实现 ip 流量限制，

http://blog.csdn.net/wind0513/archive/2010/03/02/5339127.aspx

[5]官方说明，netem

netem The Linux Foundation.mht