

Traffic control

There is a mature bandwidth supplying system, called Traffic Control, or TC for short. Traffic control can support classify, range, share and limit traffic by using many kinds of modes.

1 Brief introduction

In Linux OS, the main job of TC is to set up a queue at output port and carry out Traffic Control, the controlling method bases on route, this is traffic control bases on objective IP address or network number of objective subnet.

The basic function modules of TC are queue, class and filter. The queues that are supported in Linux kernel are: Class Based Queue, Token Bucket Flow, CSZ, First In First Out, Priority, TEQL, SFQ, ATM, RED.

The queue and class that we discuss here are bases on CBQ (Class Based Queue), but filter is based on Route.

In order to use TC well, we will explain the unit following.

- Unit of bandwidth or velocity of flow
 - kpbs: kilo-bytes per second
 - mbps: mega-bytes per second
 - kbit: kilo-bits per second
 - mbit: mega-bits per second
 - b or a number without unit: byte
- Unit of time
 - s, sec or secs: second
 - ms, msec or msecs: minute
 - us, usec, usecs or a number without unit: microsecond

2 Configuration

Configure and use traffic controller TC, it can be mainly separated to the following aspects: set up queue, set up class, set up filter and set up route. Additionally, it needs to monitor current queue, class, filter and route.

The basic using steps are as follows:

1. Bind a CBQ queue witch point to network physical device (such as Ethernet network card eth0);
2. Set up class on this queue;
3. Set up a filter witch is based on route for every class;
4. Lastly, set up a special route table cooperating with filter.

Take a simple environment for example:

The IP address of Ethernet network card eth0 on traffic controller is 192.9.200.66, set up a CBQ queue on it. We assume that the average size of packet is 1000 bytes; the size of the packet interval sanding unit is 8 bytes, and the packet quantity of packet that it can accept conflict is 20 bytes.

If there has three types of flux that needs to be controlled:

1. Flux that will be sent to host 1, its IP address is 192.9.200.11. Its flux bandwidth is controlled to be 8Mbit, priority is 2;
2. Flux that will be sent to host 2, its IP address is 192.9.200.21. Its flux bandwidth is controlled

to be 1Mbit, priority is 1:

3. Flux that will be sent to subnet 1, its subnet number is 192.9.200.0/24, subnet mask is 255.255.255.0. Its flux bandwidth is controlled to be 1Mbit, priority is 6.

2.1 Set up queue

In general situation, it only needs to set up one queue for one network card.

Bind a cbq queue to network physical interface eth0, its number is 1:0; the real bandwidth of network physical interface is 10Mbit, the average size of packet is 1000 bytes; packet interval sent unit is 8 bytes, minimum size of transmitted packet is 64 bytes.

```
# tc qdisc add dev eth0 root handle 0:1 cbq bandwidth 10Mbit avpkt 1000 cell 8 mpu 64
```

2.2 Set up class

Class is set up above queue. In general situation, it needs to set up a root class for a queue, and then set up sub-class above it. Class works according to its serial number, and smaller number has higher priority; once it accord with matching rule of some class, it will send data packet through this class, the other class will not work.

1. Establish root class 1:1; allocate bandwidth to be 10 Mbit, priority is 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 1 Mbit
```

The biggest available bandwidth of this queue is 10 Mbit, the actual allocated bandwidth is 10 Mbit, the quantity of longest packet that sent and can accept conflict at the same time is 20 bytes; the size of maximum transmitted unit with MAC head is 1514 bytes, priority is 8, the average size of packet is 1000 bytes, the size of packet interval sent unit is 8 bytes, corresponding weight velocity of actual bandwidth is 8 Mbit, its priority is 2.

2. Establish class 1:2, its parent class is 1:1, allocated bandwidth is 8 Mbit, and its priority is 2.

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

The biggest available bandwidth of this queue is 10 Mbit, the actual allocated bandwidth is 8 Mbit, the quantity of longest packet that sent and can accept conflict at the same time is 20 bytes; the size of maximum transmitted unit with MAC head is 1514 bytes, priority is 1, the average size of packet is 1000 bytes, the size of packet interval sent unit is 8 bytes, corresponding weight velocity of actual bandwidth is 800 Kbit, separate point of class is 1:0, and it can not borrow unused bandwidth.

3. Establish class 1:3, its parent class is 1:1, allocated bandwidth is 1 Mbit, and its priority is 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
```

The biggest available bandwidth of this queue is 10 Mbit, the actual allocated bandwidth is 1 Mbit, the quantity of longest packet that sent and can accept conflict at the same time is 20

bytes; the size of maximum transmitted unit with MAC head is 1514 bytes, priority is 2, the average size of packet is 1000 bytes, the size of packet interval sent unit is 8 bytes, corresponding weight velocity of actual bandwidth is 100 Kbit, separate point of class is 1:0.

4. Establish class 1:4, its parent class is 1:1, allocated bandwidth is 1 Mbit, and its priority is 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
```

The biggest available bandwidth of this queue is 10 Mbit, the actual allocated bandwidth is 64 Kbit, the quantity of longest packet that sent and can accept conflict at the same time is 20 bytes; the size of maximum transmitted unit with MAC head is 1514 bytes, priority is 1, the average size of packet is 1000 bytes, the size of packet interval sent unit is 8 bytes, corresponding weight velocity of actual bandwidth is 100 Kbit, separate point of class is 1:0.

2.3 Set up filter

Filter mainly serves class. Generally, it only needs to provide a filter for root class, and then provide route mapping for each sub-class.

1. Application route classifier to root of cbq queue, its parent number is 1:0; filtering protocol is ip, priority is 100, filter is based on route table.

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

2. Set up route mapping class 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
```

2.4 Set up route

This route corresponds one by one to the route mapping that set up before.

1. Data packet that is sent to host 192.9.200.11 will be transmitted through class 2 (the velocity of class 2 is 8 Mbit).

```
# ip route add 192.9.200.11 dev eth0 via 192.9.200.66 realm 2
```

2. Data packet that is sent to host 192.9.200.21 will be transmitted through class 3 (the velocity of class 3 is 1 Mbit).

```
# ip route add 192.9.200.21 dev eth0 via 192.9.200.66 realm 3
```

3. Data packet that is sent to host 192.9.200.0/24 will be transmitted through class 4 (the velocity of class 4 is 1 Mbit).

```
# ip route add 192.9.200.0/24 dev eth0 via 192.9.200.66 realm 4
```

To net section that is directly connected to traffic controller, it is suggested to use IP host address traffic control limit, don't use subnet traffic control. If it must use subnet traffic control limit to directly connected subnet, then it needs to delete original route that set up by system before set up route mapping of this subnet.

2.5 Monitor

It mainly includes monitoring queue, class, filter and route.

1. Display status of queue

- Simply display queue status of appointed device (eth0)

```
# tc qdisc ls dev eth0
qdisc cbq 1: rate 10Mbit (bounded, isolated) prio no-transmit
```

- Display queue status of appointed device (eth0) in detail

```
# 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
```

Here mainly displays that it has sent 13232 data packets through this queue, data flux is 7646731 bytes, and quantity of lost packet is 0, quantity of packets that over velocity limit is 0.

2. Display status of class

- Simply display class status of appointed device (eth0)

```
# 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 means priority is 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
```

- Display class status of appointed device (here is eth0) in detail

```
# 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, overtime 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
```

Here mainly displays data packets that sent through different class, data flux, quantity of discarded packets, and quantity of packets that over velocity limit. Among this, the status of root class (class cbq 1:0) should be similar with status of queue.

For example, class cbq 1:4 sends 8076 data packets, its data flux is 5552879 bytes, quantity of discarded packets is 0, and quantity of packets that over velocity limit is 0.

- Display status of filter

```
# 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
```

Here, flowid 1:2 delegate class cbq 1:2, to 2 means send through route 2.

- Display status of current route

```
# ip route
192.9.200.66 dev eth0 scope link
192.9.200.11 via 192.9.200.66 dev eth0 realm 2
202.102.24.216 dev ppp0 proto kernel scope link src 202.102.76.5
192.9.200.21 via 192.9.200.66 dev eth0 realm 3
192.9.200.0/24 via 192.9.200.66 dev eth0 realm 4
192.9.200.0/24 dev eth0 proto kernel scope link src 192.9.200.66
172.16.1.0/24 via 192.9.200.66 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 202.102.24.216 dev ppp0
default via 192.9.200.254 dev eth0
```

As it describes above, the display that end with realm is acting route filter.

2.6 Maintenance

Maintenance includes adding, modifying and deleting queue, class, filter and route.

Adding action always acts as “Queue→Class→Filter→Route” order; there has no request about modification; deleting always acts as “Route→Filter→Class→Queue” other.

1. Maintenance of queue

To a traffic controller, it usually has already configured a queue contrapose every Ethernet network card, in general situation, it does not need to add, modify and delete queue.

2. Maintenance of class

- Add

Adding is implemented through tc class add command, as it displayed above.

- Modify

Modification is implemented through tc class change command:

```
# 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
```

You should use bounded command cautiously, once added, then modify. It can only be implemented by adding after deleting.

- Delete

Deleting can only be carried before this class does not work, once it has sent data through this class, it can not be deleted. So, it needs to modify through shell mode, implement deleting through restarting.

3. Maintenance of filter

- Add

Adding is implemented through tc filter add command, as it described above.

- Modify

Modification is implemented through tc filter change command:

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

- Delete

Deleting is implemented through tc filter del command”

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

4. Maintenance of route that mapping one by one with filter

- Add

Adding is implemented through tc route ommand, as it described above.

- Modify

Modification is implemented through ip route change command:

```
# ip route change 192.9.200.21 dev eth0 via 192.9.200.66 realm 8
```

- Delete

Deleting is implemented through ip route del command:

```
# ip route del 192.9.200.21 dev eth0 via 192.9.200.66 realm 8
```

```
# ip route del 192.9.200.0/24 dev eth0 via 192.9.200.66 realm 4
```

Actually, Asianux 2.0 that constructed on 2.4.21 kernel can manage bandwidth like some high level special bandwidth, or even better than ATM.

Now, many people have not used these advanced functions yet. If you want to get more information, please visit <http://lartc.org>.