NAME

tc - show / manipulate traffic control settings

SYNOPSIS

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

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

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

tc [*OPTIONS*] [*FORMAT*] **qdisc show [dev** DEV **]**

tc [*OPTIONS*] [*FORMAT*] **class show dev** DEV

tc [*OPTIONS*] **filter show dev** DEV

OPTIONS := { **[-force] -b**[*atch*] **[filename]** | **[-n**[*etns*] name **]** | **[-nm** | **-nam**[*es*] **]** | **[** { **-cf** | **-c**[*onf*] } **[filename]]** }

FORMAT := { **-s**[*tatistics*] | **-d**[*etails*] | **-r**[*aw*] | **-p**[*retty*] | **-i**[*ec*] | **-g**[*raph*] }

DESCRIPTION

Tc is used to configure Traffic Control in the Linux kernel. Traffic Control consists of the following:

SHAPING
: When traffic is shaped, its rate of transmission is under control. Shaping may be more than lowering the available bandwidth - it is also used to smooth out bursts in traffic for better network behaviour. Shaping occurs on egress.

SCHEDULING
: By scheduling the transmission of packets it is possible to improve interactivity for traffic that needs it while still guaranteeing bandwidth to bulk transfers. Reordering is also called prioritizing, and happens only on egress.

POLICING
: Whereas shaping deals with transmission of traffic, policing pertains to traffic arriving. Policing thus occurs on ingress.

DROPPING
: Traffic exceeding a set bandwidth may also be dropped forthwith, both on ingress and on egress.

Processing of traffic is controlled by three kinds of objects: qdiscs, classes and filters.

QDISCS

qdisc is short for 'queueing discipline' and it is elementary to understanding traffic control. Whenever the kernel needs to send a packet to an interface, it is **enqueued** to the qdisc configured for that interface. Immediately afterwards, the kernel tries to get as many packets as possible from the qdisc, for giving them to the network adaptor driver.

A simple QDISC is the 'pfifo' one, which does no processing at all and is a pure First In, First Out queue. It does however store traffic when the network interface can't handle it momentarily.

CLASSES

Some qdiscs can contain classes, which contain further qdiscs - traffic may then be enqueued in any of the inner qdiscs, which are within the **classes.** When the kernel tries to dequeue a packet from such a **classful qdisc** it can come from any of the classes. A qdisc may for example prioritize certain kinds of traffic by trying to dequeue from certain classes before others.

FILTERS

A **filter** is used by a classful qdisc to determine in which class a packet will be enqueued. Whenever traffic arrives at a class with subclasses, it needs to be classified. Various methods may be employed to do so, one of these are the filters. All filters attached to the class are called, until one of them returns with a verdict. If no verdict was made, other criteria may be available. This differs per qdisc.

It is important to notice that filters reside **within** qdiscs - they are not masters of what happens.

The available filters are:

basic Filter packets based on an ematch expression. See tc-ematch(8) for details.

bpf Filter packets using (e)BPF, see tc-bpf(8) for details.

cgroup Filter packets based on the control group of their process. See tc-cgroup(8) for details.

flow, flower
Flow-based classifiers, filtering packets based on their flow (identified by selectable keys). See **tc-flow**(8) and tc-flower(8) for details.

fw Filter based on fwmark. Directly maps fwmark value to traffic class. See tc-fw(8).

route Filter packets based on routing table. See tc-route(8) for details.

rsvp Match Resource Reservation Protocol (RSVP) packets.

tcindex Filter packets based on traffic control index. See tc-tcindex(8).

u32 Generic filtering on arbitrary packet data, assisted by syntax to abstract common operations. See tc-u32(8) for details.

matchall
Traffic control filter that matches every packet. See tc-matchall(8) for details.

CLASSLESS QDISCS

The classless qdiscs are:

choke CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) is a classless qdisc designed to both identify and penalize flows that monopolize the queue. CHOKe is a variation of RED, and the configuration is similar to RED.

codel CoDel (pronounced "coddle") is an adaptive "no-knobs" active queue management algorithm (AQM) scheme that was developed to address the shortcomings of RED and its variants.

[p|b]fifo
Simplest usable qdisc, pure First In, First Out behaviour. Limited in packets or in bytes.

fq Fair Queue Scheduler realises TCP pacing and scales to millions of concurrent flows per qdisc.

fq_codel
Fair Queuing Controlled Delay is queuing discipline that combines Fair Queuing with the CoDel AQM scheme. FQ_Codel uses a stochastic model to classify incoming packets into different flows and is used to provide a fair share of the bandwidth to all the flows using the queue. Each such flow is managed by the CoDel queuing discipline. Reordering within a flow is avoided since Codel internally uses a FIFO queue.

gred Generalized Random Early Detection combines multiple RED queues in order to achieve multiple drop priorities. This is required to realize Assured Forwarding (RFC 2597).

hhf Heavy-Hitter Filter differentiates between small flows and the opposite, heavy-hitters. The goal is to catch the heavy-hitters and move them to a separate queue with less priority so that bulk traffic does not affect the latency of critical traffic.

ingress This is a special qdisc as it applies to incoming traffic on an interface, allowing for it to be filtered and policed.

mqprio The Multiqueue Priority Qdisc is a simple queuing discipline that allows mapping traffic flows to hardware queue ranges using priorities and a configurable priority to traffic class mapping. A traffic class in this context is a set of contiguous qdisc classes which map 1:1 to a set of hardware exposed queues.

multiq Multiqueue is a qdisc optimized for devices with multiple Tx queues. It has been added for hardware that wishes to avoid head-of-line blocking. It will cycle though the bands and verify that the hardware queue associated with the band is not stopped prior to dequeuing a packet.

netem Network Emulator is an enhancement of the Linux traffic control facilities that allow to add delay, packet loss, duplication and more other characteristics to packets outgoing from a selected network interface.

pfifo_fast
Standard qdisc for 'Advanced Router' enabled kernels. Consists of a three-band queue which honors Type of Service flags, as well as the priority that may be assigned to a packet.

pie Proportional Integral controller-Enhanced (PIE) is a control theoretic active queue management scheme. It is based on the proportional integral controller but aims to control delay.

red Random Early Detection simulates physical congestion by randomly dropping packets when nearing configured bandwidth allocation. Well suited to very large bandwidth applications.

rr Round-Robin qdisc with support for multiqueue network devices. Removed from Linux since kernel version 2.6.27.

sfb Stochastic Fair Blue is a classless qdisc to manage congestion based on packet loss and link utilization history while trying to prevent non-responsive flows (i.e. flows that do not react to congestion marking or dropped packets) from impacting performance of responsive flows. Unlike RED, where the marking probability has to be configured, BLUE tries to determine the ideal marking probability automatically.

sfq Stochastic Fairness Queueing reorders queued traffic so each 'session' gets to send a packet in turn.

tbf The Token Bucket Filter is suited for slowing traffic down to a precisely configured rate. Scales well to large bandwidths.

CONFIGURING CLASSLESS QDISCS

In the absence of classful qdiscs, classless qdiscs can only be attached at the root of a device. Full syntax:

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

To remove, issue

tc qdisc del dev DEV **root**

The **pfifo_fast** qdisc is the automatic default in the absence of a configured qdisc.

CLASSFUL QDISCS

The classful qdiscs are:

ATM Map flows to virtual circuits of an underlying asynchronous transfer mode device.

CBQ Class Based Queueing implements a rich linksharing hierarchy of classes. It contains shaping elements as well as prioritizing capabilities. Shaping is performed using link idle time calculations based on average packet size and underlying link bandwidth. The latter may be ill-defined for some interfaces.

DRR The Deficit Round Robin Scheduler is a more flexible replacement for Stochastic Fairness Queueing. Unlike SFQ, there are no built-in queues -- you need to add classes and then set up filters to classify packets accordingly. This can be useful e.g. for using RED qdiscs with different settings for particular traffic. There is no default class -- if a packet cannot be classified, it is dropped.

DSMARK
Classify packets based on TOS field, change TOS field of packets based on classification.

HFSC Hierarchical Fair Service Curve guarantees precise bandwidth and delay allocation for leaf classes and allocates excess bandwidth fairly. Unlike HTB, it makes use of packet dropping to achieve low delays which interactive sessions benefit from.

HTB The Hierarchy Token Bucket implements a rich linksharing hierarchy of classes with an emphasis on conforming to existing practices. HTB facilitates guaranteeing bandwidth to classes, while also allowing specification of upper limits to inter-class sharing. It contains shaping elements, based on TBF and can prioritize classes.

PRIO The PRIO qdisc is a non-shaping container for a configurable number of classes which are dequeued in order. This allows for easy prioritization of traffic, where lower classes are only able to send if higher ones have no packets available. To facilitate configuration, Type Of Service bits are honored by default.

QFQ Quick Fair Queueing is an O(1) scheduler that provides near-optimal guarantees, and is the first to achieve that goal with a constant cost also with respect to the number of groups and the packet length. The QFQ algorithm has no loops, and uses very simple instructions and data structures that lend themselves very well to a hardware implementation.

THEORY OF OPERATION

Classes form a tree, where each class has a single parent. A class may have multiple children. Some qdiscs allow for runtime addition of classes (CBQ, HTB) while others (PRIO) are created with a static number of children.

Qdiscs which allow dynamic addition of classes can have zero or more subclasses to which traffic may be enqueued.

Furthermore, each class contains a **leaf qdisc** which by default has **pfifo** behaviour, although another qdisc can be attached in place. This qdisc may again contain classes, but each class can have only one leaf qdisc.

When a packet enters a classful qdisc it can be **classified** to one of the classes within. Three criteria are available, although not all qdiscs will use all three:

tc filters
If tc filters are attached to a class, they are consulted first for relevant instructions. Filters can match on all fields of a packet header, as well as on the firewall mark applied by ipchains or iptables.

Type of Service
Some qdiscs have built in rules for classifying packets based on the TOS field.

skb->priority
Userspace programs can encode a class-id in the 'skb->priority' field using the SO_PRIORITY option.

Each node within the tree can have its own filters but higher level filters may also point directly to lower classes.

If classification did not succeed, packets are enqueued to the leaf qdisc attached to that class. Check qdisc specific manpages for details, however.

NAMING

All qdiscs, classes and filters have IDs, which can either be specified or be automatically assigned.

IDs consist of a **major** number and a **minor** number, separated by a colon - **major**:**minor**. Both**major** and **minor** are hexadecimal numbers and are limited to 16 bits. There are two special values: root is signified by **major** and **minor** of all ones, and unspecified is all zeros.

QDISCS
A qdisc, which potentially can have children, gets assigned a **major** number, called a 'handle', leaving the **minor** number namespace available for classes. The handle is expressed as '10:'. It is customary to explicitly assign a handle to qdiscs expected to have children.

CLASSES
Classes residing under a qdisc share their qdisc **major** number, but each have a separate **minor** number called a 'classid' that has no relation to their parent classes, only to their parent qdisc. The same naming custom as for qdiscs applies.

FILTERS
Filters have a three part ID, which is only needed when using a hashed filter hierarchy.

PARAMETERS

The following parameters are widely used in TC. For other parameters, see the man pages for individual qdiscs.

RATES Bandwidths or rates. These parameters accept a floating point number, possibly followed by a unit (both SI and IEC units supported).

bit or a bare number
Bits per second

kbit Kilobits per second

mbit Megabits per second

gbit Gigabits per second

tbit Terabits per second

bps Bytes per second

kbps Kilobytes per second

mbps Megabytes per second

gbps Gigabytes per second

tbps Terabytes per second

To specify in IEC units, replace the SI prefix (k-, m-, g-, t-) with IEC prefix (ki-, mi-, gi- and ti-) respectively.

TC store rates as a 32-bit unsigned integer in bps internally, so we can specify a max rate of 4294967295 bps.

TIMES Length of time. Can be specified as a floating point number followed by an optional unit:

s, sec or secs
Whole seconds

ms, msec or msecs
Milliseconds

us, usec, usecs or a bare number
Microseconds.

TC defined its own time unit (equal to microsecond) and stores time values as 32-bit unsigned integer, thus we can specify a max time value of 4294967295 usecs.

SIZES Amounts of data. Can be specified as a floating point number followed by an optional unit:

b or a bare number
Bytes.

kbit Kilobits

kb or k Kilobytes

mbit Megabits

mb or m
Megabytes

gbit Gigabits

gb or g Gigabytes

TC stores sizes internally as 32-bit unsigned integer in byte, so we can specify a max size of 4294967295 bytes.

VALUES
Other values without a unit. These parameters are interpreted as decimal by default, but you can indicate TC to interpret them as octal and hexadecimal by adding a '0' or '0x' prefix respectively.

TC COMMANDS

The following commands are available for qdiscs, classes and filter:

add Add a qdisc, class or filter to a node. For all entities, a **parent** must be passed, either by passing its ID or by attaching directly to the root of a device. When creating a qdisc or a filter, it can be named with the **handle** parameter. A class is named with the **classid** parameter.

delete A qdisc can be deleted by specifying its handle, which may also be 'root'. All subclasses and their leaf qdiscs are automatically deleted, as well as any filters attached to them.

change Some entities can be modified 'in place'. Shares the syntax of 'add', with the exception that the handle cannot be changed and neither can the parent. In other words, **change** cannot move a node.

replace Performs a nearly atomic remove/add on an existing node id. If the node does not exist yet it is created.

get Displays a single filter given the interface, parent ID, priority, protocol and handle ID.

show Displays all filters attached to the given interface. A valid parent ID must be passed.

link Only available for qdiscs and performs a replace where the node must exist already.

OPTIONS

-b, **-b filename**, **-batch**, **-batch filename**
read commands from provided file or standard input and invoke them. First failure will cause termination of tc.

-force don't terminate tc on errors in batch mode. If there were any errors during execution of the commands, the application return code will be non zero.

-n, **-net**, **-netns** <NETNS>
switches **tc** to the specified network namespace *NETNS*. Actually it just simplifies executing of:

ip netns exec *NETNS* **tc** [*OPTIONS*] *OBJECT* { *COMMAND* | **help** }

to

tc -n[etns] *NETNS* [*OPTIONS*] *OBJECT* { *COMMAND* | **help** }

-cf, **-conf** <FILENAME>
specifies path to the config file. This option is used in conjunction with other options (e.g. **-nm**).

FORMAT

The show command has additional formatting options:

-s, **-stats**, **-statistics**
output more statistics about packet usage.

-d, **-details**
output more detailed information about rates and cell sizes.

-r, **-raw**
output raw hex values for handles.

-p, **-pretty**
decode filter offset and mask values to equivalent filter commands based on TCP/IP.

-iec print rates in IEC units (ie. 1K = 1024).

-g, **-graph**
shows classes as ASCII graph. Prints generic stats info under each class if **-s** option was specified. Classes can be filtered only by **dev** option.

-nm, **-name**
resolve class name from **/etc/iproute2/tc_cls** file or from file specified by **-cf** option. This file is just a mapping of **classid** to class name:

```
# Here is comment
1:40 voip # Here is another comment
1:50 web
1:60 ftp
1:2 home
```

tc will not fail if **-nm** was specified without **-cf** option but **/etc/iproute2/tc_cls** file does not exist, which makes it possible to pass **-nm** option for creating **tc** alias.

EXAMPLES

tc -g class show dev eth0
Shows classes as ASCII graph on eth0 interface.

tc -g -s class show dev eth0
Shows classes as ASCII graph with stats info under each class.

HISTORY

tc was written by Alexey N. Kuznetsov and added in Linux 2.2.

SEE ALSO

tc-basic(8), tc-bfifo(8), tc-bpf(8), tc-cbq(8), tc-cgroup(8), tc-choke(8), tc-codel(8), tc-drr(8), tc-ematch(8), tc-flow(8), tc-flower(8), tc-fq(8), tc-fq_codel(8), tc-fw(8), tc-hfsc(7), tc-hfsc(8), tc-htb(8), tc-mqprio(8), tc-pfifo(8), tc-pfifo_fast(8), tc-red(8), tc-route(8), tc-sfb(8), tc-sfq(8), tc-stab(8), tc-tbf(8), tc-tcindex(8), tc-u32(8),
User documentation at http://lartc.org/ ", but please direct bugreports and patches to: " <netdev@vger.kernel.org>

AUTHOR

Manpage maintained by bert hubert (ahu@ds9a.nl)