NAME

HTB − Hierarchy Token Bucket

SYNOPSIS

tc qdisc ... dev dev **(parent** classid | **root)** [**handle** major:] **htb** [**default** minor-id]

tc class ... dev dev **parent** major:[minor] [**classid** major:minor] **htb rate** rate [**ceil** rate] **burst** bytes [**cburst** bytes] [**prio** priority]

DESCRIPTION

HTB is meant as a more understandable and intuitive replacement for the CBQ qdisc in Linux. Both CBQ and HTB help you to control the use of the outbound bandwidth on a given link. Both allow you to use one physical link to simulate several slower links and to send different kinds of traffic on different simulated links. In both cases, you have to specify how to divide the physical link into simulated links and how to decide which simulated link to use for a given packet to be sent.

Unlike CBQ, HTB shapes traffic based on the Token Bucket Filter algorithm which does not depend on interface characteristics and so does not need to know the underlying bandwidth of the outgoing interface.

SHAPING ALGORITHM

Shaping works as documented in **tc-tbf (8).**

CLASSIFICATION

Within the one HRB instance many classes may exist. Each of these classes contains another qdisc, by default **tc-pfifo**(8).

When enqueueing a packet, HTB starts at the root and uses various methods to determine which class should receive the data.

In the absence of uncommon configuration options, the process is rather easy. At each node we look for an instruction, and then go to the class the instruction refers us to. If the class found is a barren leaf-node (without children), we enqueue the packet there. If it is not yet a leaf node, we do the whole thing over again starting from that node.

The following actions are performed, in order at each node we visit, until one sends us to another node, or terminates the process.

(i) Consult filters attached to the class. If sent to a leafnode, we are done. Otherwise, restart.

(ii) If none of the above returned with an instruction, enqueue at this node.

This algorithm makes sure that a packet always ends up somewhere, even while you are busy building your configuration.

LINK SHARING ALGORITHM

FIXME

QDISC

The root of a HTB qdisc class tree has the following parameters:

parent major:minor | root
: This mandatory parameter determines the place of the HTB instance, either at the **root** of an interface or within an existing class.

handle major:
: Like all other qdiscs, the HTB can be assigned a handle. Should consist only of a major number, followed by a colon. Optional, but very useful if classes will be generated within this qdisc.

default minor-id
: Unclassified traffic gets sent to the class with this minor-id.

CLASSES

Classes have a host of parameters to configure their operation.

parent major:minor
: Place of this class within the hierarchy. If attached directly to a qdisc and not to another class, minor can be omitted. Mandatory.

classid major:minor
: Like qdiscs, classes can be named. The major number must be equal to the major number of the qdisc to which it belongs. Optional, but needed if this class is going to have children.

prio priority
: In the round-robin process, classes with the lowest priority field are tried for packets first. Mandatory.

rate rate
: Maximum rate this class and all its children are guaranteed. Mandatory.

ceil rate
: Maximum rate at which a class can send, if its parent has bandwidth to spare. Defaults to the configured rate, which implies no borrowing

burst bytes
: Amount of bytes that can be burst at **ceil** speed, in excess of the configured **rate.** Should be at least as high as the highest burst of all children.

cburst bytes
: Amount of bytes that can be burst at 'infinite' speed, in other words, as fast as the interface can transmit them. For perfect evening out, should be equal to at most one average packet. Should be at least as high as the highest cburst of all children.

NOTES

Due to Unix timing constraints, the maximum ceil rate is not infinite and may in fact be quite low. On Intel, there are 100 timer events per second, the maximum rate is that rate at which 'burst' bytes are sent each timer tick. From this, the mininum burst size for a specified rate can be calculated. For i386, a 10mbit rate requires a 12 kilobyte burst as 100*12kb*8 equals 10mbit.

SEE ALSO

tc(8)

HTB website: http://luxik.cdi.cz/~devik/qos/htb/

AUTHOR

Martin Devera <devik@cdi.cz>. This manpage maintained by bert hubert <ahu@ds9a.nl>