## NAME

NetEm – Network Emulator

## SYNOPSIS

**tc qdisc ... dev** *DEVICE* ] **add netem** *OPTIONS*

*OPTIONS* := [ *LIMIT* ] [ *DELAY* ] [ *LOSS* ] [ *CORRUPT* ] [ *DUPLICATION* ] [ *REORDERING* ] [ *RATE* ] [ *SLOT* ]

*LIMIT* := **limit** *packets*

*DELAY* := **delay** *TIME* [ *JITTER* [ *CORRELATION* ]]]
[ **distribution** { **uniform** | **normal** | **pareto** | **paretonormal** } ]

*LOSS* := **loss** { **random** *PERCENT* [ *CORRELATION* ] |
**state** *p13* [ *p31* [ *p32* [ *p23* [ *p14*]]]] |
**gemodel** *p* [ *r* [ *1-h* [ *1-k* ]]] } [ **ecn** ]

*CORRUPT* := **corrupt** *PERCENT* [ *CORRELATION* ]]

*DUPLICATION* := **duplicate** *PERCENT* [ *CORRELATION* ]]

*REORDERING* := **reorder** *PERCENT* [ *CORRELATION* ] [ **gap** *DISTANCE* ]

*RATE* := **rate** *RATE* [ *PACKETOVERHEAD* [ *CELLSIZE* [ *CELLOVERHEAD* ]]]]

*SLOT* := **slot** { *MIN_DELAY* [ *MAX_DELAY* ] |
**distribution** { **uniform** | **normal** | **pareto** | **paretonormal** | *FILE* } *DELAY JITTER* }
[ **packets** *PACKETS* ] [ **bytes** *BYTES* ]

## DESCRIPTION

NetEm 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. NetEm is built using the existing Quality Of Service (QOS) and Differentiated Services (diffserv) facilities in the Linux kernel.

## netem OPTIONS

netem has the following options:

### limit packets

maximum number of packets the qdisc may hold queued at a time.

### delay

adds the chosen delay to the packets outgoing to chosen network interface. The optional parameters allows to introduce a delay variation and a correlation. Delay and jitter values are expressed in ms while correlation is percentage.

### distribution

allow the user to choose the delay distribution. If not specified, the default distribution is Normal. Additional parameters allow to consider situations in which network has variable delays depending on traffic flows concurring on the same path, that causes several delay peaks and a tail.

### loss random

adds an independent loss probability to the packets outgoing from the chosen network interface. It is also possible to add a correlation, but this option is now deprecated due to the noticed bad behavior.

### loss state

adds packet losses according to the 4-state Markov using the transition probabilities as input parameters. The parameter p13 is mandatory and if used alone corresponds to the Bernoulli model. The optional parameters allows to extend the model to 2-state (p31), 3-state (p23 and p32) and 4-state (p14). State 1

corresponds to good reception, State 4 to independent losses, State 3 to burst losses and State 2 to good reception within a burst.

**loss gemodel**
adds packet losses according to the Gilbert-Elliot loss model or its special cases (Gilbert, Simple Gilbert and Bernoulli). To use the Bernoulli model, the only needed parameter is p while the others will be set to the default values r=1-p, 1-h=1 and 1-k=0. The parameters needed for the Simple Gilbert model are two (p and r), while three parameters (p, r, 1-h) are needed for the Gilbert model and four (p, r, 1-h and 1-k) are needed for the Gilbert-Elliot model. As known, p and r are the transition probabilities between the bad and the good states, 1-h is the loss probability in the bad state and 1-k is the loss probability in the good state.

**ecn**
can be used optionally to mark packets instead of dropping them. A loss model has to be used for this to be enabled.

**corrupt**
allows the emulation of random noise introducing an error in a random position for a chosen percent of packets. It is also possible to add a correlation through the proper parameter.

**duplicate**
using this option the chosen percent of packets is duplicated before queuing them. It is also possible to add a correlation through the proper parameter.

**reorder**
to use reordering, a delay option must be specified. There are two ways to use this option (assuming 'delay 10ms' in the options list).

**reorder** *25% 50%* **gap** *5*
in this first example, the first 4 (gap - 1) packets are delayed by 10ms and subsequent packets are sent immediately with a probability of 0.25 (with correlation of 50% ) or delayed with a probability of 0.75. After a packet is reordered, the process restarts i.e. the next 4 packets are delayed and subsequent packets are sent immediately or delayed based on reordering probability. To cause a repeatable pattern where every 5th packet is reordered reliably, a reorder probability of 100% can be used.

**reorder** *25% 50%*
in this second example 25% of packets are sent immediately (with correlation of 50%) while the others are delayed by 10 ms.

**rate**
delay packets based on packet size and is a replacement for *TBF*. Rate can be specified in common units (e.g. 100kbit). Optional *PACKETOVERHEAD* (in bytes) specify an per packet overhead and can be negative. A positive value can be used to simulate additional link layer headers. A negative value can be used to artificial strip the Ethernet header (e.g. -14) and/or simulate a link layer header compression scheme. The third parameter - an unsigned value - specify the cellsize. Cellsize can be used to simulate link layer schemes. ATM for example has an payload cellsize of 48 bytes and 5 byte per cell header. If a packet is 50 byte then ATM must use two cells: 2 * 48 bytes payload including 2 * 5 byte header, thus consume 106 byte on the wire. The last optional value *CELLOVERHEAD* can be used to specify per cell overhead - for our ATM example 5. *CELLOVERHEAD* can be negative, but use negative values with caution.

Note that rate throttling is limited by several factors: the kernel clock granularity avoid a perfect shaping at a specific level. This will show up in an artificial packet compression (bursts). Another influence factor are network adapter buffers which can also add artificial delay.

**slot**
defer delivering accumulated packets to within a slot. Each available slot can be configured with a minimum delay to acquire, and an optional maximum delay. Alternatively it can be configured with the distribution similar to **distribution** for **delay** option. Slot delays can be specified in nanoseconds, microseconds,

milliseconds or seconds (e.g. 800us). Values for the optional parameters *BYTES* will limit the number of bytes delivered per slot, and/or *PACKETS* will limit the number of packets delivered per slot.

These slot options can provide a crude approximation of bursty MACs such as DOCSIS, WiFi, and LTE.

Note that slotting is limited by several factors: the kernel clock granularity, as with a rate, and attempts to deliver many packets within a slot will be smeared by the timer resolution, and by the underlying native bandwidth also.

It is possible to combine slotting with a rate, in which case complex behaviors where either the rate, or the slot limits on bytes or packets per slot, govern the actual delivered rate.

## LIMITATIONS

The main known limitation of Netem are related to timer granularity, since Linux is not a real-time operating system.

## EXAMPLES

tc qdisc add dev eth0 root netem rate 5kbit 20 100 5

delay all outgoing packets on device eth0 with a rate of 5kbit, a per packet overhead of 20 byte, a cellsize of 100 byte and a per celloverhead of 5 byte:

## SOURCES

1. Hemminger S. , "Network Emulation with NetEm", Open Source Development Lab, April 2005 (http://devresources.linux-foundation.org/shemminger/netem/LCA2005_paper.pdf)

2. Netem page from Linux foundation, (http://www.linuxfoundation.org/en/Net:Netem)

3. Salsano S., Ludovici F., Ordine A., "Definition of a general and intuitive loss model for packet networks and its implementation in the Netem module in the Linux kernel", available at http://netgroup.uniroma2.it/NetemCLG

## SEE ALSO

tc(8), tc-tbf(8)

## AUTHOR

Netem was written by Stephen Hemminger at Linux foundation and is based on NISTnet. This manpage was created by Fabio Ludovici <fabio.ludovici at yahoo dot it> and Hagen Paul Pfeifer <hagen@jauu.net>