NAME

CAKE – Common Applications Kept Enhanced (CAKE)

SYNOPSIS

tc qdisc ... cake
[**bandwidth** RATE | **unlimited*** | **autorate-ingress**]
[**rtt** TIME | **datacentre** | **lan** | **metro** | **regional** | **internet*** | **oceanic** | **satellite** | **interplanetary**]
[**besteffort** | **diffserv8** | **diffserv4** | **diffserv3***]
[**flowblind** | **srchost** | **dsthost** | **hosts** | **flows** | **dual-srchost** | **dual-dsthost** | **triple-isolate***]
[**nat** | **nonat***]
[**wash** | **nowash***]
[**split-gso*** | **no-split-gso**]
[**ack-filter** | **ack-filter-aggressive** | **no-ack-filter***]
[**memlimit** LIMIT]
[**fwmark** MASK]
[**ptm** | **atm** | **noatm***]
[**overhead** N | **conservative** | **raw***]
[**mpu** N]
[**ingress** | **egress***]
(* marks defaults)

DESCRIPTION

CAKE (Common Applications Kept Enhanced) is a shaping-capable queue discipline which uses both AQM and FQ. It combines COBALT, which is an AQM algorithm combining Codel and BLUE, a shaper which operates in deficit mode, and a variant of DRR++ for flow isolation. 8-way set-associative hashing is used to virtually eliminate hash collisions. Priority queuing is available through a simplified diffserv implementation. Overhead compensation for various encapsulation schemes is tightly integrated.

All settings are optional; the default settings are chosen to be sensible in most common deployments. Most people will only need to set the **bandwidth** parameter to get useful results, but reading the **Overhead Compensation** and **Round Trip Time** sections is strongly encouraged.

SHAPER PARAMETERS

CAKE uses a deficit-mode shaper, which does not exhibit the initial burst typical of token-bucket shapers. It will automatically burst precisely as much as required to maintain the configured throughput. As such, it is very straightforward to configure.

unlimited (default)
No limit on the bandwidth.

bandwidth RATE
Set the shaper bandwidth. See **tc(8)** or examples below for details of the RATE value.

autorate-ingress
Automatic capacity estimation based on traffic arriving at this qdisc. This is most likely to be useful with cellular links, which tend to change quality randomly. A **bandwidth** parameter can be used in conjunction to specify an initial estimate. The shaper will periodically be set to a bandwidth slightly below the estimated rate. This estimator cannot estimate the bandwidth of links downstream of itself.

OVERHEAD COMPENSATION PARAMETERS

The size of each packet on the wire may differ from that seen by Linux. The following parameters allow CAKE to compensate for this difference by internally considering each packet to be bigger than Linux informs it. To assist users who are not expert network engineers, keywords have been provided to represent a number of common link technologies.

Manual Overhead Specification

overhead BYTES

Adds BYTES to the size of each packet. BYTES may be negative; values between -64 and 256 (inclusive) are accepted.

mpu BYTES

Rounds each packet (including overhead) up to a minimum length BYTES. BYTES may not be negative; values between 0 and 256 (inclusive) are accepted.

atm

Compensates for ATM cell framing, which is normally found on ADSL links. This is performed after the **overhead** parameter above. ATM uses fixed 53-byte cells, each of which can carry 48 bytes payload.

ptm

Compensates for PTM encoding, which is normally found on VDSL2 links and uses a 64b/65b encoding scheme. It is even more efficient to simply derate the specified shaper bandwidth by a factor of 64/65 or 0.984. See ITU G.992.3 Annex N and IEEE 802.3 Section 61.3 for details.

noatm

Disables ATM and PTM compensation.

Failsafe Overhead Keywords

These two keywords are provided for quick-and-dirty setup. Use them if you can't be bothered to read the rest of this section.

raw (default)

Turns off all overhead compensation in CAKE. The packet size reported by Linux will be used directly.

Other overhead keywords may be added after "raw". The effect of this is to make the overhead compensation operate relative to the reported packet size, not the underlying IP packet size.

conservative

Compensates for more overhead than is likely to occur on any widely-deployed link technology. Equivalent to **overhead 48 atm.**

ADSL Overhead Keywords

Most ADSL modems have a way to check which framing scheme is in use. Often this is also specified in the settings document provided by the ISP. The keywords in this section are intended to correspond with these sources of information. All of them implicitly set the **atm** flag.

pppoa-vcmux

Equivalent to **overhead 10 atm**

pppoa-llc

Equivalent to **overhead 14 atm**

pppoe-vcmux

Equivalent to **overhead 32 atm**

pppoe-llcsnap

Equivalent to **overhead 40 atm**

bridged-vcmux

Equivalent to **overhead 24 atm**

bridged-llcsnap

Equivalent to **overhead 32 atm**

ipoa-vcmux

Equivalent to **overhead 8 atm**

ipoa-llcsnap
Equivalent to **overhead 16 atm**

See also the Ethernet Correction Factors section below.

VDSL2 Overhead Keywords

ATM was dropped from VDSL2 in favour of PTM, which is a much more straightforward framing scheme. Some ISPs retained PPPoE for compatibility with their existing back-end systems.

pppoe-ptm
Equivalent to **overhead 30 ptm**

PPPoE: 2B PPP + 6B PPPoE +
ETHERNET: 6B dest MAC + 6B src MAC + 2B ethertype + 4B Frame Check Sequence +
PTM: 1B Start of Frame (S) + 1B End of Frame (Ck) + 2B TC-CRC (PTM-FCS)

bridged-ptm
Equivalent to **overhead 22 ptm**
ETHERNET: 6B dest MAC + 6B src MAC + 2B ethertype + 4B Frame Check Sequence +
PTM: 1B Start of Frame (S) + 1B End of Frame (Ck) + 2B TC-CRC (PTM-FCS)

See also the Ethernet Correction Factors section below.

DOCSIS Cable Overhead Keyword

DOCSIS is the universal standard for providing Internet service over cable-TV infrastructure.

In this case, the actual on-wire overhead is less important than the packet size the head-end equipment uses for shaping and metering. This is specified to be an Ethernet frame including the CRC (aka FCS).

docsis
Equivalent to **overhead 18 mpu 64 noatm**

Ethernet Overhead Keywords

ethernet
Accounts for Ethernet's preamble, inter-frame gap, and Frame Check Sequence. Use this keyword when the bottleneck being shaped for is an actual Ethernet cable.
Equivalent to **overhead 38 mpu 84 noatm**

ether-vlan
Adds 4 bytes to the overhead compensation, accounting for an IEEE 802.1Q VLAN header appended to the Ethernet frame header. NB: Some ISPs use one or even two of these within PPPoE; this keyword may be repeated as necessary to express this.

ROUND TRIP TIME PARAMETERS

Active Queue Management (AQM) consists of embedding congestion signals in the packet flow, which receivers use to instruct senders to slow down when the queue is persistently occupied. CAKE uses ECN signalling when available, and packet drops otherwise, according to a combination of the Codel and BLUE AQM algorithms called COBALT.

Very short latencies require a very rapid AQM response to adequately control latency. However, such a rapid response tends to impair throughput when the actual RTT is relatively long. CAKE allows specifying the RTT it assumes for tuning various parameters. Actual RTTs within an order of magnitude of this will generally work well for both throughput and latency management.

At the 'lan' setting and below, the time constants are similar in magnitude to the jitter in the Linux kernel itself, so congestion might be signalled prematurely. The flows will then become sparse and total throughput

reduced, leaving little or no back-pressure for the fairness logic to work against. Use the "metro" setting for local lans unless you have a custom kernel.

rtt TIME
Manually specify an RTT.

datacentre
For extremely high-performance 10GigE+ networks only. Equivalent to **rtt 100us.**

lan
For pure Ethernet (not Wi-Fi) networks, at home or in the office. Don't use this when shaping for an Internet access link. Equivalent to **rtt 1ms.**

metro
For traffic mostly within a single city. Equivalent to **rtt 10ms.**

regional
For traffic mostly within a European-sized country. Equivalent to **rtt 30ms.**

internet (default)
This is suitable for most Internet traffic. Equivalent to **rtt 100ms.**

oceanic
For Internet traffic with generally above-average latency, such as that suffered by Australasian residents. Equivalent to **rtt 300ms.**

satellite
For traffic via geostationary satellites. Equivalent to **rtt 1000ms.**

interplanetary
So named because Jupiter is about 1 light-hour from Earth. Use this to (almost) completely disable AQM actions. Equivalent to **rtt 3600s.**

FLOW ISOLATION PARAMETERS

With flow isolation enabled, CAKE places packets from different flows into different queues, each of which carries its own AQM state. Packets from each queue are then delivered fairly, according to a DRR++ algorithm which minimises latency for "sparse" flows. CAKE uses a set-associative hashing algorithm to minimise flow collisions.

These keywords specify whether fairness based on source address, destination address, individual flows, or any combination of those is desired.

flowblind
Disables flow isolation; all traffic passes through a single queue for each tin.

srchost
Flows are defined only by source address. Could be useful on the egress path of an ISP backhaul.

dsthost
Flows are defined only by destination address. Could be useful on the ingress path of an ISP backhaul.

hosts
Flows are defined by source-destination host pairs. This is host isolation, rather than flow isolation.

flows
Flows are defined by the entire 5-tuple of source address, destination address, transport protocol, source port and destination port. This is the type of flow isolation performed by SFQ and fq_codel.

dual-srchost
Flows are defined by the 5-tuple, and fairness is applied first over source addresses, then over individual flows. Good for use on egress traffic from a LAN to the internet, where it'll prevent any one LAN

host from monopolising the uplink, regardless of the number of flows they use.

dual-dsthost
Flows are defined by the 5-tuple, and fairness is applied first over destination addresses, then over individual flows. Good for use on ingress traffic to a LAN from the internet, where it'll prevent any one LAN host from monopolising the downlink, regardless of the number of flows they use.

triple-isolate (default)
Flows are defined by the 5-tuple, and fairness is applied over source *and* destination addresses intelligently (ie. not merely by host-pairs), and also over individual flows. Use this if you're not certain whether to use dual-srchost or dual-dsthost; it'll do both jobs at once, preventing any one host on *either* side of the link from monopolising it with a large number of flows.

nat
Instructs Cake to perform a NAT lookup before applying flow-isolation rules, to determine the true addresses and port numbers of the packet, to improve fairness between hosts "inside" the NAT. This has no practical effect in "flowblind" or "flows" modes, or if NAT is performed on a different host.

nonat (default)
Cake will not perform a NAT lookup. Flow isolation will be performed using the addresses and port numbers directly visible to the interface Cake is attached to.

PRIORITY QUEUE PARAMETERS

CAKE can divide traffic into "tins" based on the Diffserv field. Each tin has its own independent set of flow-isolation queues, and is serviced based on a WRR algorithm. To avoid perverse Diffserv marking incentives, tin weights have a "priority sharing" value when bandwidth used by that tin is below a threshold, and a lower "bandwidth sharing" value when above. Bandwidth is compared against the threshold using the same algorithm as the deficit-mode shaper.

Detailed customisation of tin parameters is not provided. The following presets perform all necessary tuning, relative to the current shaper bandwidth and RTT settings.

besteffort
Disables priority queuing by placing all traffic in one tin.

precedence
Enables legacy interpretation of TOS "Precedence" field. Use of this preset on the modern Internet is firmly discouraged.

diffserv4
Provides a general-purpose Diffserv implementation with four tins:
Bulk (CS1), 6.25% threshold, generally low priority.
Best Effort (general), 100% threshold.
Video (AF4x, AF3x, CS3, AF2x, CS2, TOS4, TOS1), 50% threshold.
Voice (CS7, CS6, EF, VA, CS5, CS4), 25% threshold.

diffserv3 (default)
Provides a simple, general-purpose Diffserv implementation with three tins:
Bulk (CS1), 6.25% threshold, generally low priority.
Best Effort (general), 100% threshold.
Voice (CS7, CS6, EF, VA, TOS4), 25% threshold, reduced Codel interval.

fwmark MASK
This options turns on fwmark-based overriding of CAKE's tin selection. If set, the option specifies a bitmask that will be applied to the fwmark associated with each packet. If the result of this masking is non-zero, the result will be right-shifted by the number of least-significant unset bits in the mask value, and the result will be used as a the tin number for that packet. This can be used to set policies in a firewall script that will override CAKE's built-in tin selection.

OTHER PARAMETERS

memlimit LIMIT

Limit the memory consumed by Cake to LIMIT bytes. Note that this does not translate directly to queue size (so do not size this based on bandwidth delay product considerations, but rather on worst case acceptable memory consumption), as there is some overhead in the data structures containing the packets, especially for small packets.

By default, the limit is calculated based on the bandwidth and RTT settings.

wash

Traffic entering your diffserv domain is frequently mis-marked in transit from the perspective of your network, and traffic exiting yours may be mis-marked from the perspective of the transiting provider.

Apply the wash option to clear all extra diffserv (but not ECN bits), after priority queuing has taken place.

If you are shaping inbound, and cannot trust the diffserv markings (as is the case for Comcast Cable, among others), it is best to use a single queue "besteffort" mode with wash.

split-gso

This option controls whether CAKE will split General Segmentation Offload (GSO) super-packets into their on-the-wire components and dequeue them individually.

Super-packets are created by the networking stack to improve efficiency. However, because they are larger they take longer to dequeue, which translates to higher latency for competing flows, especially at lower bandwidths. CAKE defaults to splitting GSO packets to achieve the lowest possible latency. At link speeds higher than 10 Gbps, setting the no-split-gso parameter can increase the maximum achievable throughput by retaining the full GSO packets.

OVERRIDING CLASSIFICATION WITH TC FILTERS

CAKE supports overriding of its internal classification of packets through the tc filter mechanism. Packets can be assigned to different priority tins by setting the **priority** field on the skb, and the flow hashing can be overridden by setting the **classid** parameter.

Tin override

To assign a priority tin, the major number of the priority field needs to match the qdisc handle of the cake instance; if it does, the minor number will be interpreted as the tin index. For example, to classify all ICMP packets as 'bulk', the following filter can be used:

```
# tc qdisc replace dev eth0 handle 1: root cake diffserv3
# tc filter add dev eth0 parent 1: protocol ip prio 1 \
  u32 match icmp type 0 0 action skbedit priority 1:1
```

Flow hash override

To override flow hashing, the classid can be set. CAKE will interpret the major number of the classid as the host hash used in host isolation mode, and the minor number as the flow hash used for flow-based queueing. One or both of those can be set, and will be used if the relevant flow isolation parameter is set (i.e., the major number will be ignored if CAKE is not configured in hosts mode, and the minor number will be ignored if CAKE is not configured in flows mode).

This example will assign all ICMP packets to the first queue:

```
# tc qdisc replace dev eth0 handle 1: root cake
# tc filter add dev eth0 parent 1: protocol ip prio 1 \
  u32 match icmp type 0 0 classid 0:1
```

If only one of the host and flow overrides is set, CAKE will compute the other hash from the packet as normal. Note, however, that the host isolation mode works by assigning a host ID to the flow queue; so if overriding both host and flow, the same flow cannot have more than one host assigned. In addition, it is not possible to assign different source and destination host IDs through the override mechanism; if a host ID is assigned, it will be used as both source and destination host.

EXAMPLES

```
# tc qdisc delete root dev eth0
# tc qdisc add root dev eth0 cake bandwidth 100Mbit ethernet
# tc -s qdisc show dev eth0
qdisc cake 1: root refcnt 2 bandwidth 100Mbit diffserv3 triple-isolate rtt 100.0ms noatm overhead 38 mpu
84
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 memory used: 0b of 5000000b
 capacity estimate: 100Mbit
 min/max network layer size:        65535 /       0
 min/max overhead-adjusted size:    65535 /       0
 average network hdr offset:            0

                   Bulk  Best Effort        Voice
  thresh       6250Kbit      100Mbit       25Mbit
  target          5.0ms        5.0ms        5.0ms
  interval      100.0ms      100.0ms      100.0ms
  pk_delay          0us          0us          0us
  av_delay          0us          0us          0us
  sp_delay          0us          0us          0us
  pkts                0            0            0
  bytes               0            0            0
  way_inds            0            0            0
  way_miss            0            0            0
  way_cols            0            0            0
  drops               0            0            0
  marks               0            0            0
  ack_drop            0            0            0
  sp_flows            0            0            0
  bk_flows            0            0            0
  un_flows            0            0            0
  max_len             0            0            0
  quantum           300         1514          762
```

After some use:

```
# tc -s qdisc show dev eth0

qdisc cake 1: root refcnt 2 bandwidth 100Mbit diffserv3 triple-isolate rtt 100.0ms noatm overhead 38 mpu
84
```

Sent 44709231 bytes 31931 pkt (dropped 45, overlimits 93782 requeues 0)
backlog 33308b 22p requeues 0
memory used: 292352b of 5000000b
capacity estimate: 100Mbit
min/max network layer size: 28 / 1500
min/max overhead-adjusted size: 84 / 1538
average network hdr offset: 14

	Bulk	Best Effort	Voice
thresh	6250Kbit	100Mbit	25Mbit
target	5.0ms	5.0ms	5.0ms
interval	100.0ms	100.0ms	100.0ms
pk_delay	8.7ms	6.9ms	5.0ms
av_delay	4.9ms	5.3ms	3.8ms
sp_delay	727us	1.4ms	511us
pkts	2590	21271	8137
bytes	3081804	30302659	11426206
way_inds	0	46	0
way_miss	3	17	4
way_cols	0	0	0
drops	20	15	10
marks	0	0	0
ack_drop	0	0	0
sp_flows	2	4	1
bk_flows	1	2	1
un_flows	0	0	0
max_len	1514	1514	1514
quantum	300	1514	762

SEE ALSO

tc(8), **tc-codel**(8), **tc-fq_codel**(8), **tc-htb**(8)

AUTHORS

Cake's principal author is Jonathan Morton, with contributions from Tony Ambardar, Kevin Darbyshire-Bryant, Toke Høiland-Jørgensen, Sebastian Moeller, Ryan Mounce, Dean Scarff, Nils Andreas Svee, and Dave Täht.

This manual page was written by Loganaden Velvindron. Please report corrections to the Linux Networking mailing list <netdev@vger.kernel.org>.