

NAME

tcp_bbr - TCP Bottleneck Bandwidth and Round-Trip Time Algorithm

SYNOPSIS

To use this TCP stack you have to place the following line in your kernel configuration file:

options TCPHPTS

To load the driver as a module at boot time, place the following line in loader.conf(5):

```
tcp_bbr_load="YES"
```

To enable the TCP stack you must place the following line in the sysctl.conf(5):

```
net.inet.tcp.functions_default=bbr
```

DESCRIPTION

Bottleneck bandwidth and round-trip time (BBR) is a congestion control algorithm which seeks high throughput with a small queue by probing BW and RTT. It is a round-up redesign of congestion control, which is not loss-based, delay-based, ECN-based or AIMD-based.

The core design of BBR is about creating a model graph of the network path by estimating the maximum BW and minimum RTT on each ACK.

MIB Variables

The algorithm exposes the following scopes in the *net.inet.tcp.bbr* branch of the sysctl(3) MIB:

<i>cwnd</i>	Cwnd controls, for example "target cwnd rtt measurement" and "BBR initial window".
<i>measure</i>	Measurement controls.
<i>pacing</i>	Connection pacing controls.
<i>policer</i>	Policer controls, for example "false detection threshold" and "loss threshold".
<i>probertt</i>	Probe RTT controls.
<i>startup</i>	Startup controls.

states State controls.

timeout Time out controls.

Besides the variables within the above scopes the following variables are also exposed in the *net.inet.tcp.bbr* branch:

clrlost Clear lost counters.

software_pacing Total number of software paced flows.

hdwr_pacing Total number of hardware paced flows.

enob_no_hdwr_pacing
Total number of enobufs for non-hardware paced flows.

enob_hdwr_pacing
Total number of enobufs for hardware paced flows.

rtt_tlp_thresh What divisor for TLP rtt/retran will be added (1=rtt, 2=1/2 rtt etc).

reorder_fade Does reorder detection fade, if so how many ms (0 means never).

reorder_thresh What factor for rack will be added when seeing reordering (shift right).

bb_verbose Should BBR black box logging be verbose.

sblklimit When do we start ignoring small sack blocks.

resend_use_tso Can resends use TSO?

data_after_close Do we hold off sending a RST until all pending data is ack'd.

kill_paceout When we hit this many errors in a row, kill the session?

error_paceout When we hit an error what is the min to pace out in usec's?

cheat_rxt Do we burst 1ms between sends on retransmissions (like rack)?

minrto Minimum RTO in ms.

SEE ALSO

cc_chd(4), cc_cubic(4), cc_hd(4), cc_htcp(4), cc_newreno(4), cc_vegas(4), h_ertt(4), mod_cc(4), tcp(4), tcp_rack(4), mod_cc(9)

Neal Cardwell, Yuchung Cheng, Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson, "BBR: Congestion-Based Congestion Control", *ACM Queue*, Vol. 14, September / October 2016.

Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle, "Towards a Deeper Understanding of TCP BBR Congestion Control", *IFIP Networking 2018*, <http://www.net.in.tum.de/fileadmin/bibtex/publications/papers/IFIP-Networking-2018-TCP-BBR.pdf>, May 2018.

HISTORY

The **tcp_bbr** congestion control module first appeared in FreeBSD 13.0.

AUTHORS

The **tcp_bbr** congestion control module was written by Randall Stewart <rrs@FreeBSD.org> and sponsored by Netflix, Inc. This manual page was written by Gordon Bergling <gbe@FreeBSD.org>.