NAME

cc_newreno - NewReno Congestion Control Algorithm

SYNOPSIS

#include <netinet/cc/cc_newreno.h>

DESCRIPTION

Details about the algorithm can be found in RFC5681.

Socket Options

The **cc_newreno** module supports a number of socket options under TCP_CCALGOOPT (refer to tcp(4) and mod_cc(9) for details) which can be set with setsockopt(2) and tested with getsockopt(2). The **cc_newreno** socket options use this structure defined in <sys/netinet/cc/cc_newreno.h>:

```
struct cc_newreno_opts {
        int name;
        uint32_t val;
}
```

CC_NEWRENO_BETA Multiplicative window decrease factor, specified as a percentage, applied to the congestion window in response to a congestion signal per: cwnd = (cwnd * CC_NEWRENO_BETA) / 100. Default is 50.

CC_NEWRENO_BETA_ECN Multiplicative window decrease factor, specified as a percentage, applied to the congestion window in response to an ECN congestion signal when *net.inet.tcp.cc.abe=1* per: cwnd = (cwnd * CC_NEWRENO_BETA_ECN) / 100. Default is 80.

Note that currently the only way to enable hystart++ is to enable it via socket option. When enabling it a value of 1 will enable precise internet-draft (version 4) behavior (subject to any MIB variable settings), other setting (2 and 3) are experimental.

Note that hystart++ requires the TCP stack be able to call to the congestion controller with both the *newround* function as well as the *rttsample* function. Currently the only TCP stacks that provide this feedback to the congestion controller is rack.

MIB Variables

The algorithm exposes these variables in the *net.inet.tcp.cc.newreno* branch of the sysctl(3) MIB:

beta Multiplicative window decrease factor, specified as a percentage, applied to the congestion window in response to a congestion signal per: cwnd = (cwnd * beta) / 100. Default is 50.

beta_ecn Multiplicative window decrease factor, specified as a percentage, applied to the congestion window in response to an ECN congestion signal when *net.inet.tcp.cc.abe=1* per: cwnd = (cwnd * beta_ecn) / 100. Default is 80.

SEE ALSO

cc_cdg(4), cc_chd(4), cc_cubic(4), cc_dctcp(4), cc_hd(4), cc_htcp(4), cc_vegas(4), mod_cc(4), tcp(4), mod_cc(9)

Mark Allman, Vern Paxson, and Ethan Blanton, *TCP Congestion Control*, RFC 5681.

Naeem Khademi, Michael Welzl, Grenville Armitage, and Gorry Fairhurst, *TCP Alternative Backoff with ECN (ABE)*, RFC 8511.

ACKNOWLEDGEMENTS

Development and testing of this software were made possible in part by grants from the FreeBSD Foundation and Cisco University Research Program Fund at Community Foundation Silicon Valley.

HISTORY

The **cc_newreno** congestion control algorithm first appeared in its modular form in FreeBSD 9.0.

This was the default congestion control algorithm in FreeBSD before version FreeBSD 14.0, after which cc_cubic(4) replaced it.

The module was first released in 2007 by James Healy and Lawrence Stewart whilst working on the NewTCP research project at Swinburne University of Technology's Centre for Advanced Internet Architectures, Melbourne, Australia, which was made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley. More details are available at:

http://caia.swin.edu.au/urp/newtcp/

AUTHORS

The **cc_newreno** congestion control module was written by James Healy <*jimmy@deefa.com*>, Lawrence Stewart <*lstewart@FreeBSD.org*> and David Hayes <*david.hayes@ieee.org*>.

Support for TCP ABE was added by Tom Jones <*tj@enoti.me*>.

This manual page was written by Lawrence Stewart <*lstewart@FreeBSD.org*>.