NAME

mod_cc - Modular congestion control

DESCRIPTION

The modular congestion control framework allows the TCP implementation to dynamically change the congestion control algorithm used by new and existing connections. Algorithms are identified by a unique ascii(7) name. Algorithm modules can be compiled into the kernel or loaded as kernel modules using the kld(4) facility.

The default algorithm is CUBIC, and all connections use the default unless explicitly overridden using the TCP_CONGESTION socket option (see tcp(4) for details). The default can be changed using a sysctl(3) MIB variable detailed in the *MIB Variables* section below.

Algorithm specific parameters can be set or queried using the TCP_CCALGOOPT socket option (see tcp(4) for details). Callers must pass a pointer to an algorithm specific data, and specify its size.

Unloading a congestion control module will fail if it is used as a default by any Vnet. When unloading a module, the Vnet default is used to switch a connection to an alternate congestion control. Note that the new congestion control module may fail to initialize its internal memory, if so it will fail the module unload. If this occurs often times retrying the unload will succeed since the temporary memory shortage as the new CC module malloc's memory, that prevented the switch is often transient.

MIB Variables

The framework exposes the following variables in the *net.inet.tcp.cc* branch of the sysctl(3) MIB:

available Read-only list of currently available congestion control algorithms by name.

algorithm Returns the current default congestion control algorithm when read, and changes the default when set. When attempting to change the default algorithm, this variable should be set to one of the names listed by the *net.inet.tcp.cc.available* MIB variable.

abe Enable support for RFC 8511, which alters the window decrease factor applied to the congestion window in response to an ECN congestion signal. Refer to individual congestion control man pages to determine if they implement support for ABE and for configuration details.

abe_frlossreduce If non-zero, apply standard beta instead of ABE-beta during ECN-signalled congestion recovery episodes if loss also needs to be repaired.

hystartplusplus.bblogs This boolean controls if black box logging will be done for hystart++ events. If set to zero (the default) no logging is performed. If set to one then black box logs will be generated on all hystart++ events.

hystartplusplus.css_rounds This value controls the number of rounds that CSS runs for. The default value matches the current internet-draft of 5.

hystartplusplus.css_growth_div This value controls the divisor applied to slowstart during CSS. The default value matches the current internet-draft of 4.

hystartplusplus.n_rttsamples This value controls how many rtt samples must be collected in each round for hystart++ to be active. The default value matches the current internet-draft of 8.

hystartplusplus.maxrtt_thresh This value controls the maximum rtt variance clamp when considering if CSS is needed. The default value matches the current internet-draft of 16000 (in microseconds). For further explanation please see the internet-draft.

hystartplusplus.minrtt_thresh This value controls the minimum rtt variance clamp when considering if CSS is needed. The default value matches the current internet-draft of 4000 (in microseconds). For further explanation please see the internet-draft.

Each congestion control module may also expose other MIB variables to control their behaviour. Note that both NewReno and CUBIC now support Hystart++ based on the version 3 of the internet-draft.

Kernel Configuration

All of the available congestion control modules may also be loaded via kernel configutation options. A kernel configuration is required to have at least one congestion control algorithm built into it via kernel option and a system default specified. Compilation of the kernel will fail if these two conditions are not met.

Kernel Configuration Options

The framework exposes the following kernel configuration options.

CC_NEWRENO This directive loads the NewReno congestion control algorithm.

CC_CUBIC This directive loads the CUBIC congestion control algorithm and is included in GENERIC by default.

CC_VEGAS This directive loads the vegas congestion control algorithm, note that this algorithm also requires the TCP_HHOOK option as well.

CC_CDG This directive loads the cdg congestion control algorithm, note that this algorithm also requires the TCP_HHOOK option as well.

CC_DCTCP This directive loads the dctcp congestion control algorithm.

CC_HD This directive loads the hd congestion control algorithm, note that this algorithm also requires the TCP_HHOOK option as well.

CC_CHD This directive loads the chd congestion control algorithm, note that this algorithm also requires the TCP_HHOOK option as well.

CC_HTCP This directive loads the htcp congestion control algorithm.

CC_DEFAULT This directive specifies the string that represents the name of the system default algorithm, the GENERIC kernel defaults this to CUBIC.

SEE ALSO

cc_cdg(4), cc_chd(4), cc_cubic(4), cc_dctcp(4), cc_hd(4), cc_htcp(4), cc_newreno(4), cc_vegas(4), tcp(4), config(5), config(8), mod_cc(9)

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 **mod_cc** modular congestion control framework first appeared in FreeBSD 9.0.

The framework 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 **mod_cc** facility was written by Lawrence Stewart <*lstewart@FreeBSD.org*>, James Healy

<*jimmy@deefa.com*> and David Hayes <*david.hayes@ieee.org*>.

This manual page was written by David Hayes <*david.hayes@ieee.org*> and Lawrence Stewart <*lstewart@FreeBSD.org*>.