Capping of cwnd growth in FreeBSD's NewReno over high BDP paths

Alana Huebner
Centre for Advanced Internet Architectures, Technical Report 080829A
Swinburne University of Technology
Melbourne, Australia
4087127@student.swin.edu.au

***Abstract*—This report experimentally validates that integer division in FreeBSD's NewReno implementation results in $cwnd$ being capped on large BDP paths.**

***Index Terms*—NewReno, FreeBSD**

I. Introduction

This report documents an experimental validation of a corner case involving integer maths in FreeBSD's current NewReno implementation. Readers are encouraged to read CAIA Technical Report 080818A [1] for details of the testbed and measurement tools used in this experiment.

Under FreeBSD the growth of TCP's congestion window ($cwnd$) during congestion avoidance is determined by Equation 1.

$$cwnd+ = \frac{SMSS \times SMSS}{cwnd} \quad (1)$$

As a result of integer division, if $cwnd$ is larger than $SMSS^2$ the right hand side of the expression equals zero and $cwnd$ will cease to grow. This report will show that $cwnd$ growth in congestion avoidance mode is capped when $cwnd$ becomes greater than or equal to $SMSS^2$ bytes.

RFC2581 [2] identifies this issue and suggests that implementations increase $cwnd$ by 1 byte when $cwnd > SMSS^2$. The FreeBSD implementation does not take this approach.

Section 2 will describe the experimental method.

Section 3 will present the results and show how $cwnd$ growth is capped under FreeBSD.

II. Experimental method

The tests were conducted over the NewTCP [3] testbed as detailed in [1]. A test involved a single NewReno TCP flow between two FreeBSD 7.0 hosts. A router between the hosts simulated a high $bandwidth \times delay$ product (BDP) link with dummynet [4].

Bandwidth	100Mb
Delay (ms)	100, 150, 175, 200, 210, 225, 250
Queue Size	1 BDP

TABLE I
Test variables

The $SMSS$ was FreeBSD's default value of 1448 bytes. A range of link characteristics were tested, shown in Table I. These characteristics were chosen so that the BDP of the link was large enough to allow $cwnd$ to grow beyond $SMSS^2$.

Each test lasted three minutes and was performed twice for consistency. During the tests the TCP sender's $cwnd$ was recorded with SIFTR [5].

The window used by the TCP sender is the minimum of $cwnd$, the receiver's window and the send buffer size. As the experiment dealt with large BDPs the default size of the send buffer was inadequate. It was necessary to configure the send buffer to be larger than the BDP so that $cwnd$ is the utilised window value. Under FreeBSD, $cwnd$ is updated during congestion avoidance even if it is not the minimum value. When this occurs $cwnd$ is not representative of the TCP window and it will continue to grow unbounded while packet loss does not occur.

III. Results

Figure 1 shows $cwnd$ for the duration of a three minute test. After slow start and the first packet loss occurs TCP enters congestion avoidance mode. Starting from $sshresh$, half the maximum $cwnd$, $cwnd$ increases linearly according to Equation 1 until it reaches 1448 packets near $t = 100$ seconds. $cwnd$ growth stops as a result of integer division at 1448 packets, exactly $SMSS^2$ bytes.

Figure 2 shows the values at which $cwnd$ was capped

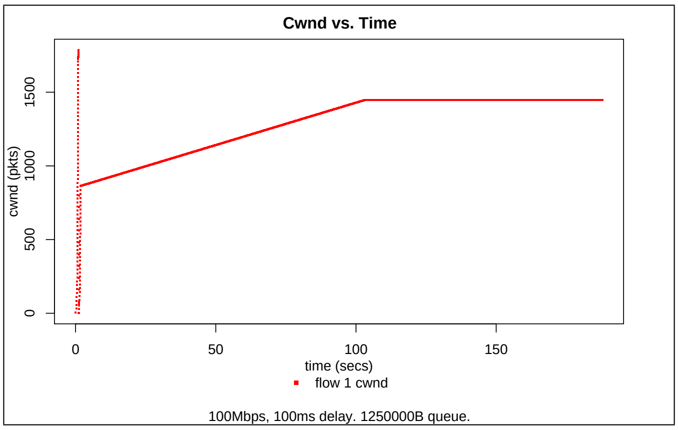

Fig. 1. NewReno Congestion Window

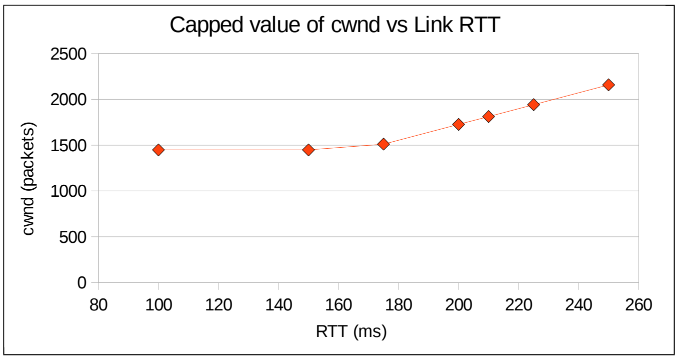

Fig. 2. Capped value of $cwnd$ as BDP increases

for all the links tested. From the 175ms test $cwnd$ is capped at increasingly large values.

Figure 3 contains a plot of $cwnd$ for the 175ms test. At this larger BDP $cwnd$ is capped as soon as it is set to $ssthresh$ after the first packet loss. In this case $ssthresh$ is greater than $SMSS^2$ and the integer division problem is encountered on the first congestion avoidance update of $cwnd$.

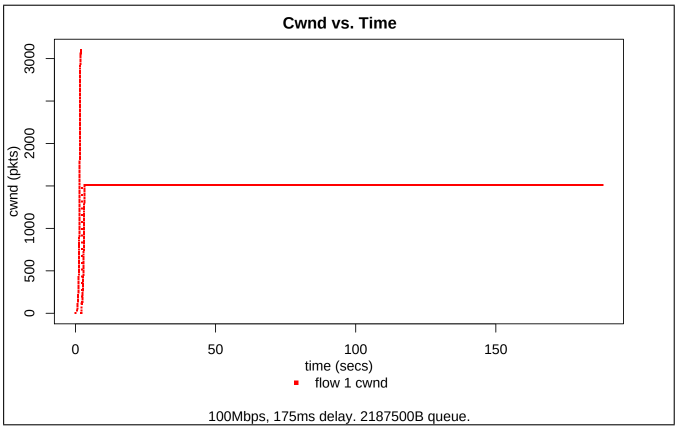

Fig. 3. NewReno Congestion Window

IV. Conclusion

This report describes experimental evaluation of a corner case in FreeBSD's NewReno implementation. Integer division in the congestion avoidance algorithm causes $cwnd$ growth to cease when $cwnd >= SMSS^2$.

When $ssthresh < SMSS^2$ $cwnd$ is capped at $SMSS^2$ bytes or $SMSS$ packets. However if $ssthresh > SMSS^2$ $cwnd$ is capped at $ssthresh$ bytes.

Acknowledgements

This report has been made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley.

References

[1] A. Huebner, "Experimental Evaluation of Latency Induced in Real-Time Traffic by TCP Congestion Control Algorithms," CAIA, Tech. Rep. 080818A, August 2008. [Online]. Available: http://caia.swin.edu.au/reports/080818A/CAIA-TR-080818A.pdf

[2] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control ," RFC 2581 (Proposed Standard), Apr. 1999, updated by RFC 3390. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

[3] "The NewTCP Project," August 2008, Accessed 8 Aug 2008. [Online]. Available: http://caia.swin.edu.au/urp/newtcp

[4] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[5] L. Stewart, J. Healy, "Characterising the Behaviour and Performance of SIFTR v1.1.0," CAIA, Tech. Rep. 070824A, August 2007. [Online]. Available: http://caia.swin.edu.au/reports/070824A/CAIA-TR-070824A.pdf