Smooth TCP

Elvis Melo Vieira[1] and Michael Bauer

Department of Computer Science
The University of Western Ontario
London Ontario, N6A5B7
{elvis, bauer}@csd.uwo.ca

Abstract. The TCP's congestion control mechanism presents two problems. The first is the delay that occurs between the time a congestion event is detected and the time the sender takes to recognize it and react to prevent further congestion. The second is related to the use of packet drops as the main congestion indicator. This is inherently imprecise since a packet loss is also detected as a packet drop due to a full queue. In order to deal with these two problems and add some other desirable characteristics to TCP, we define a new property that a congestion control mechanism should have and propose a new mechanism satisfying this requirement defined generically as the Smoothness Property.

1 Introduction

A key element in the success of the Transmission Control Protocol (TCP) is due to its congestion control mechanism developed after the first case of congestion reported in the Internet [1]. Although two very simple algorithms compose it (slow-start/congestion avoidance and fast retransmission/fast recovery), previous work [3,4,5,6] has demonstrated the existence of design problems related to the way congestion is signaled to the TCP sender, which is done through packet drops.

Indeed, packets drops should indicate to the TCP sender that there is a router queue, somewhere between the sender and the receiver, which is full. Therefore, the sender should decrease its sending rate in order to return the queue to a normal state. But this is not always true. A packet can be dropped for reasons other than a full queue, such as an electrical noise in the communication line. If such drops happen, there is no reason to decrease the sending rate. Consequently, the sender is penalized unnecessarily because of this *signal uncertainty*. Additionally, using packet drops as the main congestion indication creates another problem. Once a packet is dropped in a queue, the sender does not know immediately, but it will take a time longer than one round trip time (RTT). This ***temporal gap*** makes the TCP halve the window size, a hard penalty in high-speed networks.

Our work suggests that these two problems can be addressed by introducing the concept of smoothness. Briefly, a congestion control mechanism, which has this

[1] Sponsorship from CNPQ-Brazil

smoothness characteristic, should react to congestion signals in a less "harsh" way than just halving its window size, as does Standard TCP. This is because if the congestion control mechanism is "smooth", then there is almost no signal uncertainty and the temporal gap is minimized. Thus a TCP with smoothness should be capable of avoiding certain unnecessary throughput decreases and RTT variations, creating a better performance in some situations.

The remainder of this paper is organized as follows. Section 2 discusses the motivation behind this research and some related work. Section 3 introduces a definition of smoothness, which is given in the context of four properties. Based on these properties, a set of functions that can be used to derive smooth congestion control mechanisms is given in Section 4 and introduces Smooth TCP. Section 5 reports on some simulation results of Smooth TCP using Network Simulator (www.isi.edu/nsnam/ns). The final section provides a summary and conclusions.

2 Motivation and Related Work

The work in [2] suggests two motives for changing the behavior of Standard TCP. The first is that the threat to the stability of the Internet originates not from flows having alternative congestion controls, but from those not having any congestion control at all, such as large-scale multicasting flows or some real-time traffic. The second is that the stability of the Internet does not require that flows decrease their sending rate by halving their window sizes. In particular, to avoid congestion collapse it is only necessary to use a lower sending rate when there is a high loss rate.

Two key issues related to the TCP performance can be identified under links subjected to errors: its inability to separate the packet losses due to congestion from those due to other problems and its reliance on the timer to recover from a failed retransmission cycle [3]. One of the effects of these issues is the spurious retransmission timeouts, which reduce the performance of TCP as they can start unnecessary retransmissions of segments. These problems have been noted and addressed by others:

1. Eifel Algorithm: In [3], Ludwig and Katz proposed an algorithm, called Eifel, to deal with spurious retransmission timeouts. It proposes an alteration in TCP that uses either timestamps sent in TCP segments, or two of the reserved bits in the TCP header to determine whether there was any packet loss due to some error when an ACK arrived in the sender.
2. Westwood TCP: In [4] and [5] Westwood TCP is described. Instead of dropping the window size by half of the value it was before when a packet loss, it proposed a linear decreasing of the congestion window depending on the bandwidth measured on each ACK arrival. This results in a performance of up to four times that of the Standard TCP when spurious errors occur.
3. F-RTO : The F-RTO algorithm is described in [6]. F-RTO only affects the TCP sender when there is a retransmission timeout. Otherwise it behaves as Standard TCP.

In [7], Floyd discusses the huge impact of halving the TCP window in high-speed networks. A TCP-friendly mechanism is described in [2] to control unicast traffic using equation-based congestion control. This mechanism avoids reducing the sending rate to react to a single packet drop. Instead, the sender adjusts its sending rate according to the measured rate of loss events in a single round-trip time. The Family of Binomial Congestion Control Algorithms [8] also look at the generalization of congestion control; the work in [8] was the basis for the idea of congestion control functions in the present work.

3 Smoothness Definition

Although in [9] *smoothness* is a characteristic defined by four properties, on this work we are redefining that concept throughout five properties: a smooth curve, vertical smoothness, horizontal smoothness, proactive smoothness and precision.

1. **Smooth Curve Property**: An important characteristic of the TCP's standard congestion control algorithm is the action of halving its congestion window in face of a packet drop. This sudden decrease marks a discontinuation point on its evolution curve. The effect of discontinuation points is a fast and large increase in RRT. As a result, the first property required for a smooth congestion control is that its curve should not have discontinuation points or, at least they should have as few as possible. This requirement is called the *smooth curve property*.
2. **Vertical Smoothness Property**: Formally, the congestion control uses a function f(u) to translate events u into window sizes. When f(u) is a function of a variable where bursts with very large amplitude can occur, like RTT, this cannot be done linearly, since variations of RTT near 0 are more important or significant than variations among large values. For this reason, a hyperbolic tangent function is proposed to introduce vertical smoothness, making sure certain signals remains within a specified range.
3. **Horizontal Smoothness Property:** Sometimes, for the congestion control it is important to know the frequency of an event (or its rate) instead of its amplitude. Consequently in f(u), u should mean the rate of those events. In our work u can be the smooth average rate as described in [10][2]. A congestion control function smoothing horizontal bursts will be said to have the *horizontal smoothness property***.**
4. **Proactive Control Property**: Suppose a packet is dropped at a router because its queue is full. In order for the sender to decrease its sending rate, the receiver sends duplicate acknowledgment packets, which arrive at the sender, approximately, 1 RTT (more precisely, 1 RTT and 3 duplicate acknowledgment arrivals) later. This relatively long time, which we refer to as a *temporal gap,* is one of the principal reasons TCP reacts by halving the window [11]. We can do better if other metrics are used also to signal congestion. This work proposes the use of the following metrics: RTT ([12] details the relation between the router queue size and RTT) and ICMP-Source Quench (ICMP-SQ) Messages. Explicit Congestion Notification (ECN) [13] can be used instead of ICMP-SQ, but the latter has a smaller temporal gap since the messages are to sent directly to the sender (in the opposite direction).

[2] The smooth average of n terms is $\bar{u}(u_n)=(1-\alpha)*\ \bar{u}(u_{n-1})+\alpha*u_n$ where α is ⅛ or ¼.

5. **Precision Property**: Another problem is that a packet drop is not always the best way to signal congestion. A packet drop does not always means that the packet was discarded as a result of congestion in a router queue. For example, a packet could be dropped because of a CRC error. For that reason, we say that there is a *signal uncertainty*. Algorithms like Eifel[3] and F-RTO [6] eliminate some of the uncertainties, but not completely.

The possibility of using ICMP-SQ messages in a proactive manner also enhances the precision of the congestion signalization. A router only sends them when the threshold of a RED queue is reached and so, for example, ICMP-SQ messages are not sent or ECN bits marked because of CRC errors. On the other hand, RTT increases could be provoked by events other than congestion, like a route table update. Therefore, RTT would not be as precise as ICMP-SQ or ECN.

4 Smooth Congestion Control Algorithms

For our notion of smoothness, we would like to be able to make "appropriate" adjustments to the congestion control window. To do this, we think of the congestion window function *w* as being defined as a function of some set of variables $p_1,..,p_i,..,p_n$ describing the network state and congestion conditions. So, we are interested in changes in *w*, that is, an increment or decrement dw to the existing window size depending on changes in $p_1,..,p_i,..,p_n$. First, let us define D:

$$D = \frac{\partial w}{\partial p_1} dp_1 + ... + \frac{\partial w}{\partial p_i} dp_i ... + \frac{\partial w}{\partial p_n} dp_n \quad \textbf{(1)}$$

As per our discussion of "smoothness" previous section, we assume that each partial derivative is of the form of f(u) (hyperbolic tangent), namely:

$$\frac{\partial w}{\partial p_i} = A_{p_i} + C_{p_i} \tanh(B_{p_i}(p_i + M_{p_i})) \quad \textbf{(2)}$$

The sigmoid function tanh is proposed because it has the property of "shrinking" values of a metric into the opened interval (-1,1), as required by the vertical smoothness property. $A_{p_i}, B_{p_i}, C_{p_i}$ and M_{p_i} above are coefficients to be determined. In order to enforce all connections sharing approximately the same bandwidth, the following idea, inspired from Standard TCP gives the final increment dw in w:

$$\begin{aligned} dw &= (D + \kappa)/w \quad \text{if } D \geq 0 \text{ or} \\ dw &= (D + \kappa) \bullet w \quad \text{if } D < 0 \end{aligned} \quad \textbf{(3)}$$

Consequently, having n connections sharing different window sizes, after a certain amount of dp_i variations, they will share approximately the same window size. This guarantees that those connections having the largest window sizes suffer the smallest window increments. On the other hand, Eq. (3) multiplies the decrement (D < 0) by the current window. This guarantees that connections having the smallest window sizes suffer the smallest decrements. At last, κ is called the fair factor. It is used to adjusted the fairness of Smooth when D=0. Eq. (3) defines a family of functions.

When such a family includes only smooth functions, we call it a *Family of Smooth Congestion Control Algorithms* (or briefly, Smooth Algorithms).

4.1 Smooth TCP

We can then introduce the form of the functions for the partial derivatives that define the particular instance of Smooth Algorithms called *Smooth TCP*:

$$D = \frac{\delta w}{\delta j}dj + \frac{\delta w}{\delta x}dx + \frac{\delta w}{\delta f}df + \frac{\delta w}{\delta e}de + \frac{\delta w}{\delta q}dq \tag{4}$$

where *j* is the variation in RTT, x is the variation in the number of timeout retransmits, *f* is variation in the number of fast retransmits, e is the variation in the number of ECN acknowledgments and *q* is the variation in the number of ICMP-SQ messages. Then, the individual partial derivatives are defined to be:

$$\frac{\partial w}{\partial j} = A_j + C_j \tanh(B_j (j + M_j)) \tag{5}$$

$$\frac{\partial w}{\partial x} = A_x + C_x \tanh(B_x (x + M_x))$$

$$\frac{\partial w}{\partial f} = A_f + C_f \tanh(B_f (f + M_f))$$

$$\frac{\partial w}{\partial e} = A_e + C_e \tanh(B_e (e + M_e))$$

$$\frac{\partial w}{\partial q} = A_q + C_q \tanh(B_q (q + M_q))$$

4.2 A New Slow-Start Mechanism

The work in [10] discusses extensively the necessity of a slowly starting transfer for a TCP connection. This phase avoids a huge burst of packets following the initial moment after a TCP connection is opened. The standard Slow-Start procedure can be characterized by a very slow grow after a certain time characterized by a increment of dw=1/w. This means dw is small enough after a certain time, although different from 0. However we would like the slow-start of Smooth TCP have the characteristic, but with parameterized point of convergence. After that point, Eq. (3) can be responsible for updating the window.

A requirement imposed to this new mechanism is an adjustable growth rate since different rates can be configured to adapt to each environment. Moreover, this new mechanism should include some features of the standard TCP: an exponential and very slow growth given and window updates only at each acknowledgment arrival. Using the number of acknowledgements (*a*) arriving at the sender and A_a=-C_a in Eq. (2), the following function results:

$$dw = \left\lfloor A_a \quad A_a \tanh\left(B_a\left(a + M_a\right)\right)\right\rfloor / w \; da \qquad \textbf{(6)}$$

The new resulting equation converges to a finite value and does not have any discontinuation point like standard TCP [10]. Instead, it is continuous over all its extension. Solving the new differential equation, the non-recursive equation is obtained:

$$w = \sqrt{2\left(A_a - \frac{A_a}{B_a}\ln\left(\cosh\left(a \, . \, B_a\right)\right) + T\right)} - \sqrt{T} + 1 \qquad \textbf{(7)}$$

Where T is constant of integration, which is made to be 0 most of the time. The function given by Equation (9) converges to a fixed point given by the limit:

$$w_c = \lim_{a \to \infty} w = \sqrt{2\left(\frac{A_a}{B_a}\ln(2) + T\right)} - \sqrt{T} + 1 \qquad \textbf{(8)}$$

The objective of the new slow start algorithm, illustrated in Fig. 1.a, is to match the value of this limit to a stable window (it does not provoke any congestion signal). Initially, let w_c be the limit given by Eq. (8) for some initial A_a. Each of the iterations tests a new function. That function has the same coefficients B_a and M_a, but different coefficients A_a (being A_a=-C_a). In each of the iterations, the window size is continuously incremented according to Eq. (6) (or Eq.(7)). When any congestion signal is detected, w_c is adjusted and a new A_a is calculated which starts a new iteration. If the window size reaches w_c and no congestion signal is detected, then the slow-start function has found the stable window and the algorithm terminates. Fig. 1.b shows the evolution of the window in the slow-start algorithm. Initially, w_c is made to be a large value. As the congestion occurs in the points a′ and a″, w_c changes to w_c' and from w_c' to w_c''. Fig. 1.b shows a family of functions determined by Eq. (6) (or Eq. (7)) having the same coefficients, except A_a. So, the slow-start terminates when the correct function is discovered whose limit is the sustainable window.

For computing the new value of A_a, we proceed as follows. Let w_c' and A_a' be the new values calculated for w_c and A_a at the end of an iteration and let d = w_c-w'_c. Then the value A_a' can be calculated by:

$$A'_a = \frac{A_a \ln(2) - B_a d \sqrt{2\left(\frac{A_a}{B_a}\ln(2) + T\right)} + \frac{B_a d^2}{2}}{\ln(2)} \qquad \textbf{(9)}$$

Finally, the inverse function of w is given by:

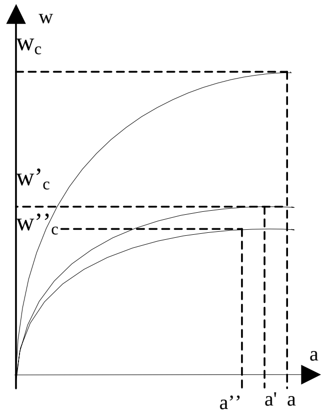
$$a = \frac{\ln\left(e^{\frac{-B_a}{2A_a}\left[\left(w + \sqrt{T} - 1\right)^2 - 2T\right] + \ln(2)} - 1\right)}{-2B_a} \qquad \textbf{(10)}$$

5 Simulations

In the following, we illustrate only characteristics of Smooth TCP related to the slow start, fair factor and the variation of RTT, mainly, due the huge set of aspects to be covered. Using these Equations and algorithms, we explored Smooth TCP in the setting shown in **Fig. 2**. In this scenario, the links between the router and the server are 10 Mbps and between the two routers is 1.5 Mbps with latency of 20ms. The link latency between the servers and the first router is 2ms and between the clients and the second router is 4ms. Modified RED queues were implemented in the routers to send ICMP-SQ messages when the average queue size reaches 8 packets or a packet is dropped, i.e., as soon as the average queue size reaches 16 packets.

```
slow_start=true
WHILE slow_start AND new_ack() DO
   IF($w<w_c$) THEN
      IF(congestion_signal)THEN
            $d = w_c – w$
            $A_a$= upt_conv (d) //Eq.(9)
            $w_c = \lim_{a\to\infty} (w)$  // Eq.(8)
            $w = w*3/4$
            $a = W^{-1}(w)$        // Eq.(10)
      ELSE
            $w = w+A_a-A_a \tanh(B_a a)$
            $a = a+1$
   ELSE
            slow_start = false
END WHILE
```

(a)



(b)

Fig. 1. The new slow-start algorithm

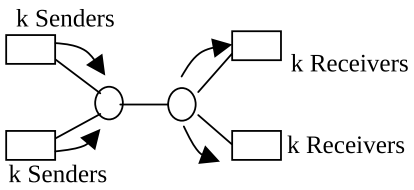


Fig. 2. Networks Simulated

Table 1 shows the behavior of the Slow-Start algorithm for various simulations having A_a initially setup to 40, while B_a is modified for each simulation. All Smooth TCP connections begin the Slow-start procedure after 1s to give enough time to initiate all them. So the convergence time in **Table 1** should discount this time. In general, we see that the time to converge and the amount of acknowledgments are inversely proportional to B_a. We also consider the precision of the converged window is directly proportional to the time or the amount of acknowledgments to converge.

B_a	k	Convergence Time	Acks to Converge	Converged Window	Steps
0.0003125	1	5.84	69943.5	7.04	10
0.0006250	1	4.81	35080.2	8.25	9
0.0012500	1	3.06	17625.5	10.1	5
0.0003125	2	6.60	69528.2	5.22	9
0.0006250	2	5.05	34891.8	6.37	9
0.0012500	2	4.50	17558.4	5.50	11
0.0003125	4	30.36	68814.4	4.00	62
0.0006250	4	9.22	34156.4	2.87	14
0.0012500	4	5.86	17539.7	6.05	16

Table 1. Performance of Slow-Start Algorithm

The coefficients of Smooth TCP are configured as showed in **Table 2**. Indications how to they were chosen can be found in [13]. The other coefficients not in **Table 2** were made to be 0.0 for these simulations. **Fig. 3**, **Fig. 4** and **Fig. 5** show the simulation results for the 300s after starting 2 standard TCP connections and 2 Smooth TCP connections (k=1). The window of Standard TCP basically has an irregular behavior, fluctuating up and down without finding a value of convergence. The consequence of this fluctuation is the irregular behavior of the RTT, inserting huge sudden variations.

Slow Start	RTT Variation	ICMP-SQ
A_a=20	A_j=0.0	A_q=0.0
B_a= 0.000625	B_j=45.02	B_q=0.5493
C_a=-20	C_j=0.1	C_q=0.1
M_a=0.0	M_j=0.00	M_q=0.00

Table 2. Coefficient values of Smooth TCP for the Simulations

Fig. 4 shows the results for Smooth TCP with a fair factor κ=0.000. Clearly its suffers from fairness since the connections do not share the same bandwidth. However this changes when κ=0.001 as showed in **Fig. 5**. Smooth TCP with κ=0.001 has approximately the same fairness as Standard TCP. However, this introduces variations in RTT caused by the fair factor different from 0, even though those variations are much smaller than in Standard TCP. **Table 3** illustrates the standard deviation of other simulations.

Another characteristic is that in both Smooth TCP simulations (**Fig. 4** and **Fig. 5**), the number of ICMP-SQ messages is smaller than those in Standard TCP. In fact, with κ=0.000, ICMP-SQ are no longer generated after a certain time. On the order hand, doing κ=0.001, Smooth TCP keeps the generated ICMP-SQ messages constant.

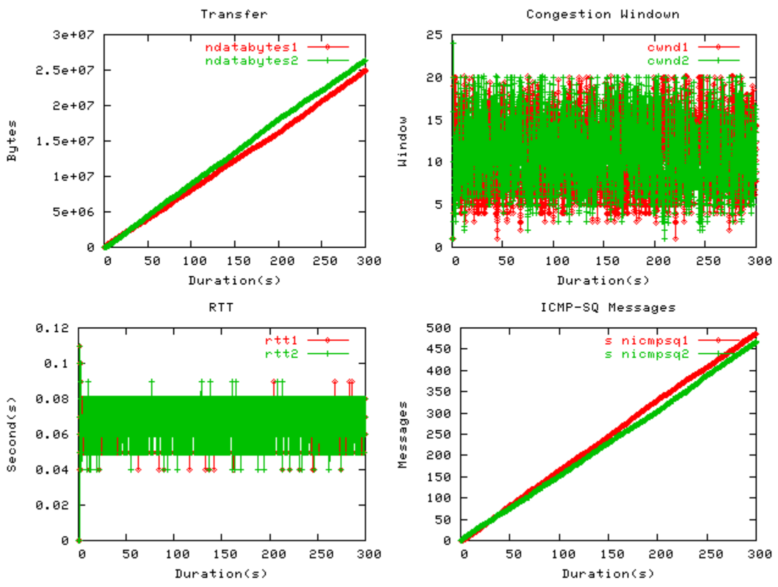


Fig. 3. Behavior of Standard TCP for k=1

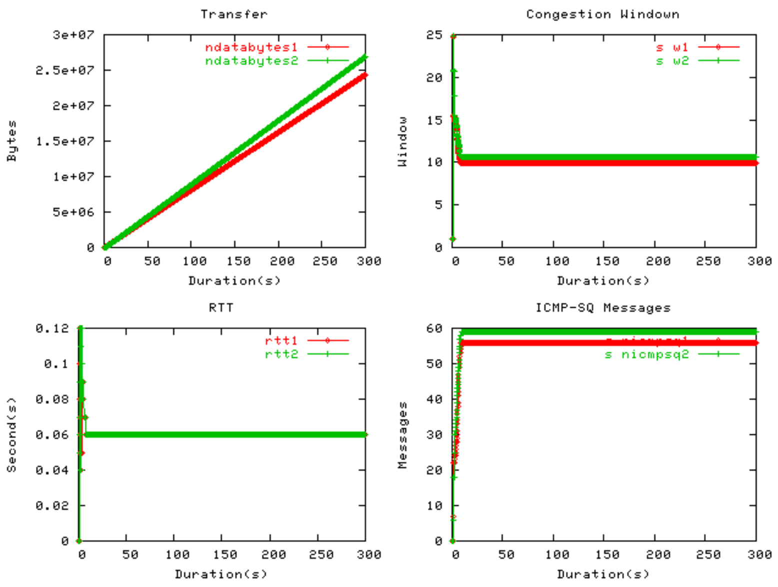


Fig. 4. Behavior of Smooth TCP for k=1 and κ=0.000

Fair Factor(κ)	**k=1**	**k=2**	**k=4**	**k=8**
0.000	1.20E+06	3.30E+04	1.20E+06	1.33E+06
0.001	3.30E+05	5.40E+05	1.50E+05	5.38E+05
0.005	3.80E+05	4.00E+05	2.90E+05	1.12E+05
0.010	1.91E+05	2.08E+05	1.74E+05	8.14E+04
Standard TCP	1.00E+06	2.50E+05	1.60E+05	9.17E+04

Table 3. Standard deviation of transfered bytes

Fig. 6 shows that all duplicate acknowledgments and timeout retransmits were generated in the initial phase when the queue was overloaded. After that, its size is kept at a level enough to generate ICMP-SQ messages without causing any packet drops (no duplicate acknowledgments/timeout retransmits). But this behavior changes, as the number of connections increases (or fair factor), approximating to the Standard TCP.

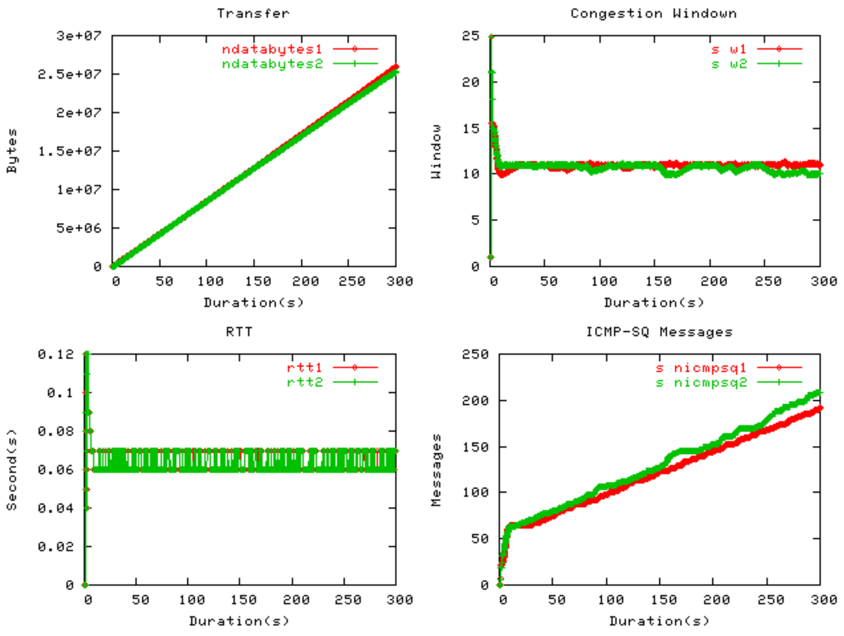


Fig. 5. Behavior of Smooth TCP for k=1 and κ=0.001

6 Final Considerations

A new requirement for new congestion control mechanisms for TCP was presented as well as an instance of such a mechanism, called Smooth TCP. This mechanism was motivated by the desire to reduce the temporal gap and to improve the precision of congestion signals. Throughout simulations, it was compared the performance of various configurations of Smooth TCP against Standard TCP. The results for these simulations were a characterization of Smooth TCP as inserting variations of RTT proportionally to its fair factor, which roughly is responsible for the fairness degree. Also, from the simulations results, Standard TCP inserted more variations than any configuration of Smooth TCP, even if the fairness characteristics are similar. These results are important for TCP used with wireless and high-speed networks, where a smooth mechanism seems to be most applicable. As ongoing work, the next phase is to determine good coefficients for Smooth TCP in each possible environment, namely wireless, high-speed network and TCP multimedia.

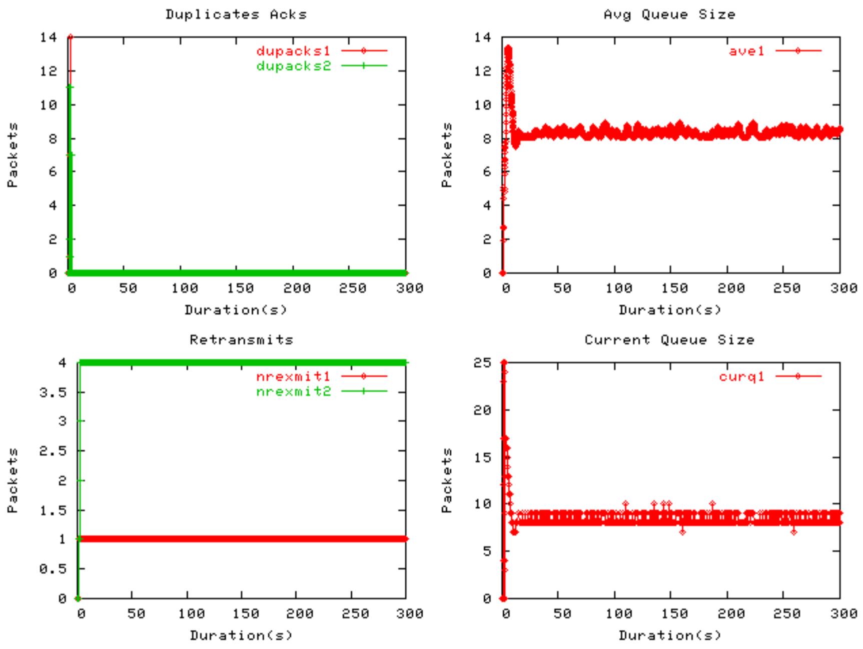


Fig. 6. Duplicates Acknowledgements and average queue size in both Standard TCP and Smooth TCP

References

1. Nagle, J.: RFC0896 - Congestion Control in IP/TCP Internetworks. Network Working Group - IETF (1984).
2. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-Based Cong. Control for Unicast Applications: the Extended Version. Technical Report, TR-00-003, ICSI Berkeley (2000).
3. Ludwig, R., Katz., R.: The Eifel Algorithm: Making TCP Robust Against Spurious Re-transmissions. Vol 30-1 (2000) 30-36.

4. Gerla, M., Sanadidi, M., Wang, R., Zanella, A., Casetti, C., Mascolo, S.: TCP Westwood: Congestion Window Control Using Bandwidth Estimation. Proceedings of IEEE Globecom 2001, Vol 3. San Antonio (2001) 1698-1702.
5. Zanella, A., Procissi, G., Gerla, M., Sanadidi, M. Y.: TCP Westwood: Analytic Model and Performance Evaluation. Proceedings of IEEE Globecom 2001, San Antonio (2001) 1703-1707.
6. Sarolahti, P., Kojo, M., Raatikainen, K.: F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts. ACM SIGCOMM 03 Computer Communication Review, Vol 33-2, ACM Press (2003) 51–63.
7. Floyd, S.: RFC3649: High-Speed TCP for Large Congestion Windows, Network Working Group. IETF (2003).
8. Bansal, D., Balakrishnam, H.: TCP-Friendly Congestion Control for Real-time Streaming Applications. Technical Report, MIT-LCS-TR-806, M.I.T Laboratory for Computer Science, Cambridge (2000).
9. Vieira, E., Bauer, M.: Smoothness Properties in Congestion Control for TCP, 3rd International Conference on Computing Communications and Control Technologies. Austin (2005) *(to appear)*.
10. Paxson, V., Allman, M.: RFC2988 - Computing TCP's Retransmission Timer. Network Working Group, IETF (2000).
11. Jacobson, V.: Congestion Avoidance and Control. ACM SIGCOMM 88 Computer Communication Review, ACM Press, (1988) 314-328.
12. Appenzeller, G., Keslassy, I., McKeown, N.: Sizing Router Buffers. ACM SIGCOMM 04 Computer Communication Review (2004) 281-291.
13. Ramakrishnan, K., Floyd, S., Black, D.: RFC3168 - The Addition of Explicit Congestion Notification (ECN) to IP. Network Working Group, IETF (2001).