Issues in Performance Evaluation of New TCP Stacks in High Speed Networks

Saverio Mascolo and Francesco Vacirca

***Abstract*— TCP congestion control has been designed to ensure Internet stability along with fair and efficient allocation of the network bandwidth. With the spreading of high-speed and long-distance networks, the standard TCP congestion control algorithm showed some limitations in exploiting full bandwidth utilization. During the last few years, many new congestion control algorithms have been proposed to improve the classic Reno/NewReno TCP congestion control in such networks.**
This paper aims at comparing some of the most promising proposals and at investigating the relation between these new congestion control proposals and different bottleneck buffer provisioning rules.

I. Introduction

In recent years, issues regarding the behavior of TCP in high-speed and long-distance networks have been extensively addressed in the networking research community, both because TCP is the most widespread transport protocol in the current Internet and because bandwidth-delay product continues to grow. The well known problem of TCP in high bandwidth-delay product networks is that the TCP Additive Increase Multiplicative Decrease *AIMD* probing mechanism is too slow in adapting the sending rate to the end-to-end available bandwidth.

To overcome this problem, many modifications have been proposed such as FAST TCP [1], STCP [2], HSTCP [3], HTCP [4], BIC TCP [5] and CUBIC TCP [6].
It is important to remark that all new proposal must preserve backward compatibility with standard TCP implementations, a requirement that is usually referred as "TCP friendliness". Friendliness enables new TCP stacks to coexists in the same networks with older stacks. Using the ns simulator [7] in a simple single bottleneck scenario with few sender entities, we show that none of the actual TCP proposal suffices in exploiting the link capacity while preserving fair bandwidth sharing with standard TCP congestion control. Moreover, little changes in the network scenario affect results in remarkable way.

The paper is organized as follows: Section II provides background on TCP and on High-Speed TCP congestion control; Section III reports simulation results, whereas Section IV draws the conclusions.

Saverio Mascolo (mascolo@poliba.it), DEE Politecnico di Bari, via Orabona 4, 70125 Bari, Italy.
Francesco Vacirca (vacirca@infocom.uniroma1.it), INFOCOM Dept. - University of Roma *"La Sapienza"*, Via Eudossiana 18, 00184, Roma, Italy.

II. Background on High Speed TCP congestion control

In this Section we briefly summarize main features of the TCP stacks we are investigating in this paper. In particular, we will consider the standard TCP (i.e. Reno/NewReno TCP), STCP, HSTCP, HTCP, Westwood+ ([8], [9]), BIC and CUBIC. In the following of the paper we refer to the set of TCP new proposals as New Generation TCPs (NGTCPs) to distinguish from standard TCP.
TCP congestion control [10] is essentially made of a probing phase and a decreasing phase. The probing phase of standard TCP consists of an exponential growing phase (i.e. the slow-start phase) and of a linear increasing phase (i.e. the congestion avoidance phase). The probing phase stops when congestion is experienced in the form of 3 duplicate acknowledgments or a timeout. At this point Reno or NewReno TCP implements a multiplicative decrease behavior. The Reno/NewReno setting of the congestion window during the congestion avoidance phase is: on ACK reception, $cwnd$ is increased by $1/cwnd$, when a loss occurs (3 DUPACKs are received), $cwnd$ is halved.
This behaviour can be generalized with the following equation:

a) On ACK reception:

$$cwnd \longleftarrow cwnd + a \tag{1}$$

b) When 3 DUPACKs are received:

$$cwnd \longleftarrow cwnd - b \cdot cwnd \tag{2}$$

where a is $1/cwnd$ and b is 0.5. Some of the NGTCP algorithms can be described as a modification of the a and b parameters.

STCP. In the Scalable TCP [2], the increase factor a is set to a constant value to make the growth of $cwnd$ independent of $cwnd$ itself. The author suggests to set a = 0.01 to permit the rate to double in about 70 round trip time for any rate. The decreasing factor b is changed to 0.125.

HSTCP. The High Speed TCP [3] congestion control algorithm is divided into two parts depending on the value of the congestion windows. If $cwnd$ is lower than a threshold $cwnd_{low}$ of 38 packets , the values of a and b are the same of NewReno. When $cwnd$ is between $cwnd_{low}$ and $cwnd_{high}$ packets, a and b are computed as:

$$b = (0.1 - 0.5) \cdot \frac{\log cwnd - \log cwnd_{low}}{\log cwnd_{high} - \log cwnd_{low}} + 0.5 \tag{3}$$

$$a = 2 \cdot cwnd^2 \cdot p(cwnd) \frac{b}{2 - b} \tag{4}$$

where $p(w) = 0.078/w^{1.2}$. The rationale behind equations 3 and 4 is the choice of a predefined response function, that maps the steady-state packet drop probability to the average sending rate. The original TCP response function ($w = 1.2/\sqrt{p}$) is modified to $w = 0.12/p^{0.835}$

HTCP. In [4] the authors propose Hamilton TCP as an enhancement for high bandwidth-delay networks. They define Δ_i as the time elapsed from the last congestion event for the i-th source. If Δ_i is lower than a threshold Δ_{th} the protocol behaves like a NewReno TCP. If $\Delta_i > \Delta_{th}$, the increment factor a is equal to:

$$a = 1 + 10(\Delta_i - \Delta_{th}) + (\frac{\Delta_i - \Delta_{th}}{2})^2 \qquad (5)$$

It can be noticed that the increment factor has a quadratic trend in relation with the time elapsed from the last congestion event. The decrement factor b is set dynamically considering that the full link utilization is achieved when the bottleneck buffer does not remain empty for a long period. To this purpose the authors propose to set $b = 1 - \frac{RTT_{min}}{RTT_{max}}$.

Westwood+. The main idea of Westwood+ TCP is to set the control windows after congestion such that the bandwidth available at the time of congestion is exactly matched (see [8] and [9]). The available bandwidth is estimated by counting and averaging the stream of returning ACK packets. In particular, when three DUPACKs are received, the congestion window ($cwnd$) is set equal to the estimated bandwidth (BWE) times the minimum measured round trip time (RTT_{min}).

BIC. The BIC protocol [5] consists of two parts: a binary search increase phase and an additive increase phase. In the binary search phase the congestion window setting is performed as a binary search problem. After a packet loss, the congestion window is reduced by a constant factor b, $cwnd_{max}$ is set to the window size before the lost and $cwnd_{min}$ is set to the value of congestion window after the loss ($cwnd_{min} = b \cdot cwnd_{max}$). If the difference between the congestion window middle point $(cwnd_{max} + cwnd_{min})/2$ and the minimum congestion window $cwnd_{min}$ is lower than a threshold S_{max} the protocol starts a binary search algorithm increasing the congestion window to the middlepoint, otherwise the protocol enters a "linear increase" phase and increments the congestion window by one for each received ACK. If BIC does not get a loss indication at this window size, the actual window size become the new minimum window, whereas if it get a packet loss the actual window size become the new maximum. The process goes on till the window increment become lower than the S_{min} threshold and the congestion window is set to $cwnd_{max}$. If the window grows more than $cwnd_{max}$, the protocol enters into a new phase ("max probing") that is specular to the previous phase; that is, it uses the inverse of the binary search phase first and then the addictive increase. The default value of b is 0.2.

CUBIC. The CUBIC algorithm [6] defines a congestion window increase analytical function that is similar to the increase behaviour of BIC, but less aggressive in situations where the link has a small bandwidth-delay product to enhance the friendliness with NewReno TCPs. The congestion window increase of the CUBIC protocol is described by the following formula:

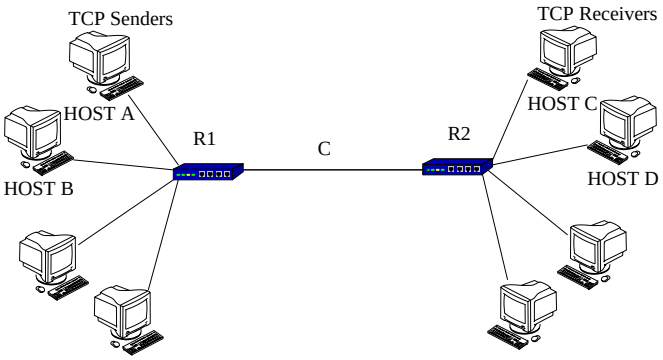


Fig. 1. Simulation scenario.

$$cwnd = C(\Delta - K)^3 + cwnd_{max} \qquad (6)$$

where C is a scaling factor, Δ is the time elapsed since the last congestion event $cwnd_{max}$ is the congestion window size before the last congestion event and K is:

$$K = \sqrt[3]{cwnd_{max} \cdot b/C} \qquad (7)$$

where b is the constant decrement factor (default value is 0.2 as BIC).

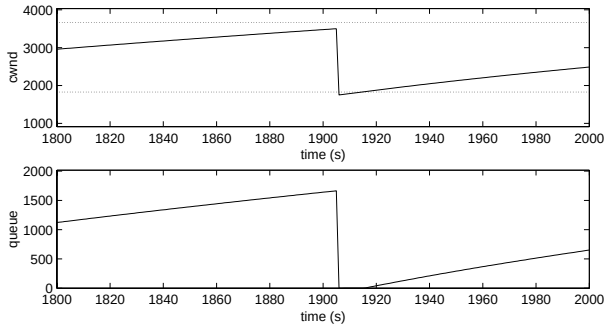
III. Simulation Results

Figure 1 shows the reference scenario employed for collecting computer simulation results. It consists of N TCP Senders establishing a connection with N TCP Receivers. The bottleneck link between the R1 and R2 routers is provisioned with capacity C and has a round trip time propagation time equal to RTT. The propagation delays of the dedicated links between R1 and the senders and between R2 and the receivers are distributed between $2.5/N$ ms and 2.5 ms. This small difference in the propagation delays is important to desynchronize flows in multiple source scenarios. Links between router R1 and TCP senders and links between router R2 and TCP receivers are provisioned with a capacity greater than C so that the link with capacity C is the bottleneck for each TCP flow. The bottleneck buffer is the queue in the R1 router and it can accommodate up to B packets; in all simulations the packet size is 1500 bytes. Simulations have been performed using the network simulation [7] version 2.28 with code and settings shown in [11], [12] and in [13]. Every simulation lasts 2000 seconds and the goodput is computed over the last 1000s of the simulation.

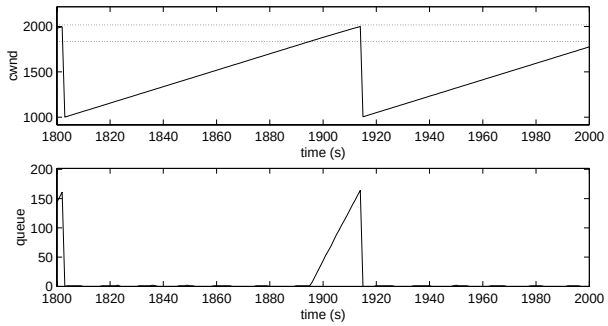
A. Single flow scenario

The goal is to analyze how different TCP congestion control algorithms behave with respect to different buffering provisioning; for this purpose we focus our analysis on the decrement factor b. To this aim we perform a simple set of simulations using a single source (host A). The bottleneck link round trip propagation time delay RTT is set equal

to 100 ms and to 300 ms and the capacity C is set equal to 200 Mbit/s. Propagation delay between Host A and R1 and between R2 and Host C is 2.5 ms and the end-to-end propagation round trip delay is $RTT+$ 10 ms. The bottleneck buffer B is set to two different values $B = C \cdot RTT$ packets and $B = 0.1 \cdot C \cdot RTT$. The first setting follows the classical "rule-of-thumb" attributed to Villamizar and Song [14] to keep the congested link as busy as possible when standard TCP congestion control is used. The second setting is more realistic in high bandwidth delay link where providing buffers using the rule-of-thumb would be too expensive [15].



(a) $B = C \cdot RTT$

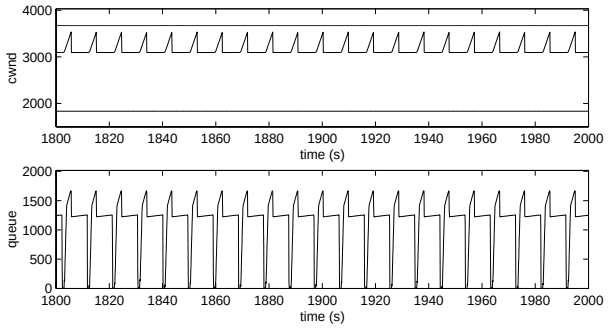


(b) $B = 0.1 \cdot C \cdot RTT$

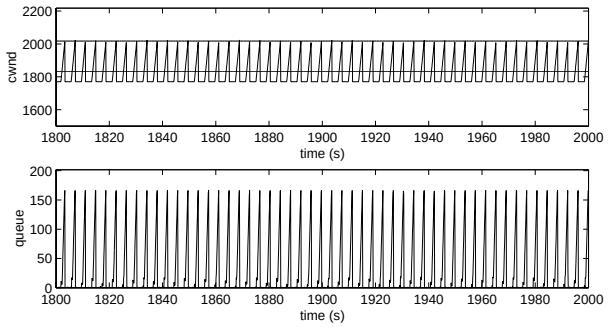
Fig. 2. NewReno TCP Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.

Figures 2(a), 3(a), 4(a), 5(a), 6(a), 7(a) and 8(a) depict the congestion window and the bottleneck queue utilization for NewReno and NGTCPs in case of $B = C \cdot RTT$, whereas Figures 2(b), 3(b), 4(b), 5(b), 6(b), 7(b) and 8(b) depict same variables in the case of $B = 0.1 \cdot C \cdot RTT$[1]. In the first case, NewReno congestion window oscillates between $C \cdot RTT$ and $C \cdot RTT + B = 2 \cdot C \cdot RTT$ and the queue is almost never empty leading to full link utilization with a goodput of 199.8 kbit/s. When the buffer is under-provisioned with respect to the "rule-of-thumb" (Figure 2(b)), TCP Reno is not able to provide full link utilization because the congestion

[1]In the congestion window plots the two dotted lines represent $C \cdot RTT$ and $B + C \cdot RTT$



(a) $B = C \cdot RTT$

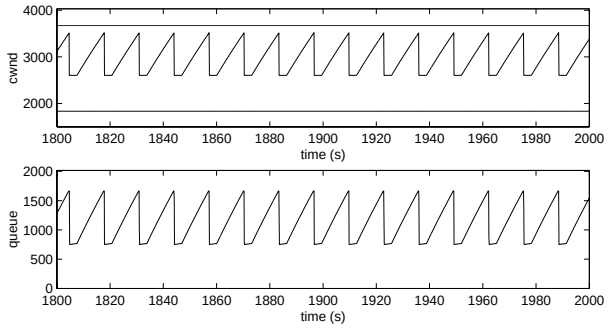


(b) $B = 0.1 \cdot C \cdot RTT$

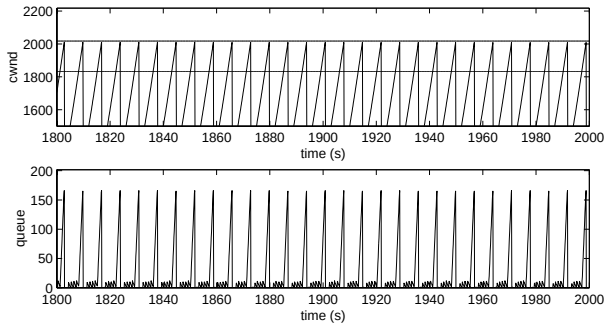
Fig. 3. STCP Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.

window oscillates between $B + C \cdot RTT = 1.1 \cdot C \cdot RTT$ and $(1.1 \cdot C \cdot RTT)/2$, which empties the queue in repeating intervals during which the link is underutilized. In this case the measured goodput is 162 Mbit/s. The underutilization of the link is due to the decrease value of the congestion window $b = 0.5$ that is independent of the bottleneck buffer. Generalizing, the link is underutilized when the bottleneck buffer B is lower than $b/(1-b) \cdot C \cdot RTT$, whereas the queue is always congested when $B > b/(1-b) \cdot C \cdot RTT$.

As far as regards NGTCP protocols, we can group the protocols according to the decreasing factor policy. In the first group (STCP, BIC and CUBIC) the decreasing factor is static, whereas in the second group (HSTCP, Westwood+ and HTCP) the decreasing factor is modified dynamically according to different rules. It is possible to note that for all protocols in the first group, when the buffer is provisioned according to the NewReno "rule-of-thumbs", b is too low to drain the bottleneck queue and the utilization of the queue is always high (Figures 3(a), 7(a) and 8(a)). When the buffer is under-provisioned with respect to the "rule-of-thumbs", b is slightly higher than the optimal one and the full-link utilization is not completely achieved (see Table III-A and Figures 3(b), 7(b) and 8(b)). However, the achieved goodput is higher than that of NewReno both because the b is lower than that of NewReno and because the increasing factor is

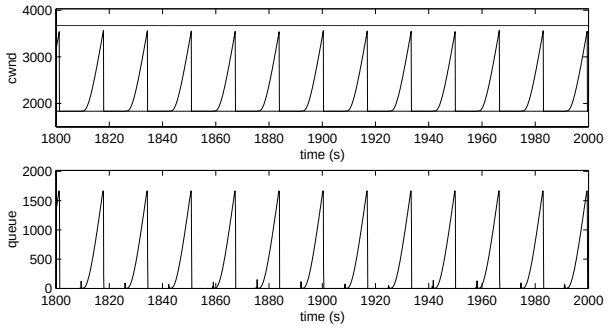


(a) $B = C \cdot RTT$

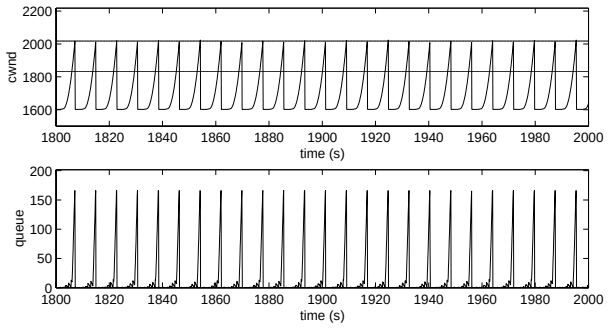


(b) $B = 0.1 \cdot C \cdot RTT$

Fig. 4. HSTCP TCP Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.



(a) $B = C \cdot RTT$



(b) $B = 0.1 \cdot C \cdot RTT$

Fig. 5. HTCP TCP Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.

higher and faster in filling the queue after congestion.

As far as regards the second group, it is possible to observe that the decreasing factor of HSTCP, computed with Eq. 3, depends on the maximum congestion window before the loss events. Although b is computed dynamically, the setting is not optimized to empty the buffer after congestion: when $B = C \cdot RTT$, the computed $b(cwnd)$ is too low and the protocol is not able to drain the queue (Figure 4(a)). When $B = 0.1 \cdot C \cdot RTT$, instead, the queue is often empty but the achieved goodput is almost equal to full link utilization, as reported in Table III-A.

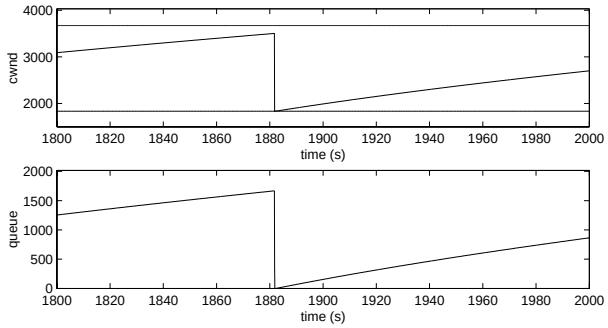
With reference to Westwood+, when $B = C \cdot RTT$, it behaves as NewReno: in fact, when a packet is lost, TCP Westwood+ decreases the congestion window to $C \cdot RTT$, which corresponds to the half of $B + C \cdot RTT$, that is, it corresponds exactly to TCP NewReno congestion window setting after loss. Also the queue dynamics behave similarly, and both protocols achieve full link utilization (Figure 6(a)). When the buffer is under-provisioned with respect to the "rule-of-thumb", behaviors of NewReno and Westwood+ are different. As it is shown in Figure 6(b), Westwood+ congestion control oscillates between $1.1 \cdot C \cdot RTT$ and $C \cdot RTT$, which provides a higher level of link utilization. In this simple scenario, Westwood+ is the protocol that obtains the best goodput for every bottleneck buffer size and for every RTT (see Table III-A); however it is important to emphasize that Westwood+ has not been designed for high bandwidth delay product networks and the increasing factor a is the same of NewReno.

	$B = 0.1 \cdot C \cdot RTT$		$B = C \cdot RTT$	
RTT	100ms	300ms	100ms	300ms
NewReno	162113.4	140719.4	199807.6	199917.5
STCP	194521.7	90597.7	187380.2	65858.3
HSTCP	178091.5	184023.4	199999.9	163408.0
HTCP	181463.4	67408.5	177873.3	43019
Westwood+	199994.6	199999.9	199992.5	199999.5
BIC	189212.6	177318.3	199996	189488.3
CUBIC	193757.9	184963.7	199997	185620.8

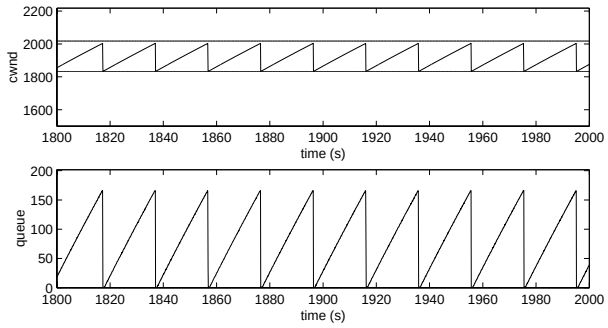
TABLE I

GOODPUT (KBIT/S) IN A SINGLE SOURCE SCENARIO FOR DIFFERENT PROTOCOLS WITH C=200 MBIT/S.

The last protocol to analyze is HTCP. From Figure 5, it is possible to notice that the congestion window increases quadratically with respect to the time elapsed from the last congestion recovery event. However it seems that the setting of congestion window after a loss does not reflect the expected value that should converge to the same value used by Westwood+. It is not clear if this malfunction is due to a bad estimation of minimum and/or maximum round trip time or to other reasons.

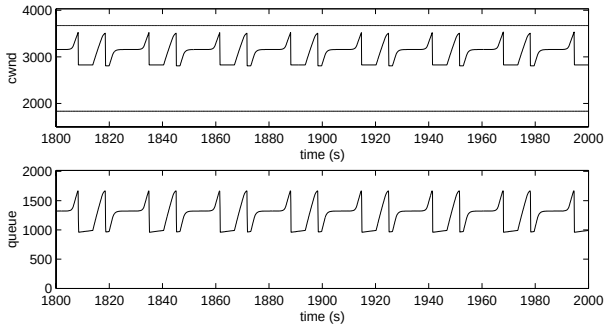


(a) $B = C \cdot RTT$

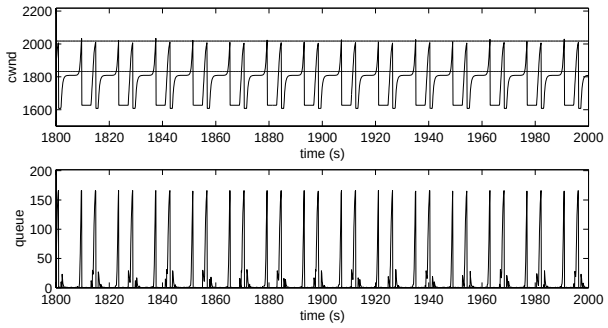


(b) $B = 0.1 \cdot C \cdot RTT$

Fig. 6. Westwood+ TCP Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.



(a) $B = C \cdot RTT$



(b) $B = 0.1 \cdot C \cdot RTT$

Fig. 7. BIC Congestion Window and queue utilization for a single TCP source scenario with $C = 200$ Mbit/s and $RTT = 100$ ms.

B. Two flow scenario

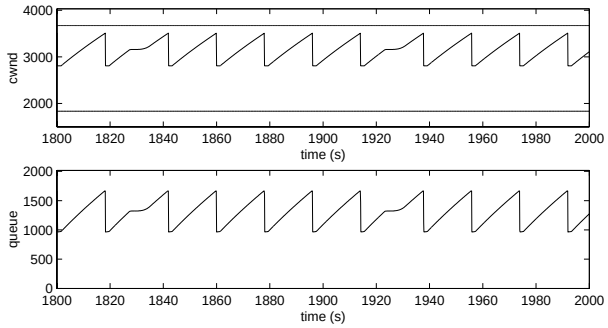
In this section we consider the scenario where two TCP flows share the same bottleneck link. The bottleneck capacity is 200 Mbit/s and RTT is 100 ms. Propagation delays between Host A and R1 and between R2 and Host C is 1.25 ms, whereas the propagation delay of the link between Host B and R1 and between R2 and Host D is 2.5 ms. The first issue we explore is how two New Generation TCP stacks share the available bandwidth of the bottleneck and how this is related to buffer provisioning. Figure 9 depicts the overall goodput of the two sources with respect to the buffer sizes provisioned in the bottleneck router (R1). In this scenario, BIC, CUBIC and STCP achieve a higher overall goodput when the buffer is smaller than the bandwidth-delay product[2], whereas HTCP, HSTCP and NewReno achieve an utilization around 0.85. When the buffer is equal to the bandwidth delay product all protocols, except HTCP, achieve the full link utilization.
Figure 10 reports the Jain fairness index [16]:

$$J_{FI} = \frac{\left(\sum_{i=1}^{N} g_i\right)^2}{N \sum_{i=1}^{N} g_i^2}$$

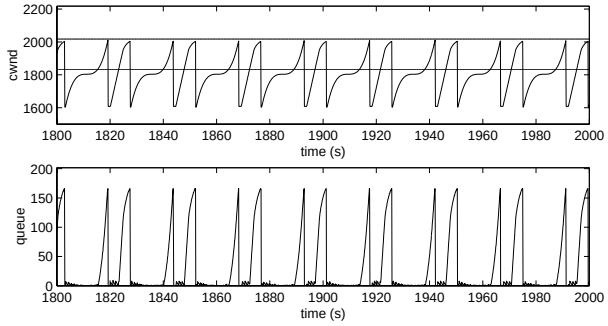
[2]When C=200Mbit/s, RTT=100ms and the packet size is 1500 byte, the bandwidth delay product is 1667 packets.

where g_i is the goodput of the i-th connection and N is the number of connections sharing the bottleneck. It is worth to notice that STCP provides a very low fairness that slightly increase with the buffer size. Other protocols provide a fairness that increases with the buffer size but in most of the considered cases is lower than that of NewReno. Moreover, it is worth to notice that in the network scenarios with RTT equal to 100 ms, such as in the one shown in Figure 9, NGTCP connection with lower end-to-end round trip time delay obtains a lower goodput than the connection with higher round trip time; in other scenarios, which are not reported, the goodput obtained by connection that experience a lower round trip time is higher than connection with higher round trip time.

The second issue we investigate is the achievable fairness between a Next Generation TCP flow and a NewReno flow. Table III-B illustrates the goodput in kbit/s when two flows share the same bottleneck link. In the first six rows the NGTCP source is on the Host A and the NewReno TCP is located on the Host B, whereas in the second six rows the NewReno is on the Host A and NGTCP is on Host B. It is worth noting that a small difference in RTT provides a large difference in the goodput achievable by the two heterogeneous TCP stacks. When the NewReno connection experiences a higher round trip time delay with respect



(a) $B = C \cdot RTT$



(b) $B = 0.1 \cdot C \cdot RTT$

Fig. 8. CUBIC TCP Congestion Window and queue utilization for a single TCP source scenario with C = 200 Mbit/s and RTT = 100 ms.

to the NGTCP one, NGTCPs (except STCP) provides a satisfactory fairness comparable to NewReno homogeneous scenarios; but when the NGTCPs experience higher delays the friendliness with NewReno is not achieved. As far as regards Westwood+, the goodput of the Host A (with lower RTT) is always lower than goodput of Host B independently of the protocol; Westwood+ does not prevail on NewReno because in scenarios with $B = C \cdot RTT$ the two protocols have similar increase and decrease factors as shown in Figures 2(a) and 6(a).

C. Multi flow scenario

In this section, we consider the case of many flows that share the same bottleneck link. Figures 11 and 12 depict the overall goodput achieved by 10 TCP flows when the bottleneck link capacity is 200 Mbit/s and the propagation round trip time delay is 100 ms and 300 ms respectively. In the first scenario, STCP, BIC, CUBIC and Westwood+ achieve the full link utilization with small buffers (between $0.1 \cdot C \cdot RTT$ and $0.3 \cdot C \cdot RTT$), whereas NewReno and HSTCP need larger buffers, between $0.6 \cdot C \cdot RTT$ and $C \cdot RTT$, to exploit the whole link capacity. HTCP is not able to exploit fully the link capacity and the utilization is a little bit lower than other protocols. In the second scenario with RTT equal to 300 ms, BIC, CUBIC and Westwood+ provide full link utilization with small and large buffers, whereas

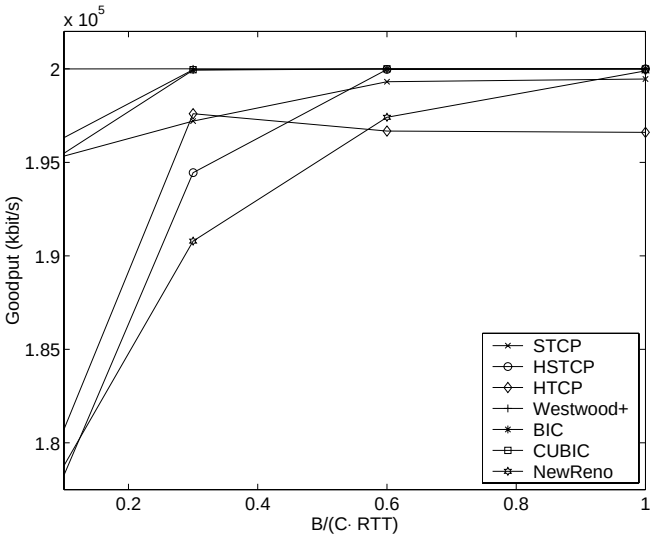


Fig. 9. Goodput in a 2 sources scenario with C = 200 Mbit/s, RTT = 100 ms.

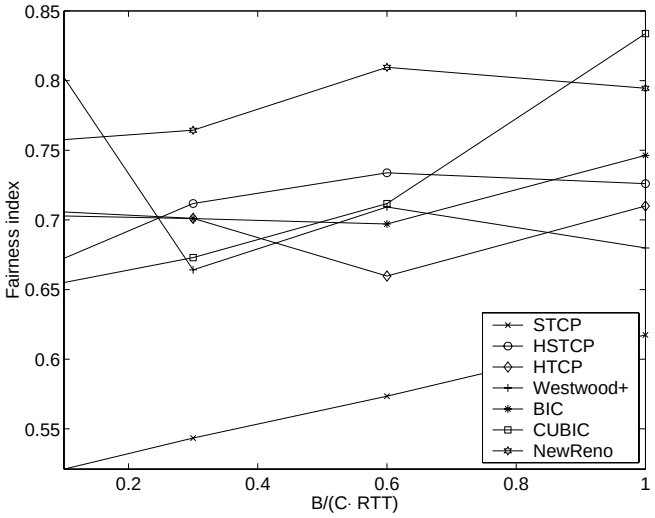


Fig. 10. Fairness index in a 2 sources scenario with C = 200 Mbit/s, RTT = 100 ms.

Host A		Host B	
STCP	185841	NewReno	12701.8
HSTCP	148327	NewReno	51461
HTCP	143233	NewReno	55246
Westwood+	40524	NewReno	159422
BIC	153381	NewReno	43295
CUBIC	103065	NewReno	96932
NewReno	1573	STCP	198425
NewReno	4234	HSTCP	195762
NewReno	41166	Westwood+	158827
NewReno	8555	HTCP	187691
NewReno	3224	BIC	196773
NewReno	6550	CUBIC	193452

TABLE II

GOODPUT (KBIT/S) IN A MIX FLOW SCENARIO FOR C=200 MBIT/S, RTT=100MS AND $B = RTT \cdot C$.

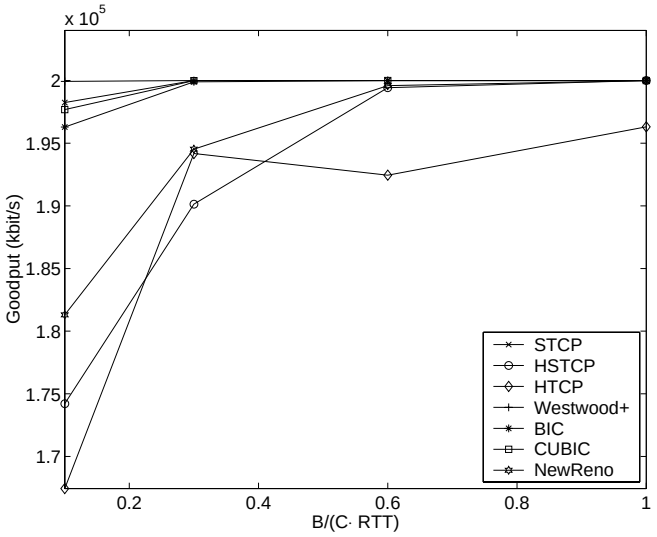


Fig. 11. Goodput in a 10 source scenario with C =200 Mbit/s, RTT = 100ms.

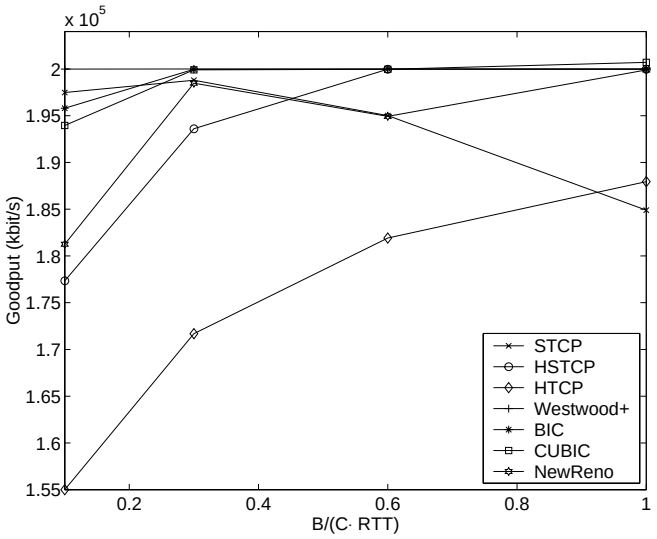


Fig. 12. Goodput in a 10 source scenario with C = 200 Mbit/s, RTT = 300ms.

STCP fails when the buffer increases and the achieved goodput decreases. HSTCP and NewReno obtain the same performance of the previous scenario and HTCP goodput is notably lower than other protocols.

The last feature we consider is the time needed by the protocols to get to the steady state. Figure 13 depicts the goodput obtained by the different NGTCP protocols in a 1 Gbit/s scenario with RTT = 100 ms and a bottleneck buffer $B = 0.1 \cdot C \cdot RTT$. For every protocol, the goodputs computed between 10s and 100s (black bar), the goodputs computed between 100s and 1000s (gray bar) and the goodputs computed over the last 1000s (white bar) are reported in the case of 5, 10 and 50 flows sharing the same bottleneck. When the number of flows that share the link is large (e.g. 50 flows), all the protocols are able to exploit the link

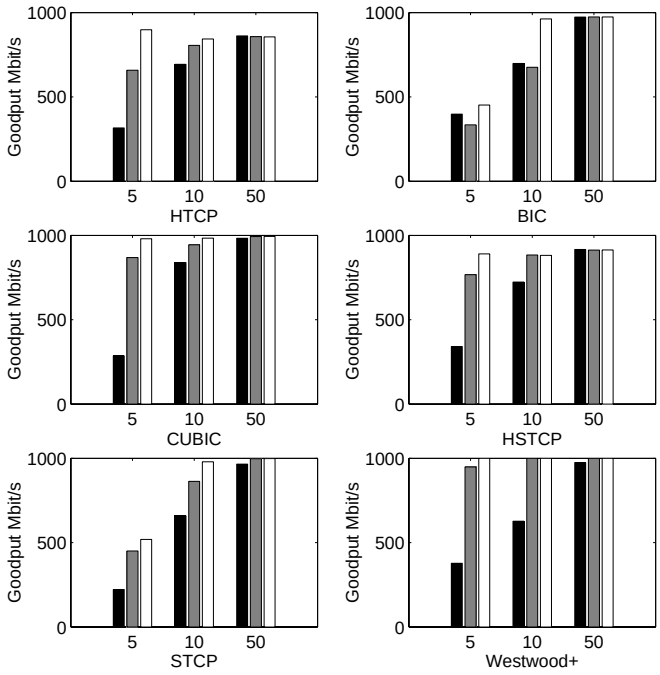


Fig. 13. Goodput in a Multisource scenario with C = 1 Gbit/s, RTT = 100ms.

capacity in few seconds. When 10 flows compete on the same link, HSTCP and CUBIC achieve the 72% and the 83% respectively of the goodput in the first 100 seconds, whereas other protocols are below the 70% with the minimum of 62% achieved by Westwood+. In the first 1000 seconds, CUBIC and Westwood+ are the only protocol to exploit the available capacity, whereas the other protocols need more time to achieve the maximum goodput. It is worth noticing that HSTCP and HTCP are not able to fully utilize the bandwidth and the goodputs computed over the last 2000 seconds are 882 Mbit/s and 843 Mbit/s respectively.
Finally, when only 5 flows compete, CUBIC and Westwood+ behave well achieving high goodput in the first 1000 seconds and full link utilization within 2000 seconds, whereas other protocol are fast in the start phase, but are not able to obtain high goodput during the 2000 seconds of simulations.

IV. Conclusions

In this paper we present a simulation study of new congestion control algorithms proposed to overcome the TCP well-known problem in high bandwidth-delay networks. Using the ns simulator [7] in a single bottleneck scenario with few or many sender entities, we analyze the performance of different proposals also with respect to different buffer size provisioning. Simulation results show interesting and complex behaviour of TCP stacks. In particular, STCP has shown significant lacking of fairness, HTCP and HSTCP have shown low goodputs in the presence of buffer with size smaller than the bandwidth delay product.
As a result of this investigation, in the next future we will focus our investigation on NewReno, BIC/CUBIC and

Westwood+ using real network measurements and more realistic scenarios.

REFERENCES

[1] Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance," Proc. of INFOCOM 2004, March 2004, Hong Kong, China.

[2] Tom Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks," Proc of PFLDnet 2003, February 2003, Geneva, Switzerland.

[3] Sally Floyd, "HighSpeed TCP for Large Congestion Windows," RFC 3649, Experimental, December 2003.

[4] R.N.Shorten, D.J.Leith, "H-TCP: TCP for high-speed and long-distance networks" Proc. PFLDnet, Argonne, 2004.

[5] L. Xu, K. Harfoush, I. Rhee, "Binary Increase Congestion Control for Fast, Long Distance Networks," Proc. of INFOCOM 2004, March 2004, Hong Kong, China.

[6] I. Rhee, L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," Prof. of PFLDnet 2005, February 2005, Lyon, France.

[7] The network simulator. URL: www.isi.edu/nsnam/ns/

[8] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks," Proceedings of ACM Mobicom, Rome, Italy, July 2001.

[9] L. A. Grieco, S. Mascolo "Performance evaluation and comparison of Westwood+, Vegas and New Reno TCP congestion control," ACM Computer Communication Review, April 2004.

[10] V. Jacobson, R. Braden, D. Borman, " TCP Extensions for High Performance ", RFC 1323, May 1992.

[11] http://dsd.lbl.gov/ evandro/hstcp/simul/simul.html

[12] http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/cubic-script/script.htm

[13] http://www.hamilton.ie/net/index.htm

[14] Villamizar, C. and Song C. (1995), "High Performance TCP in ANSNET", ACM Computer Communication Review, vol. 24, no. 5, pp. 45-60.

[15] G. Appenzeller, I. Keslassy, N. McKeown, "Sizing Router Buffers," Proc. of Sigcomm 2004, August 30 - Sept. 3, 2004, Portland, Oregon, USA.

[16] R. Jain, "The art of computer system performance analysis," John Wiley and Sons, 1999.