Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control

Luigi A. Grieco and Saverio Mascolo
Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy
{a.grieco,mascolo}@poliba.it

Abstract

TCP congestion control has been designed to ensure Internet stability along with fair and efficient allocation of the network bandwidth. During the last decade, many congestion control algorithms have been proposed to improve the classic Tahoe/Reno TCP congestion control. This paper aims at evaluating and comparing three control algorithms, which are Westwood+, New Reno and Vegas TCP, using both Ns-2 simulations and live Internet measurements. Simulation scenarios are carefully designed in order to investigate goodput, fairness and friendliness provided by each of the algorithms. Results show that Westwood+ TCP is friendly towards New Reno TCP and improves fairness in bandwidth allocation whereas Vegas TCP is fair but it is not able to grab its bandwidth share when coexisting with Reno or in the presence of reverse traffic because of its RTT-based congestion detection mechanism. Finally results show that Westwood+ remarkably improves utilization of wireless links that are affected by losses not due to congestion.

1. INTRODUCTION

Internet stability is still largely based on the congestion control algorithm proposed by Van Jacobson at the end of the eighties [1], which is known as Tahoe TCP, on its first modification, which is known as Reno TCP, and other variants described in [3]-[5]. The Van Jacobson congestion control algorithm has been designed by following the end-to-end principle and has been quite successful from keeping the Internet away from congestion collapse [18]-[20]. Two variables, congestion window (*cwnd*) and slow start threshold (*ssthresh*), are used to throttle the TCP input rate in order to match the network available bandwidth. All these congestion control algorithms exploit the *Additive-Increase/Multiplicative-Decrease* (AIMD) paradigm, which additively increases the *cwnd* to grab the available bandwidth and suddenly decreases the *cwnd* when network capacity is hit and congestion is experienced via segment losses, i.e. timeout or duplicate acknowledgments. AIMD algorithms ensure network stability but they don't guarantee fair sharing of network resources [1], [6],[7], [21].

After the introduction of the Van Jacobson algorithm research on TCP congestion control become very active and several end-to-end congestion control algorithms have been proposed since then to improve network stability, fair bandwidth allocation and resource utilization of high-speed networks and wireless networks [2,3,4,9,12,14,20,32,33,39]. In fact, today TCP is not well suited for wireless links since losses due to radio channel problems are misinterpreted as a symptom of congestion by current TCP schemes and lead to an undue reduction of the transmission rate. Thus, TCP requires supplementary link layer protocols such as reliable link-layer or split-connections approach to efficiently operate over wireless links [2,10,21,33,34].

Vegas TCP was the first attempt to depart from the loss-driven paradigm of the TCP by introducing a mechanism of congestion detection before packet losses [9]. In particular, Vegas TCP computes the difference between the *actual input rate* (*cwnd*/*RTT*) and the *expected rate* (*cwnd*/RTT_{min}), where *RTT* is the Round Trip Time and RTT_{min} is the minimum measured round trip time, to infer network congestion. In particular, if the difference is smaller than a threshold α then the *cwnd* is additively increased, whereas if the difference is greater than another threshold β then the *cwnd* is Additively Decreased; finally, if the difference is smaller than β and greater than α, then the *cwnd* is kept constant [9]. In [11] it has been shown that Vegas TCP ensures network stability but it is not able to grab its own bandwidth share when interacting with algorithms that systematically hits network queue capacity as Reno.

Westwood TCP is a new congestion control algorithm that is based on end-to-end bandwidth estimate [12]. In particular, Westwood TCP estimates the available bandwidth by counting and filtering the flow of returning ACKs and adaptively sets the *cwnd* and the *ssthresh* after congestion by taking into account the estimated bandwidth. The original bandwidth estimation algorithm fails to work properly in the presence of ACK compression [41]. Thus a slightly modified version of the bandwidth estimation algorithm has been proposed in [14] to cope with ACK compression effect. We call Westwood+ the original Westwood algorithm with the enhanced bandwidth estimate. Furthermore, in [14] it has been shown via a mathematical analysis that Westwood+ is friendly towards Reno TCP and fairer than Reno in bandwidth allocation.

This paper aims at comparing Westwood+, New Reno and Vegas TCP. New Reno is an improved version of Reno that avoids multiple reductions of the *cwnd* when several segments from the same window of data get lost [3]. New Reno TCP has been considered because it is the leading Internet congestion control protocol [27]. Vegas TCP has been considered because it also proposes, as Westwood+, a new mechanism for throttling the congestion window that is based on measuring the network congestion status via RTT measurements. Moreover, Vegas TCP provides the basic ideas behind the new Fast TCP congestion control algorithm, which has been recently proposed by researchers at Caltech [39]. In authors' words, "Fast TCP is a sort of high-speed version of Vegas" [40]. At the time of this paper Fast TCP is still in a trial phase and authors do not have released any kernel code or *ns-2* implementation. Being based on *RTT* measurements to infer congestion, it could inherit all drawbacks of Vegas that will be illustrated in this paper, mainly the incapacity to grab bandwidth when coexisting with Reno traffic or in the presence of reverse traffic.

For evaluation and comparison purposes, computer simulations using the *ns*-2 simulator [16], and measurements using a Linux implementation, over the real Internet, have been collected. In particular, ns-2 simulations have been carried out over single and multi bottleneck scenarios with link capacities ranging from 1Mbps to 100Mbps, for various buffer sizes and in the presence of homogeneous and heterogeneous traffic sources. Moreover, Medium Earth Orbit (MEO) and Geosynchronous Earth Orbit (GEO) satellite scenarios in the presence of lossy links have been simulated. Simulation results have shown that: (1) Westwood+ TCP is friendly towards New Reno TCP; (2) Weswtood+ TCP improves the fairness in bandwidth sharing with respect to New Reno TCP; (3) Vegas TCP is not able to grab its own bandwidth share when coexisting with New Reno TCP [11] or in the presence of reverse traffic; (4) Westwood+ improves the utilization of wireless (i.e. satellite) links with respect to both Vegas and New Reno in the presence of uniform or bursty losses.

The paper is organized as follows: Section 2 outlines the Westwood+ algorithm; Section 3 compares New Reno, Vegas and Westwood+ using the *ns*-2 simulator; Section 4 compares New Reno and Westwood+ using live Internet experiments; finally, the last section draws the conclusions.

2. WESTWOOD+ TCP

This section describes the Westwood+ congestion control algorithm. In particular, Section 2.1 describes the control algorithm, Section 2.2 the bandwidth estimation algorithm used by the control algorithm and section 2.3 summarizes some results on mathematical evaluation of fairness and friendliness of Reno and Westwood+ TCP.

2.1 The algorithm

The Westwood+ algorithm is based on end-to-end estimation of the bandwidth available along the TCP connection path [12],[14]. The estimate is obtained by filtering the stream of returning ACK packets and it is used to adaptively set the control windows when network congestion is experienced. In particular, when three DUPACKs are received, both the congestion window (*cwnd*) and the slow start threshold (*ssthresh*) are set equal to the estimated bandwidth (*BWE*) times the minimum measured round trip time (RTT_{min}); when a coarse timeout expires the *ssthresh* is set as before while the *cwnd* is set equal to one.

The pseudo code of the Westwood+ algorithm is reported below:

```
a)  On ACK reception:
        cwnd is increased accordingly to the Reno algorithm;
        the end-to-end bandwidth estimate BWE is computed;
b)  When 3 DUPACKs are received:
        ssthresh =max(2, $(BWE* RTT_{min})$ / seg_size);
        cwnd = ssthresh;
c)  When coarse timeout expires:
        ssthresh = max(2,$(BWE* RTT_{min})$ / seg_size);
        cwnd = 1;
```

From the pseudo-code reported above, it turns out that Westwood+ additively increases the *cwnd* as Reno, when ACKs are received. On the other hand, when a congestion episode happens, Westwood employs an adaptive setting of *cwnd* and *ssthresh* so that it can be said that Westwood+ follows an *Additive-Increase/Adaptive-Decrease* paradigm [14].

It is worth noting that the adaptive decrease mechanism employed by Westwood+ TCP improves the stability of the standard TCP multiplicative decrease algorithm. In fact, the adaptive window shrinking provides a congestion window that is decreased enough in the presence of heavy congestion and not too much in the presence of light congestion or losses that are not due to congestion, such as in the case of unreliable radio links. Moreover, the adaptive setting of the control windows increases the fair allocation of available bandwidth to different TCP flows. This result can be intuitively explained by considering that the window setting of Westwood+ TCP tracks the estimated bandwidth so that, if this estimate is a good measurement of the fair share, then the fairness is improved. Alternatively, it could be noted that the setting $cwnd = B \times RTTmin$ sustains a transmission rate $(cwnd/RTT) = (B \times RTTmin)/RTT$ that is smaller than the bandwidth B estimated at the time of congestion: as a consequence, the Westwood+ TCP flow clears out its path backlog after the setting thus leaving room in the buffers for coexisting flows, which improves statistical multiplexing and fairness.

2.2 The end-to-end bandwidth estimate

The AIMD algorithm can be viewed as an end-to-end method to obtain a "rough" but robust measurement of the best effort bandwidth that is available along a TCP path.

The first attempt to exploit ACK packets to improve bandwidth estimation is the packet pair (PP) algorithm, which tries to infer the bottleneck available bandwidth at the starting of a connection by measuring the interarrival time between the ACKs of two packets that are sent back to back [23]. Hoe proposes a refined PP method for estimating the available bandwidth in order to properly initialize the *ssthresh*: the bandwidth is calculated by using the least-square estimation on the reception time of three ACKs corresponding to three closely-spaced packets [24]. Allman and Paxson evaluate the PP techniques and show that in practice they perform less well than expected [25]. Lai and Baker propose an evolution of the PP algorithm for measuring the link bandwidth in FIFO-queuing networks [26]. The method consumes less network bandwidth while maintaining approximately the same accuracy of other methods, which is poor for paths longer than few hops. The inaccuracy of the algorithms based on the packet pair approach is due to the fact that the interarrival times between consecutive segments at the receiver can be very different from the interarrival times between the corresponding ACKs at the sender. It will be shown in the following section that this effect is much more significant in the presence of congestion along the ACK path. Jain and Dovrolis propose to use streams of probing packets to measure the end-to-end available bandwidth, which is defined as the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic. The estimate is computed over an averaging interval [36]. Finally, they focus on the relationship between the available bandwidth in a path they measure and the throughput of a persistent TCP connection. They show that the averaged throughput of a TCP connection is about 20-30% more than the available bandwidth measured by their tool due to the fact that the TCP probing mechanism gets more bandwidth than what was previously available in the path, grabbing part of the throughput of other connections. We notice that the latter result is not surprising being a consequence of the fundamental property of the TCP congestion control algorithm for which a new joining TCP flow must get its bandwidth share from existing flows.

A similar technique, based on using streams of probing packets, has been proposed by Melander et. al [37]. It uses sequences of packet pairs at increasing rates and estimates the available bandwidth by comparing input and output rates of different packet pairs.

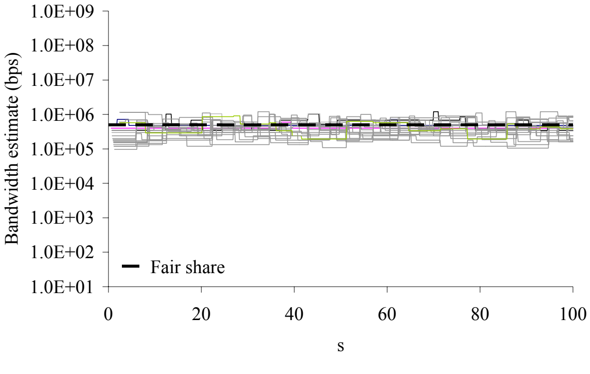
Westwood+ TCP proposes an end-to-end estimate of the "best-effort" available bandwidth by properly counting and filtering the *flow* of returning ACKs [12]. A sample of available bandwidth $b_k = d_k / \Delta_k$ is computed every *RTT*, where d_k is the amount of data acknowledged during the last $RTT = \Delta_k$. The amount d_k is determined by a proper counting procedure that considers delayed ACKs and duplicate ACKs: a duplicate ACK counts for one delivered segment, a delayed ACK for two segments, a cumulative ACK counts for one segment or for the number of segments exceeding those already accounted for by previous duplicate acknowledgements (see [12] for more details on this). Bandwidth samples b_k are low-pass filtered since congestion is due to low frequency components [31], and because of delayed ACK option [13,22]. In [42] the following time-invariant low-pass filter has been proposed as an alternative to the original time-varying filter of Westwood TCP [12]:

$$\hat{b}_k = \alpha \cdot \hat{b}_{k-1} + (1-\alpha) \cdot b_k \qquad (1)$$

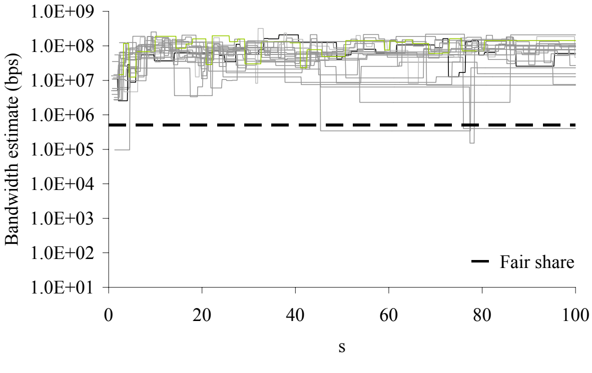
where α is a constant set equal to 0.9. The filter (1) reveals to be particularly suited for kernel code implementation, where floating point operations should be avoided [44].

It should be noted that b_k are samples of *used bandwidth* that *coincide* with the "best-effort" available bandwidth when the connection hits network capacity and experiences losses. Measuring the actual rate a connection is achieving during the data transfer as done by Westwood+ TCP is a different and much easier task than estimating the bandwidth available at the beginning of a TCP connection going over a shared FIFO queuing network.

To give an insight into the bandwidth estimation algorithm, Fig. 1 shows the bandwidth computed at congestion instants by 20 Westwood+ or 20 Westwood flows sharing a 10Mbps bottleneck in the presence of reverse traffic contributed by 10 TCP long lived New Reno connections. Fig. 1(a) shows that all the 20 Westwood+ connections estimate a best-effort available bandwidth that reasonably approaches the fair share of 0.5Mbps. On the other hand, Fig. 1(b) highlights that Westwood overestimates up to 100 times the fair share due to ACK compression.



(a)



(b)

Figure 1. Bandwidth estimates of 20 TCP flows in the presence of ACK compression: (a) Westwood+; (b) Westwood.

To conclude this section we report some considerations on the end-to-end bandwidth estimate of Westwood+ and on the backlog estimate of Vegas. The bandwidth estimate employed by Westwood+ TCP measures the low frequency components of the used bandwidth samples b_k, which coincides with the "best-effort" available bandwidth of a TCP sender at the end of the *slow-start/congestion-avoidance* probing phase. We remark that the estimate (1) is different from measuring the low frequency components of the sending rate *cwnd*/*RTT*, where *cwnd*/*RTT* is the measure of the instantaneous throughput employed by Vegas TCP. In fact, the Vegas actual rate *cwnd*/*RTT* is a measure of the available bandwidth that is based on the number of sent packets (*cwnd*) and not on the number of acknowledged packets d_k. As a consequence, Vegas samples do not take into account that a fraction of sent packets could be lost thus leading to available bandwidth overestimate. To illustrate this point, we simulate a single Westwood+ connection that sends data through a 1Mbps bottleneck link. Fig. 2 shows the bandwidth estimates obtained by filtering the b_k samples or the *cwnd/RTT* samples using the filter (1). Results confirm that the bandwidth estimate obtained by filtering the ACKs is more accurate and less oscillating than the one obtained by filtering the input rate, which, in fact, provides an overestimate of the available bandwidth.

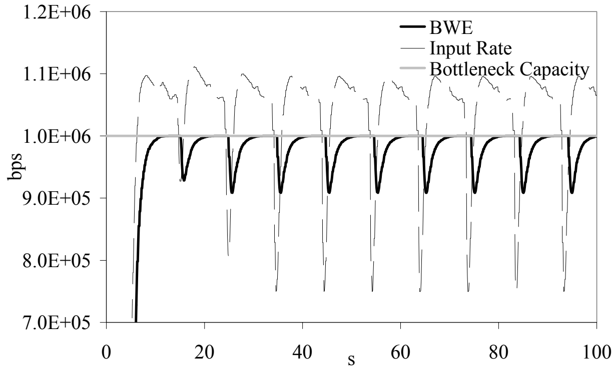


Figure 2. Bandwidth estimate and input rate.

2.3 Mathematical evaluation of fairness and friendliness

This section investigates the intra-protocol fairness in bandwidth allocation provided by Reno and Westwood+ and their inter-protocol *friendliness* using the mathematical models of the Reno and Westwood+ throughputs reported in [14,27,28]. In particular, it has been shown that when the average segment loss probability *p* is low, the Reno throughput is proportional to the inverse of the average round trip time *RTT* and to $1/\sqrt{p}$ as follows [28]:

$$T^{Reno} = \frac{1}{RTT}\sqrt{\frac{2(1-p)}{p}} \qquad (2)$$

Under the same assumptions, it has been shown that Westwood+ TCP provides the following steady state throughput [14]:

$$T^{West} = \frac{1}{\sqrt{RTT \cdot T_q}}\sqrt{\frac{1-p}{p}} \qquad (3)$$

where T_q is the average queuing time equal to the difference between *RTT* and the minimum round trip time RTT_{min}.

By comparing (2) and (3) it results that both throughputs of Westwood+ and Reno depend on $1/\sqrt{p}$, that is Westwood+ and Reno are *friendly* to each other. Moreover, since flows with different *RTTs* experience the same mean queuing time T_q, Eq. (3) shows that the throughput of Westwood+ depends on round trip time as $1/\sqrt{RTT}$ whereas the throughput of Reno as $1/RTT$, that is, Westwood+ increases fair sharing of network capacity between flows with different RTTs. Friendliness and fairness will be investigated through simulations in the next sections to confirm these theoretical results.

3. SIMULATION-BASED COMPARISON

In this section we evaluate and compare Westwood+, New Reno and Vegas TCP using the *ns*-2 simulator [16,38]. Simple scenarios are considered in order to illustrate the fundamental features of the considered protocol dynamics whereas more complex topologies are considered to test the protocols in more realistic settings. In particular, single, multi-bottleneck and mixed wired/wireless scenarios are considered. Each considered scenario is particularly useful to highlight a particular feature of the dynamic behavior of the protocols or to evaluate a specific metric. In all considered scenarios, unless otherwise specified, the timestamp option is enabled; destinations implement the delayed ACK option except when Vegas is used, since its congestion avoidance mechanism is based on RTT measurements [9, 22]. Packets are 1500 Bytes long and buffer sizes are set equal to the link bandwidth times the maximum round trip propagation time unless otherwise specified. The initial congestion window has been set equal to 3 segments [43]. The receiver window size and the initial *ssthresh* are set equal to a value greater than the pipe network capacity so that flows are always network constrained and grab the available bandwidth using slow-start when the connection starts.

3.1 A simple single-connection scenario

In order to analyze the fundamental dynamics of the considered TCP congestion control algorithms, we start by considering the single connection scenario depicted in Fig. 3. The TCP_1 connection is persistent and sends data over a 2Mbps bottleneck link. The *RTT* is 250ms. Ten ON-OFF New Reno TCP senders inject traffic along the ACK path of the single TCP connection, i.e. they generate reverse traffic for the TCP connection on the left side of Fig. 3. Reverse traffic aims at provoking congestion along the ACK path and at exciting ACK compression, which is important to be considered since it exacerbates the bursty nature of the TCP [41].

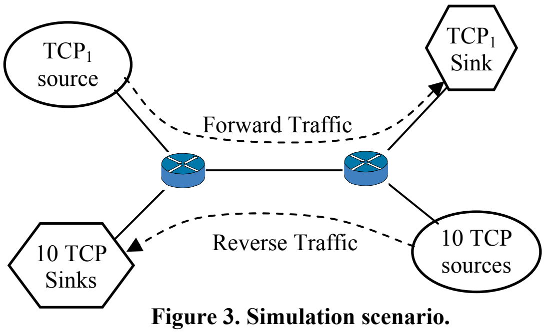


Figure 3. Simulation scenario.

The TCP_1 connection starts at *t*=0. The 10 TCP connections on the backward path follow an OFF-ON-OFF-ON pattern in order to investigate the effect of reverse traffic. In particular, the reverse traffic is ON during the intervals [250s, 500s] and [750s, 1000s] and is silent during the intervals [0s, 250s] and [500s, 750s].

Tables I reports the goodputs that have been measured during each interval.

Table I. Goodputs of a single TCP connection.

	[0s,250s] (Mbps)	[250s,500s] (Mbps)	[500,750s] (Mbps)	[750s,1000s] (Mbps)
New Reno	1.86	1.62	1.99	1.64
Vegas	1.97	0.48	1.97	0.51
Westwood+	1.86	1.68	1.99	1.69

The Goodput has been computed as follows:

$$Goodput = \frac{(sent\_data - retransmitted\_data)}{transfer\_time}$$

When the reverse traffic is OFF, goodputs of all considered TCPs approaches the bottleneck link capacity. However, when the reverse traffic is ON Vegas provides the worst goodput whereas Westwood+ obtains a slightly better goodput with respect to New Reno TCP.

To get a further insight, Figs. 4-5 plot the *cwnd* and the *ssthresh* dynamics. New Reno and Westwood+ TCP achieve a larger *cwnd* during the time intervals when the reverse traffic is off. The reason is that, when the reverse traffic is silent, New Reno and Westwood+ TCP increase the *cwnds* up to the bandwidth delay product, which is roughly 40 segments, plus the bottleneck queue size, which is 40 segments too, before experiencing congestion; this is shown in Figs. 4 and 5 where the *cwnds* of New Reno and Westwood+ systematically reach the value of 80 segments before being reduced by following the Multiplicative Decrease or Adaptive Decrease mechanism, respectively. On the other hand, when the reverse traffic is turned on, both New Reno and Westwood+ can experience a congestion episode as soon as the *cwnd* is larger than the buffer size because of the burstiness of the TCP due to ACK compression. However, it should be noted that the *ssthresh* dynamics of Westwood+ is less sensitive to congestion along the ACK path with respect to New Reno because of the bandwidth estimate used for its setting.

Regarding Vegas, Fig. 6 shows that the *cwnd* is constant and approaches the bandwidth delay product when the reverse traffic is off, thus providing an efficient link utilization. On the other hand, when the reverse traffic is on, the *cwnd* keeps very low values thus preventing link utilization. The reason is that the reverse traffic provokes queuing delays along the backward path, which increases the RTT.

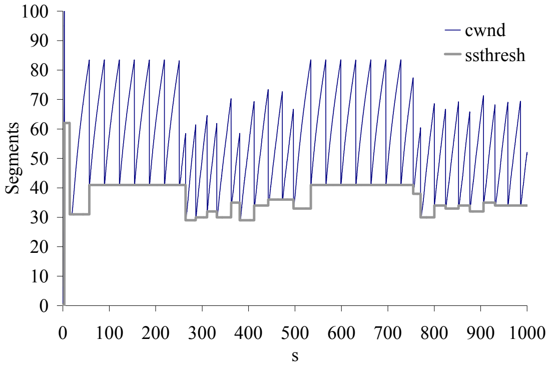


Figure 4. ***cwnd*** **and** ***ssthresh*** **of a single New Reno TCP flow with reverse traffic contributed by 10 New Reno TCP flows.**

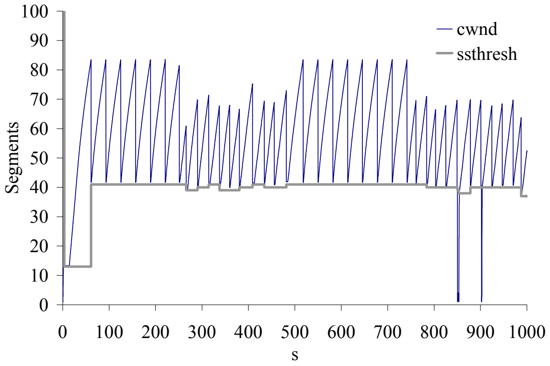


Figure 5. ***cwnd*** **and** ***ssthresh*** **of a single Westwood+ TCP flow with reverse traffic contributed by 10 New Reno TCP flows.**

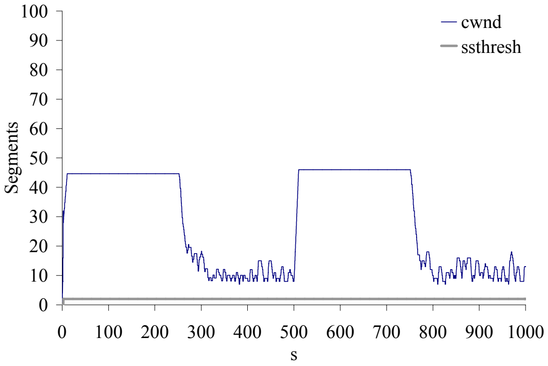


Figure 6. ***cwnd*** **and** ***ssthresh*** **of a single Vegas TCP flow with reverse traffic contributed by 10 New Reno TCP flows**

As a consequence, the Vegas connection on the forward path additively shrinks the *cwnd* thus obtaining a poor utilization of the bottleneck link. We have investigated more this point to see if the kind of used reverse traffic is of importance. Fig. 7 shows that Vegas is not able to grab the bottleneck link also when the reverse traffic is contributed by 10 Vegas connections, that is, Vegas does not provide full link utilization whatever source type of reverse traffic is considered. To complete the comparison we report, for this time only, the behavior of Reno TCP. Fig. 8 shows that Reno is heavily affected by the presence of reverse traffic. Therefore, since now on, we will always consider the New Reno TCP.

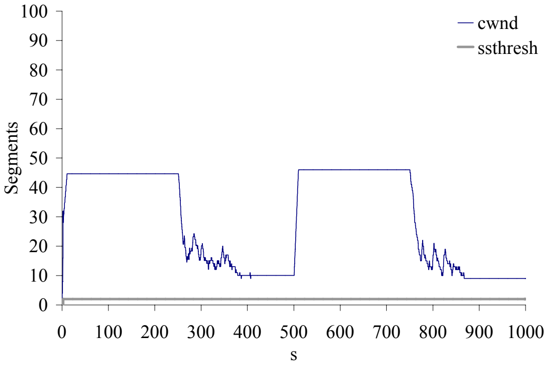


Figure 7. ***cwnd*** **and** ***ssthresh*** **of a single Vegas TCP flow with reverse traffic contributed by 10 Vegas TCP flows.**

Based on the simulations above reported and on others that we do not report here, we can conclude that the reverse traffic heavily affects protocol behaviors. Therefore, the reverse traffic will be always active in all scenarios we will consider in the sequel. Moreover, since we have seen that the effect of reverse traffic does not depend significantly on the TCP control algorithm that generates it, we will consider always New Reno type reverse traffic because it is the more efficient and today used TCP.

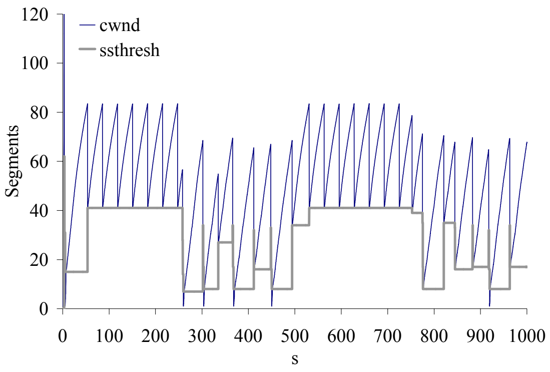


Figure 8. *cwnd* and *ssthresh* of a single Reno TCP flow with reverse traffic contributed by 10 New Reno TCP flows.

We conclude this section by noting that Fig. 4 and 5 show that Westwood+ and Reno exhibit a cyclic behavior that continuously probes for network capacity. The main consequence of this behavior is that, when Westwood+ and New Reno coexist, they are friendly to each other; on the other hand, when Vegas coexists with New Reno or Westwood+ it gives up bandwidth to New Reno or Westwood+ since Vegas lacks of this probing behavior [11].

3.2 Single bottleneck scenario

The scenario depicted in Fig. 11, where M TCP sources with different RTTs share a 10Mbps bottleneck link, is particularly suited for evaluating goodput and fairness in bandwidth allocation. The *M* TCP flows are persistent and controlled by the same algorithm in order to evaluate the intra-protocol fairness. *RTTs* are uniformly spread in the interval [20+230/*M*, 250]ms, with M ranging from 10 to 200, to investigate the fairness with respect to the round trip time. Simulations last 1000s during which all the TCP sources send data. In order to obtain ACK compression, 10 TCP New Reno senders inject traffic along the ACKs path of the *M* connections.

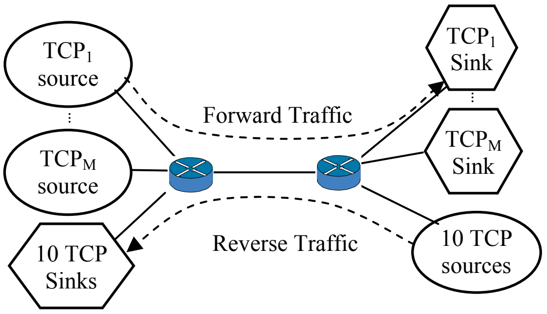


Figure 11. Single bottleneck scenario.

Fig. 12 shows the *total goodput*, which is defined as the sum of the goodputs of all the *M* TCP connections on the forward path. In particular, the figure shows that when M is larger than 40 the total goodput approaches the bottleneck link capacity. On the other hand, when M is smaller than 40, Vegas provides a very low total goodput. Again this phenomenon is due to the TCP traffic on the backward path, which has a significant impact on Vegas TCP (see also Figs. 6, 7).

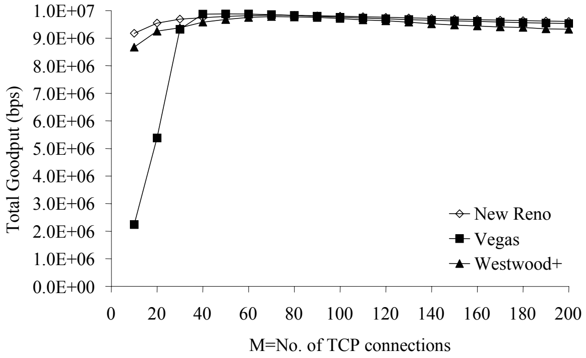


Figure 12. Total goodput of M TCP connections

Fig. 13 plots the Jain fairness index [17]:

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

where b_i is the goodput of the i^{th} connection and *M* are the connections sharing the bottleneck. The Jain fairness index belongs to the interval [0,1] and increases with fairness reaching the maximum value at one.

Fig. 13 shows that Westwood+ improves fairness in bandwidth sharing with respect to New Reno TCP when $M<60$. The reason is that RTTs are uniformly spread over the interval [20+230/*M*, 250]ms so that, for smaller M, RTTs are more distant, which increases the unfair behavior of TCP as stated by Eqs. (2) and (3). Regarding Vegas, it exhibits the best jain fairness index, but with the lowest goodput (see Fig. 12).

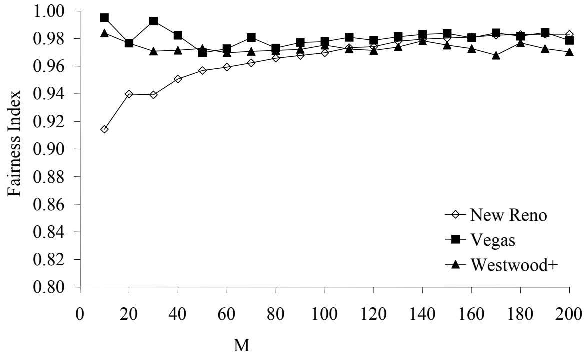


Figure. 13. Jain fairness index over a 10Mbps bottleneck.

To provide a “visual” look into the fairness issue, the sequence numbers of 20 New Reno or 20 Westwood+ or 20 Vegas connections sharing a 10Mbps bottleneck are shown in Figs. 14-16, respectively. Figs. 14 and 15 show that the New Reno final sequence numbers are spread in the interval [26693, 64238] whereas the Westwood+ ones are in the shorter interval [28822, 53423]. Fig. 16 shows that Vegas is fair but provides very low goodput due to the presence of reverse traffic (see Fig. 12).

To summarize simulation results of this section, we can say that both Westwood+ and New Reno achieve full link utilization with Westwood+ providing improved intraprotocol fairness with respect to New Reno. On the other hand, Vegas is fair but it is unable to utilize the network bandwidth.

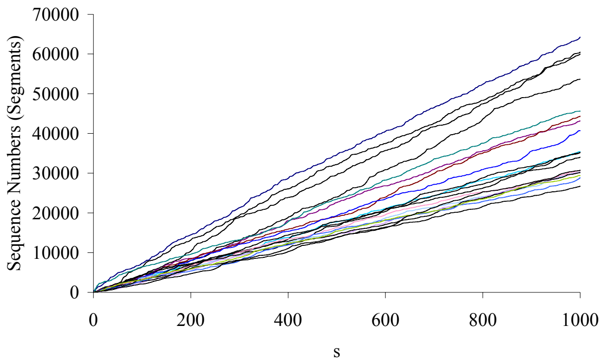


Figure 14. Sequence numbers of 20 New Reno TCP connections

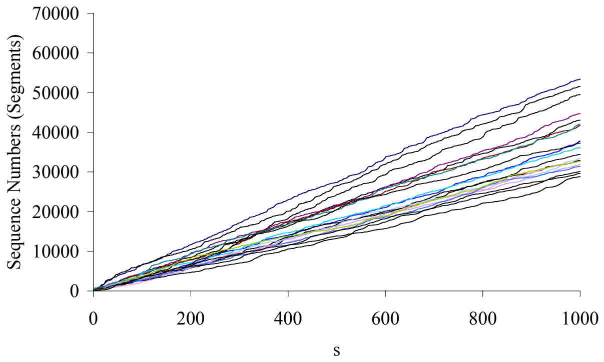


Figure 15. Sequence numbers of 20 Westwood+ TCP.

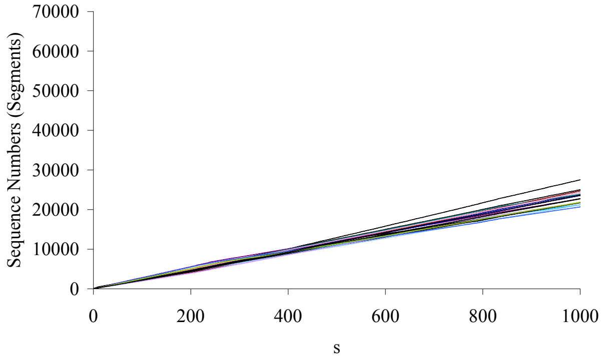


Figure 16. Sequence numbers of 20 Vegas TCP connections.

3.3 Multi bottleneck scenario

The multi-bottleneck scenario is particularly suited to investigate the inter-protocol friendliness of New Reno, Westwood+ and Vegas TCP. The topology depicted in Fig. 17 is characterized by: (a) N hops; (b) one persistent connection C_1 going through all the N hops; (c) $2N$ persistent sources $C_2;C_3;C_4$ ... C_{2N+1} transmitting cross traffic data over every single hop. The capacity of the entry/exit links is 100Mbps with 20ms propagation delay. The capacity of the links connecting the routers is 10Mbps with 10ms propagation delay. Router queue sizes have been set equal to 125 packets, which corresponds to the bandwidth delay product of a typical *RTT* of 150*ms*. Simulation lasts 1000s during which the cross traffic sources are always active. The connection $C1$ is persistent and starts at time $t = 10s$. Notice that the described scenario is a "worst case" scenario for the source C_1 since: (1) C_1 starts data transmission when the network bandwidth has been grabbed by the cross traffic sources; (2) C_1 has the longest *RTT* and experiences drops at each router it goes through.

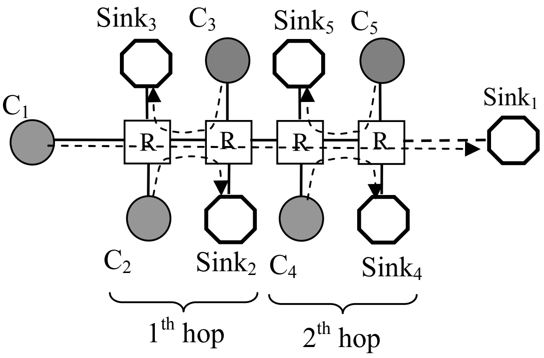


Figure 17. Multi bottleneck topology.

We will consider the following 4 scenarios:

Scenario 1. The C_2,C_3,C_4 ... C_{2N+1} sources of cross traffic are controlled by New Reno TCP whereas the C_1 connection is controlled by New Reno, Vegas or Westwood+, respectively. This scenario aims at comparing New Reno, Vegas or Westwood+, when going through an Internet dominated by New Reno traffic. In other terms, this scenario allows us to investigate the capacity of New Reno, Vegas and Westwood+ to grab network bandwidth when competing with New Reno cross traffic, which is the friendliness of New Reno TCP towards Vegas or Weswtood+ TCP.

Fig. 18 shows the goodput of the C_1 connection as a function of the number of hops; the fair share is 5Mbps. The goodput of the C_1 connection monotonically decreases with the number of hops because of increased loss ratio and *RTT*. Westwood+ roughly achieves the same goodput as New Reno, whereas Vegas is again not able to grab its bandwidth share in a "New Reno environment".

Fig. 19 shows that the total goodput, which is now computed as the *goodput of the C_1 connection + average of the $C_2,C_4..C_{2N}$ connection goodputs*, does not vary significantly with the number of hops. This is due to the fact that the total goodput mainly depends on the behavior of the cross traffic connections $C_2,C_4..C_{2N}$ whereas the C_1 connection has a negligible impact on the total goodput.

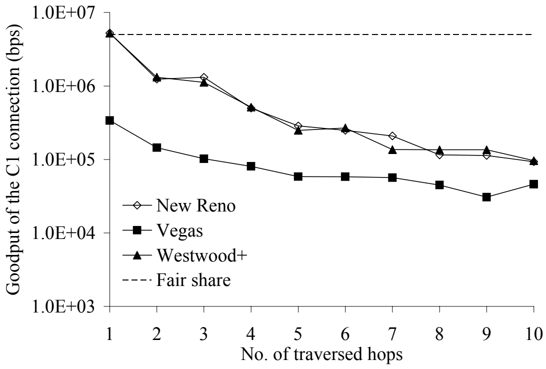


Figure 18. C_1 Goodput vs. number of traversed hops in the presence of New Reno cross traffic.

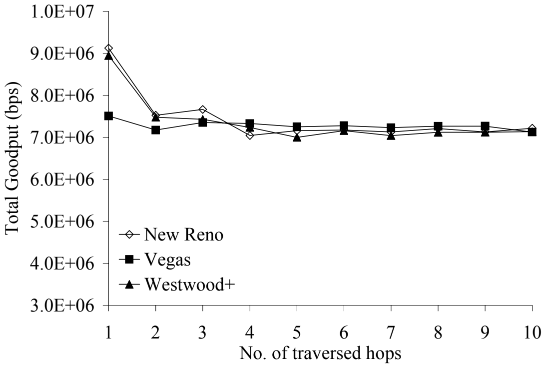


Figure 19. Total Goodput vs. number of traversed hops in the presence of New Reno cross traffic.

Scenario 2. The $C_2;C_3;C_4$ ... C_{2N+1} sources of cross traffic are controlled by Westwood+ TCP whereas the C_1 connection is alternatively controlled by New Reno, Vegas or Westwood+. This scenario allows us to investigate the friendliness of Westwood+ towards New Reno and Vegas TCP. Fig. 20 shows the goodput of the C_1 connection as a function of the number of traversed hops. Again, it shows that Vegas is not able to grab its bandwidth share. Moreover, a comparison of the New Reno curves in Fig. 18 and Fig. 20 shows that the C_1 New Reno achieves a slightly greater goodput when going through Westwood+ cross-traffic than when going through New Reno cross-traffic, that is, Westwood+ is *more than friendly* towards New Reno. Also in this case, the total goodput, which is reported in Fig. 21, does not vary significantly with N.

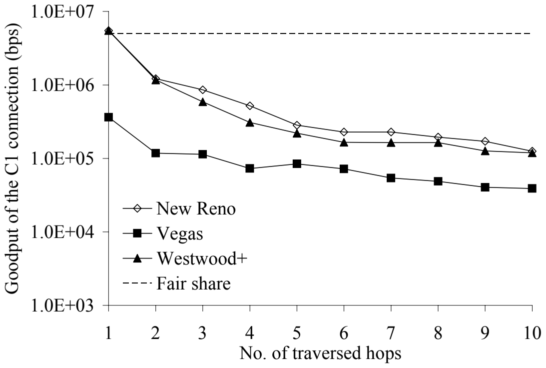


Figure 20. C_1 Goodput vs. number of traversed hops in the presence of Westwood+ cross traffic.

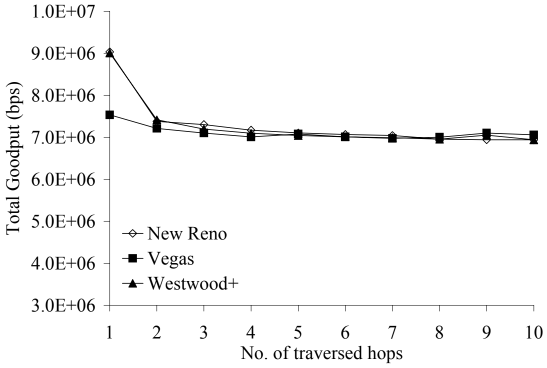


Figure 21. Total Goodput vs. number of traversed hops in the presence of Westwood+ cross traffic.

Scenario 3. The $C_2;C_3;C_4$ ... C_{2N+1} sources of cross traffic are controlled by Vegas TCP whereas the C_1 connection is alternatively controlled by Reno, Vegas or Westwood+. This scenario investigates the friendliness of Vegas towards Reno and Westwood+ TCP. Fig. 22 shows that Reno and Westwood+ basically achieve the same goodput, which is larger than the fair share for any number of traversed hops: the reason is that the Vegas cross traffic, differently from New Reno and Westwood+, avoids to systematically fill the queues thus leaving more room for the C_1 traffic. The total goodput is very low when the C_1 connection is controlled by Vegas, whereas it approaches the 10Mbps link capacity when the C_1 connection is controlled by New Reno or Westwood+ (see Fig. 23). This result again shows that the Vegas cross traffic connections $C_2,C_4..C_{2N}$ are far from providing an efficient utilization of the network capacity.

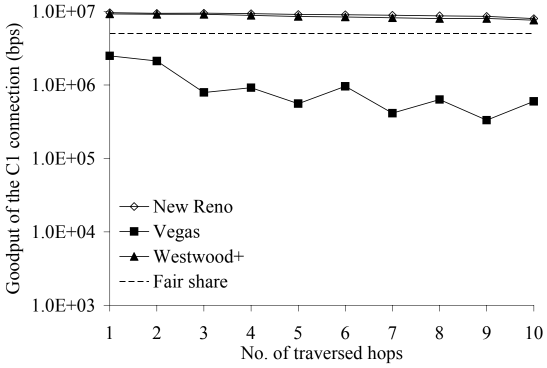


Figure 22. C_1 Goodput vs. number of traversed hops in the presence of Vegas cross traffic.

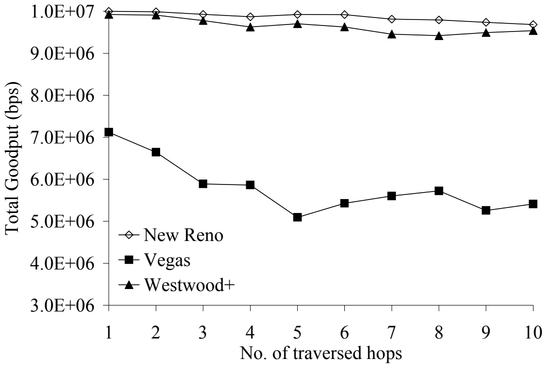


Figure 23. Total Goodput vs. number of traversed hops in the presence of Vegas cross traffic.

Scenario 4. All traffic sources are controlled by the same control algorithm. This is a homogeneous scenario aiming at evaluating New Reno, Westwood+ and Vegas TCP in absolute terms.

Fig. 24 shows that New Reno and Westwood+ provide roughly the same goodput, which monotonically decreases when the number of traversed hops increases, whereas Vegas achieves the highest goodputs when the number of traversed hops is larger than 3 because the Vegas cross traffic is not efficient as Reno or Westwood+. In fact, Fig. 25 shows that the total goodput obtained by using Vegas TCP scenario is much smaller than the total goodputs obtained by New Reno or Westwood+, which means that the Vegas cross traffic do not use the share of link capacity left unused by the C_1 connection.

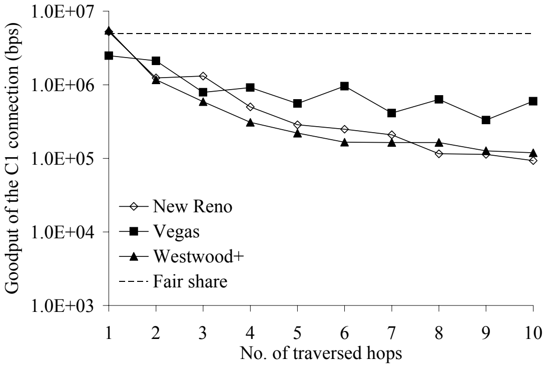


Figure 24. C_1 Goodput vs. number of traversed hops in the presence of homogeneous cross-traffic.

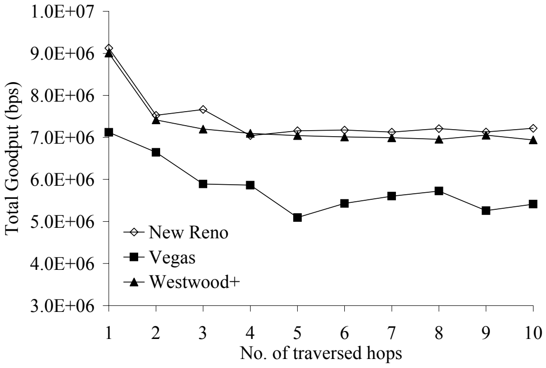


Figure 25. Total Goodput vs. number of traversed hops in the presence of homogeneous cross-traffic.

To summarize, results of this section have shown the inter-protocol friendliness of New Reno and Westwood+ towards each other that is mainly due to the fact that they employs the same probing mechanism. On the other hand, the rtt-based congestion detection mechanism of Vegas TCP turns out an unfriendly behavior of New Reno or Westwood+ towards Vegas, which is not able to fully utilize the network bandwidth. For these reasons, we will not consider Vegas in the sequel of the paper.

3.4 Wireless scenarios

This section aims at investigating the behavior of TCP over wireless links that are affected by losses not due to congestion. This case is particularly interesting since it is well known that protocols that react to losses by multiplicatively decreasing the control windows do not provide efficient utilization of lossy channels. In this scenario we consider Westwood+, New Reno and also TCP SACK to investigate the efficiency of SACK to recover from sporadic random losses. Since TCP SACK by default does not use delayed ACK, we consider Westwood+ and New Reno with delayed ACK (default case) and without delayed ACK in order to get a fair comparison.

3.4.1 Terrestrial scenario

The first scenario we consider is the hybrid wired/wireless topology shown in Fig. 26. The TCP1 connection goes through a wired path terminating with a last hop wireless link. The wireless last hop models a mobile user accessing the Internet using a radio link as in the case of a cellular telephone system. The *one way delay* of the TCP1 connection is 125ms with 20ms delay on the wireless link, which is a 2Mbps link [2]. *RTT*s of the 5 cross traffic connections and of the 10 New Reno backward traffic connections are uniformly spread in the intervals [66ms,250ms] and [46ms,250ms], respectively. We consider a wireless link affected by bursty segment losses in both directions. We use the Gilbert two state Markov chain to model the loss process [10]. In particular, we assume a segment loss probability equal to 0, when the channel is in the *Good state*, and equal to 0.1 when the channel is in the *Bad state*. The permanence time in the *Good state* is assumed deterministic and equal to 1s whereas the permanence time in the *Bad state* is assumed also deterministic but this time we consider values ranging from 0.1ms to 100 ms. When the permanence time in a state elapses, the state can transit to a *Good* or *Bad* state with a probability p=0.5. For each considered case, we run 10 simulations by varying the seed of the random loss process. For each value of the BAD state duration we report the maximum, minimum and average goodputs.

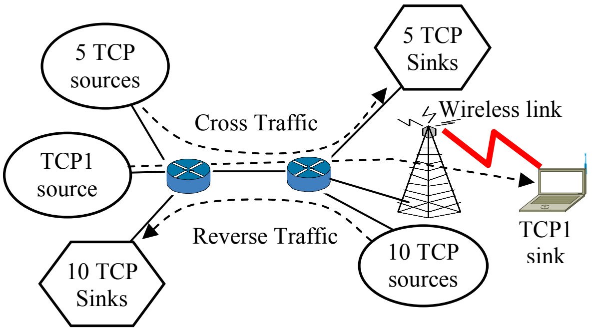
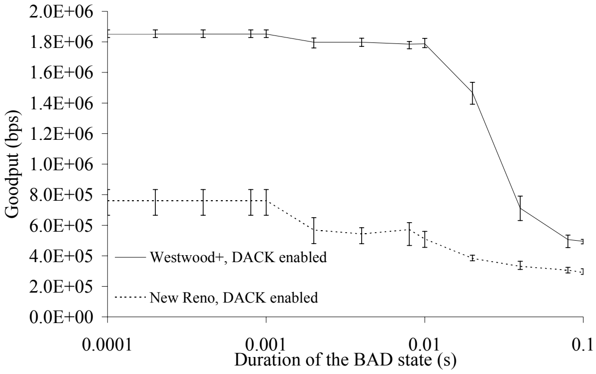
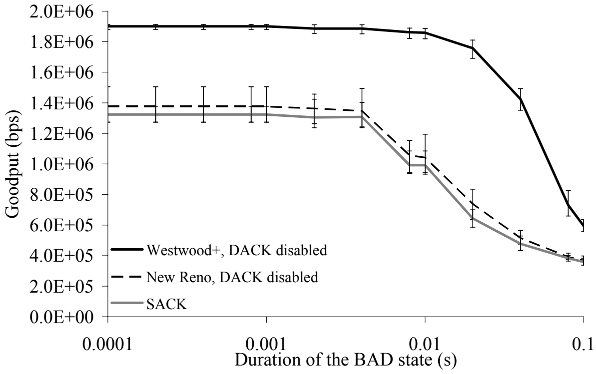


Figure 26. Mixed wired/wireless scenario.

In order to analyze only the impact of bursty losses on the TCP behavior, we have first turned off both the cross and reverse traffic sources. This simple scenario is particularly useful to investigate the effectiveness of the adaptive decrease paradigm when losses not due to congestion are experienced by the TCP. Fig. 27 (a) shows the goodput of TCP1 Westwood+ and New Reno, when the delayed ACK is enabled (default case), as a function of the duration of the BAD state. It turns out that Westwood+ improves the goodput for a large set of channel conditions. In particular, when the delayed ACK option is enabled, Westwood+ increases the utilization of the wireless link from 70% to 230% with respect to New Reno. Fig. 27 (b) shows the goodput of Westwood+, New Reno and TCP SACK when the delayed ACK is disabled (default case for TCP SACK). In this case SACK TCP provides a goodput similar to that of New Reno, whereas Westwood+ improves the link utilization with respect to SACK and New Reno from 34% up to 177%. The reason is that the adaptive setting of *cwnd* and *ssthresh* performed by Westwood+ takes into account the bandwidth used at time of congestion, so that the TCP sender does *not lose ground* when in the presence of losses not due to congestion. To get a further insight into this feature, Figs. 28 and 29 report the *cwnd* dynamics of Westwood+ and New Reno, respectively, obtained when the duration of the BAD state is 0.01s and the delayed ACK is enabled. In this case, Westwood+ TCP provides a *ssthresh* approaching the bandwidth-delay product of 40 packets, whereas New Reno TCP provides a *ssthresh* that is smaller than one fourth the bandwidth-delay product.



(a)



(b)

Figure 27. Goodput of the TCP1 connection without reverse traffic: (a) DACK enabled; (b) DACK disabled.

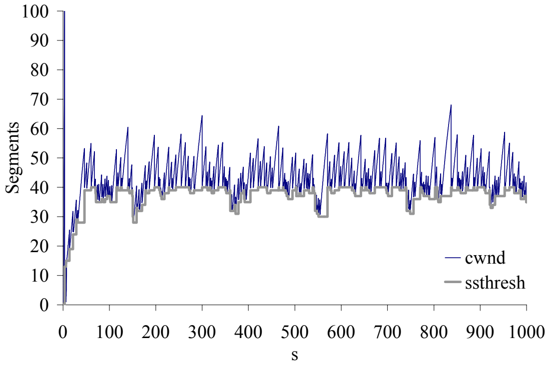


Figure 28. *Cwnd* and *ssthresh* of Westwood+ when the duration of the BAD state is 0.01s.

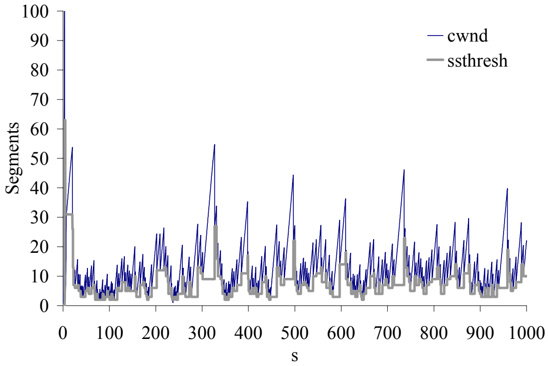


Figure 29. *Cwnd* and *ssthresh* of New Reno when the duration of the BAD state is 0.01s.

One further point valuable of investigation is when Westwood+ shares the wired portion of the network with several TCP flows on the forward and backward paths. For that purpose, we turn on the cross and reverse traffic in Fig. 26 and we measure the goodput of the TCP1 connections for various values of the BAD state duration. Fig. 30 shows that the delayed ACK option plays a major role in this scenario. In fact, protocols that do not employ the delayed ACK option provides goodputs that are roughly two times larger than those obtained when the delayed ACK option is enabled. The reason is that the delayed ACK option slows down the TCP probing phase. In these scenarios Westwood+ TCP (DACK disabled) still improves the goodput with respect to New Reno (DACK disabled) and SACK TCP, but the improvement is now only up to roughly 20%. The reason is that in this case the TCP1 connection loses bandwidth in favor of the cross traffic that, being wired, is not penalized by losses not due to congestion.

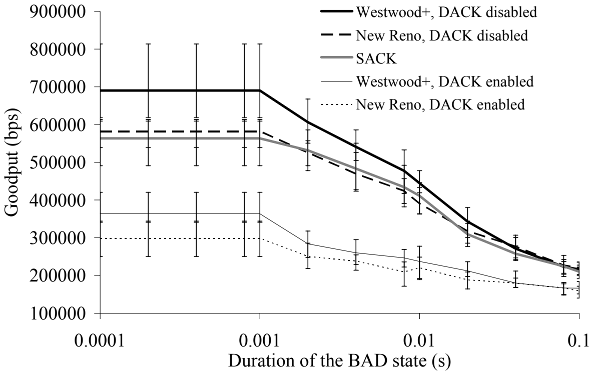


Figure 30. Goodput of the TCP1 connection in presence of cross and reverse traffic.

3.4.2 Satellite scenario

This section investigates the performances of New Reno and Westwood+ over a large leaky pipe such as in the case of a satellite scenario. For that purpose we consider the scenario in Fig. 31 where a 10mbps bottleneck link has a one-way delay equal to 275ms, which corresponds to a GEO satellite connection [32].

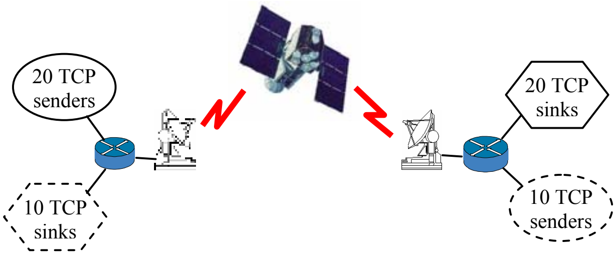


Figure 31. GEO satellite scenario.

We consider 20 TCP forward connections in the presence of reverse traffic contributed by 10 long-lived New Reno connections. RTTs of the forward connections are equal to 590ms. Simulations last 1000s. We assume the same error model used in the previous sub-section except for the BAD state duration that has been increased up to 1s. The bottleneck link experiences segment losses in both directions. Fig. 32 shows the goodput provided by the two considered TCP control algorithms. When delayed ACK is enabled, Westwood+ TCP provides a goodput improvement with respect to New Reno TCP that ranges from 20% to 160%, whereas, when the delayed ACK is disabled, the improvements of Westwood+ with respect to SACK TCP and New Reno are up to 80%. Again, the reason is that Westwood+ adaptively reduces the *cwnd* and *sstresh* by taking into account an estimate of the available bandwidth: this mitigates the impact of random losses not due to congestion that provokes multiplicative reductions of New Reno control windows.

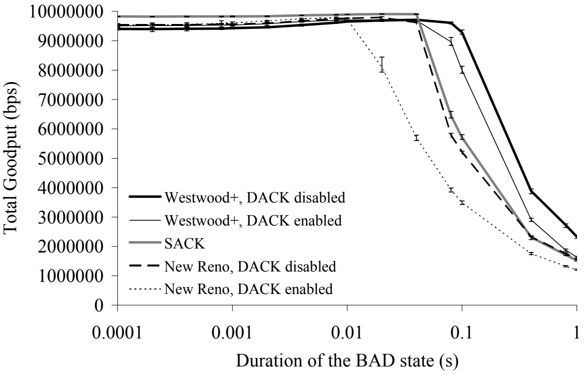


Figure 32. Goodput over the GEO satellite scenario.

4. LIVE MEASUREMENTS

When a new protocol is proposed, it is necessary to collect a large set of simulation and experimental results in order to assess its validity and the advantages of its deployment in the real Internet [29]. In this section we test Linux implementations of New Reno and Westwood+ [44] over the real Internet. More than 4000 files, with different sizes, have been uploaded via ftp from a host at the Laboratory of Communication and Control at Politecnico di Bari (South of Italy) to three remote servers, which are located at Parma (North of Italy), Uppsala University (Sweden) and University of California, Los Angeles (Ucla). For each upload we have measured the goodput, the number of retransmitted segments, and important variables such as *cwnd*, *sshtresh*, *RTT* and *bandwidth estimate*. Each measurements session collects data of many file uploads, which are alternatively executed by using Westwood+ or New Reno. Table 2 summarizes the main characteristics of the sessions from Bari to the FTP server at UCLA, whereas Fig. 33 shows the average goodputs achieved during data transfer.

Table 2: FTP uploads from Politecnico of Bari, Italy to UCLA, Los Angeles.

date	No. of Uploads	Size of uploaded files (MBytes)
Feb,21 2003	117	32
Feb,26 2003	197	3.2
Feb,28 2003	702	3.2
Mar,14 2003	54	32
Mar,19 2003	79	32
Mar,21 2003	47	32

The average goodput of a measurements session is obtained by averaging the goodputs of the session uploads. It turns out that Westwood+ TCP provides goodput improvements ranging from 23% to 53% with respect to New Reno. It is also worth noting that improvements provided by Westwood+ TCP are not due to a more aggressive behaviour, since Fig. 34 shows that Westwood+ and New Reno TCP have similar retransmission ratios.

Table 3 summarizes the characteristics of the measurement sessions from Bari to the server at the Uppsala University. Figs. 35 and 36 show the average goodputs and the average retransmission ratio in this case. Westwood+ TCP provides goodput improvements ranging from 4% to 40% with respect to New Reno with similar retransmission ratios.

Table 3: FTP from Politecnico of Bari, Italy to Uppsala University (Sweden)

date	No. of Uploads	Size of uploaded files (MBytes)
Dec,14 2002	253	32
Dec,17 2002	200	3.2
Jan,10 2003	100	3.2
Jan,12 2003	100	3.2
Jan,13 2003	150	32
Jan,17 2003	1000	3.2
Feb,3 2003	278	32
Feb,7 2003	500	32

Table 4 summarizes the characteristics of the measurement sessions from Bari to the FTP server located at Parma. Figs. 37 and 38 show the goodputs and the retransmission ratios that have been measured during these data transfer. In this case the connection has a national extension and Westwood+ and New Reno TCP provide similar goodputs.

Table 4: FTP uploads towards the server located at Parma (Italy)

date	No. of Uploads	Size of uploaded files (MBytes)
Mar,26 2003	100	32
Mar,27 2003	98	3.2
Mar,28 2003	1000	3.2
Apr,4 2003	390	32
Apr,7 2003	200	3.2
Apr,9 2003	200	3.2
Apr,11 2003	1000	3.2

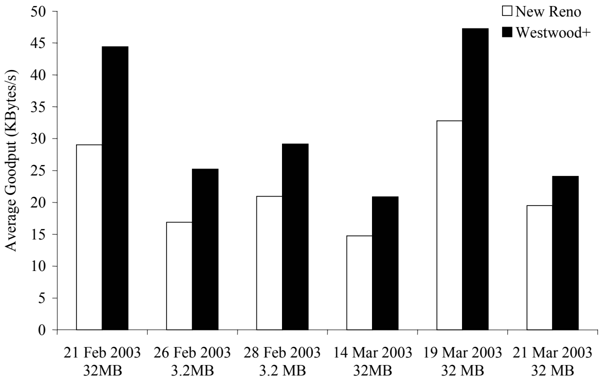


Figure 33. Average goodput during *ftp* to Los Angeles.

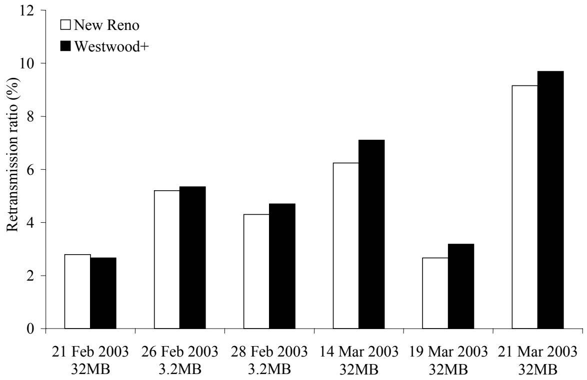


Figure 34. Retransmission ratio during *ftp* to UCLA.

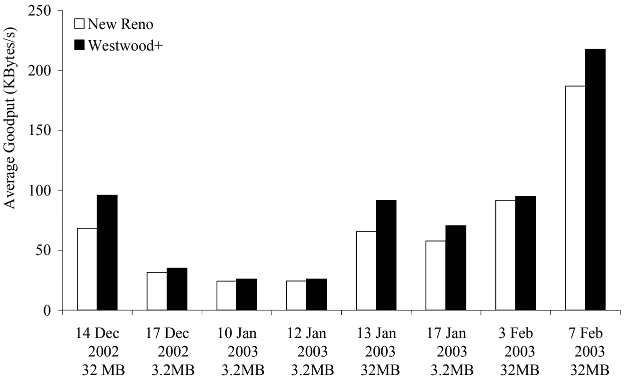


Figure 35. Average goodput during *ftp* to Uppsala.

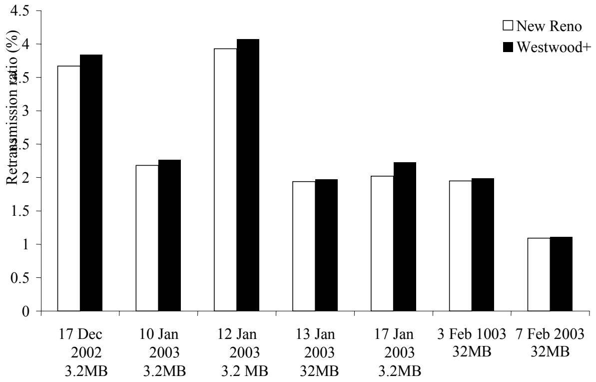


Figure 36. Retransmission ratio during *ftp* to Uppsala.

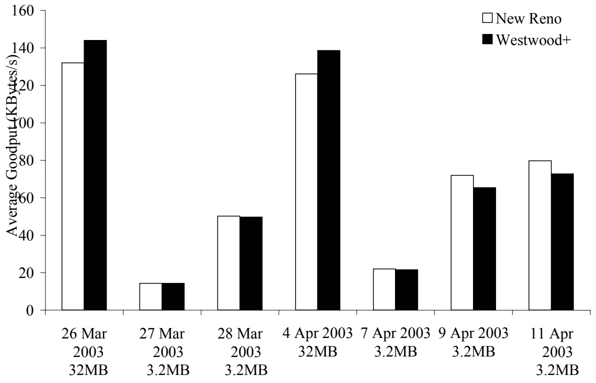


Figure 37. Average goodput during *ftp* to Parma.

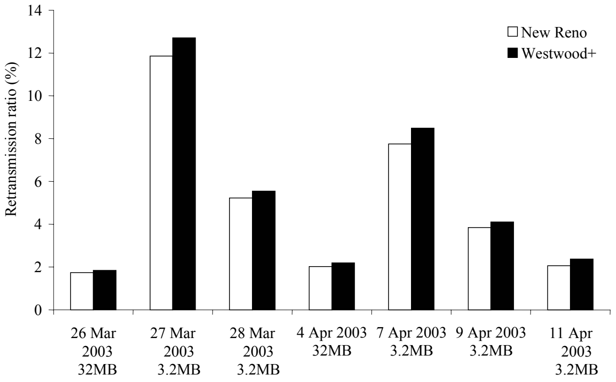


Figure 38. Retransmission ratio during *ftp* to Parma.

To conclude this section, it is worth to take a look at the pipe size of the considered connections. The minimum measured *RTT* has been equal to 190ms, 70ms and 50 ms during transfers to Ucla, Uppsala and Parma, respectively. By computing the average Westwood+ TCP goodput times the minimum RTT, we get the pipe sizes reported in Table 5. This Table shows that when the pipe size is larger than few MSS, Westwood+ improves the goodput with respect to Reno up to 53%. We were not able to execute measurements over larger bandwidth-delay paths where we expect that Westwood+ will provide larger goodput improvements.

Table 5: Performance analysis (MSS=1500Bytes)

Server	Goodput Improvement	Pipe size (MSS)
UCLA	23% ÷ 53%	3-6
Uppsala	4% ÷ 40%	1-10
Parma	–9% ÷ 10%	0.5-5

5. Conclusions

A detailed evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control algorithms has been developed through this paper using the ns-2 simulator. Results have shown: (1) the inter-protocol friendliness of Westwood+ and New Reno whereas Vegas is not able to grab its bandwidth share when coexisting with New Reno or Westwood+; (2) the increased intra-protocol fairness in bandwidth allocation of Westwood+ TCP w.r.t. New Reno; (3) the improved utilization of lossy links provided by Westwood+ wrt New Reno. Finally, measurements collected over the real Internet have shown that Westwood+ improve the goodput with respect to New Reno TCP when the pipe size is larger than few segments.

6. Acknowledgments

We thank anonymous reviewers and John Wroclawski who allowed us to greatly improve the quality of the paper.

7. References

[1] Jacobson, V. Congestion avoidance and control, in Proceedings of ACM SIGCOMM '88 (Stanford CA, August 1988), 314-329.

[2] Krishnan, R., Allman, M., Partridge, C., and Sterbenz, J. P. G. Explicit Transport Error Notification (ETEN) for Error Prone Wireless and Satellite Networks, BBN Technical Report No. 8333, March 22, 2002.

[3] Floyd, S., Henderson, T. New Reno Modification to TCP's Fast Recovery, RFC 2582, April 1999.

[4] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. TCP Selective Acknowledgement Options, RFC 2018, April 1996.

[5] Allman, M., Paxson, V., Stevens, W. R. TCP congestion control, RFC 2581, April 1999.

[6] Dah-Ming Chiu, Jain, R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, 17(1), (1989), 1-14.

[7] Padhye, J., Firoiu, V., Towsley, D., Kurose, J. Modeling TCP Throughput: A Simple Model and its Empirical Validation, in Proceedings of ACM Sigcomm 1998, (Vancouver BC, Canada, September 1998), 303-314.

[8] Morris, R. TCP behavior with Many Flows, in Proceedings of IEEE International Conference on Network Protocols, (Atlanta, Georgia, October 1997), 205-211.

[9] Brakmo, L. S., O'Malley, S.W., and Peterson, L. TCP Vegas: End-to-end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications (JSAC)*,* 13(8), (1995), 1465-1480.

[10] C. Barakat, E. Altman, Bandwidth tradeoff between TCP and link-level FEC, Computer Networks, 39, (2002), pp. 133-150.

[11] Mo, J., La, R. J., Anantharam, V., Walrand, J. Analysis and comparison of TCP Reno and Vegas, in Proceedings of IEEE Infocom 1999, (New York NY, March 1999), 1556-1563.

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

[13] Mascolo, S., and Grieco, L. A., Additive increase early adaptive decrease mechanism for TCP congestion control. *IEEE ICT 2003*, Papeete, French Polynesia, February 2003..

[14] Grieco, L. A., and Mascolo, S., Westwood TCP and easy RED to improve Fairness in High Speed Networks, in Proceedings of IFIP/IEEE Seventh International Workshop on Protocols For High-Speed Networks, PfHSN02, (Berlin, Germany, April, 2002).

[15] *Aström* , K. J. and B. Wittenmark (1997). Computer controlled systems, Prentice Hall, Englewood Cliffs, N. J, 1995.

[16] Ns-2 network simulator (ver 2). LBL, URL: http://www-mash.cs.berkeley.edu/ns.

[17] Jain, R. The art of computer systems performance analysis, John Wiley and Sons, 1991.

[18] Clark, D. The design philosophy of the DARPA Internet protocols, in Proceedings of ACM Sigcomm'88 (Stanford CA, August 1988), 106-114.

[19] Floyd, S., Fall, K. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking, 7(4), (1999), 458-72.

[20] Mascolo, S. Congestion control in high-speed communication networks. Automatica, Special Issue on Control Methods for Communication Networks, vol. 35, no. 12, Dec. 1999, pp. 1921-35.

[21] Lakshman, T.V. and Madhow, U. The Performance of TCP/IP for

Networks with High Bandwidth-Delay Products and Random Loss. IEEE/ACM Transactions on Networking, 5(3), (1997).

[22] Stevens, W. TCP/IP illustrated, Addison Wesley, Reading, MA, 1994.

[23] Keshav, S. A Control-theoretic Approach to Flow Control, in Proceedings of ACM Sigcomm 1991, (Zurich, Switzerland, September 1991), 3-15.

[24] Hoe, J. C. Improving the Start-up Behavior of a Congestion Control Scheme for TCP, in Proceedings of ACM Sigcomm'96, (Palo Alto, CA, August 1996), 270-280

[25] Allman, M. and Paxson, V. On Estimating End-to-End Network Path Properties, in Proceedings of ACM Sigcomm 1999. (Cambridge, Massachusetts, August 1999), 263-276.

[26] Lai, K. and Baker, M. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay, in Proceedings of ACM Sigcomm 2000, (Stockholm, Sweden, August 2000), 283-294.

[27] Padhye, J., and Floyd, S. On inferring TCP behavior, in Proceedings of ACM Sigcomm 2001. (San Diego CA, August 2001).

[28] Kelly, F. Mathematical modeling of the Internet, in Proceedings of the Fourth International Congress on Industrial and Applied Mathematics, (July 1999).

[29] Floyd, S., Paxson, V. Difficulties in simulating the Internet, IEEE/ACM Trans. Networking, 9(4), (2001), 392-403.

[30] Rizzo, L. Dummynet: a simple approach to the evaluation of network protocols, ACM Computer Communication Review, 27(1), (1997), 31-41.

[31] Li, S. Q., and Hwang, C. Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks, IEEE/ACM Transaction on Networking, 3(1), (1995), 10-25.

[32] Akyildiz, I., Morabito, G., and Palazzo S. TCP-Peach: A new Congegestion Control Scheme for Satellite IP Networks. IEEE/ACM Transaction on Networking, 6, (2001), 307-321.

[33] Balakrishnan, H.; Padmanabhan, V.N.; Seshan, S.; Katz, R.H. A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking, 5(6), (1997), 756-769.

[34] Chaskar, H.M.; Lakshman, T.V.; and Madhow, U. TCP Over Wireless with Link Level Error Control: Analysis and Design Methodology, IEEE/ACM Transactions on Networking, 7(5), (1999), 605-615.

[35] Allman, M., Falk, A. On the Effective Evaluation of TCP, ACM Computer Communication Review, 29(5), October 1999.

[36] Jain, M., Dovrolis, C. End to End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput, in Proceedings of ACM Sigcomm 2002.

[37] Melander, B., Bjorkman, M. and Gunningberg, P. A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks, in Proceedings of Global Internet Symposium, 2000.

[38] ns-2 Westwood+ implementation, available at http://www-ictserv.poliba.it/mascolo/tcp%20westwood/modules.htm

[39] Fast AQM Scalable TCP, http://netlab.caltech.edu/FAST/

[40] Communication to the e2e mailing list, from: Steven Low Wednesday, August 06, 2003 8:04 AM Subject: Re: [e2e] Question about FAST TCP.

[41] J.C. Mogul, “Observing TCP dynamics in real networks,” in Proceedings of ACM Sigcomm 1992, 305-317.

[42] Grieco, L. A. , and Mascolo, S., End-to-End Bandwidth Estimation for Congestion Control in Packet Networks. *Second International Workshop, QoS-IP 2003, Milano, Italy, February 2003*.

[43] Allman, M., Floyd, S., and Partridge, C. Increasing TCP's Initial Window, PFC 2414.

[44] Linux Implementation of Westwood+ TCP, available at http://buffer.antifork.org/westwood/westwood.html.