Performance Evaluation of Westwood+ TCP Congestion Control [⋆,⋆⋆]

S. Mascolo [∗], L. A. Grieco, R. Ferorelli, P. Camarda, G. Piscitelli

Dipartimento di Elettrotecnica ed Elettronica,
Politecnico di Bari,
Via Orabona, 4 – 70125 BARI, Italy
{*mascolo;a.grieco;ferorelli;camarda;piscitel*}*@poliba.it*

Abstract

Westwood+ TCP is a sender-side only modification of the classic Tahoe/Reno TCP that has been recently proposed to improve fairness and efficiency of TCP. The key idea of Westwood+ TCP is to perform an end-to-end estimate of the bandwidth available for a TCP connection by properly counting and filtering the stream of ACK packets. This estimate is used to adaptively decrease the congestion window and slow start threshold after a congestion episode. In this way Westwood+ TCP substitutes the classic *multiplicative decrease* paradigm with the *adaptive decrease* paradigm. In this paper we report experimental results that have been obtained running Linux 2.2.20 implementations of Westwood+, Westwood and Reno TCP to ftp data over an emulated WAN and over Internet connections spanning continental and intercontinental distances. In particular, collected measurements show that the bandwidth estimation algorithm employed by Westwood+ nicely tracks the available bandwidth, whereas the TCP Westwood bandwidth estimation algorithm greatly overestimates the available bandwidth because of ACK compression. Live Internet measurements also show that Westwood+ TCP improves the goodput w.r.t. TCP Reno. Finally, computer simulations using *ns*-2 have been developed to test Westwood, Westwood+ and Reno in controlled scenarios. These simulations show that Westwood+ improves fairness and goodput w.r.t. Reno.

Key words: TCP Congestion control, Network emulation, Network measurement, Network simulation.

1 Introduction

Due to the fundamental end-to-end principle [1–3], classic Tahoe/Reno TCP congestion control implements a self-clocked *additive increase multiplicative decrease* (AIMD) sliding window algorithm to probe the network bandwidth capacity [4–7]. It employs two control variables: (1)the *congestion window* (*cwnd*), which limits the number of segments transmitted by the source and not yet acknowledged by the receiver, (i.e. the outstanding segments), and (2) the *slow start threshold* (*ssthresh*), which varies the way the *cwnd* is increased. In order to probe the network capacity, the *cwnd* is increased until a congestion episode happens to indicate that the network capacity has been hit. The way the *cwnd* is increased consists of two phases: (1) the slow-start phase, which exponentially increases the congestion window when $cwnd < ssthresh$, aims at quickly grabbing the available bandwidth; (2) the congestion avoidance phase, which linearly increases the congestion window when $cwnd \geq ssthresh$, aims at gently probing for new available bandwidth. The probing phase ends when the sender becomes aware of congestion via the reception of duplicated acknowledgements (3 DUPACKs) or when a timeout expires. The sender reacts to light congestion (i.e. 3 DUPACKs) by halving the congestion window (fast recovery mechanism) and by re-transmitting the missing packet (fast retransmit mechanism) whereas it reacts to heavy congestion (i.e. timeout) by reducing the congestion window to one. The *ssthresh* is always set equal to $cwnd/2$ after a congestion episode.

The implementation of the mechanisms described above are at the core of today Internet stability [4], [8] . However, while the AIMD paradigm can ensure that the network capacity is not exceeded, it cannot guarantee fair sharing of that capacity[4]. Moreover today TCP is not well suited for lossy links since segment losses due to channel interference and fading are misinterpreted as symptom of congestion thus leading to an undue reduction of the transmission rate and poor utilization of wireless links [9]. To overcome this problem, TCP requires supplementary ARQ and FEC algorithms implemented at the data link layer to efficiently operate over unreliable wireless links [10–12].

TCP Westwood [13] proposes an *additive increase/adaptive decrease* (AIAD) paradigm to enhance the classic AIMD algorithm. The AIAD paradigm leaves unchanged the slow-start and congestion avoidance probing phases and em-

* This work was partially supported by the FIRB Research Project ”Traffic models and Algorithms for Next Generation IP networks Optimization (TANGO)” and by the MIUR Research Project 488/92 Cluster 22 entitled ”E-service to the Citizens and to the Environment”

**This is an extended version of the paper ”Live Internet Measurements Using Westwood+ TCP Congestion Control”, IEEE Globecom 2002 conference.

* Corresponding author

ploys an end-to-end estimation of the available bandwidth to adaptively set the control windows after a congestion episode. TCP Westwood significantly improves fairness in wired networks and utilization of wireless links [13,14].

The first attempt to exploit ACK packets for 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 [15]. Hoe proposes a refined PP method for estimating the available bandwidth in order to properly initialize the *ssthresh* [16]: the bandwidth is calculated by using the least-square estimation on the reception time of three ACKs corresponding to three closely-spaced packets. Allman and Paxson evaluate the PP techniques and show that in practice they perform less well than expected [17]. Lai and Baker propose an evolution of the PP algorithm for measuring the link bandwidth in FIFO-queuing networks [18]. This method consumes less network bandwidth while maintaining approximately the same accuracy of other methods, which is poor for paths longer than few hops. 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 [19]. 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 note that the latter result is not surprising: in fact, a fundamental property of the TCP congestion control algorithm is to allow a new joining TCP flow to grab its bandwidth share from existing flows. A technique that is similar to the one proposed in [19] has been proposed in [20]. 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 tries to measure the actual rate a connection is achieving during the data transfer by properly counting and filtering the stream of returning ACKs [13]. It should be noted that this task is easier than trying to estimate the available bandwidth at the beginning of a TCP connection. Moreover, Westwood TCP does not use out of band probing packets since the additive increase phase probes the network by itself.

In this paper we show that the bandwidth estimation algorithm proposed in [13] does not work properly in real Internet because of ACK compression [21]. In particular, Westwood may greatly overestimate the available bandwidth, which is potentially disruptive of fairness and can lead to starvation of coexisting connections. A slightly modified version of the Westwood bandwidth estimation algorithm is proposed and tested over the real Internet. We call the old Westwood TCP congestion control with the new bandwidth estimation algorithm Westwood+ TCP. A Linux 2.2.20 implementation of Westwood+ has been developed for testing purposes. Measurements collected from an emulated WAN, obtained using a LAN with Dummynet [22], and from live Internet have shown that the bandwidth estimation algorithm used by Westwood TCP overestimates the available bandwidth up to 300% because of ACK compression. On the other hand, Westwood+ nicely tracks the available bandwidth and properly sets the congestion window and slow-start threshold.

To conclude the performance evaluation, computer simulations using *ns*-2 [23] have been also reported to test Westwood+ in controlled experimental conditions.

The paper is organized as follows. Section 2 gives a brief background of Westwood TCP and introduces the new bandwidth estimation algorithm employed by Westwood+ TCP; Section 3 reports emulation and Internet measurements; Section 4 shows computer simulation results obtained using *ns*-2 whereas the last section draws the conclusions.

2 TCP Westwood+

TCP Westwood+ differs from TCP Westwood [13] in that it employs a new bandwidth estimation algorithm that works properly also in the presence of ACK compression. We first summarize the TCP Westwood algorithm and then we introduce the new bandwidth estimation algorithm.

2.1 TCP Westwood

A TCP connection is characterized by the following variables: (1) congestion window (*cwnd*); (2) slow start threshold (*ssthresh*); (3) round trip time of the connection (*RTT*); (4): minimum round trip time measured by the sender (RTT_{min}); (5) size of the delivered segments (*seg_size*).

The key idea of TCP Westwood is to exploit the stream of returning acknowledgment packets to estimate the bandwidth B that is available for the TCP connection. When a congestion episode happens at the end of the TCP probing phase, the used bandwidth corresponds to the definition of best effort available bandwidth in a connectionless packet network. The bandwidth estimate is used to adaptively decrease the congestion window and the slow-start threshold after a timeout or three duplicate ACKs as it is described below:

```
- When 3 DUPACKs are received by the sender:
    ssthresh =(B*RTTmin)/seg_size;
    if ssthresh<2 then ssthresh=2;
    cwnd = ssthresh;
- When a coarse timeout expires:
    ssthresh =(B*RTTmin)/seg_size;
    if ssthresh<2 then ssthresh=2;
    cwnd = 1;
- When ACKs are successfully received:
    cwnd increases following the Reno algorithm.
```

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 Westwood window setting tracks the estimated bandwidth B so that, if this estimate is a good measurement of the fair share, then the fairness is improved. In other terms, it should be noted that the setting $cwnd = B * RTTmin$ after a congestion episode sustains a transmission rate $cwnd/RTT = B * RTTmin/RTT$ that is less than the bandwidth B that is measured at the time of congestion: as a consequence, the TCP flow clears out its path backlog after the setting thus leaving room in the buffers for coexisting and joining flows, which improves statistical multiplexing and fairness. On the other hand, the setting $cwnd = B \cdot RTT$ would produce an input rate that is exactly equal to the used bandwidth, which would lead to an unstable system because an input rate equal to the experienced output rate would not drain the queues. Thus, an increasing RTT due to an increasing backlog would increase the input rate causing a positive feedback that would lead to instability.

A further insight into TCP Reno and Westwood can be obtained by comparing the long-term throughputs of these congestion control algorithms using a mathematical model. In [24,25] it has been shown that when the average segment loss probability $p \ll 1$, the long-term Reno throughput is

$$Througput^{Reno} \propto \frac{1}{RTT \cdot \sqrt{p}} \tag{1}$$

where RTT is the mean round trip time. In [14], the Westwood TCP long term throughput has been derived using arguments a la Kelly [24]. In particular, the following formula has been derived:

$$Througput^{Westwood} \propto \frac{1}{\sqrt{p \cdot RTT \cdot T_q}} \tag{2}$$

where $T_q = RTT - RTTmin$ is the average queuing time. By comparing Eqs. (1) and (2) it can be noted that both Westwood and Reno throughputs depend on $1/\sqrt{p}$, that is Westwood and Reno are friendly to each other. Regarding the fairness, it can be noted that Westwood flows with different $RTTs$ sharing the same bottleneck queue will experience the same queuing time T_q. Therefore, from Eq. (1) it derives 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 $RTTs$.

The Westwood TCP congestion control is heavily based on the bandwidth estimation algorithm. In fact, the correctness of the bandwidth estimate is a key feature in order to provide a properly working adaptive decrease mechanism. In the sequel of the paper we will show that the original Westwood algorithm overestimates the available bandwidth because of ACK compression and we will introduce a modification of the bandwidth estimation algorithm that forms what we have called Westwood+ TCP.

2.2 The Westwood TCP Bandwidth Estimation Algorithm

We have seen that the idea in Westwood TCP is to measure the actual rate a connection is achieving during the data transfer by properly counting and filtering the stream of returning ACKs. At the end of a TCP probing cycle, i.e. when the path capacity is hit and a congestion episode happens, the achieved connection rate is, by definition, the best-effort available bandwidth. This estimate of the available bandwidth is used to set the control windows after congestion. In details, TCP Westwood computes a sample of the available bandwidth $b_k = d_k/(t_k - t_{k-1})$ every time t_k the sender receives an ACK, where $t_k - t_{k-1}$ is the last ACK interarrival time, d_k is the amount of data acknowledged by an ACK, which is determined by a proper counting procedure that considers delayed ACKs, duplicate ACKs and selective ACKs. A duplicate ACK counts for one delivered segments, a delayed ACK for two segments, whereas a cumulative ACK counts for 1 segment or for the number of segments exceeding those already accounted for by previous duplicate acknowledgments (see [13] for details).

Bandwidth samples b_k are filtered using a discrete time-varying low-pass filter that provides the bandwidth estimate as follows:

$$\hat{b}_k = \frac{2\tau - \Delta_k}{2\tau + \Delta_k}\hat{b}_{k-1} + \frac{\Delta_k}{2\tau + \Delta_k}(b_k + b_{k-1}) \tag{3}$$

where Δ_k is the inter-arrival time between the $(k-1)^{th}$ and the k^{th} sample and τ is the cut-off frequency of the filter(a typical value is $0.5s$). Low-pass filtering is necessary since congestion is due to low frequency components [21], and because of delayed ACK option. The filter coefficients are time-varying to counteract the fact that sampling intervals Δ_k are not constant.

2.3 The Westwood+ TCP bandwidth estimation algorithm

The filter (3) overestimates the available bandwidth in the presence of ACK compression provoked by reverse traffic. In fact, a consequence of ACK compression is that bandwidth samples contain high frequency components that are aliased by a low-pass discrete time filter. As a consequence, ACK compression causes a systematic bandwidth overestimate, which may disrupt the fairness between TCP connections and even may lead to starvation of some connections. This effect was not evident in the original paper on TCP Westwood [13] because the phenomena of ACK compression was negligible in the considered scenarios.

In order to avoid the effects of ACK compression, we propose to compute a sample of bandwidth b_k every RTT instead of every time an ACK is received by the sender. More precisely, we count the amount of data D_k acknowledged during the last RTT to compute the bandwidth sample $b_k = D_k/\Delta_k$, where Δ_k is the last RTT. This operation corresponds to evenly spread the bandwidth samples over the RTT interval, that is, it corresponds to filter out high frequency components due to ACK compression. From a signal processing perspective, it corresponds to employ an anti-aliasing filter as it is shown in Fig. 1 [26]. Anti-aliased samples b_k are then low-pass filtered using the filter (3). To conclude, we add a few words on the sampling interval Δ_k. The Nyquist-Shannon sampling theorem requires an interarrival time $\Delta_k \leq \tau/2$. To be conservative, we require $\Delta_k \leq \tau/4$. Thus, if it happens that $\Delta_k > \tau/4$, then we interpolate and re-sample by creating N = integer($4\Delta_k/\tau$) virtual samples that arrive with inter-arrival time $\tau/4$. When $\triangle_k$ is not an exact multiple of $\tau/4$, an additional sample b_k arriving with inter-arrival time $\Delta_T = \Delta_k - 4 \cdot N\tau$ is considered.

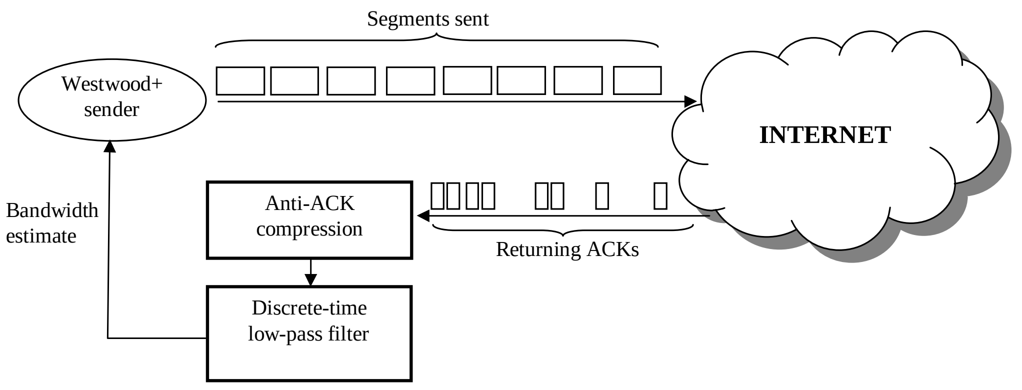


Fig. 1. End-to-end bandwidth estimation framework.

3 Performance Evaluation using a WAN emulated scenario and live Internet Experiments

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 [27]. We have developed a Linux 2.2.20 implementation of Westwood+ and we have set up a set of tests in order to evaluate and compare Westwood+ with respect to classic TCP Reno. Measurements have been collected from a WAN emulated environment and from live Internet. In both cases we have evaluated the goodput as well as the correctness of bandwidth estimation algorithm. Fig. 2 depicts the topology of the first set of experiments where two Ethernet segments are connected by a router, which emulates delays by using Dummynet [22]. Two Linux PCs are connected with the first Ethernet segment and one Linux PC with the second segment.

The second set of measurements has been collected from Internet live experiments. TCP Linux implementations of Westwood, Westwood+ and Reno located at the *Computer, Control and Communication* (C^3) laboratory at the Politecnico di Bari, Italy, have been employed to upload files with different sizes to the *signserv.signal.uu.se* server located at University of Uppsala, Sweden, and to the *panther.cs.ucla.edu* server located at University of California, Los Angeles.

3.1 Bandwidth Estimation

This subsection reports bandwidth estimates obtained by Westwood and Westwood+, respectively. In order to test the bandwidth estimation algorithm, Dummynet has been configured to provide a 500Kbytes/s link between the two Ethernet segments with a 100ms round trip time. An FTP transfer from

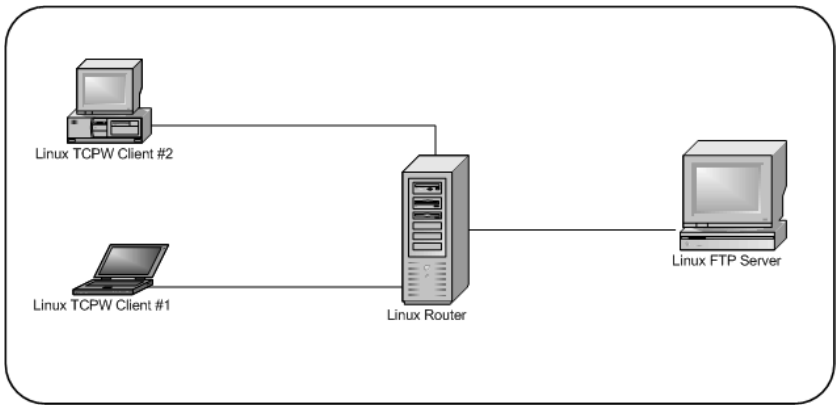


Fig. 2. Emulating a WAN using a LAN and Dummynet emulator .

PC#1 to the FTP server shows that both Westwood and Westwood+ nicely estimate the link capacity when there is no ACK compression (see. Fig. 3). In order to evaluate Westwood and Westwood+ in the presence of ACK compression, another FTP transfer has been launched along the reverse path going from the FTP server to the PC#2. Fig. 4(a) shows that the bandwidth estimation algorithm used by TCP Westwood greatly overestimates the available bandwidth up to 300%. On the other hand, Fig. 4(b) shows that Westwood+ nicely tracks the 500Kbytes/s channel capacity (the slight difference is due to the presence of protocols headers).

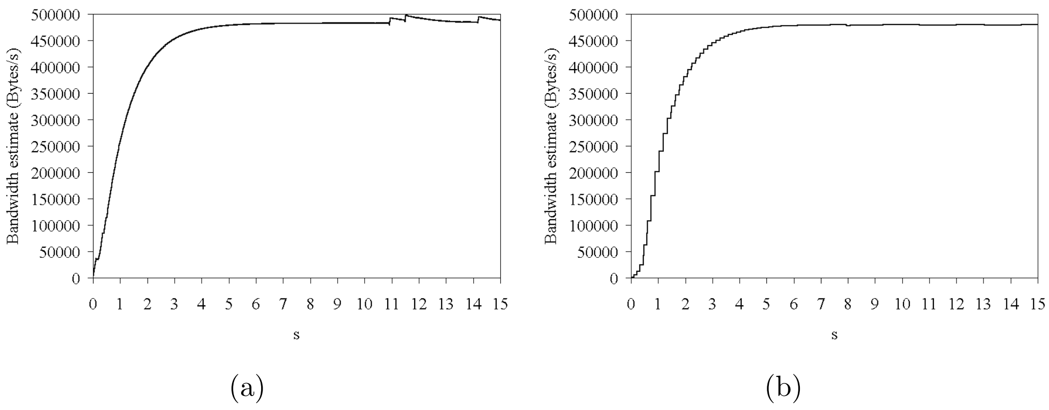


Fig. 3. Bandwidth estimates without ACK compression in the emulated scenario: (a) Westwood; (b) Westwood+.

Figs. 5 and 6 plot the bandwidth estimates obtained using Westwood and Westwood+ during two ftp uploads to UCLA, Los Angeles executed in the same day. Figs. 7 and 8 show analogous data obtained during two ftp uploads to Uppsala University, Sweden. The thick line shown in Figs. 5-8 represents the average goodput at the end of the transfer. It is a useful reference line because a good bandwidth estimate must be close to the average goodput. It can be noted that also in the case of live Internet measurements, TCP Westwood overestimates the available bandwidth up to 400%, whereas Westwood+ nicely tracks the actual used bandwidth.

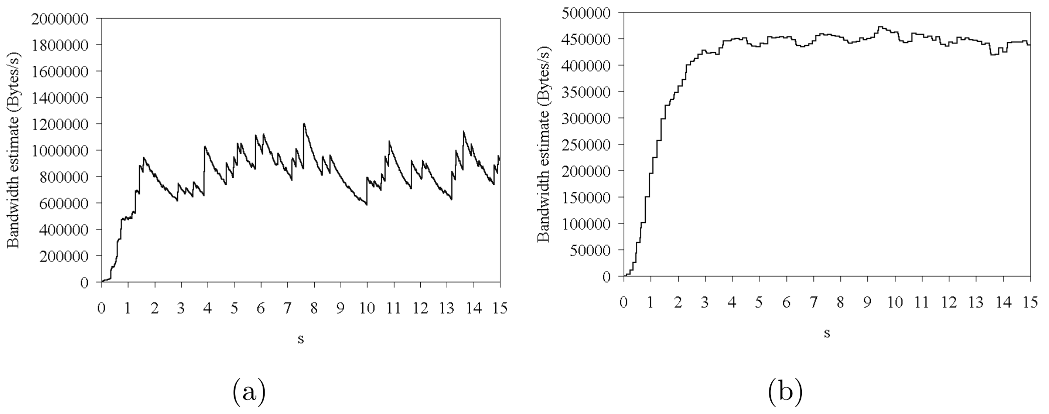


Fig. 4. Bandwidth estimates in the presence of ACK compression in the emulated scenario: (a) Westwood; (b) Westwood+.

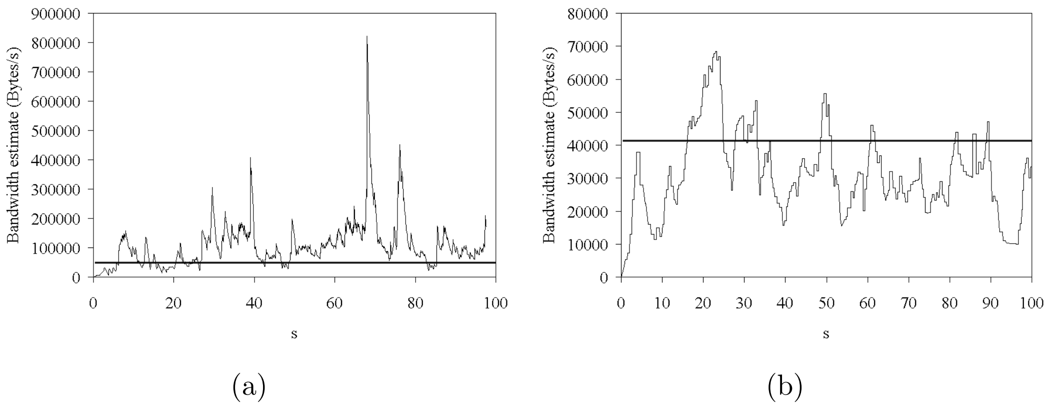


Fig. 5. Bandwidth estimate during the first ftp to UCLA, Los Angeles, California: (a) Westwood; (b) Westwood+.

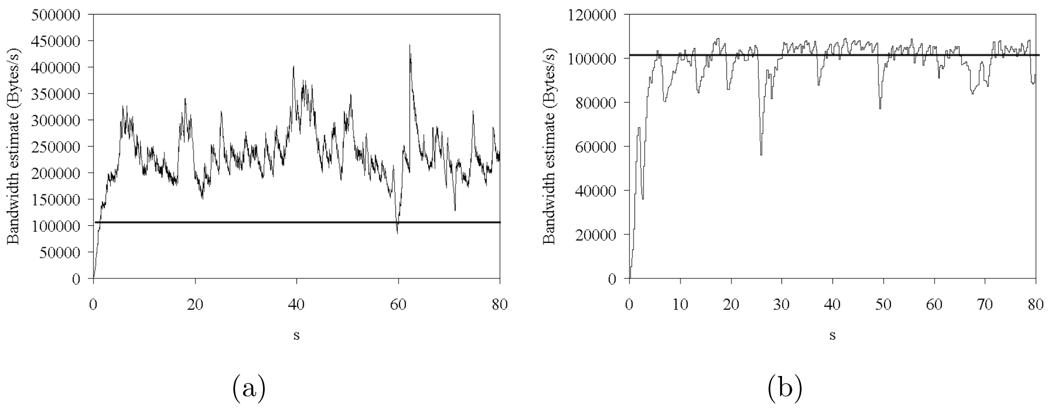


Fig. 6. Bandwidth estimate during the second ftp to UCLA, Los Angeles, California: (a) Westwood; (b) Westwood+.

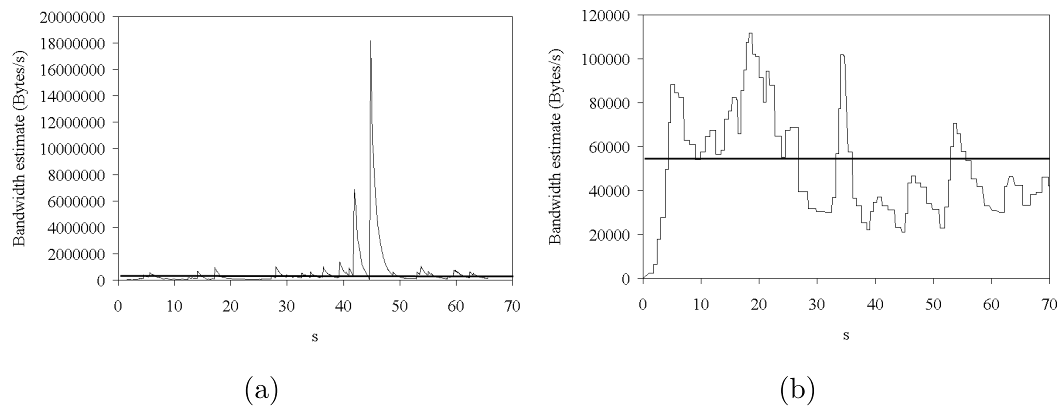


Fig. 7. Bandwidth estimate during the first ftp to Uppsala University, Sweden: (a) Westwood; (b) Westwood+.

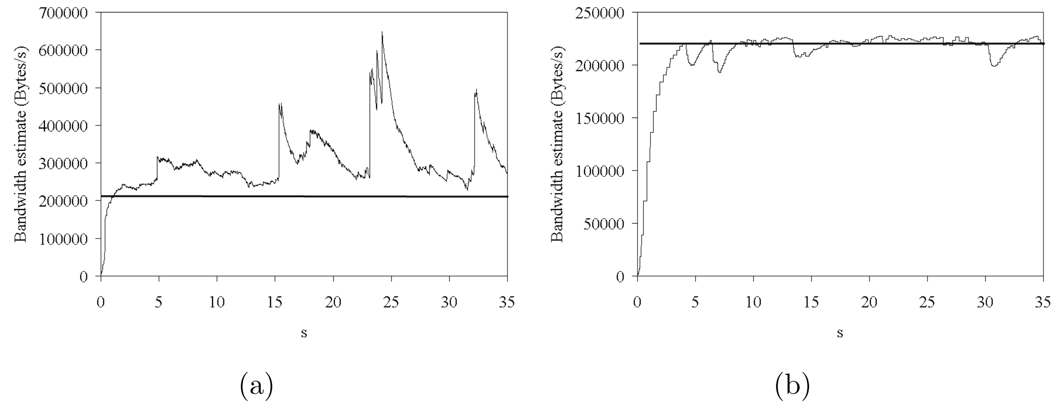
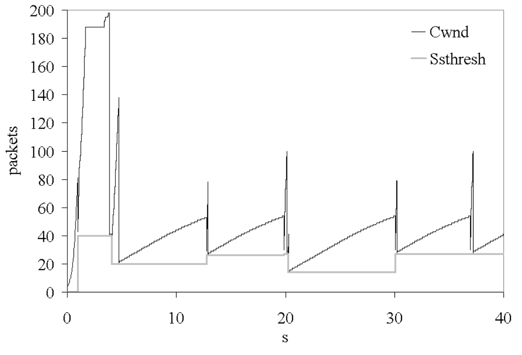


Fig. 8. Bandwidth estimate during the second ftp to Uppsala University, Sweden: (a) Westwood; (b) Westwood+.

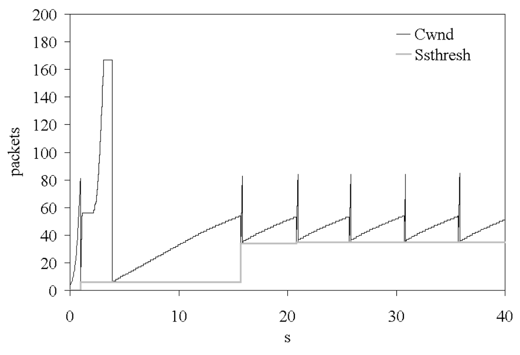
3.2 Congestion Window and Slow Start Threshold

In order to get a further insight into the behaviors of Westwood+ and Reno TCP, Fig. 9 plots the *cwnd* and the *ssthresh* dynamics during the FTP transfer in the WAN emulated scenario depicted in Fig. 2. After a short transient, TCP Westwood+ sets the *ssthresh* equal to a constant value that is greater than the corresponding value set by Reno: this setting provides a more efficient use of network bandwidth.

Similar results have been shown by Internet measurements. Fig. 10 plots the *cwnd* and the *ssthresh* of Reno and Westwood+ during an FTP transfer to UCLA, Los Angeles, whereas Fig. 11 shows analogous data during an FTP transfer to Uppsala University, Sweden.

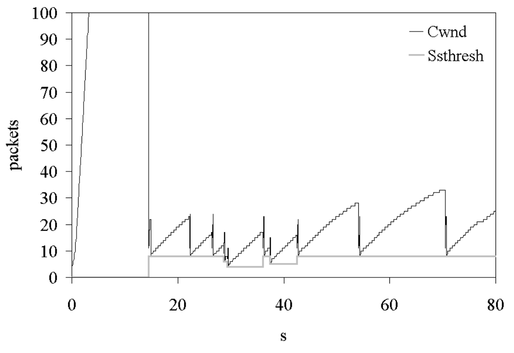


(a)

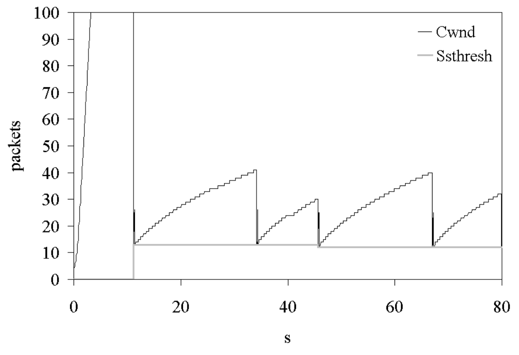


(b)

Fig. 9. *cwnd* and *ssthreh* during the ftp in the emulated scenario in the presence of ACK compression: (a) Reno; (b) Westwood+.

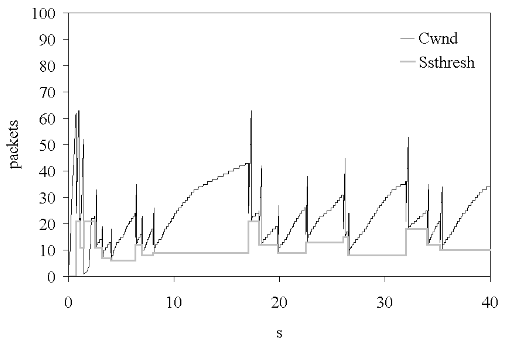


(a)

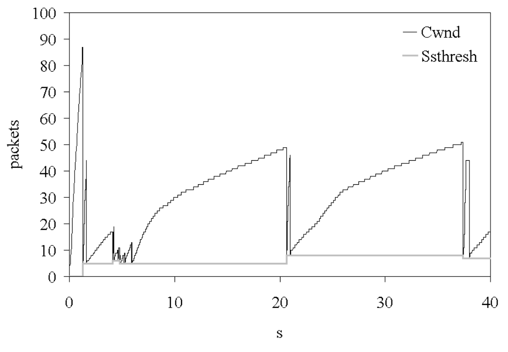


(b)

Fig. 10. *cwnd* and *ssthresh* during the ftp transfer to UCLA, Los Angeles: (a) Reno; (b) Westwood+.



(a)



(b)

Fig. 11. *cwnd* and *ssthresh* during the ftp transfer to Uppsala University, Sweden: (a) Reno; (b) Westwood+.

3.3 Goodput

To compare the efficiency of Westwood+ with respect to Reno TCP, we have repeated 50 file uploads to the server in Los Angeles and to the server in Uppsala. Moreover we have used different file sizes to investigate the impact of the transient time on the goodput. The goodput has been measured as $File\_size/transfer\_time$. The average values of the goodput achieved in each series of 50 uploads are reported in Fig. 12 for the server in Los Angeles, and in Fig. 13 for the server in Uppsala. The last column represents the average goodput over all the transfers. TCP Westwood+ provides goodput increment ranging approximately from 10% to 46% with respect to Reno.

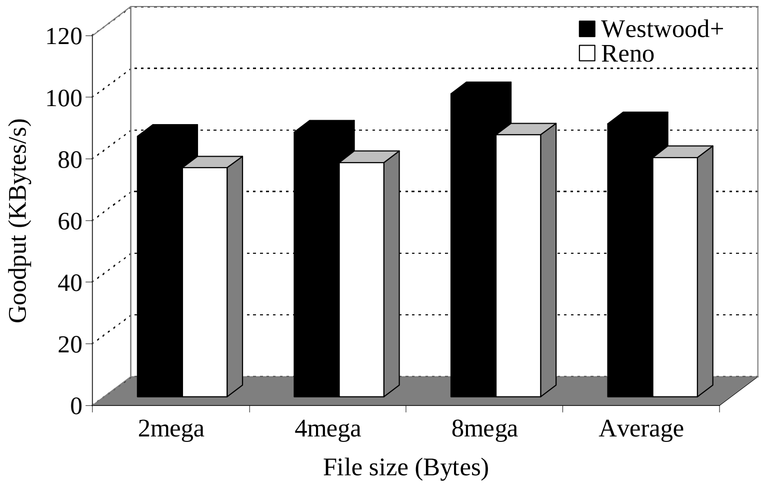


Fig. 12. Average goodput during ftp transfers to UCLA, Los Angeles.

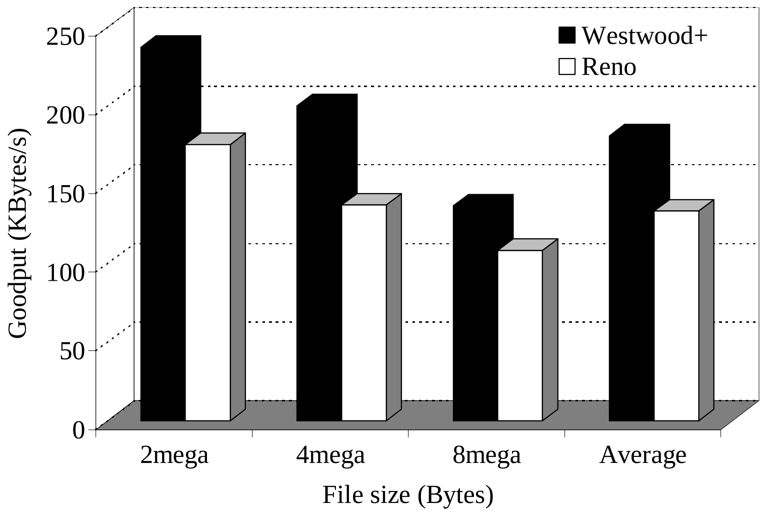


Fig. 13. Average goodput during ftp transfers to Uppsala University, Sweden.

4 Ns-2 simulation results

Testing a protocol is a complex and delicate issue [27]. A major advantage of using simulations is that experimental conditions can be controlled and results carefully analyzed. In this section Westwood+ is evaluated and compared with Reno using the *ns*-2 simulator [23]. Single and multi-hop scenarios with wired as well as wireless links are considered. Sender and receiver socket sizes are set large enough so that connections are limited always by network capacity. Segments are 1500 bytes long and,unless otherwise specified, routers buffer capacities are set equal to the bandwidth delay product as recommended in [28]. When the bandwidth delay product is less than 10 segments, the buffer capacity is set equal to 10 segments. Simulations last 1000s unless otherwise specified.

4.1 A single TCP connection scenario

In order to clearly compare the behavior of different TCP algorithms, we start by considering the single connection scenario depicted in Fig. 14 ($N = 1$), where a persistent TCP source, with $RTT = 250ms$, transmits data over a bottleneck link. We investigate the impact of buffer capacity and random losses on the goodputs of Westwood+ and Reno.

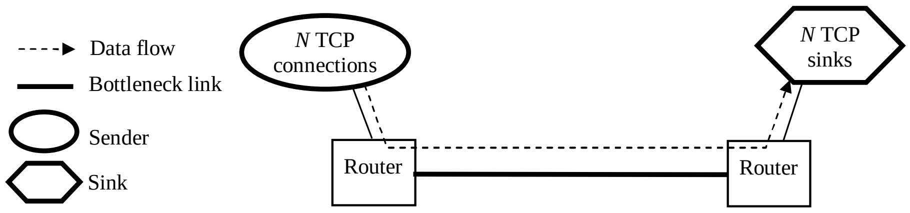


Fig. 14. Single bottleneck scenario.

4.1.1 The impact of the buffer size

To investigate the impact of the bottleneck buffer size on the performance of Reno and Westwood+ TCP, the scenario in Fig. 14 has been considered where the bottleneck link capacity has been set equal to 10Mbps and the bottleneck buffer size has been varied from 0.1 to 1.5 times the *bandwidth-delay product.* Fig. 15 shows the goodput achieved by Reno and Westwood+ TCP.

Westwood+ achieves a goodput that is roughly constant for all the considered buffer sizes and close to full link utilization. On the other hand the goodput of Reno depends on the buffer size. In particular, the goodput of Reno increases with the bottleneck buffer size and achieves the 90% of the link capacity

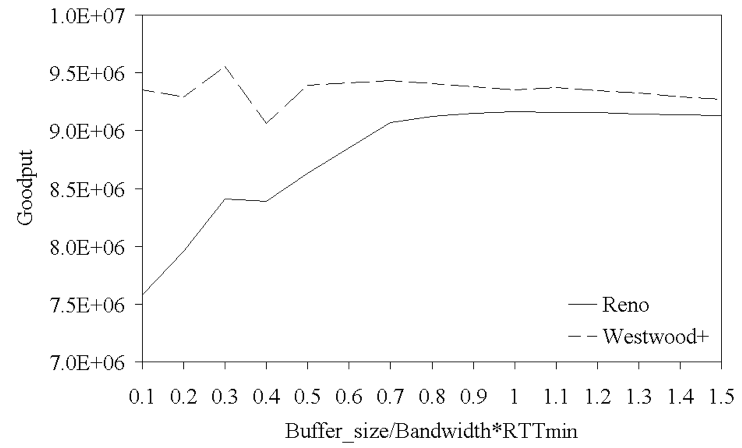


Fig. 15. Goodput as a function of the bottleneck buffer size.

when the buffer size is 0.7 times the bandwidth-delay product. For smaller buffer sizes, TCP Reno is not able to provide a satisfying link utilization. An explanation of this effect is the following: by letting C denote the bottleneck capacity, Q the buffer size and $RTTmin$ the round trip propagation time without queuing delays, it results that Reno inflates the *cwnd* up to the value $C \cdot RTTmin + Q$, when a congestion episode happens. At this point Reno TCP reduces by half the *cwnd* to the value $0.5 \cdot C \cdot RTTmin + 0.5 \cdot Q$. If Q is less than $C \cdot RTTmin$, then the setting of *cwnd* is less than $C \cdot RTTmin$, which means that the *cwnd* is not big enough to keep the pipe full and provide full link utilization. On the other hand, the *cwnd* setting of Westwood+ does not depend on the bottleneck buffer size because of the bandwidth estimation mechanism that sets $cwnd = C \cdot RTTmin$ for any value of Q.

4.1.2 The impact of link losses

To evaluate the impact of random segment losses on the goodputs of Reno and Westwood+, we assume that the bottleneck link is affected by uniformly distributed random losses ranging from 0.1% to 1% in both directions. The bottleneck link capacity is varied from 100Kbps to 100Mbps. Fig. 16 shows that, when the bottleneck link is not lossy, both Reno and Westwood+ achieves full link utilization. This confirms results in Fig. 15, since we are assuming a buffer capacity equal to the bandwidth delay product. On the other hand, Fig. 17 shows that Westwood+ improves the goodput with respect to Reno TCP in the presence of losses not due to congestion such as in the case of lossy links.

To give a further insight into the improvements provided by Westwood+ in the presence of lossy links, Fig. 18 (a) and (b) show the *cwnd* and *ssthresh* variables of Westwood+ and Reno, respectively, in the case of a 4Mbps link affected by 0.4% loss rate. A comparison of Figs. 18 (a) and (b) shows that

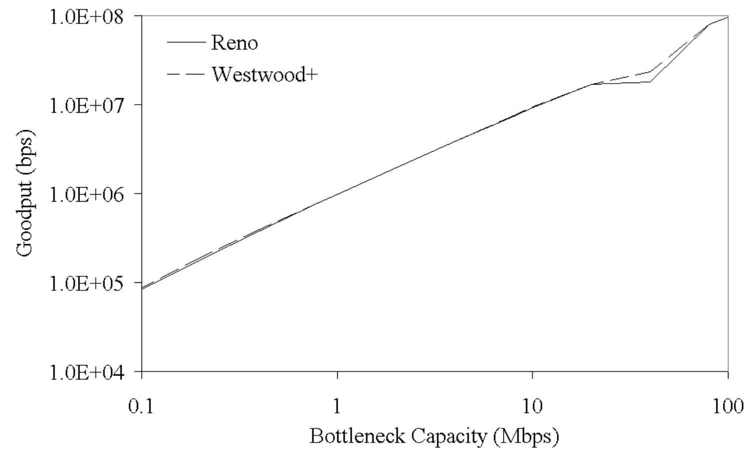


Fig. 16. Goodput of Westwood+ and Reno in the presence of reliable link.

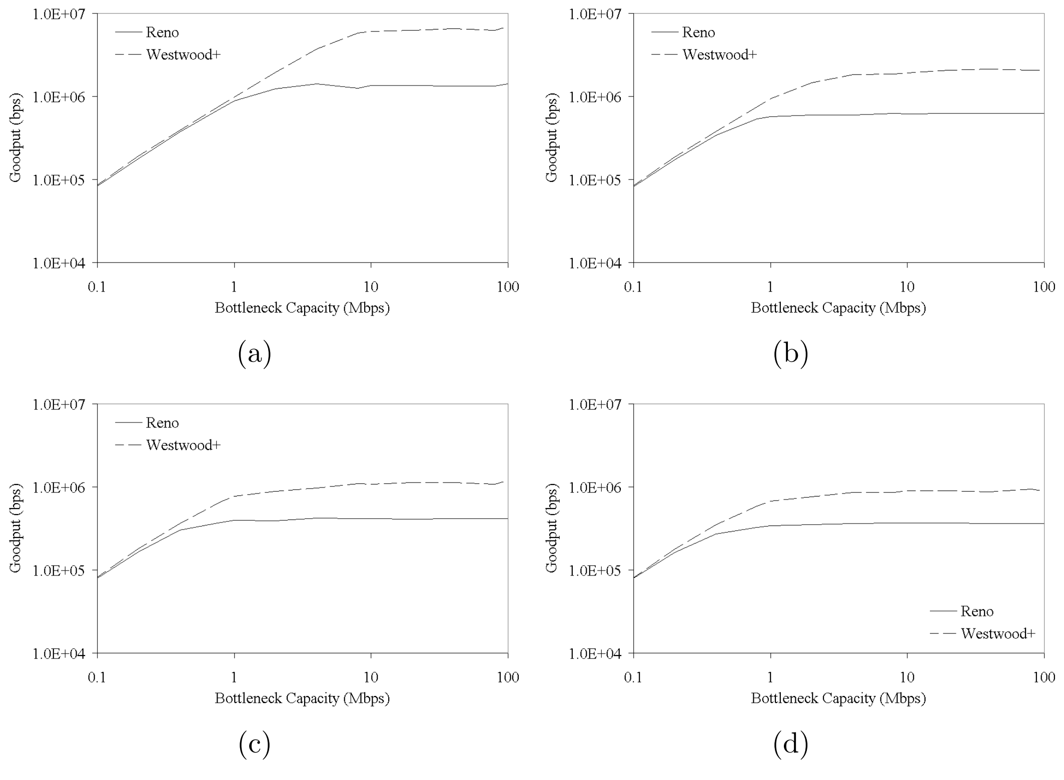
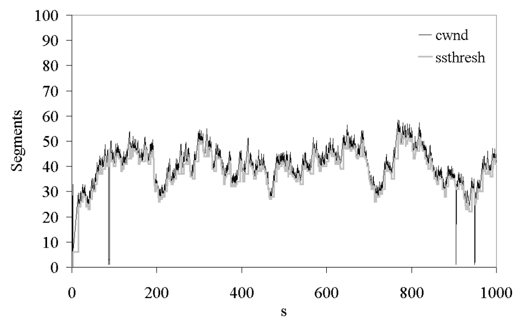
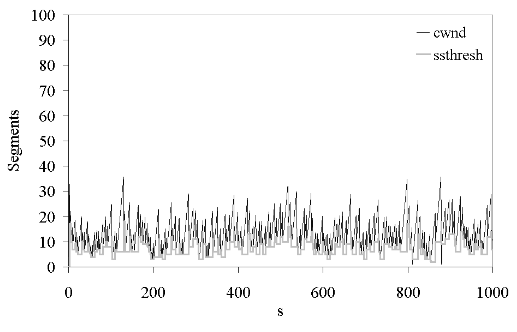


Fig. 17. Goodput of Westwood+ and Reno in the presence of lossy link with: (a) 0.1% loss rate; (b) 0.4% loss rate; (c) 0.8% loss rate; (d) 1% loss rate.

Westwood+ provides control windows that are roughly four times larger than those of Reno.



(a)

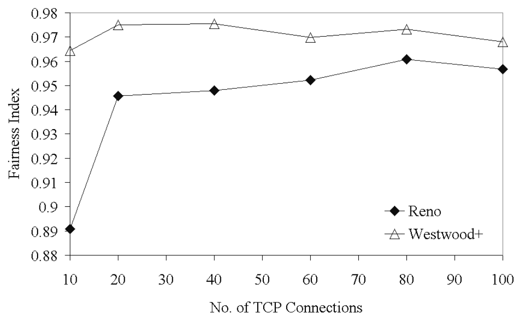


(b)

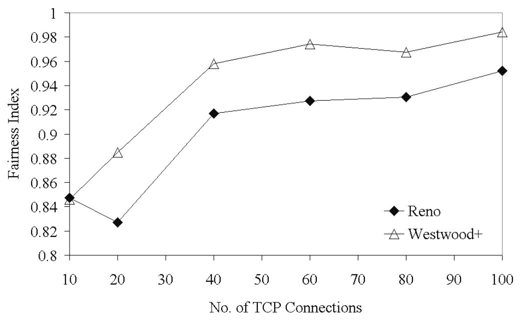
Fig. 18. *cwnd* and *ssthresh* in the presence of a lossy bottleneck link with 0.4% loss rate: a) Westwood+ TCP; b) Reno TCP.

4.2 Single Bottleneck scenario

In order to evaluate how a set of N TCP flows of the same flavor with different RTTs share the capacity of a common bottleneck, we consider N=10,20,40,60, 80,100 infinite greedy connections sharing a FIFO bottleneck with RTTs ranging uniformly from $(10+240/N)ms$ to 250ms (see Fig. 14). To provide a single numerical measure reflecting the fair share distribution across the various connections we use the Jain fairness index defined as $F.I. = \dfrac{(\sum_{i=1}^{N} b_i)^2}{N \cdot \sum_{i=1}^{N} {b_i}^2}$, where b_i is the throughput of the i^{th} connection [29]. The fairness index belongs to the interval $[0, 1]$; a unitary index represents maximum degree of fairness. Simulation results have shown that both Westwood+ and Reno achieve high utilization of the bottleneck link because the buffer size has been set equal to the *bandwidth delay product* (see Fig. 15). Figs. 19(a) and (b) show the fairness index of Westwood+ and Reno when the bottleneck capacity is equal to 10Mbps and 100Mbps, respectively.



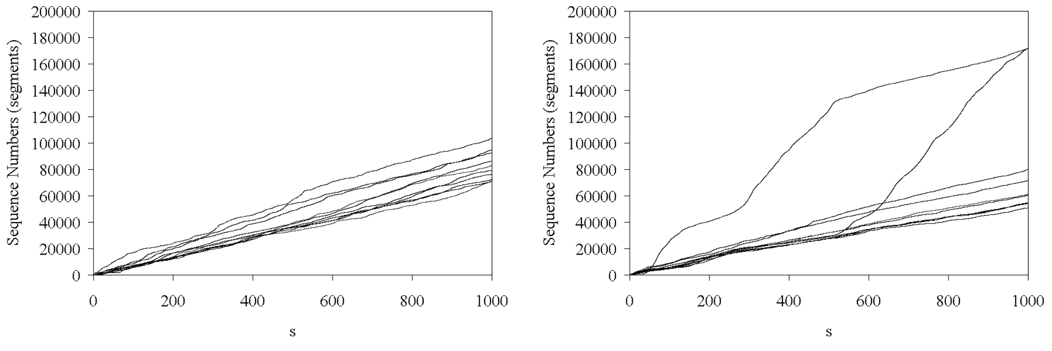
(a)



(b)

Fig. 19. Fairness Index of Reno and Westwood+ over a single bottleneck of capacity: a) 10Mbps; b) 100Mbps.

To give a visual explanation of what the fairness index represents, Fig. 20 (a) plots the sequence numbers of 10 TCP Westwood+ flows and Fig. 20 (b) the sequence numbers of 10 Reno connections sharing a 10Mbps bottleneck link. These figures show that the sequence numbers of the Reno connections are much more spread than those of Westwood+ connections, meaning that Westwood+ is fairer than Reno.



(a) (b)

Fig. 20. Sequence numbers vs. time of 10 TCP connections sharing a 10Mbps bottleneck: a) 10 Westwood+ connections; b) 10 Reno connections.

Finally, to give a further insight into the fairness issue and its correlation with the bandwidth estimation mechanism, Fig. 21 shows the bandwidth estimates obtained by 10 Westwood+ flows sharing a 10Mbps bottleneck. The estimates oscillate around the fair-share that is equal to 1Mbps, thus showing that the end-to-end bandwidth estimation algorithm improves the ability of TCP congestion control to allocate the bandwidth in a fair way.

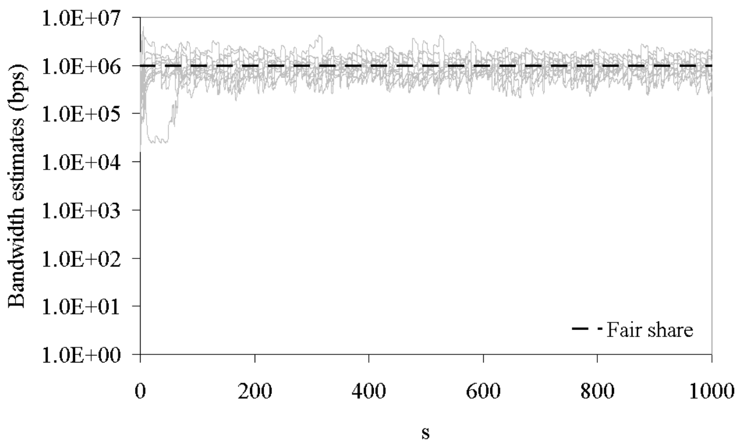


Fig. 21. Bandwidth estimates of 10 TCP Westwood+ flows.

4.3 Multiple congested gateways

We consider the multihop topology depicted in Fig. 22, which is characterized by: (a) N hops; (b) one greedy TCP connection C_1 going through all the N

hops; (c) $2N$ cross traffic greedy TCP sources $C_2 - C_{2N+1}$ transmitting data over each single hop. The simulation lasts 1000s during which the C_1 connection always sends data. Each cross traffic source $C_2 - C_{2N+1}$ starts randomly in the interval [0, 500]s, and stop randomly in the interval [500, 1000]s. The round trip time of the C_1 connection is 250ms, whereas round trip times of the $C_2 - C_{2N+1}$ connections are equal to 50ms. The link capacity between routers is equal to 10Mbps. The capacity of entry/exit links is 100Mbps. Fig. 23 shows the goodput achieved by Westwood+ and Reno TCP as a function of the number of traversed hops: Westwood+ achieves a sligthly better goodput w.r.t. Reno for any considered number of hops.

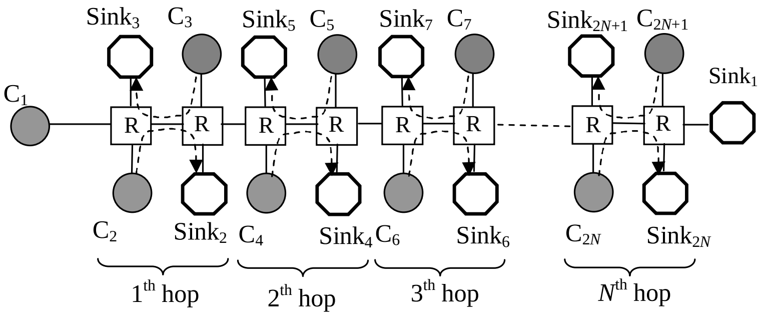


Fig. 22. Multibottleneck topology.

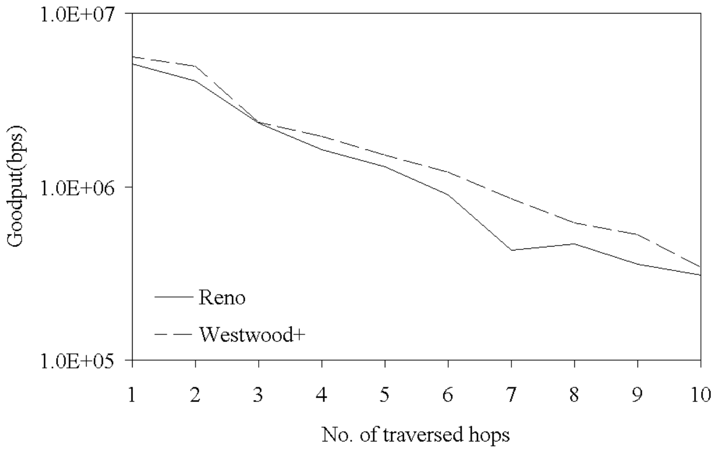


Fig. 23. Goodput over the multihop topology as a function of the number of traversed gateways.

4.3.1 A multihop scenario with a wireless lossy link

In this section we compare Westwood+ and Reno in the presence of unreliable links such as in the case of radio links. The topology shown in Fig. 22 is considered where $N = 10$ hops, and the last hop is a 2Mbps lossy link. We assume that the lossy link is affected by an independent uniform packet loss probability ranging from 0.1% to 1% in both link directions. Fig. 24 shows that Westwood+ provides a higher goodput of the C_1 connection w.r.t Reno TCP for any value of the drop rate.

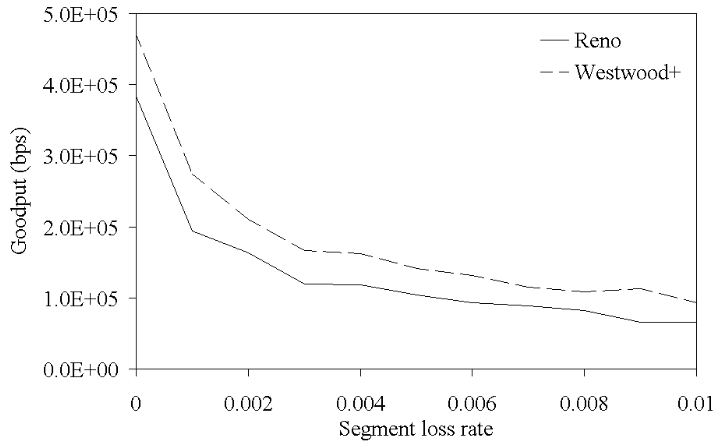


Fig. 24. Goodput over a 10 hop topology with a last hop wireless link.

5 Conclusion

In this paper we have tested the behavior of the TCP Westwood+ algorithm using WAN emulation, live Internet measurements and ns-2 simulations. Collected results have shown that the new bandwidth estimation algorithm employed by Westwood+ nicely estimates the connection available bandwidth whereas the estimation algorithm used by Westwood overestimates the available bandwidth in the presence of ACK compression. Collected results have shown that Westwood+: (a) slightly (significantly) improves the goodput with respect to Reno in wired (wireless) scenarios; (b) significantly improves fairness in bandwidth allocation.

6 Acknowledgements

We thank Profs. Mario Gerla and Mikael Sternad who allowed us to collect live Internet measurements using servers located at their Universities in Los Angeles, and Uppsala, respectively.

References

[1] D. Clark. The design philosophy of the darpa internet protocols. In *ACM Sigcomm '88*, pages 106–114, Stanford, CA, USA, August 1988.

[2] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[3] S. Mascolo. Congestion control in high-speed communication networks using the smith principle. *Automatica, Special Issue on Control methods for communication networks*, 35:1921–1935, December 1999.

[4] V. Jacobson. Congestion avoidance and control. In *ACM Sigcomm '88*, pages 314–329, Stanford, CA, USA, August 1988.

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

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

[7] L. Peterson, B. Davie, and D. Clark. *Computer Networks a Systems Approach*. Morgan Kaufmann, 2 edition, October 1999.

[8] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, June 1989.

[9] H. Balakrishnan, V. N. Padmanabhan, and S. Seshan andR. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, December 1997.

[10] C. Barakat and E. Altman. Bandwidth tradeoff between TCP and link-level FEC. *Computer Networks*, 39(2):133–150, June 2002.

[11] H. M. Chaskar, T. V. Lakshman, and U. Madhow. TCP over wireless with link level error control: Analysis and design methodology. *IEEE/ACM Transactions on Networking*, 7(5):605–615, October 1999.

[12] R. Krishnan, M. Allman, C. Partridge, and J. P. G. Sterbenz. Explicit transport error notification (ETEN) for error prone wireless and satellite networks. Technical Report 8333, BBN Technologies, March 2002.

[13] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP westwood: End-to-end bandwidth estimation for efficient transport over wired and wireless networks. In *ACM Mobicom 2001*, Rome, Italy, July 2001.

[14] L. A. Grieco and S. Mascolo. Westwood TCP and easy RED to improve fairness in high speed networks. In *IFIP/IEEE Seventh International Workshop on Protocols For High-Speed Networks*, pages 130–146, Berlin, Germany, April 2002.

[15] S. Keshav. A control-theoretic approach to flow control. In *ACM Sigcomm '91*, pages 3–6, Zurich, Switzerland, September 1991.

[16] J. C. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *ACM Sigcomm '96*, pages 270–280, Stanford University, CA, USA, August 1996.

[17] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *ACM Sigcomm '99*, pages 263–276, Harvard University Science Center, Cambridge, Massachusetts, USA, August 1999.

[18] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *ACM Sigcomm '00*, pages 283–294, Stockholm, Sweden, August 2000.

[19] M. Jain and C. Dovrolis. End to end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *ACM Sigcomm 2002*, Pittsburgh, PA, USA, August 2002.

[20] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *IEEE Global Telecommunications Conference*, pages 415–420, San Francisco, CA, November 2000.

[21] J. C. Mogul. Observing TCP dynamics in real networks. In *ACM Sigcomm '92*, pages 305–317, Baltimore, MD, USA, August 1992.

[22] L. Rizzo. Dummnynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27:31–41, January 1997.

[23] Ns-2. network simulator, 2002.

[24] F. Kelly. Mathematical modeling of the internet. In *Fourth International Congress on Industrial and Applied Mathematics*, Edinburgh, July 1999.

[25] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM Sigcomm '98*, pages 303–314, Vancouver BC, Canada, 1998.

[26] L. A. Grieco and S. Mascolo. End-to-end bandwidth estimation for congestion control in packet networks. In *Second International Workshop, QoS-IP 2003*, pages 645–658, Milano, Italy, February 2003.

[27] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transaction on Networking*, 9:392–403, August 2001.

[28] C. Villamizar and C. Song. High performance TCP in ANSNET. *ACM Computer Communication Review*, 24(5):45–60, 1995.

[29] R. Jain. *The Art of Computer Systems Performance Analysis Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley and Sons, April 1991.