Linux 2.4 Implementation of Westwood+ TCP with rate-halving: A Performance Evaluation over the Internet *

A. Dell'Aera, L. A. Grieco, S. Mascolo

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

Abstract: The additive increase/multiplicative decrease probing paradigm is at the core of TCP congestion control and allows TCP flows to regulate the transmission rate in order to match the network capacity. To improve classic Reno/New Reno congestion control algorithms, the recent Westwood+ TCP proposes to substitute the multiplicative decrease phase with an adaptive decrease phase: in particular, Westwood+ TCP passes through a filter the stream of returning ACKs to obtain an estimate of the used bandwidth (BWE), so that when a congestion episode happens, Westwood+ adaptively reduces the congestion window and the slow-start threshold to the value BWE times the minimum round trip time.

This paper aims at evaluating the performance of Westwood+ TCP over the real Internet. To this purpose, a Linux 2.4.19 implementation of Westwood+ TCP has been developed and compared with an implementation of New Reno. To test and compare the two implementations more than 4000 files, with different sizes, have been uploaded via ftp from a host at the Politecnico of 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, California). Experimental results indicate that Westwood+ TCP improves the goodput with respect to New Reno TCP over paths with a bandwidth delay product larger than few segments. In particular, goopdput improvements up to 40-50% have been measured when transmitting data from Politecnico of Bari to Uppsala or UCLA servers.

*This work has been partially supported by the FIRB Research Project "Traffic models and Algorithms for Next Generation IP networks Optimization (TANGO)"

1 Introduction

The *additive increase/multiplicative decrease* probing paradigm is at the core of the sliding window TCP congestion control algorithms and allows TCP flows to regulate the transmission rate in order to match the network capacity [1, 2, 3, 4, 5]. It employs two variables: the *congestion window (cwnd)*, which is the sliding window, and the *slow start threshold (ssthresh)*, which varies the way *cwnd* is increased. In order to probe the network capacity, *cwnd* is increased until a congestion episode happens. The *cwnd* increasing policy depends on *ssthresh* value. When $cwnd < ssthresh$, the *cwnd* is incremented by one segment for every received acknowledgement. This is the slow start phase that aims at quickly grabbing the network available bandwidth. When $cwnd \geq ssthresh$, the congestion avoidance phase is entered during which the *cwnd* is incremented by one segment per RTT. The congestion avoidance phase gently probes the network for extra available bandwidth. When 3 three duplicate acknowledgments (3 DUPACKs) are received *ssthresh* is set to *cwnd/2* and *cwnd* is set equal to *ssthresh*, whereas, if the re-

transmit timeout expires *ssthresh* is set to *cwnd/2* and *cwnd* is set equal to 1. The *ssthresh* represents an estimate of the current available bandwidth. In fact, when the *cwnd* is less than *ssthresh*, the congestion control algorithm increases *cwnd* to quickly reach the *ssthresh* value and when *cwnd=ssthresh*, it increments the *cwnd* in a more gentle way in order to probe for extra available bandwidth. TCP congestion control ensures that the network capacity is never exceeded but it cannot guarantee fair sharing of that capacity [1].

To improve classic Reno/New Reno congestion control algorithms [2, 6], the recent Westwood TCP proposes to substitute the multiplicative decrease with an adaptive decrease phase: in particular, Westwood TCP passes through a filter the stream of returning ACKs to obtain an estimate of the used bandwidth (BWE) so that when the network capacity is hit and a congestion episode happens, Westwood adaptively reduces the congestion window and the slow-start threshold to the value BWE times the minimum round trip time [7]. The original bandwidth estimation algorithms proposed in [7] does not work properly in the presence of ACK compression [8]. Westwood+ TCP differs from Westwood TCP in that it employs a new bandwidth estimation algorithm that works properly also in the presence of ACK compression. Details concerning the filtering algorithm can be found in [9, 10].

This paper aims at evaluating the performance of Westwood+ TCP over the real Internet. To this purpose, a Linux 2.4.19 implementation of Westwood+ TCP has been developed and compared with an implementation of New Reno [11]. To test and compare the two implementations more than 4000 files, with different sizes, have been uploaded via ftp from a host at the Politecnico of 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, California). Experimental results indicate that Westwood+ TCP improves the goodput with respect to New Reno TCP over paths with a bandwidth delay product larger than few segments. In particular, goodput improvements up to 40-50% have been measured when transmitting data from Politecnico of Bari to Uppsala and UCLA servers.

It is important to point out that GNU Linux 2.4 TCP implements the rate-halving algorithm [12] during the Fast Recovery/Fast Retransmit phase. In order to avoid abrupt interruption of data sending, which would be dangerous for the *self-clocking* [1], we also implement the rate-halving algorithm in Westwood+ TCP.

The paper is organized as follows: Section 2 gives a brief explanation of TCP Westwood+ and describes an implementation in Linux 2.4.19; Section 3 reports Internet measurements collected using this implementation; Finally, Section 4 draws the conclusions.

2 TCP Westwood+

TCP Westwood+ differs from TCP Westwood in that it employs a different bandwidth estimation algorithm that works properly even in presence of ACK compression. In this section, we briefly describe the TCP Westwood algorithm. Subsequently, we introduce the new bandwidth estimation algorithm of Westwood+ TCP and its implementation in GNU Linux kernel.

2.1 TCP Westwood

TCP Westwood congestion control algorithm employs 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}). TCP Westwood basic idea is to exploit the stream of returning ACK packets in order to obtain an estimate of connection available bandwidth (BWE). The bandwidth estimate is used when a congestion episode is detected to properly set the *cwnd* and *ssthtresh*. In absence of congestion episodes, the dynamics of these variables is conform to RFC2581 [2]. A pseudo-code of the Westwood TCP algorithm follows:

```
- When 3 DUPACKs are received by the sender :
    ssthresh = (BWE * RTTmin) / MSS;
    if (ssthresh<2) ssthresh=2;
```

```
    cwnd = ssthresh;

- When coarse timeout expires :
    ssthresh = (BWE * RTTmin) / MSS;
    if (ssthresh<2) ssthresh=2;
    cwnd = 1;

- When ACKs are successfully received :
    cwnd increases as stated in RFC2581.
```

It is worth noting that the adaptive decrease mechanism employed by TCP Westwood improves the stability of the standard TCP multiplicative decrease. Infact, the adaptive windowd shrinking provides a congestion window that is decreased enough in presence of heavy congestion and not too much in presence of light congestion or losses not due to a real congestion episode, such as in the case of unreliable radio links. Moreover the setting $cwnd = BWE * RTT_{min}$ injects a transmission rate equal to $B*RTT_{min}/RTT$, which is less than bandwidth used at the time of congestion. As a consequence, a Westwood TCP flow clears out its path backlog after the setting thus leaving room in the routers' buffers for coexisting flows. This obviously improves network stability, statistical multiplexing and fairness.

TCP Westwood congestion control is heavily based on bandwidth estimation algorithm. As stated before, bandwidth estimation algorithm proposed in [7] overestimates the available bandwidth in the presence of ACK compression [8]. In order to avoid this undesirable behavior, Westwood+ TCP proposes a new bandwidth estimation algorithm that works properly also in the presence of ACK compression [9, 10].

2.2 TCP Westwood+ bandwidth estimation algorithm

The new bandwidth estimation algorithm proposed along with Westwood+ TCP relies on bandwidth samples which are collected every RTT instead that every time an ACK is received by the sender. Basically, the Westwood+ TCP sender counts the amount of data D_k, which is acknowledged during the last $RTT = T_k$ and then computes a bandwidth samples as $B_k = D_k/T_k$. In order to avoid potential numerical instabilities due to too small RTT values, a minimum value of 50ms has been set for the *catching time interval* T_k. Details concerning the ACK counting can be found in [7]. Briefly, 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. Since congestion depends on low frequency components of available bandwidth [13], B_k samples are filtered using a discrete time low-pass filter. In [10] it has been shows that the B_k samples can be effectively filtered using a simple exponential filter as the following:

$$BWE_k = \frac{7}{8} BWE_{k-1} + \frac{1}{8} B_k \qquad (1)$$

The GNU Linux implementation of Westwood+ TCP slightly modifies the rate-halving mechanism to incorporate the adaptive setting of Westwood+ TCP. In particular, we exploit the gracefully *cwnd* decreasing phase of the rate-halving mechanism but we do not allow the *cwnd* to become smaller than the value $BWE * RTT_{min}$. Moreover, when the fast-recovery phase ends we set *cwnd* to $BWE * RTT_{min}$.

2.3 Implementation details

In order to implement TCP Westwood+ on GNU Linux kernel 2.4.19 a new structure named *westwood* has been inserted in structure *tcp_opt*. The *tcp_opt* is a per-socket structure that contains data that are useful for maintaining the state of a TCP connection. For example, it contains variables identifying the current window of data the sender is allowed to transmit, variables that maintain options of the TCP headers, variables for congestion control such as *cwnd* and *ssthresh* and so on. This structure is employed only if the kernel is compiled with the TCP Westwood+ support enabled. Notice that it is not possible to compile this support as a loadable kernel module but it can be compiled only statically in the kernel image. This feature simplifies the design since it provides access to not exported kernel symbols. The

patch was written having in mind the idea of being as less intrusive as possible. The patch allows the superuser to enable/disable TCP Westwood+ at run-time. Moreover, the patch provides the option to enable statistics logging support for testing purposes. It also parses log files in such a way to easily manipulate experimental results [11].

3 Experimental Results

When a new protocol is proposed, it is necessary to collect a large set of experimental results in order to assess its validity and the advantages of its deployment in the Internet [14]. To test and compare the Linux 2.4.19 implementations of Westwood+ and New Reno TCP more than 4000 files, with different sizes, have been uploaded via ftp from a host at the Politecnico of 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, California). Tests were realized exploiting the patch feature which allows to switch between TCP GNU Linux standard implementation and TCP Westwood+. For each upload we have recorded the goodput, the number of retransmitted segments, and important variables such as *cwnd*, *sstresh*, RTT and BWE. Each session of measurements collects data of many file uploads, which are alternatively executed by using Westwood+ or New Reno. Table 1 summarize the main characteristics of the measurement sessions involving the FTP server at the UCLA. Fig. 1 shows the average goodputs achieved during data transfer from Politecnico of Bari to UCLA. The average goodput of a measurement session is obtained by averaging the goodputs of all the uploads of that session. It is interesting to note that Westwood+ TCP provides goodput improvements from 23% to 53% *wrt* New Reno. Moreover the goodput improvements provided by Westwood+ TCP are not due to a more aggressive behavior, since, Fig. 2 shows that the retransmission ratio is roughly the same for both Westwood+ and New Reno TCP.

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

Table 1: FTP uploads from Politecnico of Bari to UCLA, Los Angeles.

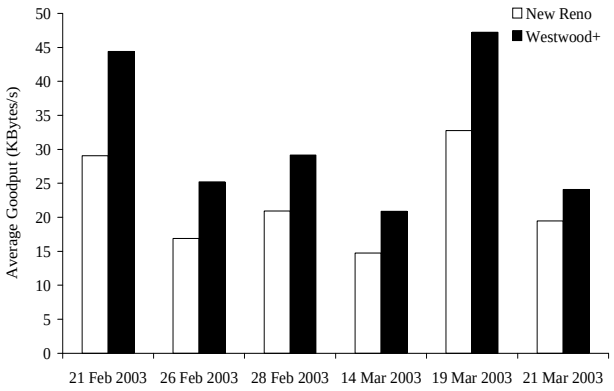

Figure 1: Average Goodput of ftps from Politecnico of Bari to UCLA.

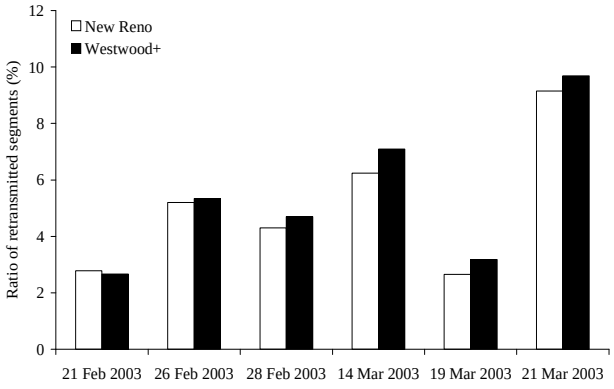

Figure 2: Retransmission ratios when transferring data from Politecnico of Bari to UCLA.

Fig. 3 shows the typical dynamics of the congestion window and the slow start threshold of Westwood+ TCP obtained during a file transfer from the Politecnico of Bari to UCLA. Fig. 4 shows the same variables obtained during the next upload using New Reno. The main difference between Westwood+ and New Reno consists in the *ssthresh* behavior, which is larger when Westwood+ is employed. This effect is due to the adaptive decrease mechanism of Westwood+, which exploits the end-to-end bandwidth estimate. Fig. 5 shows the end-to-end bandwidth estimate obtained during the data transfer. Finally, Fig. 6 plots the *RTT* experienced by the Westwood+ flow during the data transfer.

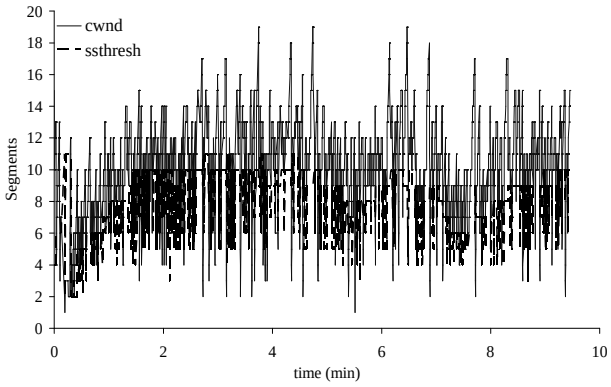

Figure 3: Congestion window and slow start threshold of Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to UCLA on 19/03/03.

Table 2 summarizes the main characteristics of the measurement sessions involving the FTP server at the Uppsala University. Figs. 7 and 8 show the average goodputs and the average retransmission percentage when transferring data from Politecnico of Bari to the Uppsala University.

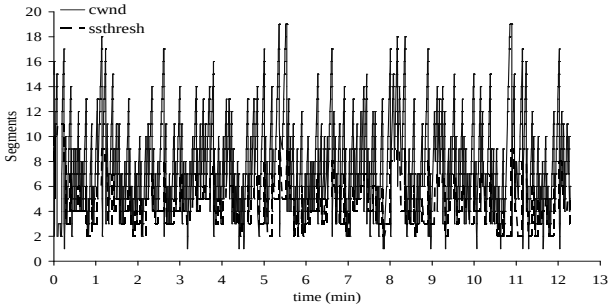

Figure 4: Congestion window and slow start threshold of New Reno TCP while transferring a 32MByte file from Politecnico of Bari to UCLA on 19/03/03.

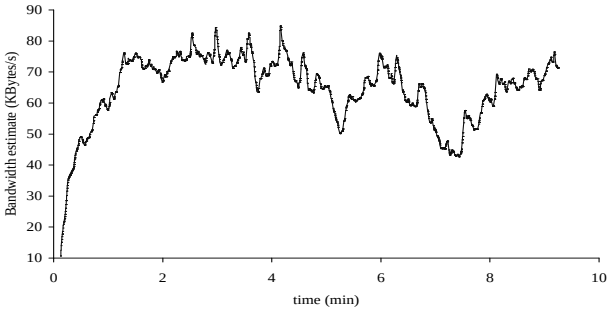

Figure 5: Bandwidth estimate of Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to UCLA on 19/03/03.

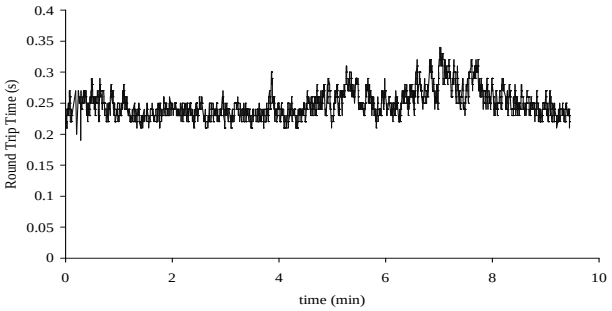

Figure 6: Round Trip Time experienced by Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to UCLA on 19/03/03.

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 2: FTP from Politecnico of Bari to the Uppsala University (Sweden)

It is interesting to note that Westwood+ TCP provides goodput improvements from 4% to 40% *wrt* New Reno, whereas the percentages of retransmitted segments are similar.

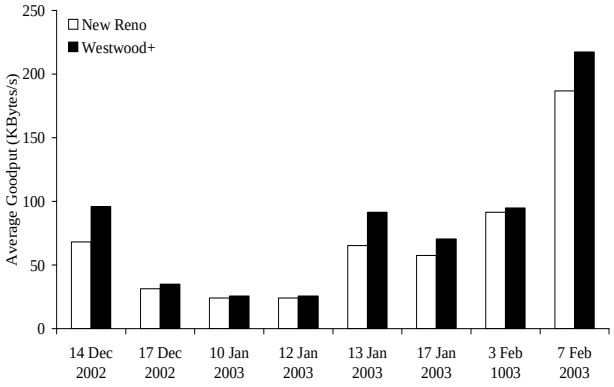

Figure 7: Average Goodput of ftps from Politecnico of Bari to Uppsala.

Table 3 summarizes the main characteristics of the measurement sessions involving the FTP server located at Parma. Figs. 9 and 10 show the goodputs and the retransmission ratios that have been measured while transferring data from Politecnico of Bari to Parma. In this case the connection has a national extension and Westwood+ and New Reno TCP basically achieve the same goodput.

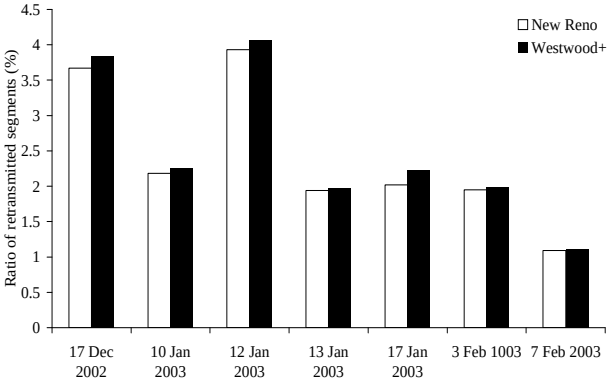

Figure 8: Retransmission ratios when transferring data from Politecnico of Bari to Uppsala.

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

Table 3: FTP uploads towards the server located at Parma (Italy)

Fig. 11 shows the congestion window and the slow start threshold of Westwood+ TCP during a file transfer from the Politecnico of Bari to Parma. Fig. 12 shows the same variables obtained using New Reno. In this case *cwnd* and *ssthtresh* exhibit a similar behavior. This is due to the small RTT that makes less drastic the effect of the Westwood+ TCP algorithm. Finally, Fig. 13 plots the bandwidth estimate obtained by Westwood+ whereas Fig. 14 shows the RTT measured during the data transfer.

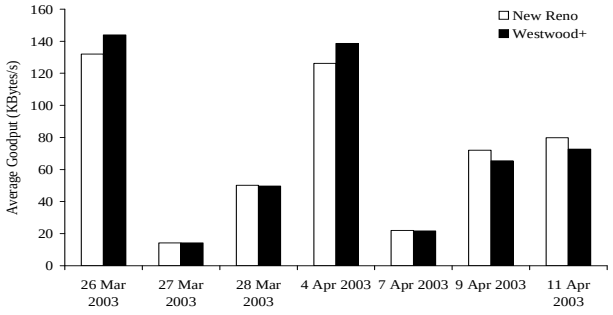

Figure 9: Average Goodput of ftps from Politecnico of Bari to Parma.

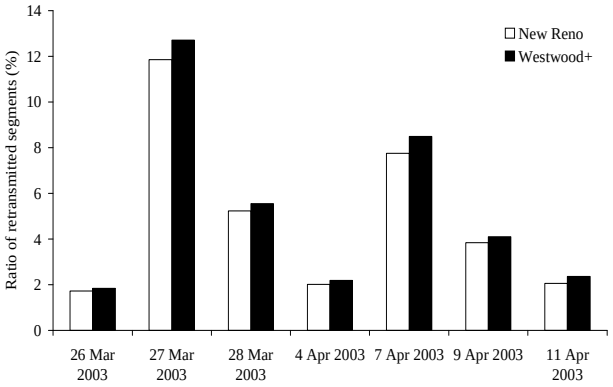

Figure 10: Retransmission ratios when transferring data from Politecnico of Bari to Parma.

It is worth noting that minimum measured RTT is equal to 190ms for the connection with the UCLA server, 70ms for the connection with the Uppsala

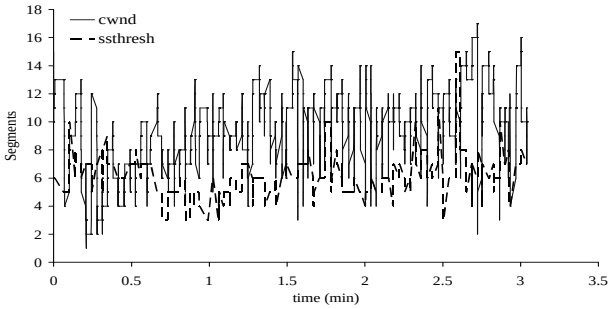

Figure 11: Congestion window and slow start threshold of Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to Parma on 4/04/03.

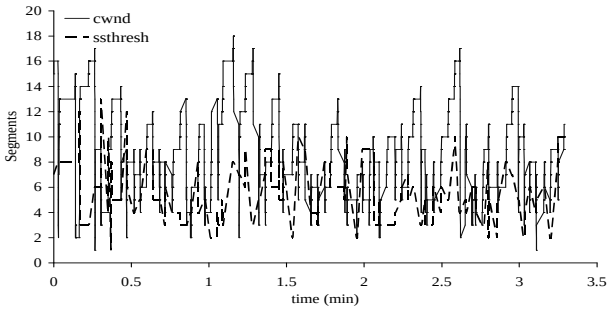

Figure 12: Congestion window and slow start threshold of New Reno TCP while transferring a 32MByte file from Politecnico of Bari to Parma on 4/04/03.

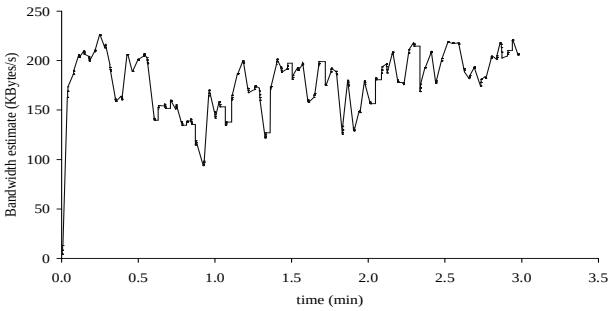

Figure 13: Bandwidth estimate of Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to Parma on 4/04/03.

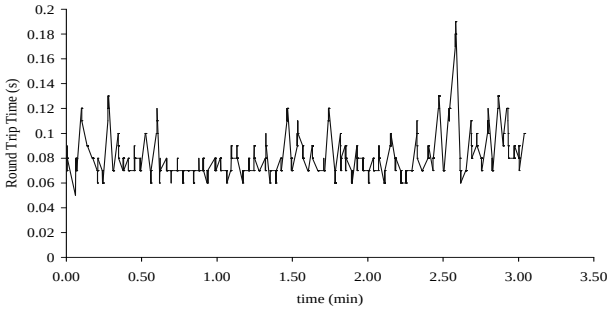

Figure 14: Round Trip Time experienced by Westwood+ TCP while transferring a 32MByte file from Politecnico of Bari to Parma on 4/04/03.

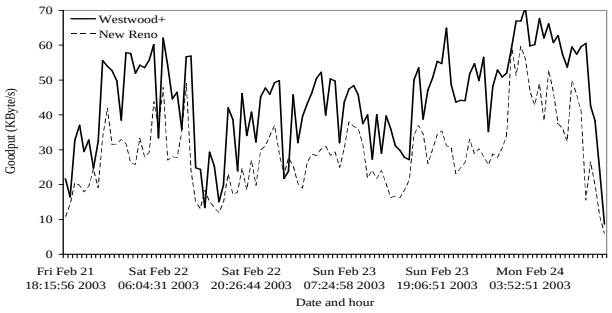

Figure 15: Goodputs of 117 ftps executed on 21/02/2003 from Politecnico of Bari to UCLA (file size = 32MBytes).

server and 50ms for the connection with the server located at Parma. Table 4 shows for each server the pipe size, which is computed as the average Westwood+ TCP goodput times the minimum RTT, and the related average goodput improvement. Table 4 shows that when the pipe size is larger than few MSS, Westwood+ improves the goodput *wrt* New 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.

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

Table 4: Performance analysis, (MSS=1500Bytes)

To give a further insight, Figs. 15-20, report the goodputs achieved by New Reno and Westwood+ TCP during each transfer of the measurement session transmitting data from Politecnico of Bari to UCLA. Figs. 27-26 report analogous data for the uploads from Politecnico of Bari to the Uppsala University and Figs. 29-35 for the uploads from Politecnico of Bari to Parma.

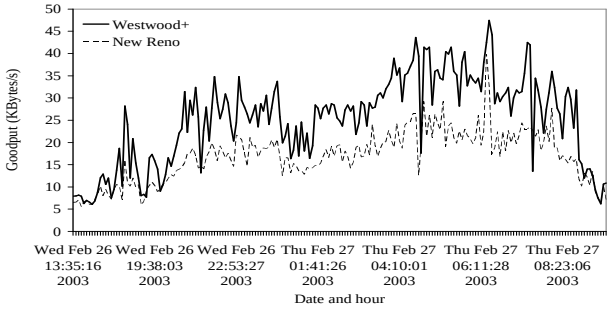

Figure 16: Goodputs of 197 ftps executed on 26/02/2003 from Politecnico of Bari to UCLA (file size = 3.2MBytes).

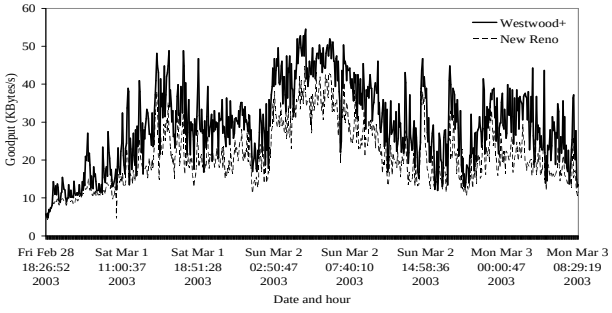

Figure 17: Goodputs of 702 ftps executed on 28/02/2003 from Politecnico of Bari to UCLA (file size = 3.2MBytes).

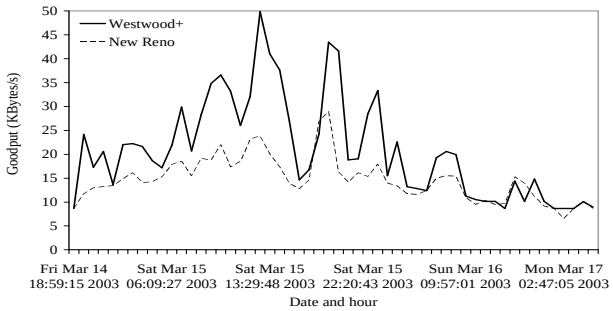

Figure 18: Goodputs of 54 ftps executed on 14/03/2003 from Politecnico of Bari to UCLA (file size = 32MBytes).

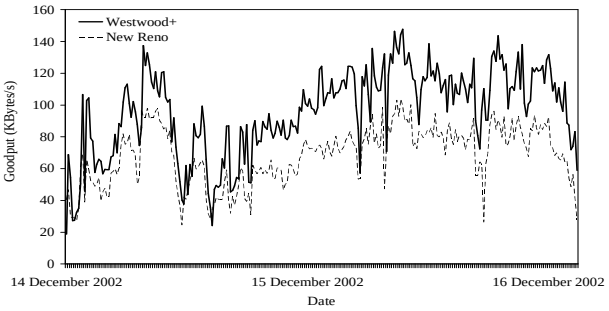

Figure 21: Goodputs of 253 ftps executed on 14/12/2002 from Politecnico of Bari to the Uppsala University (file size = 32MBytes).

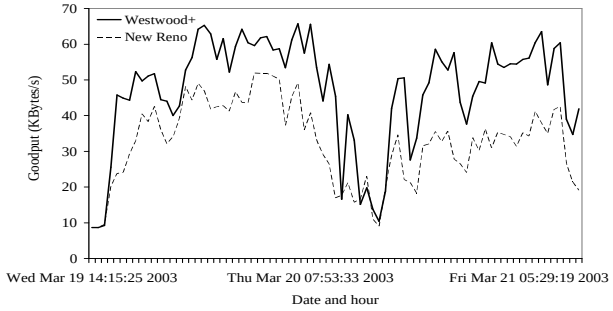

Figure 19: Goodputs of 79 ftps executed on 19/03/2003 from Politecnico of Bari to UCLA (file size = 32MBytes).

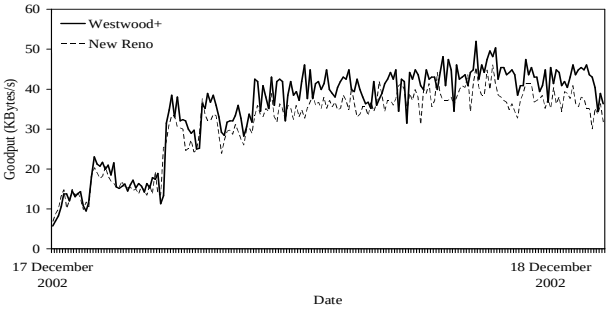

Figure 22: Goodputs of 200 ftps executed on 17/12/2002 from Politecnico of Bari to the Uppsala University (file size = 3.2MBytes).

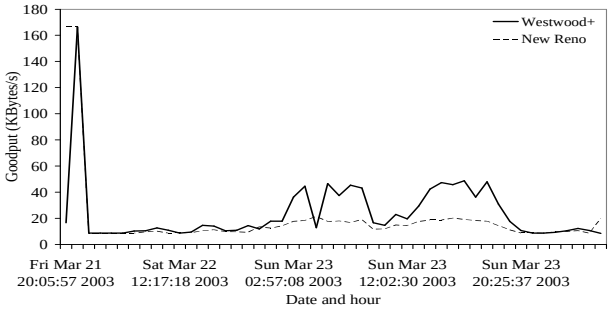

Figure 20: Goodputs of 47 ftps executed on 21/03/2003 from Politecnico of Bari to UCLA (file size = 32MBytes).

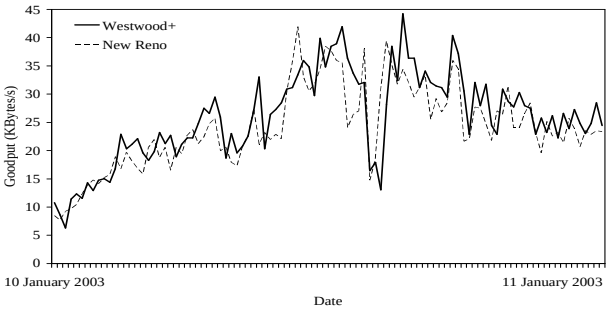

Figure 23: Goodputs of 100 ftps executed on 10/01/2003 from Politecnico of Bari to the Uppsala University (file size = 3.2MBytes).

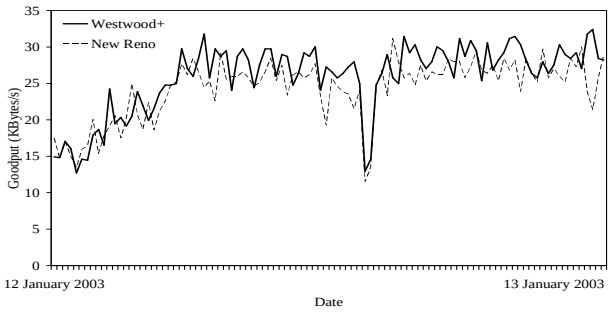

Figure 24: Goodputs of 100 ftps executed on 12/01/2003 from Politecnico of Bari to the Uppsala University (file size = 3.2MBytes).

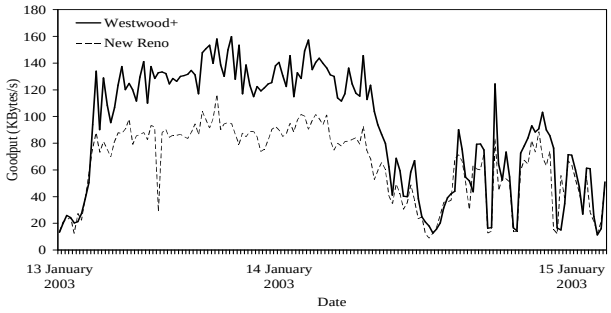

Figure 25: Goodputs of 150 ftps executed on 13/01/2003 from Politecnico of Bari to the Uppsala University (file size = 32MBytes).

4 Conclusions

Westwood+ TCP with the rate-halving feature has been implemented in Linux 2.4.19 and experimented over continental and international Internet connections. Extensive measurements have shown that Westwood+ TCP improves the goodput *wrt* New Reno over connections with a bandwidth-delay product larger than few MSS. In particular, over an Internet connection from Bari, Italy to Los Angeles, California, goodput improvements up to 50% have been measured.

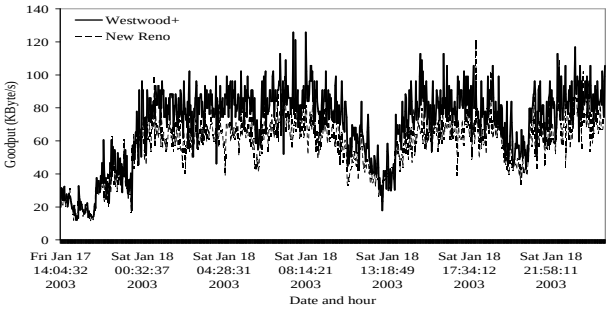

Figure 26: Goodputs of 1000 ftps executed on 17/01/2003 from Politecnico of Bari to the Uppsala University (file size = 3.2MBytes).

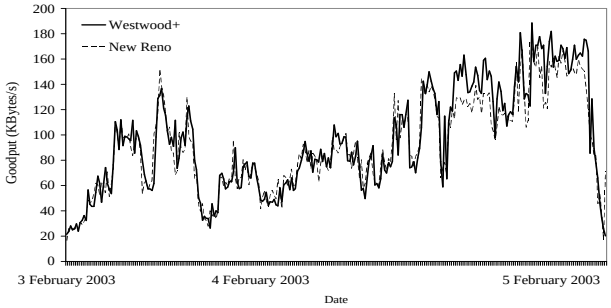

Figure 27: Goodputs of 278 ftps executed on 3/02/2003 from Politecnico of Bari to the Uppsala University (file size = 32MBytes).

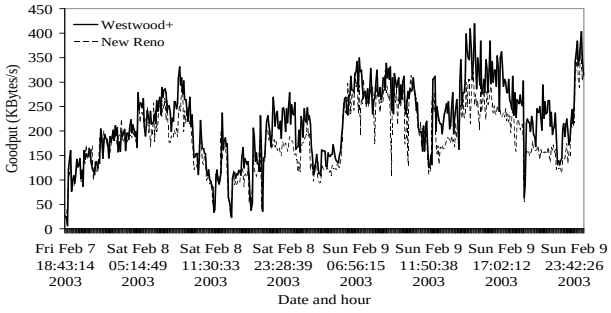

Figure 28: Goodputs of 500 ftps executed on 7/02/2003 from Politecnico of Bari to the Uppsala University (file size = 32MBytes).

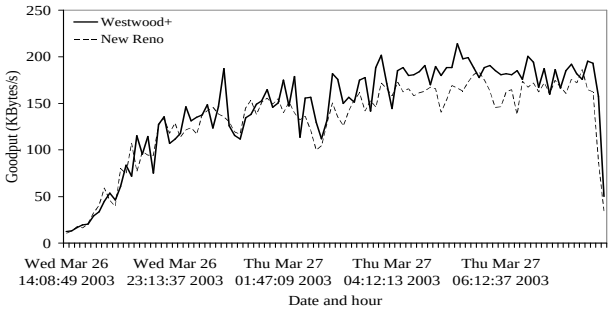

Figure 29: Goodputs of 100 ftps executed on 26/03/2003 from Politecnico of Bari to Parma (file size = 32MBytes).

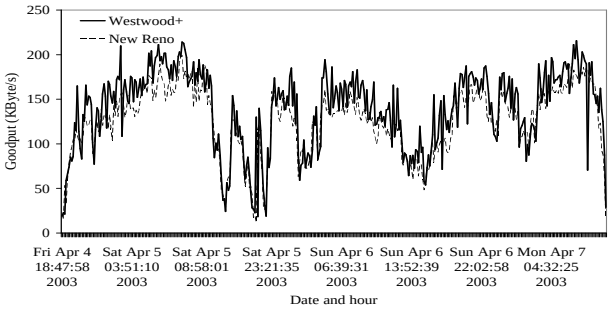

Figure 32: Goodputs of 390 ftps executed on 4/04/2003 from Politecnico of Bari to Parma (file size = 32MBytes).

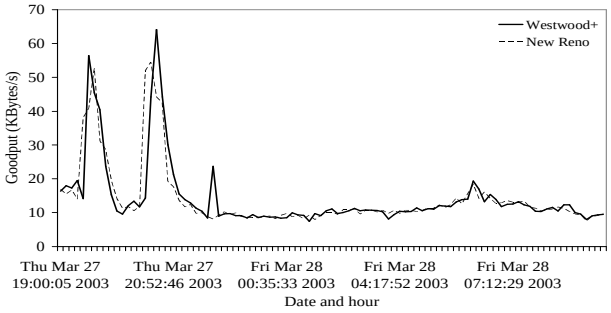

Figure 30: Goodputs of 98 ftps executed on 27/03/2003 from Politecnico of Bari to Parma (file size = 3.2MBytes).

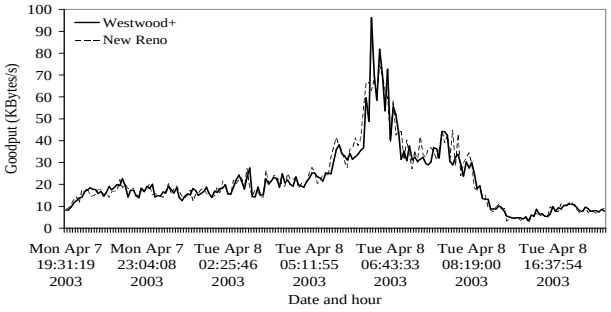

Figure 33: Goodputs of 200 ftps executed on 7/04/2003 from Politecnico of Bari to Parma (file size = 3.2MBytes).

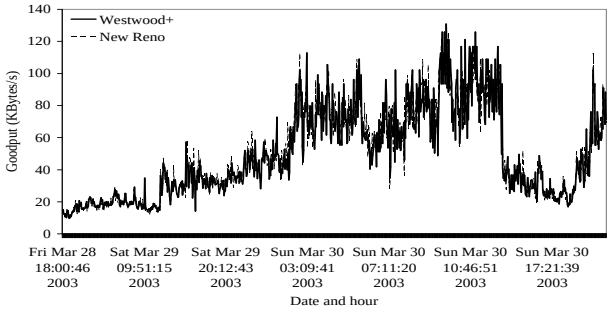

Figure 31: Goodputs of 1000 ftps executed on 28/03/2003 from Politecnico of Bari to Parma (file size = 3.2MBytes).

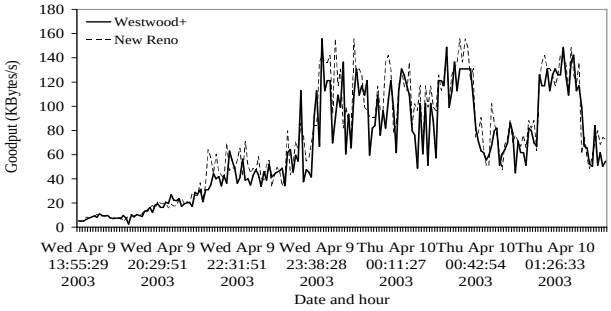

Figure 34: Goodputs of 200 ftps executed on 9/04/2003 from Politecnico of Bari to Parma (file size = 3.2MBytes).

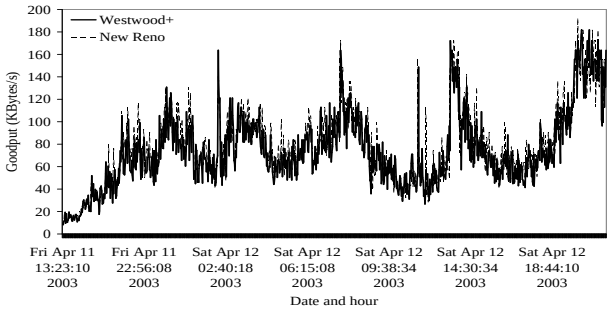

Figure 35: Goodputs of 1000 ftps executed on 11/04/2003 from Politecnico of Bari to Parma (file size = 3.2MBytes).

5 Acknowledgements

We thank Prof. Mario Gerla, Prof. Mikael Sternad and Daniele Verzelloni who allowed us to collect live Internet measurements using FTP servers located at Universities of Los Angeles (USA), Uppsala (Sweden) and Parma (Italy).

References

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

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

[3] W. Stevens. *TCP/IP Illustrated vol.1 : The Protocols*. Addison Wesley, Reading, MA, 1994.

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

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

[6] S. Floyd and T. Henderson. NewReno modification to TCP's fast recovery. RFC 2582, April 1999.

[7] 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.

[8] 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.

[9] R. Ferorelli, L. A. Grieco, S. Mascolo, G. Piscitelli, and P. Camarda. Live internet measurements using Westwood+ TCP congestion control. In *IEEE Globecom 2002*, Taipei, Taiwan, November 2002.

[10] 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.

[11] Linux kernel v2.4.19. Available at http://www.kernel.org/pub/linux/kernel/v2.4/.

[12] Matt Mathis, Jeff Semke, Jamshid Mahdavi, and Kevin Lahey. The Rate-Halving Algorithm for TCP Congestion Control. draft-ratehalving.txt, June 1999.

[13] S. Q. Li and C. Hwang. Link capacity allocation and network control by filtered input rate in high speed networks. *IEEE/ACM Transactions on Networking*, 3(1):10–25, February 1995.

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