Live Internet Measurements Using Westwood+ TCP Congestion Control*

R. Ferorelli[1], L. A. Grieco[2], S. Mascolo[1], G. Piscitelli[1], P. Camarda[1]
(1)Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy
(2)Dipartimento d'Ingegneria dell'Innovazione, Università di Lecce, Italy.

***Abstract*- Westwood+ TCP is a sender-side only modification of the classic TCP that is based on end-to-end estimation of the bandwidth available to the connection in order to adaptively shrink the TCP *congestion window* and *slow start threshold* after congestion. In this paper we report measurements obtained using Linux implementations of Westwood+, Westwood and Reno to *ftp* data over Internet connections spanning continental and intercontinental distances. In particular we show that the bandwidth estimation algorithm employed by Westwood+ nicely tracks the available bandwidth, whereas the previous bandwidth estimation algorithm used by TCP Westwood fails to work in the real Internet due to ACK compression. Collected live Internet measurements also show that Westwood+ improves the goodput, from 10 to 46 % w.r.t. TCP Reno.**

I. INTRODUCTION

Classic TCP congestion control [1,2,7,8,17] follows the end-to-end principle [4], which does not allow the network layer to supply any explicit feedback, and the *additive increase multiplicative decrease* paradigm (AIMD) [11]. The *additive increase* phase aims at increasing the connection input rate until the network available bandwidth is hit and a congestion episode happens. The sender becomes aware of congestion via the reception of duplicate acknowledgments (DUPACKs) or the expiration of a timeout. Then it 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) and to heavy congestion (i.e. timeout) by reducing the congestion window to one. This behavior is the *multiplicative decrease* phase, which aims at reducing the input rate after congestion.

The increasing phase has two parts: (1) the *slow*-start phase, which exponentially increases the congestion window, and (2) the *congestion avoidance* phase that linearly increases the congestion window. The slow-start phase aims at quickly grabbing the available bandwidth whereas the congestion avoidance phase at gently probing for new available bandwidth. The implementation of these mechanisms forms the core of classic TCP that is known as TCP Reno [1,2].

TCP Westwood [3] departs from the AIMD paradigm by proposing the *additive increase adaptive decrease* (AIAD) paradigm. The slow-start and congestion avoidance probing phases are left unchanged, whereas the adaptive decrease phase substitutes the multiplicative decrease phase. The adaptive decrease phase exploits the end-to-end bandwidth estimation method to set the control windows after congestion. TCP Westwood significantly improves fairness in wired networks and utilization of wireless links [3].

In this paper we show that the bandwidth estimation algorithm proposed in [3] does not work properly in real Internet because of ACK compression [10]. In particular, Westwood may greatly overestimate the available bandwidth, which is potentially disruptive of fairness and can lead to starvation of coexisting connections.

The aim of this paper is to test over the real Internet a slightly modified version of the bandwidth estimation algorithm proposed in Westwood [3]. The old TCP Westwood plus the new bandwidth estimation algorithm is called Westwood+. A Linux implementation of Westwood+ has been developed for testing purposes.

Measurements collected from a local area network using the Dummynet emulator [5] and Internet live tests show that the bandwidth estimation algorithm used by TCP Westwood overestimates the available bandwidth up to 300% because of ACK compression. On the other hand, Westwood+ nicely tracks the available bandwidth and properly set the congestion window and slow-start threshold, which exhibit a less oscillating dynamics.

The paper is organized as follows. Section 2 describes TCP Westwood+, Section 3 collects emulation and Internet measurements showing that Westwood+ achieves a greater goodput than TCP Reno, with increment ranging from 10 to 46%. Finally the last section draws the conclusions.

II. TCP WESTWOOD+

TCP Westwood+ enhances the TCP Westwood algorithm by employing a slightly modified bandwidth estimation algorithm that works properly also in the presence of ACK compression. We first resumes TCP Westwood algorithm and then we introduce the new bandwidth estimation algorithm.

A. 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}); (4) size of the delivered segments (*seg_size*).

TCP Westwood performs an end-to-end estimate of the bandwidth B available along a TCP connection by measuring and low-pass filtering the rate of returning ACKs. The estimate is then used to adaptively set the congestion window

* This work was partially supported by the MIUR Research Project 488/92 C22 entitled "E-service to the Citizens"

0-7803-7632-3/02/$17.00 ©2002 IEEE

and the slow-start threshold after a congestion episode as described below:

When 3 DUPACKs are received by the sender:
ssthresh = (B*RTT$_{min}$)/seg_size;
cwnd = ssthresh;
When coarse timeout expires:
ssthresh = (B*RTT$_{min}$)/seg_size;
cwnd = 1;
When ACKs are successfully received:
increases *cwnd* accordingly to Reno algorithm.

The AIAD mechanism can clearly improve networks stability with respect to the AIMD mechanism since reductions of the *cwnd* and the *ssthresh* can be large when heavy congestion happens and small when light congestion happens.

B. The New Bandwidth Estimation Algorithm

In the literature various schemes have been proposed to measure network capacity. The packet pair (PP) mechanism tries a more precise method to infer the bottleneck available bandwidth at the starting of a connection by measuring the interarrival time between the ACKs of two packets sent back to back [16]. The PP mechanism requires per-flow queuing, a feature not available in Internet routers, and no delayed ACK. Hoe proposes a refined PP method for estimating the available bandwidth in order to properly initialize the *ssthresh* [9]: 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 [12]. Lai and Baker propose an evolution of the PP property for measuring the link bandwidth in FIFO-queuing networks [13]. The method consumes less network bandwidth while maintaining approximately the same accuracy of other methods, which is poor for paths longer than few hops.

Estimating the available bandwidth at the beginning of a TCP connection over a FIFO-queuing network is a very different and much more difficult task than measuring the actual rate a connection is achieving during the data transfer as it is done by Westwood TCP [3]. In [3] the idea is to estimate the available bandwidth by properly filtering the *flow* of returning ACKs. A sample of available bandwidth $b_k = d_k/(t_k - t_{k-1})$ is computed every time t_k the sender receives an ACK, where the amount d_k of data acknowledged by an ACK is determined by a proper counting procedure that considers delayed ACKs, duplicate ACKs and selective ACKs [3]. Bandwidth samples b_k are low-pass filtered by employing the following time-varying low-pass filter to obtain the bandwidth estimate:

$$\hat{b}_k = \frac{2\tau_f - \Delta_k}{2\tau_f + \Delta_k}\hat{b}_{k-1} + \Delta_k \frac{b_k + b_{k-1}}{2\tau_f + \Delta_k}, \qquad (1)$$

where $1/\tau_f$ is the filter cut-off frequency (a typical value is $\tau_f = 0.5s$), and $\Delta_k = (t_k - t_{k-1})$. Low-pass filtering is necessary since congestion is due to low frequency components [10], and because of delayed ACK option. The filter coefficients are time-varying to counteract the fact that the sampling interval Δ_k is not constant. The bandwidth estimate obtained using the filter (1) is affected by ACK compression, which happens in the presence of reverse traffic [15]. In fact, a consequence of ACK compression is that bandwidth samples contain high frequency components that cannot be filtered out by using a low-pass discrete time filter. Consequently, ACK compression causes a systematic bandwidth overestimate. This overestimate may disrupt the fairness between TCP connections and even may lead to starvation of some connections.

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 propose to 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 now the last *RTT*. This operation corresponds to evenly spread the bandwidth samples $d_j/(t_j - t_{j-1})$ over the *RTT* interval, that is, it corresponds to low-pass filtering high frequency components due to ACK compression. Samples b_k are then low-pass filtered using (1). To conclude, we say a few words on the sampling interval $\mathrm{RTT} = \Delta_k$. In order to get a properly working filter, it is necessary to satisfy the Nyquist sampling theorem, that is, it must be $\Delta_k \le \tau_f/2$. To be conservative, we assume $\Delta_k \le \tau_f/4$. Thus, if it happens that $\Delta_k > \tau_f/4$ then we *interpolate and re-sample* by creating $N = \mathrm{int}(4 \cdot RTT/\tau_f)$[1] virtual samples $b_k = D_k/\Delta_k$ that arrives with the interarrival time $\tau_f/4$.

III. Performance 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 [6]. We have developed a Linux 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. At first, we have used an emulated environment, later we have collected live Internet measurements. In both cases we evaluate the goodput as well as the correctness of bandwidth estimation algorithm.

Fig.1 depicts the topology adopted in the first set of measurements collected from the emulated scenario. A router connects two different Ethernet segments. The router is implemented by the computer 3, which emulates delays and packet drop probabilities by using the Dummynet network emulator [5]. Two Linux PCs (One and Two in Fig. 1) are connected with the first Ethernet segment and the Linux PC

[1] int(·) stands for the integer part of (·)

Four with the second segment.

The second set of measurements has been collected over the real Internet. TCP Linux implementations of Westwood and Westwood+ located at the networking laboratory of the Politecnico di Bari, Italy, have been used 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.

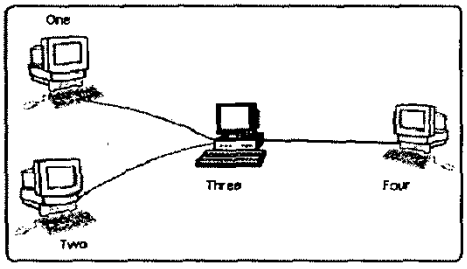

Fig. 1. LAN scenario with the Dummynet Emulator.

A. Bandwidth Estimation

This subsection reports bandwidth estimates obtained by Westwood and Westwood+, respectively. In order to test the bandwidth estimate algorithm, Dummynet has been configured to provide a 500Kbytes/s link between the two Ethernet segments with a 100ms RTT. While an FTP transfer was running from PC One to PC Four, a secondary FTP transfer was launched along the reverse path to generate ACK compression. Figure 2 shows that the filter used in TCP Westwood does not work properly in the presence of ACK compression. In fact, it overestimates the available bandwidth up to 300%. On the other hand, Figure 3 shows that the bandwidth estimation algorithm employed by Westwood+ nicely tracks the 500Kbytes/s channel capacity (the slight difference is due to the presence of protocols headers).

Figures 4 and 5 plot the bandwidth estimates obtained using Westwood and Westwood+, respectively, during *ftp* uploads to UCLA, Los Angeles. Figure 6 and 7 show analogous data using Westwood and Westwood+ during *ftp* uploads to Uppsala University, Sweden.

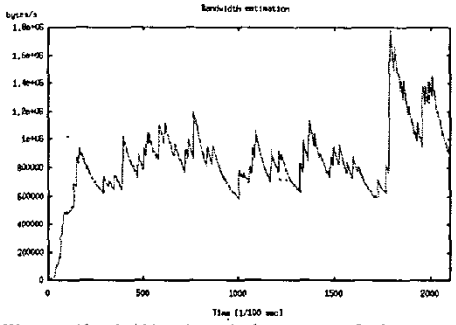

Fig. 2. Westwood bandwidth estimate in the presence of ACK compression.

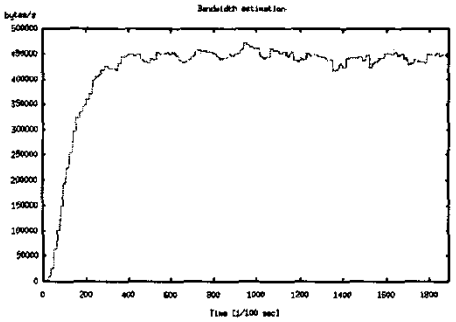

Fig. 3. Westwood+ bandwidth estimate in the presence of ACK compression.

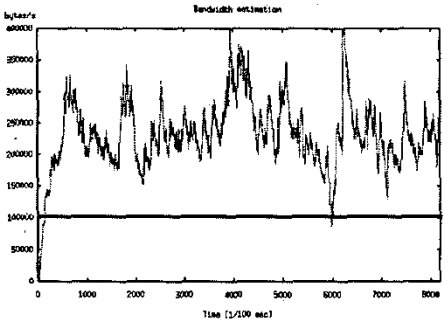

Fig. 4. Westwood bandwidth estimate during the ftp to UCLA, Los Angeles, California.

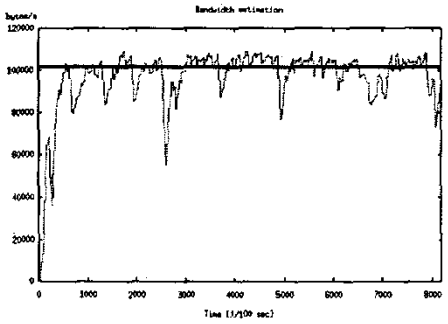

Fig. 5. Westwood+ bandwidth estimate during the ftp to Los Angeles, California.

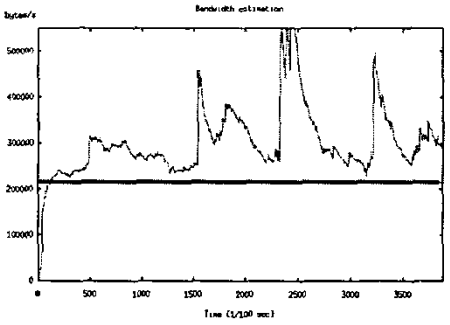

Fig. 6. Westwood bandwidth estimate during the ftp to Uppsala University, Sweden.

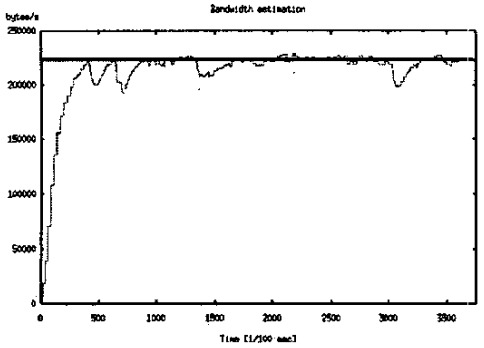

Fig. 7. Westwood+ bandwidth estimate during the ftp to Uppsala University, Sweden.

Note that Figs. 4-7 show a thick line that represents the average throughput. In fact, a good estimate must be close to the average throughput. Figs. 4-7 show 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.

B. Congestion Window and Slow Start Threshold

In order to get a further insight into the behaviours of Westwood+ and Reno TCP, figures 8 and 9 plot the *cwnd* and the *ssthresh* during the FTP transfer performed in the emulated scenario in Fig. 1. While TCP Reno exhibits an oscillating *ssthresh* (see Fig. 8), TCP Westwood+ sets the *ssthresh* equal to a constant and greater value than the one of Reno after a short transient (see Fig. 9). This provides a more efficient use of network bandwidth. Similar results have been shown by Internet live measurements.

Figures 10 and 11 plot the *cwnd* and the *ssthresh* of Reno and Westwood+ during the FTP transfers to UCLA, Los Angeles, whereas Figures 12 and 13 show the analogous plots during the FTP transfer to Uppsala University, Sweden. Figures 10-13 show that TCP Reno exhibits a more oscillating behaviour of *cwnd* and *ssthresh* because of its "blind" multiplicative window shrinking, whereas TCP Westwood+ correctly sets the right value of *ssthresh*, thus significantly reducing the number of congestion episodes.

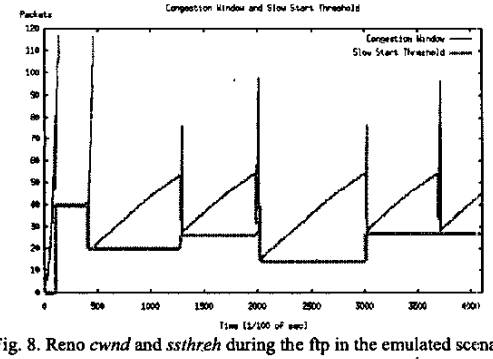

Fig. 8. Reno *cwnd* and *ssthreh* during the ftp in the emulated scenario.

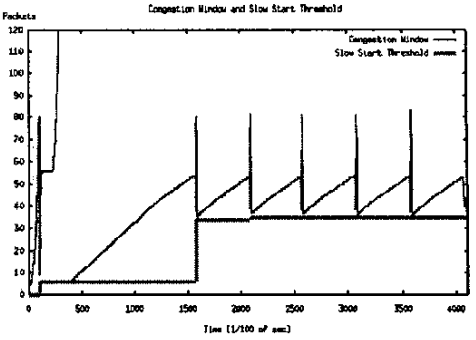

Fig. 9. Westwood+ *cwnd* and *ssthreh* during the ftp in the emulated scenario.

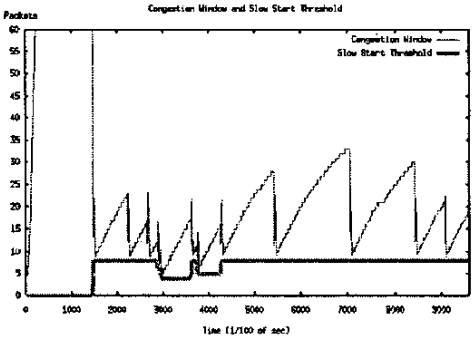

Fig. 10. Reno *cwnd* and *ssthresh* during the ftp transfer to UCLA, Los Angeles.

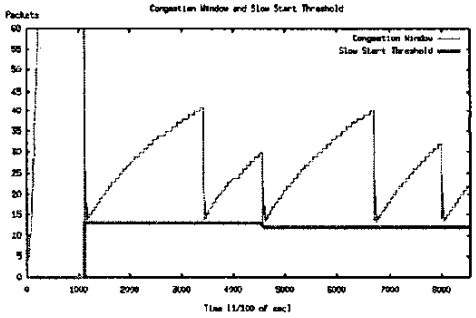

Fig. 11. Westwood+ *cwnd* and *ssthreh* during the ftp transfer to UCLA, Los Angeles.

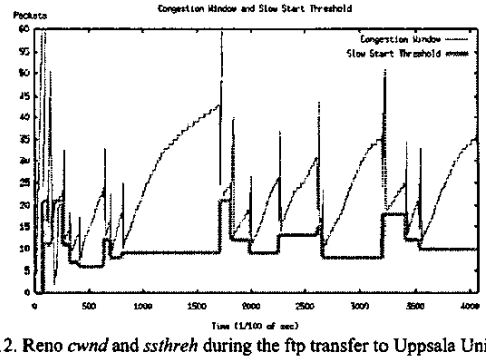

Fig. 12. Reno *cwnd* and *ssthreh* during the ftp transfer to Uppsala University, Sweden.

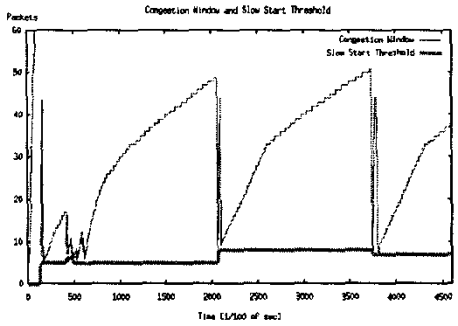

Fig. 13. Westwood+ *cwnd* and *ssthreh* during the ftp transfer to Uppsala University, Sweden.

C. Goodput

To compare the efficiency of the Westwood+ protocol with respect to Reno, we have repeated 50 file uploads to each of the two servers in Los Angeles and in Uppsala. Moreover we have used different file size to investigate the impact of the transient time on the goodput, which has been measured as *(transferred_data−retransmitted_data)/transfer_time*.

The average value of the goodput achieved in each series of 50 uploads are reported in Figure 14 for the server in Los Angeles, and in Figure 15 for the server in Uppsala. TCP Westwood+ provides goodput increment ranging approximately from 10% to 46% with respect to Reno.

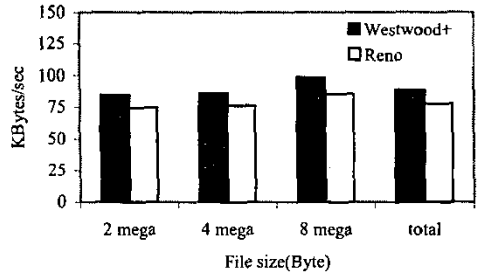

Fig. 14. Average goodput achieved during *ftp* to Los Angeles.

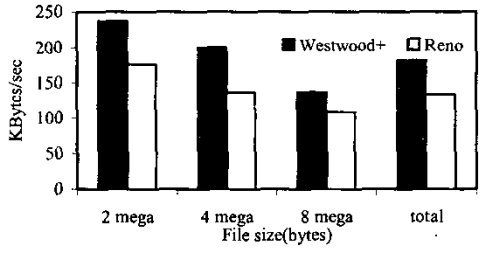

Fig. 15. Average goodput achieved during *ftp* to Uppsala.

IV. Conclusions

In this paper we have tested the behavior of the TCP Westwood+ algorithm via both emulation and live Internet measurements. Collected results have shown that the new bandwidth estimate algorithm nicely estimates the connection available bandwidth. Moreover, they have shown that Westwood+ improves the goodput from 10% to 46% with respect to Reno because of its adaptive window setting that takes into account the available bandwidth. On the other hand, Reno requires many consecutive window reductions to find the available network capacity after congestion. This increases the number of congestion episodes and reduces the throughput.

Acknowledgments

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.

References

[1] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of ACM Sigcomm '88*, Stanford, CA, pp. 314-329, August 1988.

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

[3] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks," in *Proceedings of ACM Mobicom 2001*, July, Rome, Italy. To appear in ACM Journal on Wireless Networks (WINET), Special Issue with selected papers from MOBICOM2001.

[4] D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proceedings of ACM Sigcomm'88*, Stanford, CA, pp. 106-114, August 1988.

[5] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, pp. 31-41, Jan. 1997.

[6] S. Floyd, V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transaction Networking*, vol. 9, pp 392-403, Aug. 2001.

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

[8] Peterson & B. Davie, "Computer Networks a Systems Approach," Morgan Kaufmann, 1996

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

[10] J. C. Mogul, "Observing TCP dynamics in real networks," in *Proceedings of ACM Sigcomm 1992*, Baltimore, Maryland, USA, pp. 305-317, August 1992.

[11] Dah-Ming Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol.17, pp. 1-14, June 1989.

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

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

[14] S. Q. Li and C. Hwang, "Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks," *IEEE/ACM Transaction Networking*, vol. 3, pp. 10-25, Feb. 1995.

[15] L. A. Grieco and S. Mascolo, "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, 130-146, April, 2002.

[16] S. Keshav, "A Control-theoretic Approach to Flow Control," in *Proceedings of ACM Sigcomm 1991*, Zurich, Switzerland, pp. 3-6, September 1991.

[17] S. Mascolo, "Congestion control in high-speed communication networks," *Automatica*, Special Issue on Control Methods for Comm. Networks, Eds. V. Anantharam J. Warland, Dec. 1999.