Simulation-based Comparisons of Reno, Vegas and Westwood+ TCP

LUIGI ALFREDO GRIECO
Dipartimento di Ingegneria dell' Innovazione
Universita' di Lecce
Via Monteroni, 73100 Lecce
ITALY

SAVERIO MASCOLO, PIETRO CAMARDA
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari
Via Orabona 4, 70125 Bari
ITALY

Abstract: – During the last 20 years, several congestion control algorithms have been proposed to achieve network stability, fair bandwidth allocation and high resource utilization. This paper aims at comparing three well-known control schemes that are Reno, Vegas and Westwood+ TCP. To the purpose, an extensive set of ns-2 simulation results has been collected. In particular, single and multi bottleneck scenarios with link capacities ranging from 1Mbps to 100Mbps and in the presence of homogeneous and heterogeneous traffic sources have been considered. The following main results have been found: (1) Westwood+ TCP fairly behaves when interacting with Reno TCP; (2) Westwood+ TCP improves the fairness in bandwidth sharing with respect to Reno TCP; (3) Vegas TCP is not able to grab its own bandwidth share when interacting with Reno TCP.

Keywords:- Internet Congestion Control, Computer Networks, Performance Evaluation, Computer Simulation, Networking

1. Introduction

Internet stability is still largely based on the congestion control algorithm proposed by Van Jacobson in [1], which is known as Thaoe TCP, on its first modification, which is known as Reno TCP [2], and other variants described in [3],[4],[5]. The Van Jacobson congestion control algorithm has been designed by following the end-to-end principle and has been quite successful from keeping the Internet away from congestion collapse [18],[19],[20]. After that the Van Jacobson's algorithm has been introduced at the end of the eighties, several end-to-end congestion control algorithms have been proposed to improve network stability, fair bandwidth allocation and high resource utilization. Such a control schemes are sliding window algorithms since they use ACKs reception to trigger the transmission of new data into the network (i.e. *self-clocking* principle [1]). The variables congestion window (*cwnd*) and slow start threshold (*ssthresh*) are employed to adapt the flow input rate to the network status. In particular, congestion control algorithms in [1]-[5] exploit the Additive Increase Multiplicative Decrease (AIMD) paradigm, which additively increases the *cwnd* to grab the available bandwidth and suddenly decrease the *cwnd* when network congestion is detected. Segment losses are used as implicit indication of network congestion and when they happen, the AIMD algorithms decrease the *cwnd* and the *ssthresh* via a multiplicative factor [6]. The *cwnd* variable is used as a sliding window to regulate the data transmission, whereas, the *ssthresh* is employed to vary the way the *cwnd* is increased: in particular, on ACK reception, if *cwnd* is less than *ssthresh*, then it is exponentially increased, else it is linearly increased. AIMD algorithms ensure network stability but they don't guarantee fair sharing of network resources [6], [7], [21].

Vegas TCP is the first algorithm that proposes a new paradigm for the Internet congestion control. It employs a mechanism to early detect network congestion. In particular, it first computes the difference between the input rate ($cwnd/RTT$) and the expected rate ($cwnd/RTT_{min}$), where RTT is the Round Trip Time and RTT_{min} is the minimum measured round trip time. Such a difference is then considered to infer network congestion. In particular, if the difference is less than a threshold α then the *cwnd* is additively increased, whereas if the difference is greater than another threshold β then the *cwnd* is Additively Decreased; finally, if the difference is less than β and greater than α, then the *cwnd* is kept constant [9]. Vegas TCP ensures network stability [10] but it is not able to grab its own bandwidth share when interacting with Reno TCP sources [11].

Westwood TCP is a new congestion control algorithm based on end-to-end bandwidth estimate [12]. It mainly differs from Reno TCP since it estimates the used bandwidth by filtering the flow of the returning ACKs: when network congestion is

detected, the *cwnd* and the *ssthresh* are adaptively set by taking into account the estimated bandwidth. The bandwidth estimation algorithm proposed in [12] critically behaves in the presence of ACK compression. Thus a new version of the Westwood algorithm, which we call Westwood+ TCP, has been proposed in [14] to cope with ACK compression effects. Furthermore, in [15] has been shown via a mathematical analysis that Westwood+ is friendly to Reno TCP.

This paper aims at comparing the performances of Reno, Vegas and Westwood+ TCP via the *ns-2* simulator [16]. To the purpose, an extensive set of simulation results has been collected. In particular, single and multi bottleneck scenarios with link capacities ranging from 1Mbps to 100Mbps and in the presence of homogeneous and heterogeneous traffic sources have been considered. The following main results have been found: (1) Westwood+ TCP fairly behaves when interacting with Reno TCP; (2) Westwood+ TCP improves the fairness in bandwidth sharing with respect to Reno TCP; (3) Vegas TCP is not able to grab its own bandwidth share when interacting with Reno TCP.

The paper is organized as follows: Section 2 outlines the Westwood+ algorithm; in Section 3 Reno, Vegas and Westwood+ are compared; finally, the last section draws the conclusions.

2. Westwood+ TCP

The Westwood+ TCP algorithm is based on end-to-end bandwidth estimation. In particular, it filters the average rate of the returning ACKs [12],[14]. The obtained estimate is then used to set the control windows *cwnd* and *ssthresh* when network congestion is experienced. In particular, when three DUPACKs are received, both the congestion window (*cwnd*) and the slow start threshold (*ssthresh*) are set equal to the estimated bandwidth (*BWE*) times the minimum measured round trip time (RTT_{min}); when a coarse timeout expires the *ssthresh* is set as stated above while the *cwnd* is set equal to 1.

These settings drastically reduce the control windows in the presence of heavy network congestion whereas gently reduce these windows in the presence of light congestion. On the other hand, Reno implements a blind window reduction that does not take into account the congestion status.

It is worth noting that the bandwidth estimate employed by Westwood+ TCP measures the low pass component of the used bandwidth. This is much more different from measuring the low pass component of the sending rate *cwnd*/*RTT*. In particular, under dynamic condition, a sudden reduction of network bandwidth due to a sudden change in network load can be quickly discovered by monitoring the flow of the returnig ACKs whereas it cannot be discovered by monitoring the rate *cwnd*/*RTT*.

The pseudo code of the Westwood+ algorithm is:

a) On ACK reception:
   the end-to-end bandwidth estimate *BWE* is computed and *cwnd* is increased accordingly to the Reno algorithm;

b) When 3 DUPACKs are received:
   ssthresh = (*BWE** RTT_{min}) / *seg_size*;
   cwnd = *ssthresh*;

c) When coarse timeout expires:
   ssthresh = (*BWE** RTT_{min}) / *seg_size*;
   cwnd = 1;

2.1 End-to-end bandwidth estimate

The AIMD algorithm can be viewed as end-to-end method to obtain a "rough" but robust measurement of the best effort bandwidth that is available along a TCP path. 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 that are consecutively sent [23]. Hoe proposes a refined PP method for estimating the available bandwidth in order to properly initialize the *ssthresh* [24]: 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 [25]. Lai and Baker propose an evolution of the PP property for measuring the link bandwidth in FIFO-queuing networks [26]. The method consumes less network bandwidth while maintaining approximately the same accuracy of other methods, which is poor for paths longer than few hops.

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 in [12]. In [12] 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. Bandwidth samples b_k are low-pass filtered by employing the following time-varying low-pass filter to obtain the bandwidth estimate:

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

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 [25], and because of delayed ACK option [13].

The bandwidth estimate obtained using the filter (1) is negatively affected by ACK compression, which happens in the presence of reverse traffic [14,15]. In particular, 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, in [14] we have proposed 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 $D_k = \sum d_j$ acknowledged during the last RTT to compute the bandwidth sample $b_k = D_k / \Delta_k$, where Δ_k is now the last *RTT*.

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 $\Delta_k = \tau_f / 4$.

To give an insight into the bandwidth estimation algorithms employed by Westwood and Westwood+, Fig. 1 shows the bandwidth estimate obtained by one Westwood or one Westwood+ flow over a 1Mbps bottleneck in the presence of reverse traffic made by several Reno connections. It is straightforward noting that the new filtering scheme avoids ACK compression effects. Whereas, the Westwood algorithm overestimates the bandwidth up to ten times with respect to the bottleneck bandwidth that is equal to 1Mbps.

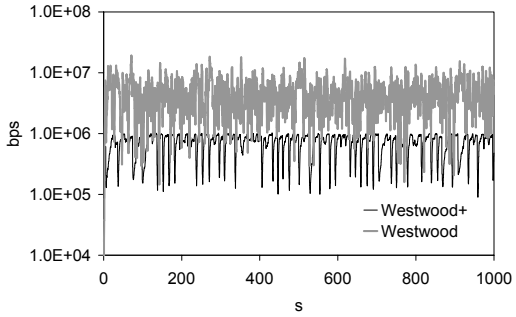

Figure 1. Bandwidth estimates over a 1Mbps bottleneck.

3. Performance Evaluation

3.1 Single bottleneck scenario

This section aims at evaluation the fairness in bandwidth sharing achieved by Reno, Vegas and Westwood+ TCP when several connections share a FIFO bottleneck. To the purpose, the scenario in Fig. 2 has been considered. It consists of a single FIFO bottleneck shared by a set of *M* TCP flows of the same flavor with different *RTTs*. In particular, we consider *M*=10,40,70,100,120,140,170,200 infinite greedy connections with *RTTs* ranging uniformly from (252/*M*)ms to 252ms. Simulations last 100s during which all the TCP sources send data.

The bottleneck bandwidth is set equal to 1Mbps or 10Mbps or 100Mbps and the bottleneck queue capacity is set equal to the link capacity times the maximum RTT, hence, it is equal to 21, 210 and 2100 segments, respectively. The segment size is 1500 bytes long. All the considered sinks implements the delayed ACK option [13],[22].

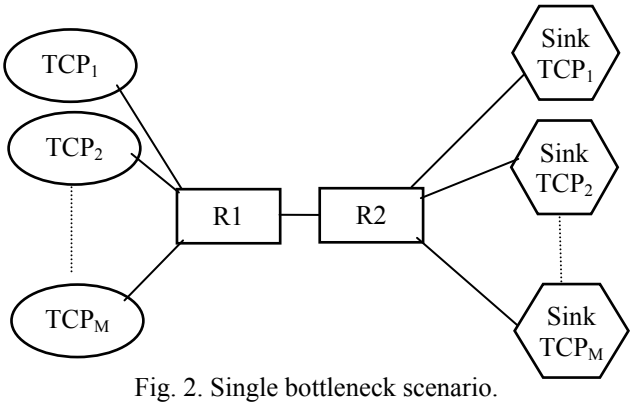

Fig. 2. Single bottleneck scenario.

To measure the fairness in bandwidth sharing achieved by each TCP flavor, we compute the Jain's fairness index defined as follows:

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

$$F.I. = \frac{(\sum_{i=1}^{M} b_i)^2}{M \sum_{i=1}^{M} b_i^2}$$

where b_i is the goodput of the i^{th} connection and M is the number of connections sharing the bottleneck [17]. Figs. 3, 4 and 5 show the fairness indexes of Reno, Vegas and Westwood over a single bottleneck of capacity 1Mbps or 10Mbps or 100Mbps, respectively.

Fig. 3 shows that Vegas exhibits a very poor fairness index with respect to Reno and Westwood+ over the 1Mbps bottleneck. The main reason of this behavior is that the early congestion detection employed by Vegas fails because of the small queue size of 21 packets. On the other hand the Jain fairness indexes of Reno and Westwood+ TCP decrease when the number of connections M increases from 40 to 200. Such a behavior is due to the *many flow effect*, which happens when the number of connections sharing a bottleneck is larger than the size of the bottleneck queue measured in segments [8].

Fig. 4 shows that when the number of connections M is larger than 40, Westwood+ exhibits the best fairness index whereas Vegas has the worst fairness index since its congestion detection mechanism does not work properly because the buffer size is not large enough.

Fig. 5 show that Vegas achieves the best fairness index since its congestion detection algorithm is able to avoid buffer overflow. Fig. 5 shows that Westwood+ algorithm improves the fairness index of Reno TCP over the 100Mbps link.

Figs. 6, 7 and 8 show the average goodput computed as the total goodput over the number of connections sharing the bottleneck when the bottleneck capacity is 1Mbps or 10Mbps or 100Mbps, respectively. From Figs. 6, 7, and 8, it turns out that in all the considered scenarios, Reno, Vegas and Westwood+ achieve high link utilization.

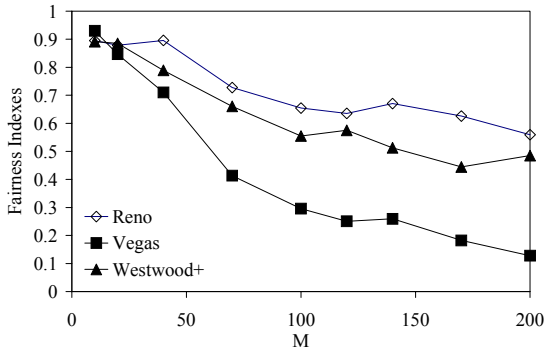

Fig. 3. Fairness Indexes over a 1Mbps bottleneck.

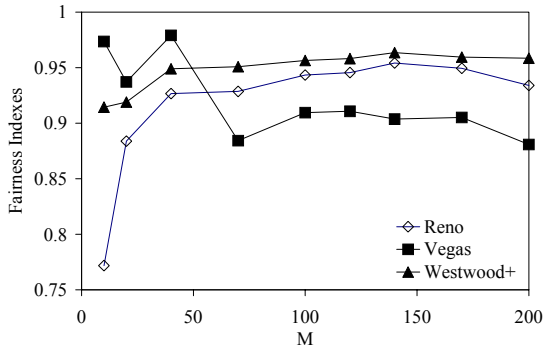

Fig. 4. Fairness Indexes over a 10Mbps bottleneck.

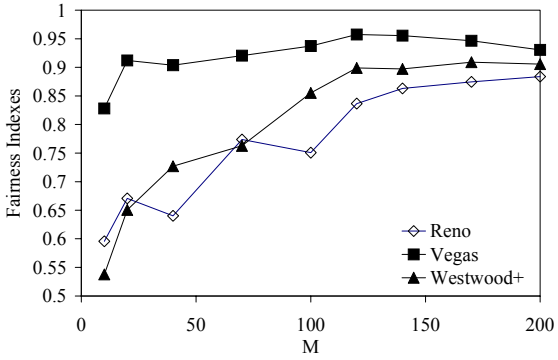

Fig. 5. Fairness Indexes over a 100Mbps bottleneck.

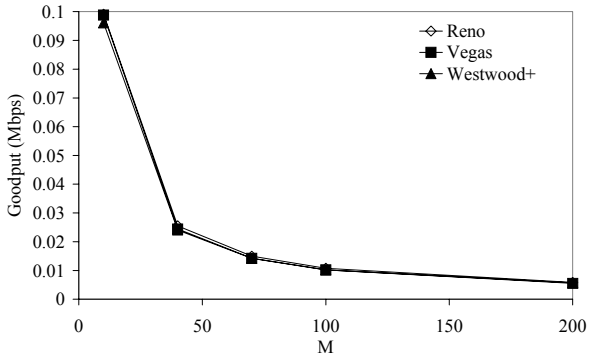

Fig. 6. Average Goodput over a 1Mbps bottleneck.

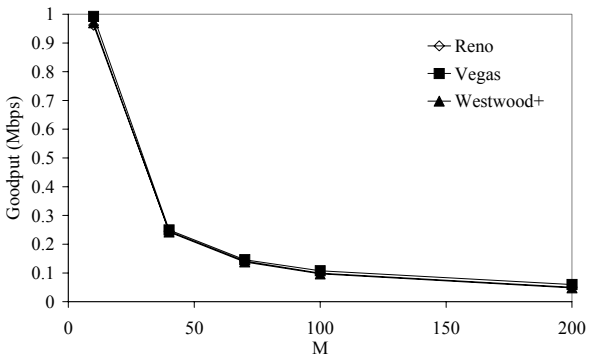

Fig. 7. Average Goodput over a 10Mbps bottleneck.

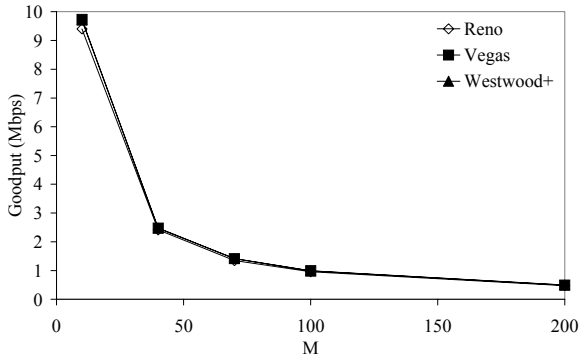

Fig. 8. Average Goodput over a 100Mbps bottleneck.

Finally, this section investigates the performances of the three TCP flavor when the bottleneck buffer size is varied. In particular, the goodput versus the (buffer_size/bandwdith*delay) ratio in the case of a 10Mbps link shared by 40 TCP flows is shown in Fig. 9. It is worth noting that Vegas achieves the best goodput since it is able to avoid segment losses via the early congestion detection mechanism that it employs, whereas Westwood+ and Reno TCP exhibit about the same goodput.

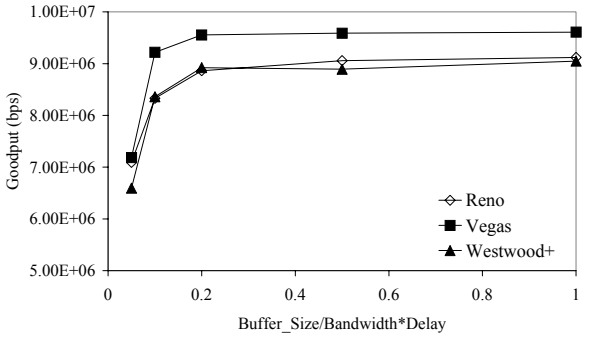

Fig. 9. Goodput over a 10Mbps bottleneck with variable buffer size.

3.2 Multi bottleneck scenario

The previous section has shown that the early congestion detection mechanism employed by Vegas TCP guarantees high fairness in bandwidth sharing when the queue size of the bottleneck is large enough. However, the congestion detection mechanism employed by Vegas causes throughput degradation when a Vegas flow shares the same bottlenecks with Reno flows. This section aims at evaluating the performances of Reno, Vegas and Westwood+ TCP over a multi bottleneck scenario in the presence of Reno traffic sources. To the purpose, the topology depicted in Fig. 10 has been considered.

The topology in Fig. 10 is characterized by: (1) N hops; (2) one greedy TCP connection C_1 going through all the N hops; (3) $2N$ cross traffic greedy TCP sources C_2-C_{2N+1} transmitting data over each single hop. The simulation lasts 110s during which the cross traffic sources always send data. The connection C_1 starts data transmission at the time t=10s when all the network bandwidth has been grabbed by the cross traffic. The segments are 1500 bytes long. The round trip times of all the connections are set equal to 50ms. The link bandwidth between the routers is equal to 10Mbps and the router queues are set equal to the link capacity times the round trip time.

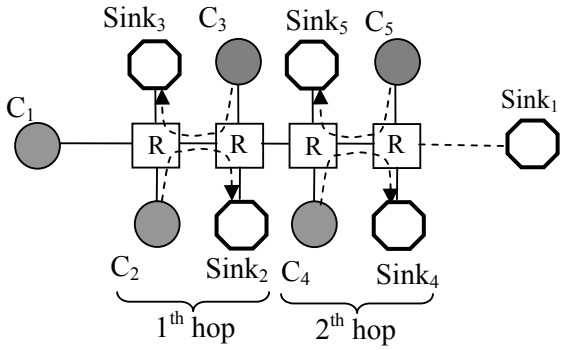

Fig. 10. Multi bottleneck topology.

Fig. 11 compares the friendliness of Reno with respect to Westwood+ and Vegas. It shows the goodputs of the connection C_1 as a function of the number of traversed hops; the C_1 source is controlled by Reno or Vegas or Westwood and the cross traffic is contributed by Reno sources. Fig. 11 clearly highlights that Westwood+ and Reno achieve about the same goodput, whereas Vegas is not able to grab its own share of bandwidth. Similar results have been obtained when the Westwood+ algorithm controls the cross sources C_2-C_{2N+1}.

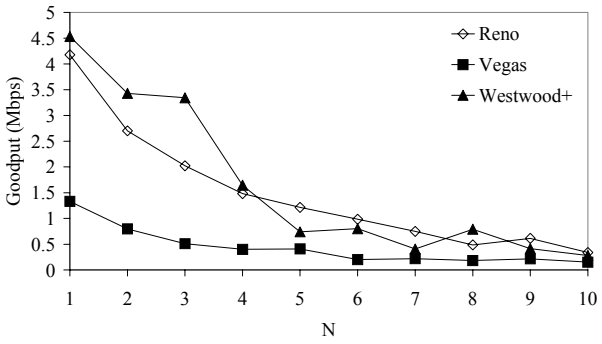

Fig. 11. Goodput vs. number of traversed hops.

4. Conclusions

We have investigated a simulation-based comparison of Reno, Vegas and Westwood+ TCP. Simulation results have shown that Westwood+ TCP fairly behaves when interacting with Reno

TCP whereas, on the other hand, Vegas is not able to get its bandwidth share when coexisting with Reno. Moreover, simulations have shown that Westwood+ TCP improves the fairness in bandwidth sharing with respect to Reno TCP.

References

[1] V. Jacobson, Congestion Avoidance and Control, *ACM Computer Communications Review*, Vol.18, No.4, 1988, pp. 314 – 329.

[2] V. Jacobson, Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno, in Proceedings of *the 18th Internet Engineering Task Force*, University of British Colombia, Vancouver, BC, Sept. 1990.

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

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

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

[6] 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, No.1, 1989, pp. 1-14.

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

[8] R. Morris, TCP behavior with Many Flows, *IEEE International Conference on Network Protocols*, October 1997, Atlanta, Georgia, pp. 205-211.

[9] L.S. Brakmo, S.W. O'Malley, and L. Peterson, TCP Vegas: End-to-end congestion avoidance on a global Internet, *IEEE Journal on Selected Areas in Communications (JSAC),* Vol.13, No.8, 1995, pp. 1465-1480.

[10] S. H. Low, L. Peterson and L. Wang, Understanding Vegas: A Duality Model, in Proceedings of *ACM Sigmetrics*, Boston, MA, June 2001.

[11] J. Mo, R. J. La, V. Anantharam, J. Walrand, Analysis and comparison of TCP Reno and Vegas, in Proceedings of *IEEE Infocom* 1999.

[12] 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, *ACM Mobicom 2001*, July, Rome, Italy. To appear in *ACM Wireless Networks (WINET)*, Special Issue on Wireless Networks with selected papers from MOBICOM 2001.

[13] L. Peterson & B. Davie, *Computer Networks a Systems Approach*, Morgan Kaufmann, 1996.

[14] L.A. Grieco, 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, April 22-24, 2002 Berlin, Germany.

[15] L.A. Grieco, S. Mascolo, R. Ferorelli, Additive Increase Adaptive Decrease Congestion control: a mathematical model and its experimental validation, to appear on Proceedings of *IEEE Symposium on Computers and Communications*, Taormina, Italy, July 1-4, 2002.

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

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

[18] D. Clark, The design philosophy of the DARPA Internet protocols, in Proceedings of *ACM Sigcomm'88* in *ACM Computer Communication Review*, Vol.18, No.4, 1988, pp. 106 - 114.

[19] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, *IEEE/ACM Transactions on Networking*, Vol.7, No.4, 1999, pp. 458-72.

[20] S. Mascolo, Congestion control in high-speed communication networks, *Automatica*, Special Issue on Control Methods for Communication Networks, Dec. 1999.

[21] T.V. Lakshman and U. Madhow, The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss, *IEEE/ACM Transactions on Networking*, Vol.5, No.3, 1997.

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

[23] S. Keshav, A Control-theoretic Approach to Flow Control, in Proceedings of *ACM Sigcomm* 1991.

[24] J. C. Hoe, Improving the Start-up Behavior of a Congestion Control Scheme for TCP, in Proceedings of *ACM Sigcomm'96.*

[25] M. Allman and V. Paxson, On Estimating End-to-End Network Path Properties, in Proceedings of *ACM Sigcomm* 1999.

[26] K. Lai and M. Baker, Measuring Link Bandwidths Using a Deterministic Model of Packet Delay, in Proceedings of *ACM Sigcomm* 2000.