

End-to-End Bandwidth Estimation Algorithms for Westwood TCP Congestion Control

Luigi A. Grieco and Saverio Mascolo

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

Abstract. Westwood TCP improves efficiency of classic TCP congestion control by introducing an innovative end-to-end bandwidth estimation algorithm. It leaves unchanged the probing phase of classic TCP but it substitutes the *multiplicative decrease* phase with an *adaptive decrease* phase, which sets the control windows by taking into account the bandwidth estimate. This paper shows that bunched ACKs can heavily affect the bandwidth estimate and therefore it is necessary to implement an anti ACK compression algorithm to properly filter out their effects. Moreover, a comparison of time-invariant and time-varying discrete filters to be used after the anti ACK compression algorithm is developed.

1 Introduction

Today TCP congestion control is based on the additive increase/multiplicative decrease probing paradigm (AIMD) [1,2]. The *additive increase* phase linearly increases the transmission rate until the network capacity is hit and the sender becomes aware of congestion via the reception of duplicate acknowledgments (DUPACKs) or the expiration of a timeout. Then the sender reacts to 3 DUPACKs reception by halving the congestion window (fast recovery) and sending again the missing packet (fast retransmit), and to timeout expiration by reducing the congestion window to one (*multiplicative decrease phase*). After a timeout or at the beginning of the connection, the TCP enters a faster increasing phase that is exponential and aims at grabbing the best-effort available bandwidth faster. This mechanism can insure that network capacity is not exceeded, but it cannot insure fair sharing of that capacity [1,13].

Westwood TCP (TCP-W) improves fairness and efficiency of congestion control by introducing an innovative end-to-end bandwidth estimation algorithm. In particular, Westwood TCP leaves unchanged the probing phase of classic TCP but it substitutes the multiplicative decrease phase with an adaptive decrease phase, which sets the control windows by taking into account the bandwidth estimate. It has been shown that Westwood TCP increases the TCP throughput over wireless links, increases the fairness in bandwidth allocation w.r.t. Reno TCP and is friendly towards Reno TCP [12,19].

It is worth remarking that the end-to-end bandwidth estimation mechanism proposed in

Westwood TCP, and that we are going to investigate in this paper, is built on the standard probing mechanism of today TCP. A congestion episode at the end of a probing phase points out that all the best-effort available bandwidth has been grabbed. Therefore, an estimate of the used bandwidth at the end of a probing phase is, by definition, an estimate of the available best effort bandwidth in a statistically multiplexed packet network. The end-to-end bandwidth estimation algorithm is a critical issue. The estimation scheme used by Westwood is much more different from the algorithms proposed and evaluated in [3,6,7,11,16,21], where, out of band probing packet streams are injected into the networks to measure the available bandwidth.

The basic idea reported in [12] is to get an estimate of the used bandwidth by counting and low-pass filtering the flow of ACK packets during the data transfer. In particular, when an ACK arrives it can be argued that a certain amount of data has been delivered since the previous ACK arrival time and a sample of the used bandwidth can be computed. Bandwidth samples must be low-pass filtered to obtain the low frequency components of the used bandwidth because congestion is only due to the low-frequency components of the traffic Φ . Low-pass filtering using discrete-time filter is a challenging goal due to the fact that ACK packets do not arrive at constant sampling intervals since packets networks are asynchronous systems. In particular, ACK packets experience congestion along the backward path they traverse and arrive bunched. The latter phenomena, known as ACK compression [14], provokes considerable bandwidth overestimate that is disruptive of the adaptive decrease mechanism and leads to connection starvation. This paper is entirely devoted to the challenging issue of end-to-end bandwidth estimation in packet networks. In particular it is shown that: (1) ACK compression generates aliased bandwidth samples that lead to greatly overestimate the used bandwidth; (2) it is necessary to implement an anti ACK compression algorithm, which plays the role of a classic anti-aliasing filter, before using any discrete-time filter in packet networks. Moreover, time-invariant and time-varying discrete filters are compared. Simulation results show that it is possible to obtain an estimate of available

bandwidth which enhances the performance of TCP congestion control.

The paper is organized as follows: Section 2 describes the Westwood TCP congestion control algorithm; Section 3 focuses on time-varying and time-invariant discrete-time filters for low-pass filtering and bandwidth estimation; Section 4 introduces the anti-aliasing filter and test time-invariant and time-varying discrete-time filters using the ns-2 simulator [15]; finally, Section 5 draws the conclusions.

2 Westwood TCP overview

TCP-W is a sender-side only modification of the TCP stack. It exploits an end-to-end bandwidth estimation algorithm that is built on the standard probing mechanism of today TCP. It implements classic slow-start and congestion avoidance phases to probe the network but, after congestion, it employs the estimate of the best-effort available bandwidth B to adaptively set the congestion window and the slow-start threshold. In particular, when a TCP-W sender receives 3 DUPACKs, it sets both $ssthresh$ and $cwnd$ equal to $\max[2, (B \cdot RTT_{min}) / seg_size]$, where RTT_{min} is the minimum measured RTT and seg_size is the size of the sent segments. On the other hand, when a timeout expires, $ssthresh$ is set as in the previous case whereas $cwnd$ is set equal to 1 segment.

It is important to notice that the setting $ssthresh = (B \cdot RTT_{min}) / seg_size$ provides a slow-start threshold that follows exactly the best-effort available bandwidth as it is computed by the TCP Westwood sender (RTT_{min} / seg_size is a scale factor). Therefore, the effective ability of Westwood TCP to track the available bandwidth can be tested by plotting the $ssthresh$.

The *adaptive decrease* mechanism improves both stability and fairness with respect to the standard TCP *multiplicative decrease* algorithm. In fact the setting $cwnd = B \cdot RTT_{min}$ sustains a transmission rate $cwnd / RTT = B \cdot RTT_{min} / RTT$ that is less than the bandwidth B estimated at the time of congestion: as a consequence, the considered TCP flow clears out its path backlog after a congestion episode thus leaving room for coexisting flows, and improving stability, statistical multiplexing and fairness.

3 Low pass filters for bandwidth estimation

This section focuses on the issue of estimating the used bandwidth by counting ACK packets and by filtering the information they convey. In particular,

when an ACK is received at time t_k , it means that a certain amount of data d_k has been received by the TCP receiver. In particular, on ACK reception, the following *sample* of bandwidth used by the TCP connection can be computed:

$$b_k = \frac{d_k}{t_k - t_{k-1}} = \frac{d_k}{\Delta_k} \quad (1)$$

where t_{k-1} is the time the previous ACK was received and $\Delta_k = t_k - t_{k-1}$ is the last interarrival time. Since congestion is due to low-frequency components of used bandwidth [9], samples (1) must be low-pass filtered by using a discrete-time filter. The latter is a delicate task because samples (1) contain high frequency components due to the fact that ACK packets can come back to the sender bunched as well as equally spaced. As a consequence, any discrete-time low-pass filter will remarkably overestimate the available bandwidth due to the phenomena of aliasing. In this section we will first show through simulations that both time-invariant and time-varying discrete-time filters fail to estimate the used bandwidth due to aliasing effects. Then, we will introduce an anti ACK-compression algorithm that plays the classic role of an anti-aliasing filter in packet networks [10].

In order to evaluate the effectiveness of the proposed approach, we employ a single bottleneck topology shown in Fig. 1, which consists in a duplex 5Mbps bottleneck link shared by one persistent TCP Westwood (TCP-W) source implementing the New Reno feature [17], and one ON/OFF constant bit rate UDP source transmitting data at 1Mbps during the ON period that lasts 100s and is silent during the OFF period that lasts 100s too. On the reverse path, 40 TCP-W sources implementing the New Reno option provoke congestion along the TCP Westwood ACK path and excite ACK compression. Packets are 1500 Bytes long whereas ACKs are 40 Bytes long. The bottleneck buffer size is set equal to the link capacity times the round trip time of the forward TCP connection whose value is 250ms. All TCP sinks implement the delayed ACK option [2]. The initial congestion window has been set equal to 3 segments [18].

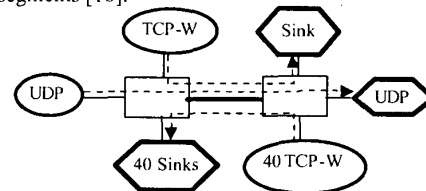


Fig. 1. Single bottleneck scenario to test bandwidth estimation algorithms.

In the first set of simulations, the UDP source is turned OFF to test various estimation algorithms in the presence of constant available bandwidth environment, then it is turned ON to investigate the estimate behavior in the presence of time varying available bandwidth.

3.1 A time-varying discrete time filter using a ZOH

The first filter we consider is obtained discretizing a first order low-pass continuous filter with time constant τ by using a zero order holder (ZOH). Such discretization lead to the following discrete time filter [10]:

$$\hat{b}_k = \alpha_k \cdot \hat{b}_{k-1} + (1 - \alpha_k) \cdot b_k \quad (2)$$

where $\alpha_k = e^{-\Delta_k / \tau}$. The Infinite Impulse Response filter (2) is identical to the one proposed in [8] to estimate the arrival rate of a flow at network edges. Fig. 2(a) plots the used bandwidth samples and Fig. 2(b) the used bandwidth estimate obtained using a TCP-W sender implementing the filter (2). The filter time constant τ has been set equal to 0.1s to better highlight the overestimate of the available bandwidth.

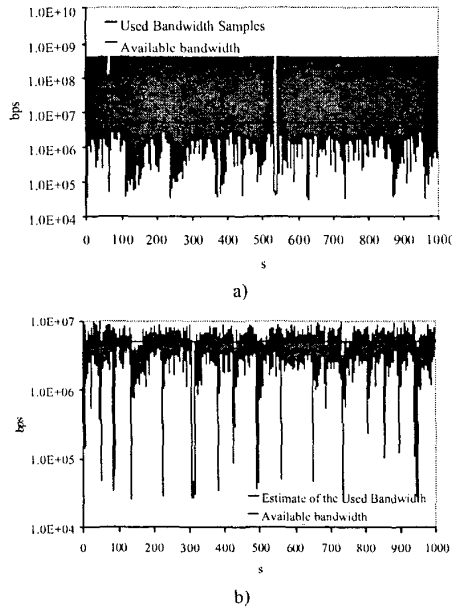


Fig. 2. Used Bandwidth: (a) Samples; (b) Output of the filter (2).

It is worth noticing that the filter (2) overestimates the available bandwidth due to aliasing effects. In order to illustrate high frequency components due to ACK compression, Fig. 2(a) shows bandwidth samples feeding the filter (2). It should be noticed that bandwidth samples exhibit a peak value equal to 375 Mbps. Such a value is equal to the returning link capacity (5Mbps) times the ratio between the segments size (1500Bytes) and the ACKs size (40Bytes), times two to take into account that each ACK acknowledges two segments because of the delayed ACK option [2].

3.2 A discrete time invariant filter

By considering a constant interarrival time $\Delta = \Delta_k$, the filter (2) gives the following time-invariant filter:

$$\hat{b}_k = \alpha \cdot \hat{b}_{k-1} + (1 - \alpha) \cdot b_k \quad (3)$$

where

$$\alpha = e^{-\Delta / \tau} \quad (4)$$

Fig. 3 plots the output of the filter (3) with $\alpha = 0.9$ when fed by these samples. It shows that the filter overestimates the available bandwidth up to more than 10 times. The reason is that bandwidth samples contain aliased frequency components due to ACK compression.

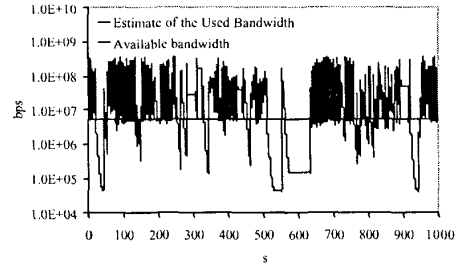


Fig. 3. Estimate of the used bandwidth using the filter (3).

3.3 A time varying filter using the bilinear transformation

As in the previous section, a sample of used bandwidth (1) is computed every time t_k the sender receives an ACK. By assuming, for the time being, that inter-arrival time between samples is uniform and equal to Δ , the continuous first order low-pass filter is discretized using the bilinear transformation [10], which leads to the following discrete-time filter:

$$\hat{b}_k = \frac{2\tau - \Delta}{2\tau + \Delta} \hat{b}_{k-1} + \Delta \frac{b_k + b_{k-1}}{2\tau + \Delta} \quad (5)$$

3.3.1 The filter of Westwood TCP

To take into account that the inter-arrival time of bandwidth samples is not uniform, the following time varying form of the filter (5) has been employed in Westwood TCP [12]:

$$\hat{b}_k = \frac{2\tau - \Delta_k}{2\tau + \Delta_k} \hat{b}_{k-1} + \Delta_k \frac{b_k + b_{k-1}}{2\tau + \Delta_k} \quad (6)$$

where Δ_k is the inter-arrival time between the $(k-1)$ -th sample and the k -th sample. In order to satisfy the Nyquist-Shannon sampling Theorem, the inter-arrival time Δ_k must be less than $\tau/2$. Therefore if it happens that $\Delta_k > \tau/2$, then the filter is fed by $N = \text{integer of } (2 \cdot \Delta_k / \tau)$ virtual samples with inter-arrival time $\tau/2$ and amplitude equal to 0. Moreover, an additional samples b_k feeds the filter with inter-arrival time equal to $\Delta_k - N \cdot \tau/2$. In the following, we set $\tau = 1s$ unless otherwise specified. We have tested the filter (6) over the single bottleneck scenario already depicted. Results are very similar to those obtained for filter (3) and show that filter (6) greatly overestimates the bandwidth because of ACK compression.

4 Anti Aliasing Filter

We have seen that ACK compression generates bandwidth samples containing aliased frequency components that cannot be low-pass filtered using a digital filter. Therefore, similarly to the case of filtering an analog signal, which requires the analog signal be pre-filtered by an anti-aliasing filter before using a discrete-time filter, also in packet networks it is necessary to implement a sort of anti-aliasing filter. The basic idea to implement an anti-aliasing filter in packet networks is to group the ACKs received in a sufficiently large time interval T and compute a unique corresponding bandwidth sample. This operation has the effect of smoothing single ACKs, which corresponds to filter out high frequency components. In particular, ACKs received during the last RTT are grouped into a unique bandwidth sample that is computed by considering the total data acknowledged over the RTT . To show that the proposed algorithm avoids ACK compression effects, we consider the same scenario investigated in the previous simulations. It is worth noting that the anti-aliasing algorithm generates one bandwidth sample for each RTT . In order to employ the time-varying filter (6), it is still necessary to satisfy the Nyquist-Shannon sampling theorem, that is, it must be $\Delta_k \leq \tau/2$ [10]. To be conservative, we assume $\Delta_k \leq \tau/4$. Thus, if it happens that $\Delta_k > \tau/4$ then we interpolate and re-sample by creating $N = \text{integer}(4 \cdot \Delta_k / \tau)$ virtual samples b_k

that arrive with the inter-arrival time $\tau/4$ and one more sample b_k arriving with inter-arrival time $\Delta T = \Delta_k - N \cdot \tau/4$. In the following we will always assume $\tau = 1s$. It should be noticed that filter (2) already implements interpolation and resampling through the ZOH that keeps constant the signal during the last interarrival time. However, the coefficient of the filter (2) requires the computation of an exponential that is harder than computing a ratio as in the case of filter (6). In the following of the paper we will test the behavior of filters (3), (6) and (2) after an anti-aliasing filtering stage.

4.1 Bandwidth estimates over a static scenario

We consider the scenario pictured in Fig. 1. Fig. 4(a) shows the bandwidth samples computed using the anti-aliasing procedure, whereas Fig. 4(b) shows the output of the filter (3) which is fed by anti-aliased samples. A comparison of Fig. 4(a) and Fig. 2(a) shows that now there is no overestimate of the bandwidth samples due to aliasing. Once the bandwidth samples are obtained as shown in Fig. 4(a), a further stage of discrete-time filtering extracts the low frequency components of the used bandwidth as it is shown in Fig. 4(b). Analogous results have been obtained with filters (6) and (2).

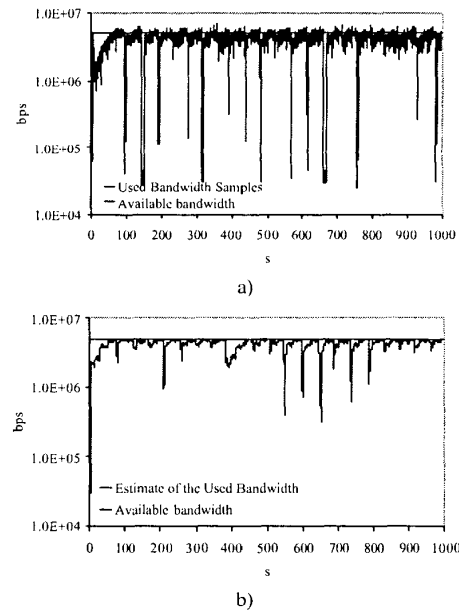


Fig. 4. Used bandwidth: (a) Anti-aliased samples; (b) Output of the filter (3).

4.2 Bandwidth estimates over a dynamic scenario

In this section we investigate the behavior of the three low pass filters introduced above in the presence of a time-varying available bandwidth when bandwidth samples are properly prefiltered using the anti-aliasing procedure. To the purpose, the scenario in Fig.1 has been considered with the UDP source turned ON. The *RTT* of the considered TCP connection is 100ms. At first, the reverse traffic has been turned OFF to observe the output of various filters without disturbances, and then the reverse traffic has been turned ON to test the filters in a realistic scenario with ACK compression.

Figs. 5(a) and (b) report the estimate of the used bandwidth obtained using the filter (6) with and without reverse traffic, respectively. As it can be viewed, the reverse traffic affects the estimate of the used bandwidth because of ACK congestion.

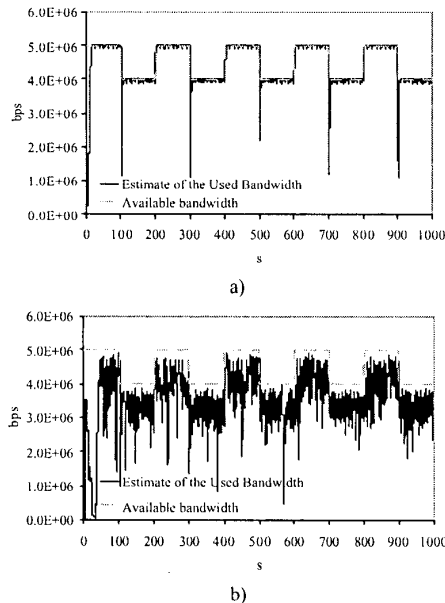


Fig. 5. Estimate of the used bandwidth with filter (6): (a) without reverse traffic; (b) with reverse traffic.

However, it should be considered that Westwood TCP employs the estimate of the used bandwidth for setting the *slow start threshold* (*ssthresh*) only after a congestion episode, that is when the used bandwidth represents the best effort available bandwidth. For this reason, the effect of reverse traffic is much weaker on the *ssthresh* dynamics, which represents the estimate of the best-effort available bandwidth. This is shown in Fig. 6. Similar

results have been obtained by employing the filters (3) and (2).

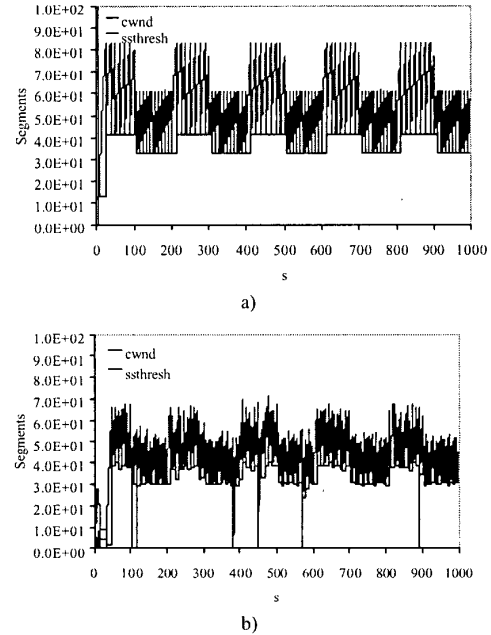


Fig. 6. Congestion window and slow start threshold with filter (6): (a) without reverse traffic; (b) with reverse traffic.

It is interesting to compare the *ssthresh* of Westwood TCP w.r.t. the *ssthresh* of Reno TCP. Figs. 7 and 8 show *cwnd* and *ssthresh* obtained with Reno TCP when the reverse traffic sources are turned OFF and ON, respectively.

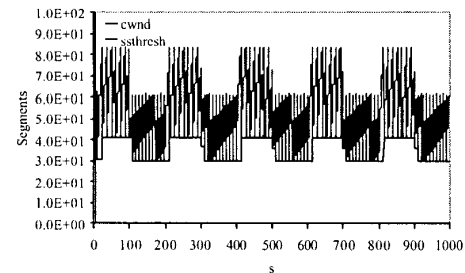


Fig. 7. Congestion window and slow start threshold of Reno when the reverse traffic sources are turned OFF.

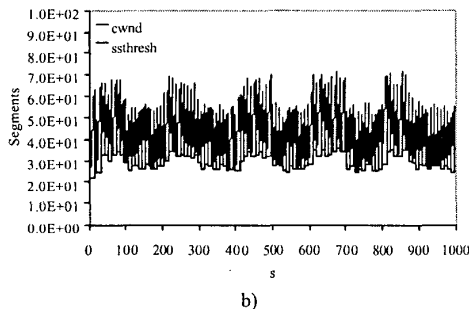


Fig. 8. Congestion window and slow start threshold of Reno when the reverse traffic sources are turned ON.

The comparison with *ssthresh* of Westwood (see Fig. 6) highlights that the proposed bandwidth estimate enhances the ability of TCP congestion control to match the available bandwidth, which provides better utilization of network capacity.

5 Conclusions

This work has shown that: (1) ACK compression generates aliased bandwidth samples that lead to greatly overestimate the used bandwidth; (2) it is necessary to implement an anti ACK compression algorithm, which plays the role of a classic anti-aliasing filter, before using any discrete-time filter in packet networks. Simulation results have shown that anti-aliasing plus low-pass discrete time filtering provide a reliable estimate of the used bandwidth that, when coupled with a probing congestion control algorithm, gives also a reliable estimate of the best-effort available bandwidth.

References

1. Jacobson, V.: Congestion avoidance and control, in Proceedings of ACM Sigcomm '88, Stanford CA, August (1988) 314–329.
2. Allman, M., Paxson, V. and Stevens W.R.: TCP congestion control, RFC 2581, April 1999.
3. Melander, B., Bjorkman, M. and Gunningberg, P.: A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks, in Proceedings of Global Internet Symposium, 2000.
4. Clark, D.: The design philosophy of the DARPA Internet protocols, in Proceedings of ACM Sigcomm '88, Stanford CA, August (1988) 106–114.
5. Floyd, S., Fall, K.: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking, Vol. 7(4), (1999), 458–472.
6. Hoe, J. C.: Improving the Start-up Behavior of a Congestion Control Scheme for TCP, in Proceedings of ACM Sigcomm'96, Palo Alto, CA, August (1996) 270–280.
7. Jain, M., Dovrolis, C.: End to End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput, in Proceedings of ACM Sigcomm 2002.
8. Stoica, I., Shenker, S. and Zhang, H.: Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks, in Proceedings of ACM Sigcomm '98, Vancouver, Canada, August (1998) 118–130.
9. Li, S. Q., and Hwang, C.: Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks, IEEE/ACM Transaction on Networking, Vol. 3(1), (1995) 10–25.
10. Åström, K. J. and B. Wittenmark (1997). Computer controlled systems, Prentice Hall, Englewood Cliffs, N. J, 1995.
11. Allman, M. and Paxson, V.: On Estimating End-to-End Network Path Properties, in Proceedings of ACM Sigcomm 1999, Cambridge, Massachusetts, August (1999) 263–276.
12. Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., Wang, R.: TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks, in Proceedings of ACM Mobicom 2001, Rome, Italy, July (2001).
13. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation, in Proceedings of ACM Sigcomm 1998, Vancouver BC, Canada, September (1998) 303–314.
14. Mogul, J. C.: Observing TCP dynamics in real networks, in Proceedings of ACM Sigcomm 1992, 305–317.
15. Ns-2 network simulator (ver 2). LBL, URL: <http://www-mash.cs.berkeley.edu/ns>.
16. Lai, K. and Baker, M.: Measuring Link Bandwidths Using a Deterministic Model of Packet Delay, in Proceedings of ACM Sigcomm 2000, Stockholm, Sweden, August (2000) 283–294.
17. Floyd, S., Henderson, T.: NewReno Modification to TCP's Fast Recovery, RFC 2582, April 1999.
18. Allman, M., Floyd, S., Partridge, C.: Increasing initial TCP's initial window, RFC 2414, September 1998.
19. Grieco, L. A., and Mascolo, S.: Westwood TCP and easy RED to improve Fairness in High Speed Networks, in Proceedings of IFIP/IEEE Seventh International Workshop on Protocols For High-Speed Networks, PfHNS02, Berlin, Germany, April (2002) 130–146.
20. Chaskar, H.M., Lakshman, T.V. and Madhow, U.: TCP Over Wireless with Link Level Error Control: Analysis and Design Methodology, IEEE/ACM Transactions on Networking, Vol. 7(5), (1999). 605–615.
21. Keshav, S.: A Control-theoretic Approach to Flow Control, in Proceedings of ACM Sigcomm 1991, Zurich, Switzerland, September (1991) 3–6.