End-to-End Bandwidth Estimation for Congestion Control in Packet Networks

Luigi Alfredo Grieco[1] and Saverio Mascolo[2]

[1] Dipartimento d' Ingegneria dell' Innovazione, Università di Lecce
Via Monteroni, 73100 Lecce, Italy
alfredo.grieco@unile.it

[2] Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari
Via Orabona 4, 70125 Bari, Italy
mascolo@poliba.it

Abstract. Today TCP/IP congestion control implements the *additive increase/multiplicative decrease* (AIMD) paradigm to probe network capacity and obtain a "rough" but robust measurement of the best effort available bandwidth. Westwood TCP proposes an *additive increase/adaptive decrease* paradigm that adaptively sets the transmission rate at the end of the probing phase to match the bandwidth used at the time of congestion, which is the definition of best-effort available bandwidth in a connectionless packet network. This paper addresses the challenging issue of estimating the best-effort bandwidth available for a TCP/IP connection by properly counting and filtering the flow of acknowledgments packets using discrete-time filters. We show that in order to implement a low-pass filter in packet networks it is necessary to implement an anti ACK compression algorithm, which plays the role of a classic anti-aliasing filter. Moreover, a comparison of time-invariant and time-varying discrete filters to be used after the anti-aliasing algorithm is developed.

1 Introduction

The today dominant Internet is a global packet network that implements resource sharing through statistical multiplexing. Packets are delivered hop by hop by a connectionless network layer that employs store and forward switching using *first in first out* (FIFO) queuing. The stability of the Internet and in particular the prevention of congestion requires that flows use some form of end-to-end congestion control to adapt the input rate to the available bandwidth [1,2,4,5]. Thus the goal of obtaining an end-to-end estimate of the bandwidth available for a TCP connection is crucial in order to design more efficient and fair congestion control algorithms.

Today TCP congestion control was introduced in the late eighties [1,2] and follows the additive increase/multiplicative decrease probing paradigm (AIMD) [6]. The additive 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 light congestion (i.e. 3 DUPACKs) by halving the congestion window (fast recovery) and sending again the missing packet (fast retransmit), and to heavy congestion (i.e. timeout) 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.

The AIMD paradigm can be viewed as an endless series of cycles, having a linear or exponential increasing phase and a multiplicative decreasing phase, which aim at continuously probing the network capacity to obtain a "rough" but robust measurement of the best effort bandwidth that is available for a TCP connection. As reported in [1], "This mechanism can insure that network capacity is not exceeded, but it cannot insure fair sharing of that capacity". In fact, it has been shown that the throughput of a TCP connection is proportional to the inverse of its round trip time [13], which favors shorter connections. Furthermore, today TCP is not well suited for wireless links since losses due to unreliable radio channels are misinterpreted as a symptom of congestion thus leading to an undue reduction of the transmission rate and low utilization of a wireless path. As a consequence, today TCP requires supplementary link layer protocols such as reliable link-layer or split-connections approach to efficiently operate over wireless links [20].

Westwood TCP proposes to improve 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, that is, it is based on probing the network capacity using the slow-start and congestion avoidance phases. 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 latter observation is valuable to be kept in mind because the used bandwidth exhibits high variability with time in packet networks due to statistical multiplexing, whereas the available best-effort bandwidth might not.

The end-to-end bandwidth estimation algorithm is a critical issue. 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 [9]. 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.

The disruptive effects of ACK compression on the bandwidth estimation algorithm were not evident in the original paper on TCP Westwood because the phenomena of ACK compression was negligible in the considered scenarios. They were shown in [19] along with a new filtering technique and a mathematical model for the long-term Westwood TCP throughput.

This paper is entirely devoted to the challenging issue of end-to-end bandwidth estimation in packet networks. In particular it is shown that ACK compression generates aliased bandwidth samples that lead to greatly overestimate the used bandwidth. Therefore, 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. 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 summarizes the related work; Section 3 describes the Westwood TCP congestion control algorithm; Section 4 focuses on time-varying and time-invariant discrete-time filters for low-pass filtering and bandwidth estimation; Section 5 introduces the anti-aliasing filter and test time-invariant and time-varying discrete-time filters using the ns-2 simulator [15]; finally, Section 6 draws the conclusions.

2 Related Work

TCP Vegas is the first significant example of congestion control algorithm that partially departs from the AIMD paradigm by proposing two estimates: (a) the expected connection rate $cwnd/RTT_{min}$ and (b) the actual connection rate $cwnd/RTT$, where RTT is the round trip time and RTT_{min} is the minimum measured RTT [11]. When the *difference* between the expected and the actual rate is less than a threshold $\alpha>0$, the *cwnd* is additively increased. When the *difference* is greater than a threshold $\beta>\alpha$ then the *cwnd* is additively decreased. When the difference is between α and β, *cwnd* is kept constant [11]. Vegas can be viewed as the first attempt to use a bandwidth estimation scheme to improve the internet congestion control. However, it should be noted that Vegas employs the actual sending rate $cwnd/RTT$ rather than the rate of the returning ACKs to infer congestion; the sending rate $cwnd/RTT$ is a measure of bandwidth that is based on the number of sent packets (*cwnd*) and not on the number of acknowledged packets. As a consequence, it does not take into account that a fraction of sent packets could be lost and could not correspond to an actual bandwidth capacity, that is, the Vegas actual rate overestimates the used bandwidth.

The first attempt to exploit ACK packets for bandwidth estimation is the packet pair (PP) algorithm, which tries to infer the bottleneck available bandwidth at the starting of a connection by measuring the interarrival time between the ACKs of two packets that are sent back to back [21]. Hoe proposes a refined PP method for estimating the available bandwidth in order to properly initialize the *ssthresh* [22]: 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 [23]. Lai and Baker propose an evolution of the PP algorithm for measuring the link bandwidth in FIFO-queuing networks [16]. The method consumes less network bandwidth while maintaining approximately the same accuracy of other methods, which is poor for paths longer than few hops. Jain and Dovrolis proposes to use streams of probing packets to measure the end-to-end available bandwidth, which is defined as the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic [7]. Finally, they focus on the relationship between the available bandwidth in a path they measure and the throughput of a persistent TCP connection. They show that the averaged throughput of a TCP connection is about 20-30% more than the available bandwidth measured by their tool due to the fact that the TCP probing mechanism gets more bandwidth than what was previously available in the path, grabbing part of the throughput of other connections. A similar technique has been proposed in [3]. It uses sequences of packet pairs at increasing rates and estimates the available bandwidth by comparing input and output rates of different packet pairs. Estimating the available bandwidth at the beginning of a TCP connection is a different and much more difficult task than measuring the actual rate a TCP connection is achieving during the data transfer as it is done by Westwood TCP [12]. In particular, Westwood TCP low-pass filters the flow of returning ACKs to get an estimate of the bandwidth a TCP connection is using. However, the filter proposed in [12] does not work properly in the presence of ACK compression because of aliasing. In particular, when in the presence of ACK compression, the estimation algorithm described in [12] causes an overestimate of the bandwidth that is disruptive of the Westwood adaptive decrease mechanism and leads to starvation and fairness disruption [19].

3 Westwood TCP Congestion Control

TCP-W is a sender-side only modification of the TCP stack. It proposes an end-to-end bandwidth estimation algorithm that is built on the standard probing mechanism of today TCP. It implements slow-start and congestion avoidance phases such as classic Reno TCP to probe the network but, after congestion, it employs the estimate of the best-effort available bandwidth B to properly 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 $max[2,(B*RTT_{min})/seg\_size]$, 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 $ssthresh=(B*RTT_{min})/seg\_size$ 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 the stability of the standard TCP *multiplicative decrease* algorithm since it can ensure that the congestion window is reduced enough in the presence of heavy congestion and not too much in the presence of light congestion or losses not due to congestion, such as in the case of radio links. Clearly, a key requirement in order to have a properly working adaptive decrease mechanism is that the bandwidth estimate is correct.

To conclude this background section we also notice that the adaptive setting of the control windows increases the fair allocation of available bandwidth to different TCP flows. This can be intuitively explained by considering that the window setting of TCP Westwood tracks the estimated bandwidth so that, if the estimate is a good measurement of the fair share, then the fairness is improved (for a mathematical proof of this results see [19]). Moreover, the setting $cwnd=B*RTT_{min}$ sustains a transmission rate $cwnd/RTT=B*RTT_{min}/RTT$ 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 statistical multiplexing and fairness.

4 Bandwidth Estimation Algorithms

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

Fig. 1 depicts the end-to-end sender-based bandwidth estimation framework. It shows a Westwood TCP sender injecting data segments into the Internet and receiving ACKs from the receiver. 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:

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

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, 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].

We consider the following three types of discrete-time low-pass filters, which are all obtained by discretizing a first-order low-pass continuous filter:

– A time-varying filter obtained by discretizing the continuous filter using a Zero Order Holder (ZOH) [10];
– A time-invariant filter obtained using a ZOH;
– A time-varying filter obtained by discretizing the continuous filter using the bilinear approximation [10].

In order to compare the three filters above, we employ the topology in Fig. 2.

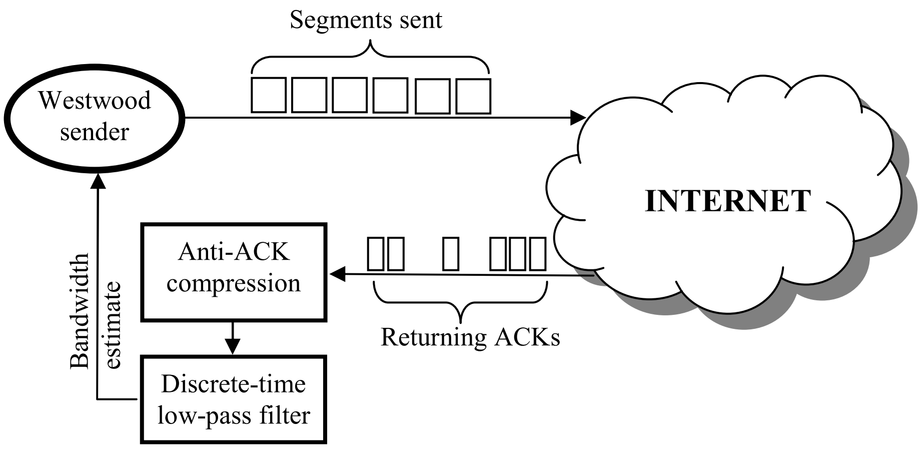


Fig. 1. Bandwidth estimation scheme.

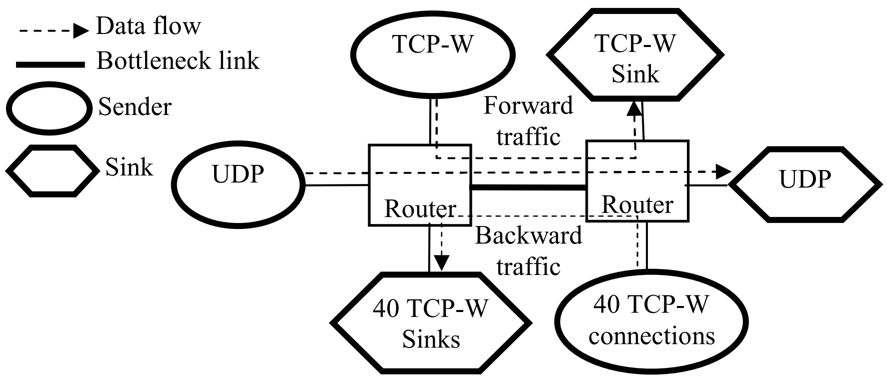


Fig. 2. Topology for testing the discrete-time filters.

It consists of 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]. 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.

4.1 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). The discretization procedure is described below:

A sample of used bandwidth (1) is considered as an impulse, which arrives every time t_k the sender receives an ACK. The impulsive sample is interpolated via a zero order holder, which generates a piecewise constant signal. The continuous piecewise constant signal is filtered by a first order single pole low-pass continuous filter with time constant τ. The output of the time-continuous filter is sampled at t_k times. Such steps lead to the following discrete time filter:

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

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. 3 (a) plots the used bandwidth samples and Fig. 3 (b) the used bandwidth estimate obtained using a TCP-W sender implementing the filter (2). It is worth noticing that, in this case, time-varying coefficients counteract the non-uniform sampling time and mitigates the effects of ACK compression. In order to illustrate high frequency components due to ACK compression, Fig. 3 (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].

Even though filter (2) mitigates ACK compression, it does not work for every value of τ due to aliasing effects. In fact, Fig. 4 shows the over estimate that is obtained by using a filter with time constant τ=0.1s.

4.2 A discrete time invariant filter

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

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

where

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

Fig. 5 (a) plots samples (1) of the used bandwidth and Fig. 5 (b) plots the output of the filter (3) with $\alpha = 0.9$ when fed by these samples. Fig. 5 (b) shows that the filter overestimates the available bandwidth up to more than 10 times. The reason is that bandwidth samples of Fig. 5 (a) contain aliased frequency components due to ACK compression.

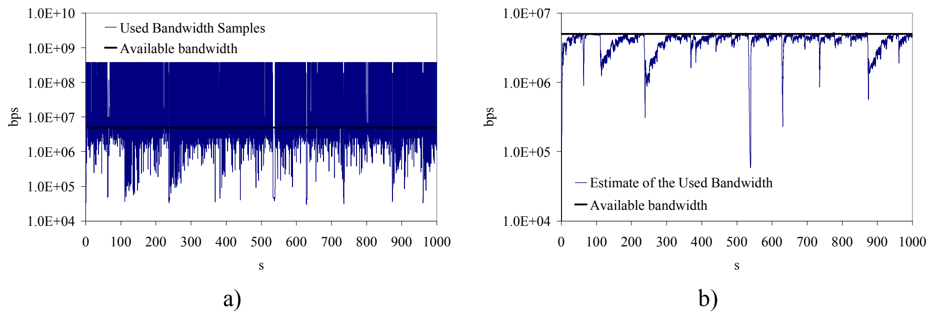


Fig. 3. Used Bandwidth: (a) Samples; (b) Output of the filter (2).

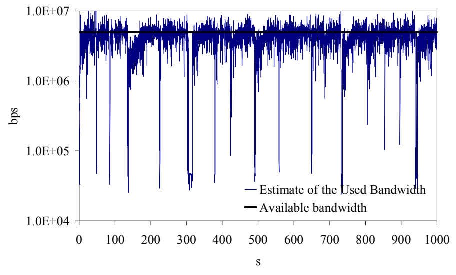


Fig. 4. Estimate of the used bandwidth using the filter (2) with τ=0.1s.

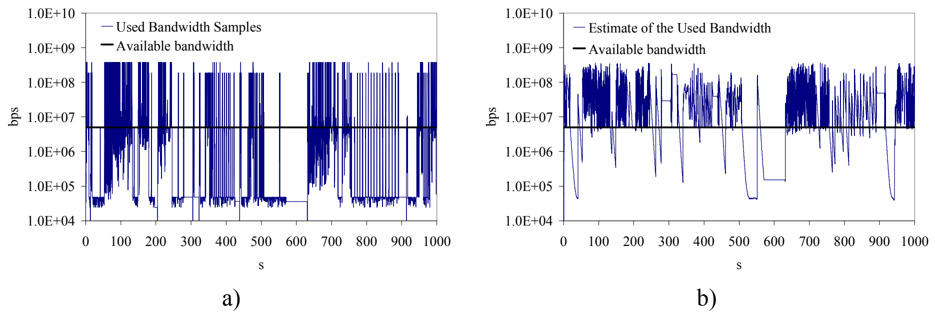


Fig. 5. Used bandwidth: (a) Samples; (b) Output of the filter (3).

4.3 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:

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

4.3.1 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] :

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

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 virtual samples with inter-arrival time $\tau/2$ and amplitude equal to 0, where $N = \text{integer of } (2 \cdot \Delta_k / \tau_f)$. 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.

Fig. 6 reports the used bandwidth samples and the estimated used bandwidth obtained using the filter (6). As it can be viewed, the filter greatly overestimates the bandwidth because of ACK compression that generates aliased samples. In the original Westwood TCP paper [12] this phenomena was not evident because the ACK compression was weak in the considered scenarios.

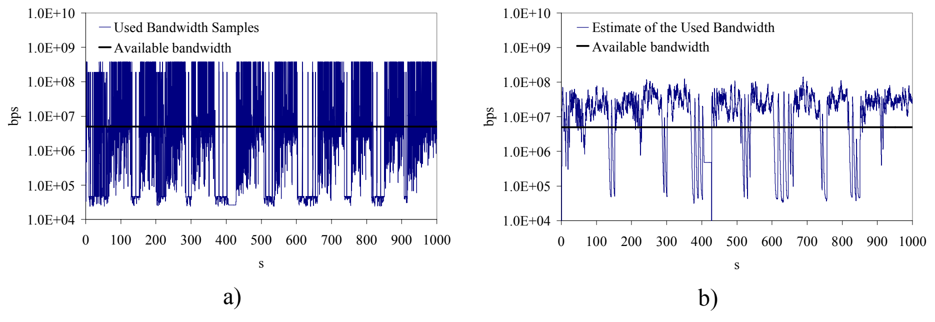


Fig. 6. Used Bandwidth: (a) Samples; (b) Output of the filter (6).

5 Anti Aliasing Filter in Packet Networks

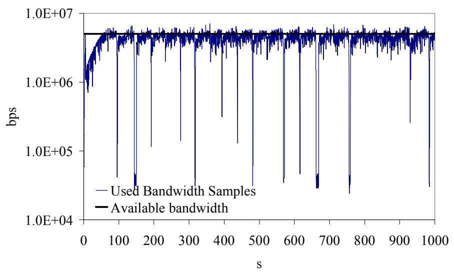
We have seen that ACK compression generates bandwidth samples containing aliased frequency components that cannot be low-pass filtered using a digital filter. Therefore, similarly to the case of filtering an analog signal, which requires the analog signal be pre-filtered by an anti-aliasing filter before 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* (that is the sampling time Δ is equal to *RTT*). Assuming a constant *RTT*, the α coefficient in filter (3) can be set accordingly to (4). The difference between the time-invariant filter (3) and the time-varying filters (2) and (6) is that filter (3) employs a constant coefficient α, whereas filters (2) and (6) dynamically adjust their coefficients by taking into account the last samples interarrival time. The latter feature is important to provide proper filtering also in scenarios with large *RTT* variance.

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 $\Delta_k \le \tau/2$ [10]. To be conservative, we assume $\Delta_k \le \tau_f/4$. 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 $\Delta T = \Delta_k - N \cdot \tau/4$.

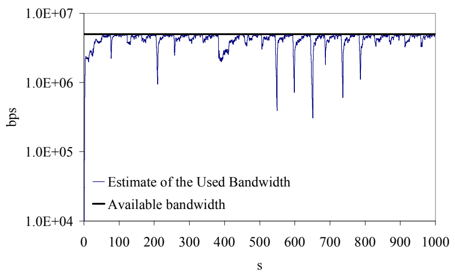
It should be noticed that filter (2) already implements interpolation and resampling through the ZOH that keeps constant the signal during the last interarrival time. However, the coefficient of the filter (2) requires the computation of an exponential that is harder than computing a ratio as in the 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.

5.1 Bandwidth estimates in the presence of constant available bandwidth

We consider the same scenario described in Fig. 2. Fig. 7 (a) shows the bandwidth samples computed using the anti-aliasing procedure, whereas Fig. 7 (b) shows the output of the filter (3) which is fed by anti-aliased samples.



a)



b)

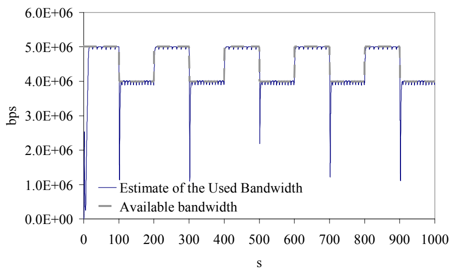
Fig. 7. Used bandwidth: (a) Anti-aliased samples; (b) Output of the filter (3).

A comparison of Fig. 7 (a) and Fig. 3 (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. 7 (a), a further stage of discrete-time filtering extracts the low frequency components of the used bandwidth as it is shown in Fig. 7 (b). Analogous results have been obtained with filters (6) and (2).

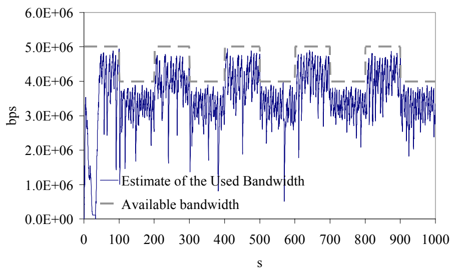
5.2 Bandwidth estimates in the presence of time varying available bandwidth

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. 2 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. 8 (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.



a)



b)

Fig. 8. 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. 9. Similar results have been obtained by employing the filters (3) and (2).

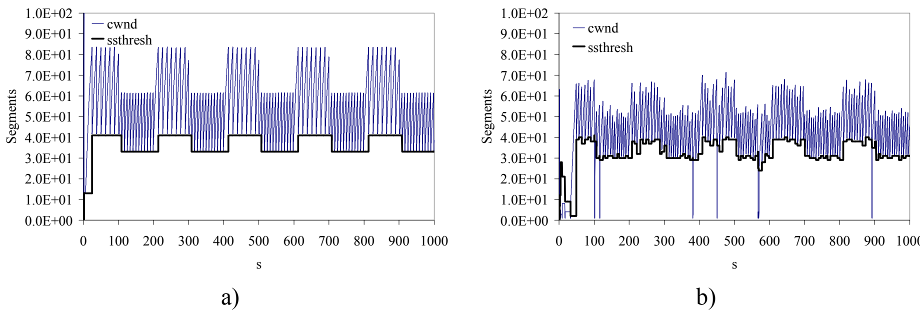


Fig. 9. Congestion window and slow start threshold with filter (6): (a) without reverse traffic; (b) with reverse traffic.

It is interesting to compare the *ssthresh* of Westwood TCP w.r.t the *ssthresh* of Reno TCP, which is plotted in Fig. 10. The comparison points out that the *ssthresh* of Westwood is larger than the one of Reno. This proves that the proposed bandwidth estimate enhances the ability of TCP congestion control to match the available bandwidth, which provides better utilization of network bandwidth.

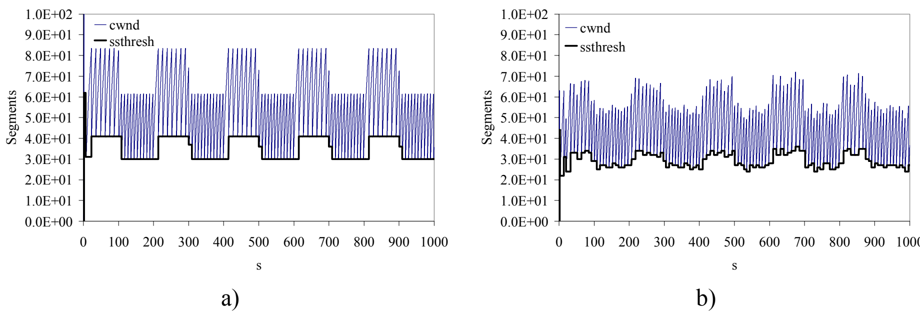


Fig. 10. Congestion window and slow start threshold of Reno: (a) without reverse traffic; (b) with reverse traffic.

5.3 Bandwidth estimates obtained by many concurrent TCP flows

In this section, we investigate the bandwidth estimates of many Westwood TCP flows competing for the same bottleneck capacity. We would like that, given a link capacity C shared by N flows each flow obtains an estimate of the used bandwidth oscillating around the fair-share C/N and a *ssthresh* which is close to the available bandwidth. This would prove that the end-to-end bandwidth estimation algorithm can improve the fairness of TCP congestion control in bandwidth allocation. We consider a single bottleneck scenario similar to the one depicted in Fig. 2, where a 10Mbps FIFO bottleneck is shared by 10 TCP-W persistent flows with *RTTs* ranging uniformly from 25ms to 250ms. Figs. 11 (a) and (b) show the estimates of the used bandwidth and of the best effort available bandwidth $ssthresh*Seg\_size/RTT_{min}$, respectively, obtained by the 10 TCP-W connections using the filter (6). Similar results have been obtained for filters (2) and (3). Fig. 11(a) shows that the used bandwidth estimates oscillate around the fair-share *C/N*, which is denoted by the dashed line. Furthermore Fig. 11(b) shows that the estimates of the best effort available bandwidth are less oscillating and closer to the fair share *C/N*. Similar results have been obtained by using the filters (3) and (2).

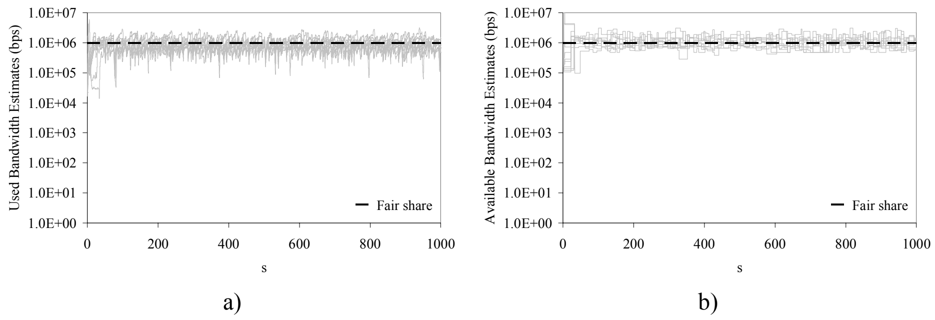


Fig. 11. Bandwidth estimates of 10 Westwood flows implementing the filter (6): (a) Used Bandwidth; (b) Available Bandwidth.

6 Conclusions

This work has focused on end-to-end bandwidth estimation schemes to be used for TCP congestion control. It has been shown that it is necessary to implement an anti-aliasing filter before using discrete-time filters 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. This estimate enhances the efficiency of TCP congestion control [12,19].

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. Dah-Ming Chiu, Jain, R.: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, Vol. 17(1), (1989) 1–14.
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 Queueing: 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. *Aström* , K. J. and B. Wittenmark (1997). Computer controlled systems, Prentice Hall, Englewood Cliffs, N. J, 1995.
11. Brakmo, L. S., and Peterson, L.: TCP Vegas: End-to-end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications (JSAC), Vol. 13(8), (1995) 1465–1480.
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, PfHSN02, 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.
22. 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.
23. 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.