TCP Westwood+ Enhancement in High-Speed Long-Distance Networks

Dzmitry Kliazovich and Fabrizio Granelli
DIT - University of Trento
Via Sommarive 14, I-38050 Trento, Italy
E-mail: [klezovic,granelli]@dit.unitn.it

Daniele Miorandi
CREATE-NET
Via Solteri 38, I-38100 Trento, Italy
E-mail: daniele.miorandi@create-net.it

***Abstract* — In this paper, mechanisms to enhance the performance of TCP Westwood+ in the presence of a large Bandwidth-Delay Product are studied. In particular, the employment of a logarithmic function for congestion window increase in the absence of packet losses is proposed. Extensive numerical simulations, carried out by using ns-2, show that the use of such logarithmic function can lead to relevant performance improvements with respect to standard Westwood+ protocol.[1]**

Keywords: congestion control, large bandwidth-delay product networks, TCP performance, TCP Westwood+

I. INTRODUCTION

The TCP/IP reference model defines a suite of protocols for enabling end-to-end reliable communications over the Internet. According to such model, no layer has complete and real-time information about available network resources over the multi-hop path where communications take place. Inability to locate a bottleneck present on the data communication path motivates the necessity of controlling the amount of data traffic delivered to the network in order to avoid network congestion and prevent network collapse.

The Transmission Control Protocol (TCP) [1] is the *de facto* standard for ensuring end-to-end reliable delivery of packets in the Internet and is used by a large variety of applications [2,3]. The TCP congestion control algorithm introduced by Van Jacobson [4] is based on a sliding window mechanism and employs an Additive Increase Multiplicative Decrease (AIMD) algorithm for updating transmission rate to the available network resources. The receiver provides an acknowledgement (ACK) feedback, based on which the sender detects packets losses and consequently updates the transmission rate. Since TCP was originally designed for wired low error-rate networks, it assumes that all packet losses are caused by network congestion.

The congestion window is the key parameter in TCP behaviour, and defines the amount of unacknowledged data (in terms of TCP segments) which can be delivered to the network. The AIMD mechanism specifies a linear congestion window increase over time. Upon loss detection via reception of duplicate ACKs[2] the sender halves the congestion window value. The overall result of the conservative increase and aggressive decrease strategy is that AIMD-based mechanisms greatly underutilize available bandwidth in high-speed networks in the presence of large delays [5,6]. The main reason for poor performance of AIMD in such scenarios is rooted in the fact that the additive increase mechanism of standard TCP is highly inefficient in the presence of large bandwidth-delay product (BDP). In other words, a large amount of time is required to "restore" the congestion window after a packet drop detection. The larger the BDP, the longer the time needed for window restoration. A detailed study of this problem is presented in [6], where the author proposes a new protocol, called Scalable TCP, based on the adoption of a multiplicative increase algorithm for overcoming such limitation. The exponential window increase of Scalable TCP leads to remarkable performance improvement in large BDP links, with bandwidth larger than 100 Mb/s and round trip time propagation delay above 50 ms. However, in low BDP networks, the window increase algorithm proposed in Scalable TCP is less aggressive than AIMD mechanism, leading to poor performance if compared with TCP NewReno.

Other proposed modifications to standard TCP congestion control mechanisms include High Speed TCP (HSTCP) proposed in [5], FAST TCP [7], and Binary Increase Congestion Control (BIC) [8]. In particular, the latter one employs a logarithmic function for window increase in congestion avoidance.

In [9], the authors propose an alternative transport protocol, called TCP Westwood, which aims at enhancing TCP performance in lossy environments. The main idea behind Westwood is to keep an estimate of the available end-to-end capacity (obtained from an appropriate filtering of returning ACK flow) and to exploit such information in order to reduce transmission rate, instead of the blind halving approach implemented in standard TCP. TCP Westwood's operations are hence based on an Additive Increase Adaptive Decrease (AIAD) window evolution mechanism. The problems

[1] This work is partially supported by National (MIUR) project Wireless 8O2.16 Multi-antenna mEsh Networks (WOMEN) under grant number 2005093248.

[2] We are here assuming that packet loss detection takes place after the sender receives three duplicate acknowledgements from the receiver, neglecting the impact of timeout occurrences.

encountered by TCP Westwood in the presence of compressed/delayed ACKs motivated a modification in the ACK filter used in bandwidth estimation, leading to the introduction of Westwood+ protocol [10].

While Westwood+ is able to outperform standard TCP in a wide range of application scenarios, it still suffers in the presence of a large BDP, due to the low responsiveness after a packet loss. Even if some recent works have tried to enhance the performance of Westwood in such scenarios [11, 12], the main function of the window increase algorithm is left to be additive with its conservative increase behavior.

In this paper, we investigate possible enhancements of TCP Westwood+ in the presence of high-speed large-delay networks. While leaving bandwidth estimate and adaptive window decrease untouched, we study the impact of different window evolution algorithms, and show how remarkable improvements over Westwood+ can be achieved in networks characterized by a large BDP by means of a logarithmic increase function.

The remainder of the paper is organized as follows. In Sec. II, we review the Westwood+ congestion control mechanism and present the proposed modifications. Sec. III reports the outcome of extensive ns-2 [13] simulations, presenting the performance improvements achievable by means of the proposed solutions. Sec. IV concludes the paper pointing out the most promising directions for future research in the field.

II. Congestion Window Evolution Strategies

TCP connection is initiated with congestion window (W) equal to one packet. Then, W is increased by one for every received non-duplicate ACK until the Slow Start Threshold is reached. The basic idea behind the slow start phase is to provide a fast (roughly exponential over time) window increase till the capacity of the transmission pipe is reached.

Once the slow start threshold is exceeded, TCP passes to the congestion avoidance phase. This phase, which aims at gently probing the network for available bandwidth, is characterized by a roughly linear increase of W over time. In the case three consecutive duplicate ACKs are received, the congestion window value W is halved. Summarizing, the actions taken upon reception of an ACK and upon packet loss detection can be represented as:

$$W \leftarrow W + \frac{1}{W}; \quad (1)$$

$$W \leftarrow \frac{W}{2}. \quad (2)$$

Overall, this determines an Additive Increase Multiplicative Decrease (AIMD) mechanism.

Scalable TCP [6] tries to overcome the shortcomings of TCP's AIMD mechanism in large BDP networks by adopting a Multiplicative Increase Multiplicative Decrease (MIMD) algorithm, whose operations can be summarized as:

$$W \leftarrow W + \alpha; \quad (3)$$

$$W \leftarrow \beta \cdot W. \quad (4)$$

The speed of window increase is controlled by parameter α, while β is responsible for window decrease dynamics. In [6], the author suggests setting α=0.01 and β=0.875 as default values for calibrating the proposed MIMD strategy.

On the other hand, TCP Westwood+ uses linear additive window increase function fully consistent with (1). However, in case a packet loss is detected, the congestion window value W is updated as follows:

$$W \leftarrow \max(2, \frac{BWE \cdot RTT_{min}}{seg\_size}). \quad (5)$$

where BWE is the estimated available end-to-end bandwidth, $RTTmin$ is the minimum round trip time measured over the duration of the connection and seg_size is the size of a TCP segment, in bits.

A combination of TCP Westwood+ with the multiplicative increase algorithm proposed in Scalable TCP creates a Multiplicative Increase Adaptive Decrease (MIAD) algorithm for congestion control. In principle, it should bring scalability to TCP Westwood+ by reducing the time spent in loss recovery for large bandwidth-delay networks. However, this would require (as with Scalable TCP) dynamic reconfiguration of parameters α and β for networks with low BDP or in case of timeout occurrence, in order to make it at least as aggressive as traditional additive increase strategy.

Another unfavorable property presented by MIAD algorithms consists in the large steps of window increase while approaching the available capacity threshold. Indeed, the larger the base for the exponent (i.e., the current window value), the larger the increase step. This means that multiple packet drops are likely to happen when the protocol approaches the available capacity, as result of buffer overflow at the edge bottleneck router.

Multiple drops of packets waste network resources on the path between source and bottleneck link, while at the same time have a negative effect on the bandwidth estimation algorithm of Westwood+. The overall result is an underestimation of the bandwidth available for the connection, with an overall negative effect on the protocol performance. This phenomenon has been confirmed by extensive numerical simulations (not reported in the paper due to lack of space).

Given such considerations, two approaches are possible. The first one would consist in the design of a novel filter for bandwidth estimation, able to cope with the behaviour induced by MIAD algorithms. However, we do not believe this is the right way to go, in that it would lead to the introduction of a less responsive bandwidth estimation mechanism. The other approach, the one we will pursue in the following sections, consists in introducing a different function for increasing the congestion window, able to fit well the features of the Westwood+ bandwidth estimation mechanism.

Based on the aforementioned observations, we propose a congestion window increase mechanism able to:

(1) be more aggressive than traditional additive increase algorithm at any moment of time, in order to guarantee equal or better throughput performance;

(2) provide a fast window increase for low W values, while being accurate in approaching the available pipe capacity;

(3) present a small sensitivity with respect to the RTT value.

In this paper, we investigate the logarithmic function as a candidate for replacement of linear congestion window evolution. The proposed function operates during the congestion avoidance phase of TCP, while the slow start phase is left unchanged. The algorithm specifies that, for each acknowledgement packet received, the congestion window should be updated according to:

$$W \leftarrow W + \frac{W_{max} - W}{\alpha \cdot W}. \qquad (6)$$

The maximum window size W_{max} is defined as the congestion window value at which the last packet loss event was detected. It basically represents a (rough) estimate of the overall available pipe size.[3] Eq. (6) selects the next W value choosing a point between the current W value and W_{max} scaled by parameter α. Parameter α controls the level of aggressiveness of the increase algorithm dynamically: α is increased by a factor of two in case a packet drop is detected before W reaches W_{max} value, while it is decreased by the same factor for every errorless increase on the same interval. We lowerbound α by 2 (initial value) from the considerations that values less than 2 will make the increase too aggressive.

According to (6) the growth of the congestion window value is aggressive in the presence of low W values, while becoming more prudent when approaching $W_{max,}$, thus satisfying design objectives 2 and 3. However, in order to ensure that the designed function is always at least as aggressive as standard TCP mechanism, we bound the minimum window increase to one packet per window:

$$W \leftarrow W + \max\left(\frac{W_{max} - W}{\alpha \cdot W}, \frac{1}{W}\right). \qquad (7)$$

The proposed approach may recall the Binary Increase Congestion Control (BIC) function presented in [6], which has a conceptual similarity with the proposed function (letting α = 2). However, in contrast to BIC, the proposed window increase always follows a logarithmic behaviour with $W<W_{max}$, while the level of aggressiveness is controlled by scaling parameter α. As a result it behaves equally well for both small and large window values.

In Fig. 1 we reported the congestion window increase trend for additive, multiplicative and logarithmic increase strategies, respectively. Since the decrease part of the Westwood+ congestion control mechanism is left untouched, such increase functions should (in the case of a perfect bandwidth estimation) be active only when W is in the range comprised between the BDP actually provided by end-to-end link and the total available BDP (including buffering).

The level of aggressiveness of the discussed strategies corresponds to the speed of window increase. The performance of TCP flow is proportional the window size in every moment of time, i.e., larger amounts of time spent at higher window value result in increased throughput performance. Thus, different congestion window strategies can be compared by integral squares of the increase functions.

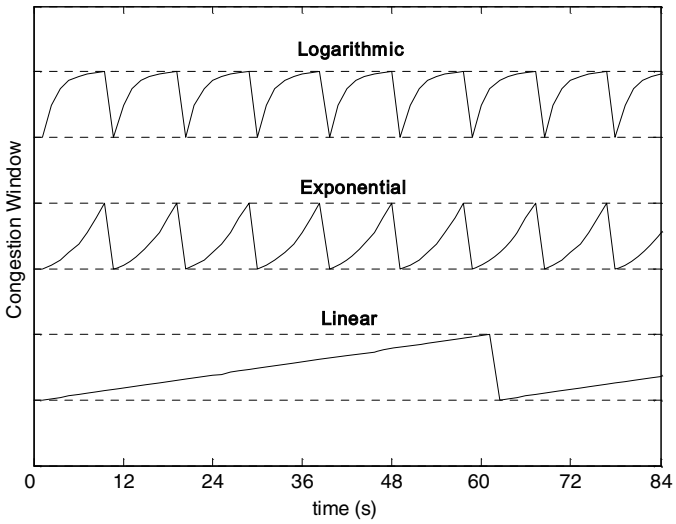

Fig. 1: Window increase trends for various candidate functions.

III. Performance Evaluation

The performance of the proposed TCP Westwood+ enhancements has been analyzed using the ns-2 network simulator [10] in its current (2.28) version. TCP Westwood+ modules were obtained from [14], on top of which the proposed logarithmic increase function was implemented with α equal to 4.

In order to evaluate the performance of the proposed algorithm, we decided to limit ourselves to a simple scenario, comprising a single bottleneck. The proposed simulation scenario is presented in Fig. 2. It consists of three nodes, connected by two links with different capacity characteristics. The source node (SRC) is connected to the destination node (DST) through a two-hop path.

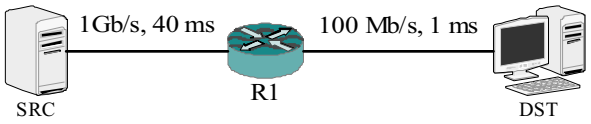

Fig. 2: The proposed simulation scenario.

The link between the SRC and the router R1 presents a capacity of 1 Gb/s and a delay of 40 ms. On the other hand, the link between the destination and the router presents a bandwidth of 100 Mb/s and a delay of 1 ms, figures roughly corresponding to the capacity of a LAN Ethernet network.

[3] The study of more efficient algorithms for estimating such value is out of the main scope of the paper, and is therefore left to future works.

Such configuration allows 4.1 Mb of data to be outstanding in the network, corresponding to a BDP value of 683 1500-byte TCP packets. The outgoing queue size of network nodes is fixed to 50 packets, corresponding to 75 Kbytes. One persistent TCP connection is simulated, corresponding to the download (from SRC to DST) of an infinitely long file. Each simulation is stopped after 2000 seconds. In order to understand the performance improvements achievable with the proposed solution, the Packet Error Rate (PER) is varied on the bottleneck link.

The proposed protocol, called *LogWestwood+*, is compared with both its ancestor Westwood+ as well as with the standard TCP NewReno with selective acknowledgements (SACK), the most widespread TCP version currently employed in the Internet.

The amount of data delivered during the entire simulation time, which corresponds to the TCP sequence number, is chosen as the main metric for comparing the performance of the three protocols.

In Fig. 3, the dynamics of the congestion window value are reported, observed over a 50s time interval. From the graph, it is easy to see that LogWestwood+ and Westwood+ outperform standard TCP. Furthermore, NewReno halves the congestion window value upon overflowing the network, TCP Westwood+ as well as LogWestwood+ reduce to the bandwidth estimate (which corresponds to the actual link capacity, excluding buffer resources present on the link).

In our evaluation scenario, NewReno's AIMD algorithm requires 60 seconds to recover from a loss event. The AIAD mechanism introduced by TCP Westwood+ takes 12 seconds for increasing the window from bandwidth estimate to the total network capacity, while LogWestwood+ requires only 1.6 seconds for performing the same operation.

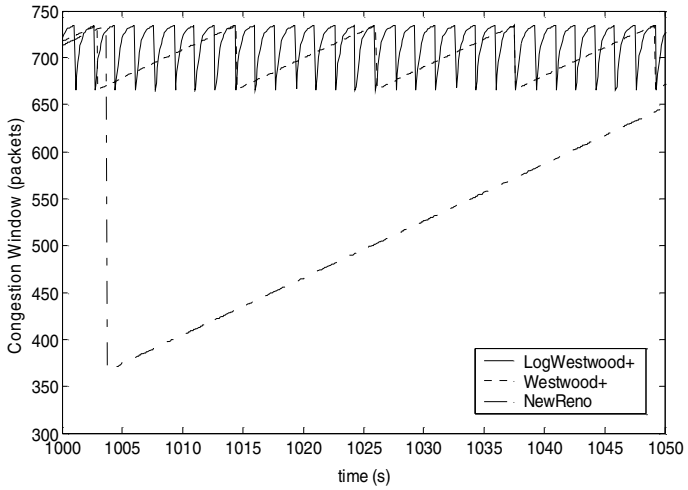

Fig. 3: Window increase of NewReno, Westwood+ and LogWestwood+.

Both LogWestwood+ and Westwood+ keep their congestion windows in the interval between estimated capacity of the link and its total capacity (which includes buffering along the data path). However, the fact that the minimum congestion window value does not fall below the link capacity makes them perform equally in case of no errors present on the link.

In Fig. 4, we plotted the performance achieved by the three solutions in the absence of packet losses, measured in terms of number of acknowledged packets versus simulation time. The number of acknowledged packets corresponds to sequence number of TCP flow in our simulations which, differently from real implementations, always starts from zero.

The poor performance obtained by TCP NewReno comes from the fact that the "blind" congestion halving reduces congestion window below the threshold of the available link capacity. After that, it underutilizes the bandwidth for the time needed by the conservative additive increase algorithm to reach again the link capacity.

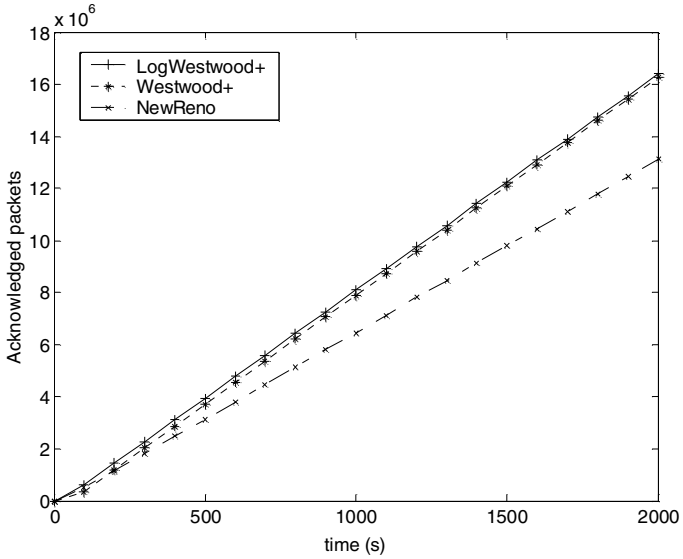

Fig. 4: Number of acknowledged packets versus simulation time for the NewReno, Westwood+ and LogWestwood+.

In order to evaluate the performance gain obtained from the proposed logarithmic window function, an error generator was connected to the bottleneck link between router R1 and DST node (see Fig. 2). Inserted errors are at the packet level and follow a memoryless pattern, each packet being lost independently with probability PER=10^{-7}, 10^{-5} and 10^{-3}, respectively. The results obtained are reported in Fig. 5 in terms of 95% confidence interval averaged from 10 runs. In the scenarios with low or no errors the confidence intervals are negligibly small and thus are not reported.

The obtained results show a LogWestwood+ performance gain increase with the larger PERs which is the consequence of faster window recovery from the occurred packet loss.

In more details, while traditional TCP NewReno assumes that all packet losses are caused by network congestion, TCP Westwood+ relies on an estimation of the available bandwidth for window reduction after the reception of duplicate ACKs. The algorithm for bandwidth estimation is based on the rate of incoming ACKs, which means that it does not take into account packets which were not successfully transmitted, but for which transmission channel resources were consumed.

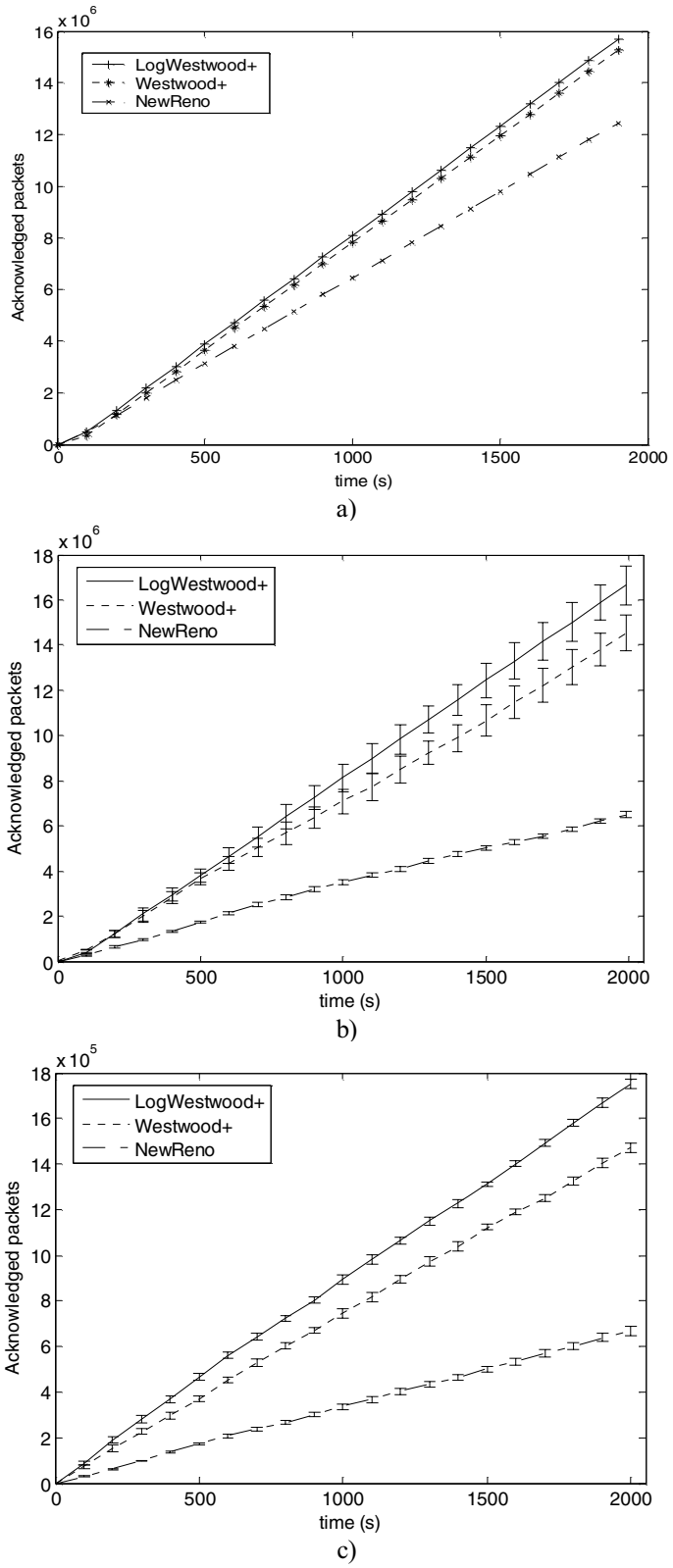

Fig. 5: Performance comparison in the presence of random errors, PER equal to a) 10^{-7}, b) 10^{-5}, and c) 10^{-3}.

As a result, TCP Westwood+ reduces its window to a value which is lower than available channel resources. Thus, the channel is underutilized for the time spent in the additive window increase phase up to link bandwidth.

The proposed LogWestwood+ algorithm allows a faster congestion window recovery after a packet loss event. The fewer time spent in the period of channel underutilization leads to a corresponding performance enhancement.

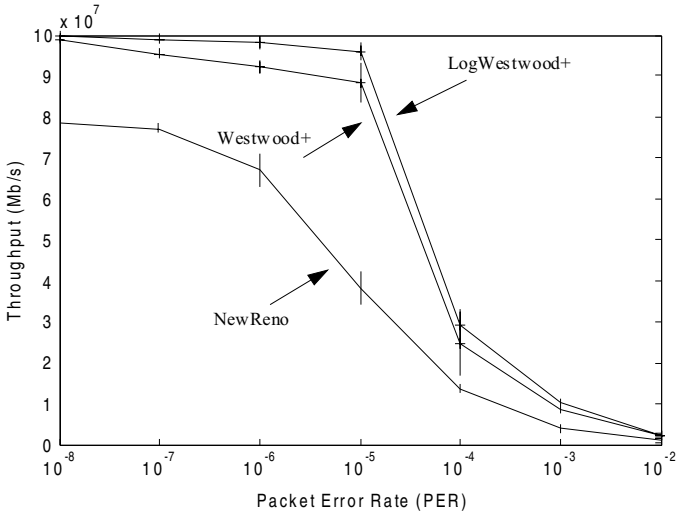

Fig. 6: Throughput performance comparison in the presence of random errors.

In Fig. 6, we reported the average throughput achieved by the three considered TCP implementations operating at different error rates. The performance of LogWestwood+ closely approximates original Westwood+ algorithm showing a good level of utilization of the end-to-end data link, while TCP NewReno achieves a lower throughput in all considered simulation settings, utilizing a maximum of 80% of the capacity of the link.

In Fig. 7, we illustrate the performance enhancements, in terms of throughput ratio, achieved by LogWestwood+ over both Westwood+ and NewReno. The results show that the improvement level over Westwood+ is highly dependent on PER, being equal to 0 (for the single flow scenario with no errors) up to 20% (for a PER in the interval 10^{-5} – 10^{-2}). If compared with NewReno, the proposed LogWestwood+ achieves an improvement of up to 160% for a PER in the interval 10^{-5} – 10^{-2}.

For high PER (>10^{-1}), all the three evaluated solutions achieve a similar level of performance. This can be explained by considering that, in such situation, most packet losses are detected by timeout expiration. In this case, the protocols spend most of the time in the slow start phase, which explains the similar performance figures achieved.

Fig. 8 reports a performance comparison for the case of multiple concurrent flows. We considered the aggregated throughput as performance metric, and varied the number of flows (started at the simulation beginning) from 1 to 30. From the results therein, we can see that LogWestwood+ keeps the throughput close to the bottleneck capacity showing good (95 – 99%) link utilization, outperforming both Westwood+ and NewReno TCP implementations.

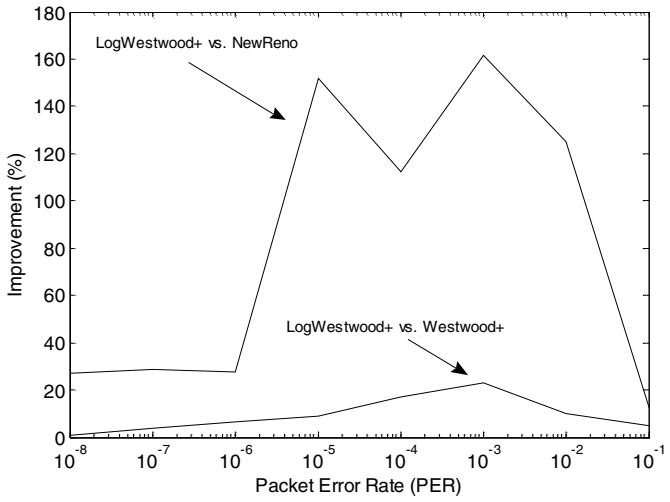

Fig. 7: LogWestwood+ performance improvement.

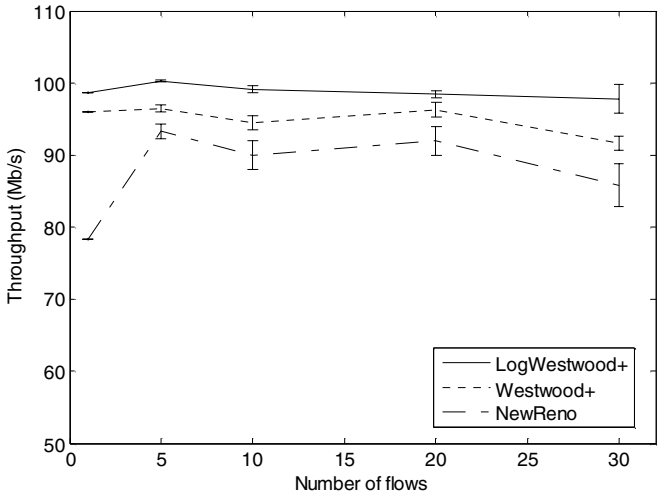

Fig. 8: Throughput performance comparison in multi-flow communications.

IV. CONCLUSIONS

In this paper, we have presented congestion control algorithms for improving the performance of TCP Westwood+ in large BDP networks. The analysis of the solutions available in the literature motivated us to address the design of a non-linear function for replacing the standard additive increase mechanism. The following design objectives have been specified for the target function: (i) to be more aggressive than traditional additive increase algorithm; (ii) to provide a fast window increase for low *W* values, while being accurate in approaching the available pipe capacity; and (iii) to ensure small sensitivity with respect to RTT. We showed how a logarithmic function is able to achieve the aforementioned design goals, leading us to the introduction of a strategy we termed LogWestwood+.

Extensive ns-2 simulations were performed to test the performance and robustness of the proposed algorithm, Numerical results show that the proposed approach is able to outperform Westwood+ and standard NewReno protocols in a wide range of simulation settings.

Two research directions appear of major interest for further investigations on the subject: the first one concerns the study of mechanisms able to dynamically adapt the value of the α parameter to the current network operating conditions; the second one consists in implementing the proposed mechanism on Web servers and performing tests in a real Internet environment.

REFERENCES

[1] J. B. Postel, "Transmission Control Protocol," *RFC 793*, September 1981.

[2] G. Miller, K. Thompson, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network*, pp. 10 – 23, November/ December 1997.

[3] N. Brownlee and K. Claffy, "Understanding Internet traffic streams: Dragonflies and tortoises," *IEEE Comm. Mag*., vol. 40, pp. 110 – 117, Oct. 2002.

[4] V. Jacobson, "Congestion Avoidance and Control", in *Proc. of ACM SIGCOMM*, pp. 314 – 329, August 1988, Standford, CA.

[5] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, December 2003.

[6] Tom Kelly, " Scalable TCP: improving performance in highspeed wide area networks," *ACM Comp. Comm. Rev*., vol. 33 , pp. 83 – 91, April 2003.

[7] Cheng Jin, D.X. Wei, S.H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proc. of IEEE INFOCOM,* vol. 4, pp. 2490 – 2501, March 2004, Hong Kong.

[8] Lisong Xu; K. Harfoush, and Injong Rhee, "Binary increase congestion control (BIC) for fast long-distance networks", in *Proc. of IEEE INFOCOM,* vol. 4, pp. 2514 – 2524, March 2004, Hong Kong.

[9] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: End-to-end Bandwidth Estimation of Efficient Transport over Wired and Wireless Networks", in *Proc. of ACM Mobicom*, July, 2001, Rome, Italy.

[10] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control", *ACM Comp. Comm. Rev*.,vol. 34, pp. 25 – 38, April 2004.

[11] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi, M. Gerla "TCP Startup Performance in Large Bandwidth Delay Networks", in *Proc. of INFOCOM 2004*, vol. 2, pp. 796 – 805, March 2004, Hong Kong.

[12] R. Wang, K. Yamada, M. Y. Sanadidi, and M. Gerla "TCP with sender-side intelligence to handle dynamic, large, leaky pipes," *IEEE J. Sel. Ar. Comm*., vol. 23, n. 2, pp. 235-248, 2005.

[13] NS-2 Simulator. http://www.isi.edu/nsnam/ns

[14] TCP Westwood+ modules for ns2, http://193.204.59.68/mascolo/tcp%20westwood/modules.htm