Logarithmic window increase for TCP Westwood+ for improvement in high speed, long distance networks

Article *in* Computer Networks · August 2008

DOI: 10.1016/j.comnet.2008.04.018 · Source: DBLP

3 authors:

Dzmitry Kliazovich
University of Luxembourg
171 PUBLICATIONS **4,293** CITATIONS
SEE PROFILE

Fabrizio Granelli
University of Trento
376 PUBLICATIONS **5,334** CITATIONS
SEE PROFILE

Daniele Miorandi
U-Hopper srl
184 PUBLICATIONS **8,288** CITATIONS
SEE PROFILE

Computer Networks 52 (2008) 2395–2410

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Logarithmic window increase for TCP Westwood+ for improvement in high speed, long distance networks [☆],[☆☆]

Dzmitry Kliazovich [a], Fabrizio Granelli [a,*], Daniele Miorandi [b]

[a] DIT, University of Trento, Via Sommarive 14, I-38050 Trento, TN, Italy
[b] CREATE-NET, Via Solteri 38, 38100 Trento, Italy

A R T I C L E   I N F O

Article history:
Received 14 November 2006
Received in revised form 19 March 2008
Accepted 29 April 2008
Available online 8 May 2008

Responsible Editor: V. Tsaoussidis

Keywords:
Congestion control
High-speed long-distance networks
TCP performance
TCP Westwood+

A B S T R A C T

The majority of current Internet applications uses Transmission Control Protocol (TCP) for ensuring reliable end-to-end delivery of data over IP networks. The resulting path is, generally speaking, characterized by fairly large propagation delays (of the order of tens to hundreds of milliseconds) and increasing available bandwidth. Current TCP[1] performance is far from representing an optimal solution in such operating conditions. The main reason lies in the conservative congestion control strategy employed, which does not let TCP to exploit the always increasing available path capacity. As a consequence, TCP optimization has been an active research topic in the research community over the last 25 years, boosted in the last few years by the widespread adoption of high-speed optical fiber links in the backbone and the emergence of supercomputing networked applications from one side and tremendous growth of wireless bandwidth in network access from another. This has led to the introduction of several alternative proposals for performing congestion control. Most of them focus on the effectiveness of bandwidth utilization, introducing more "aggressive" congestion control strategies. However, such approaches result often in unfairness among flows with substantially different RTTs, or do not present the inter-protocol fairness features required for incremental network deployment.

In this paper, we propose TCP LogWestwood+, a TCP Westwood+ enhancement based on a logarithmic increase function, targeting adaptation to the high-speed wireless environment. The algorithm shows low sensitivity with respect to RTT value, while maintaining high network utilization in a wide range of network settings. The performance, fairness and stability properties of the proposed TCP LogWestwood+ are studied analytically, and then validated by means of an extensive set of experiments including computer simulations and wide area Internet measurements.

1. Introduction

The TCP/IP reference model defines a suite of protocols for enabling end-to-end communications over the Internet.

The Transmission Control Protocol (TCP) [2] is the de facto standard in Internet used by the majority of applications [3,4]. Reliability and congestion control are two fundamental functions which motivated wide acceptance of TCP.

The TCP congestion control mechanism introduced by Van Jacobson [5] is based on a sliding window mechanism

☆ Part of this work has been presented at IEEE ICC 2006, Istanbul, Turkey [1].
☆☆ This work is partially supported by the Italian Ministry for Education, University and Research (MIUR) in the framework of the *WOMEN* project.
* Corresponding author. Tel.: +39 0461882062.
E-mail addresses: klezovich@dit.unitn.it (D. Kliazovich), granelli@dit.unitn.it (F. Granelli), daniele.miorandi@create-net.it (D. Miorandi).
[1] With *TCP* or *Standard TCP* we refer to TCP NewReno, unless specified.

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.04.018

and employs an additive increase multiplicative decrease (AIMD) algorithm with the purpose to fit transmission rate to the available network resources. Based on acknowledgement (ACK) feedback from the receiver, the sender detects packet losses and consequently adjusts its sending rate. TCP treats all packet losses as if they were caused by congestion, and as a consequence drastic reduction of sending rate is the only reaction on the packet loss detection.

At connection startup, the available path capacity is probed with an exponential window increase (by one for every ACK), giving rise to what is referred to as "slow start" phase. Then, the protocol enters the "congestion avoidance" phase, where the congestion window value is increased approximately linearly, by one for every Round Trip Time (RTT). As a result, TCP increases its sending rate proportionally to 1/RTT [6,7] making low-RTT flows more aggressive than ones with large RTTs. Such a behavior leads the two following performance issues:

- *Underutilization* of path capacity arises even with a single TCP flow operating on a path with long propagation delays. The main reason for poor performance in such scenarios is rooted in the fact that the additive increase mechanism is highly inefficient in presence of large bandwidth-delay product (BDP) [8,9]. In other words, a large amount of time is required to "restore" the congestion window back to the BDP after a packet drop detection: the larger the BDP, the longer the time needed for window restoration.
- *Unfairness* between flows sharing a portion of the path appears when one of the flows increases its window more aggressively than others when the cumulative throughput approaches the path capacity. In this scenario, the more aggressive flow tends to fill the bottleneck's buffer with its packets, forcing the less aggressive flow to reduce its sending rate reacting to the congestion loss. Therefore, an RTT-based linear increase always favors small RTT flows (which grab a larger portion of bottleneck bandwidth) greatly degrading long RTT flows performance [10,11]. RTTs depend on the employed network technology and can vary greatly: while the typical RTT in wired Local Area Networks (LANs) is less than 1 ms, it can be the order of 200 ms for intercontinental connections and as high as 600 ms for satellite links [12,13].

Currently, it is widely considered an open question which of the technologies will be the winner. Among available alternatives for fixed connections are high-speed Ethernet or PON with access speeds exceeding 1 Gbps. On the other hand, the growing number of wireless subscribers with the respect to Internet users is turning network access towards advanced wireless technologies. Such technologies like WiMAX [14] or 3G LTE [15] are already about to hit 100 Mbps barrier while in several years 4G networking [16] will bring 1 Gbps to the end-user.

Consequently, the TCP protocol, whose design dates back to the early days of ARPANET, requires an adaptation accounting networks characterized by large BDP in order to maintain high bandwidth utilization while being fair with existing widely deployed TCP solutions.

In order to overcome these limitations, the targeted window increase function needs to provide low sensitivity with respect to RTT values. This can be accomplished using non-linear functions, such as logarithmic or exponential ones. Moreover, in order to maintain a high utilization of the network resources, data transmission needs to be aggressive when the sending rate is below the available capacity of the path. However, in order to provide good intra-protocol fairness, the sending rate should smoothly approximate the capacity, which consists of the BDP of the bottleneck link plus the available buffer resources on the path.

After reviewing some TCP congestion control modifications proposed by the research community (Section 2) we identified the logarithmic increase function to suit well the aforementioned design objectives. The decrease strategy we have chosen is the one initially implemented in TCP Westwood [17] and further improved in TCP Westwood+ [18] with a modified bandwidth estimation algorithm [19]. The main idea behind TCP Westwood is to keep an estimate of the available end-to-end capacity (obtained from appropriate filtering of returning acknowledgement flow) and to exploit such information in order to reduce transmission rate, instead of the blind window halving implemented in standard TCP as well as in other proposed algorithms.

The resulting congestion control strategy, called *logarithmic increase, adaptive decrease (LIAD)*, can be considered as an enhancement of TCP Westwood+, from which it inherits the adaptive decrease strategy (as well as the bandwidth estimation technique) and from which it differs for the actions undertaken in response to the reception of an ACK packet, leading to an approximately logarithmic increase in absence of loss events. The design of LIAD is presented in Section 3 together with mathematical model of the protocol operations.

Extensive ns-2 [20] simulations have been run to preliminarily estimate the performance of the proposed protocol. The obtained results are presented and discussed in Section 4. We also implemented, in a Linux system, the proposed protocol; performance measurements over a wide-area network are reported in Section 5. Section 6 concludes the paper pointing out some promising directions for future research in the field.

2. Congestion control strategies

TCP congestion control proposals can be broadly divided into two groups: loss-based and delay-based approaches. The first group treats packet losses as an indication of the network congestion and reacts reducing the sending rate. On the contrary, delay-based approaches measure RTT variations indicating growing bottleneck queue, and thus, reacting before congestion occurs.

While delay-based approaches achieve high performance in terms of network utilization and intra-protocol fairness, they can not compete in case the majority of flows in the network is loss-based reacting to congestion post-factum. This fact was first discovered for TCP Vegas [21] and presented in [22], while more general discussion on the usability of delay-based protocols can be found in [23]. For that reason, we mostly focus on loss-based approaches within the following overview.

Conventional additive increase, multiplicative decrease (AIMD) strategy implemented in standard TCP is loss-based. TCP connection is initiated with congestion window (W) equal to one packet and Slow Start Threshold (*ssthresh*) set to a maximum possible value. Then, W is increased by one for every received non-duplicate ACK until the *ssthresh* is reached. This represents the slow start phase and its basic idea is to provide a fast (roughly exponential over time) window increase till the capacity of the transmission pipe is reached.

Once the capacity is reached and the packet loss is detected by three consecutive duplicate ACKs, W is halved and TCP enters congestion avoidance phase. In this phase, the TCP sender gently probes the network for available bandwidth with a linear increase.

With the main target to make congestion control scalable to the pipe capacity, Kelly proposed Multiplicative Increase, Multiplicative Decrease (MIMD) mechanism introduced in Scalable TCP [9]. Due to multiplicative increase phase, Scalable TCP takes an approximately constant time to recover the window after a packet loss, regardless of the pipe capacity. In contrast to standard TCP, this time is still dependant on RTT but not on the pipe capacity, making the scheme scalable to high speed networks.

Scalable TCP leads to a remarkable performance improvement over paths with bandwidth, but more important with high RTTs delaying the feedback sent by the receiver considerably, such as over satellite links [24]. However, as it is shown by multiple studies [25], exponential increase is too aggressive for medium and low BDP networks.

Another important drawback of MIMD-based protocols in large BDP networks is that they present very high increase rates when approaching the path capacity. This causes multiple packet losses when the bottleneck buffer gets in overflow, affecting stability and intra- and inter-protocol fairness.

A milestone in congestion control evolution is associated with the presentation of Binary Increase Congestion Control TCP (BIC-TCP) [26]. The authors propose a combination of linear and logarithmic increase functions. Upon packet loss, BIC-TCP reduces its window by a factor of β and updates $W_{\max}$ with the window size experienced just before the loss detection as well as $W_{\min}$ with the window size reduced after detection. Then, for every non-duplicate ACK reception BIC-TCP sets the window to a midpoint between $W_{\max}$ and $W_{\min}$. However, in case the distance between $W_{\min}$ and the midpoint becomes greater than the predefined *maximum increment* $S_{\max}$, the window is increased linearly by $S_{\max}$. Such a combination of the logarithmic and linear increase continues until the increment becomes less than another predefined constant $S_{\min}$, called *minimum increment*, and the window is set directly to $S_{\max}$.

BIC-TCP maintains high capacity utilization in the networks with routers implementing drop-tail as well as active queue management techniques (such as RED [27]). It improves RTT fairness while operating in logarithmic increase interval. Overall, BIC-TCP performs well in a large variety of evaluation scenarios [28]. Presently, BIC-TCP is a part of Linux 2.6 kernels enabled by default [29].

Changes of TCP/IP protocol stack are also considered for the next generation of the Microsoft Windows operating system. Along with other performance improvements, a Compound TCP (CTCP) approach [11] will be introduced in the kernel [30]. CTCP compounds traditional loss-based AIMD implemented in standard TCP with the delay-based approach derived from TCP Vegas [21,31].

All approaches presented above focus mainly on window increase strategy while window reduction is performed in a blind way. CTCP and BIC-TCP reduce their windows by $\beta = 0.125$, since window halving is considered to be too aggressive. The dynamics of window reduction define the tradeoff between capacity utilization and convergence of the protocol. An intelligent strategy for window reduction is a key component in TCP Westwood [17] design.

TCP Westwood keeps an estimate of the available capacity of the end-to-end path performing an appropriate filtering of the returning ACK flow. Then, upon loss detection, it adjusts the window fitting available BDP instead of blind halving. The window increase mechanism is left untouched and follows the conventional additive increase. Thus, congestion control strategy of TCP Westwood is classified as additive increase, adaptive decrease (AIAD).

The problems of bandwidth estimation accuracy encountered by TCP Westwood in the presence of compressed/delayed ACKs motivated a modification in the ACK filter used in bandwidth estimation algorithm, leading to the introduction of the TCP Westwood+ protocol [18].

While Westwood+ is able to outperform standard TCP in a wide range of application scenarios, it still suffers in the presence of large BDPs, 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 [32,33], the main function of the window increase algorithm is left to be additive with its conservative increase behavior.

3. TCP LogWestwood+

3.1. Logarithmic increase, adaptive decrease

After understanding the existing approaches, their features and limitations, we design a TCP modification targeting next generation TCP/IP protocol suite. Our design is based on a TCP Westwood+ modification as the only protocol bringing intelligent window reduction strategy, while the increase function is designed according to the following requirements:

1. to be more aggressive than traditional additive increase at any moment of time, in order to guarantee equal or better throughput performance and network utilization than other existing schemes;
2. to provide a fast window increase for low W values, while being accurate in approaching the available pipe capacity;
3. to present small sensitivity with respect to RTT.

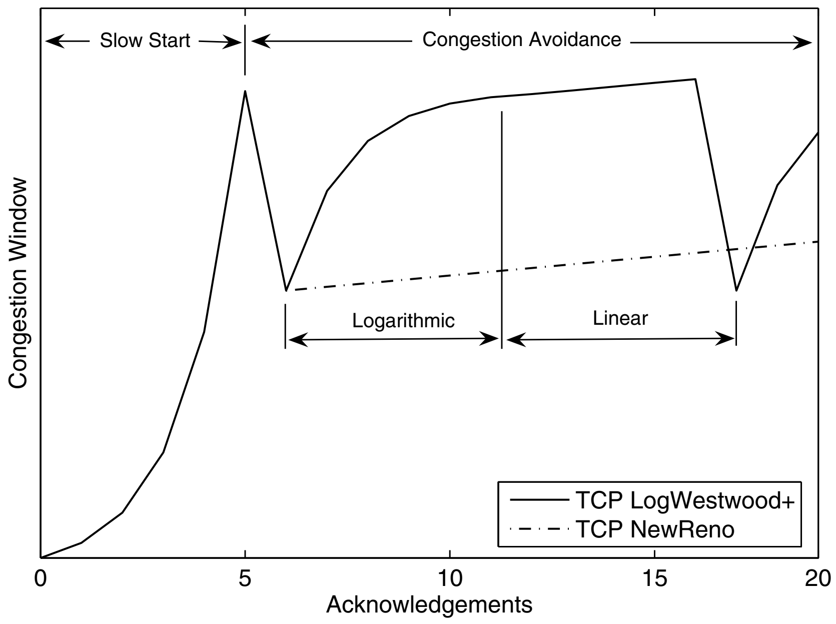


Fig. 1. Logarithmic increase, adaptive decrease (LIAD) in TCP LogWestwood+.

Given such considerations, a good match with the requirements is represented by the logarithmic function. Such set of functions require a value which upper-bounds the function. Ideally, the logarithmic increase must approach the BDP of the path which includes associated buffer resources as well. However, the difficulty of estimation of such a value using available techniques [34,35] leads us to choose the size of the window observed right before the last reduction W_{max} to serve as a target for the logarithmic function being a rough estimate of total capacity. Therefore, the use of logarithmic function is limited to congestion avoidance phase, while TCP slow start is left exponential. The proposed algorithm specifies that, for each acknowledgement packet received, the congestion window should be updated according to:

✓

$$W \leftarrow \frac{W_{max} - W}{\alpha W}, \qquad (1)$$

where W_{max} is defined as the window size at which the last packet loss event was detected.[2] Eq. (1) 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 lower-bound α by 2 (initial value) from the considerations that values less than 2 will make the increase too aggressive. According to (1) the growth of the congestion window value is aggressive for low W values, while becoming more prudent when approaching W_{max}, thus, satisfying design objectives 2 and 3. The nature of the obtained increase function is inline with the theoretical result contained in [36], which shows the optimality of a concave increase function for congestion control algorithms. However, in order to ensure that the designed function is always at least as aggressive as standard TCP mechanism (objective 1), we lower-bound the minimum window increase to one packet per RTT:

$$W \leftarrow \max\left(\frac{W_{max} - W}{\alpha W}, \frac{1}{W}\right). \qquad (2)$$

The proposed approach may recall binary increase function of BIC-TCP presented in [26], which has a conceptual similarity with the proposed function (letting $\alpha = 2$). However, in contrast to BIC-TCP, the proposed window increase always follows a logarithmic behavior with $W < W_{max}$, approaching a linear function when $W \to W_{max}$, and increasing linearly when probing for additional bandwidth with $W > W_{max}$.

As a result, the proposed window increase strategy (2), combined with the adaptive windows decrease performed by TCP Westwood+ constitutes a logarithmic increase, adaptive decrease (LIAD) mechanism. We refer to the resulting protocol as TCP LogWestwood+. Fig. 1 depicts the behavior of the LIAD mechanism and compares it with the conventional AIMD (TCP NewReno) approach. Both functions follow exponential slow start until the packet is lost. Then, after the first window reduction TCP LogWestwood+ climbs back to W_{max} using logarithmic increase and then continues to increase window linearly.

In summary, TCP LogWestwood+ is a loss-based approach which upon loss detection reduces its window to the BDP actually provided by the end-to-end link and then restores it back to the total BDP (including available on the path buffers) first using logarithmic and then linear functions.

The estimation of W_{max} in the proposed algorithm may not be accurate on the paths with rapidly increasing congestion. However, the algorithm for scaling α protects from performance degradation by controlling the level of aggressiveness of window increase strategy.

[2] Here we assume the loss is detected by the reception of three duplicated ACKs, while, in case of timeout expiration TCP initiates the slow start with $W = 1$.

3.2. Mathematical modeling

In this section, we aim at providing a simple fluid-flow model of TCP LogWestwood+ operations. This will be done under some simplifying assumptions, mostly following the analysis in [37], and aimed at providing insight into the impact of different system parameters on the protocol performance.

Let us consider a single TCP LogWestwood+ flow and assume that

i. each segment is lost with probability p;
ii. segment losses are detected by means of the reception of 3 dupACKs;
iii. the bandwidth estimate is $\widehat{B}$;
iv. the maximum window estimate is $W_{\max}$;
v. the mean round-trip time is RTT and the minimum round-trip time is $\mathrm{RTT}_{\min}$;
vi. the system operates in the large BDP regime, so that $\frac{W_{\max}-W}{\alpha W} \geqslant \frac{1}{W}$.

We start considering the sequence of events which modify the congestion window value. Given assumptions (i)–(vi), we can define a sequence of i.i.d. events, $\{\Delta(W)_n\}_{n\in\mathbb{N}}$, where $\Delta(W)_n$ describes the n-th variation of the congestion window value. The variation can be done to either the reception of an ACK or the detection of a loss. Upon the reception of a non-duplicated ACK packet, the congestion window is increased by a factor $\frac{W_{\max}-W}{\alpha W}$, which happens with probability $(1-p)$. Upon the detection of a packet loss, which takes place with probability p, the congestion window is shrinked down to $\frac{B\cdot \mathrm{RTT}_{\min}}{\mathrm{seg\_size}}$. The mean of $\Delta(W)_n$ is therefore given by

$$\mathbb{E}[\Delta(W)] = (1-p)\frac{W_{\max}-W}{\alpha W} + p\cdot\left[W - \frac{\widehat{B}\cdot \mathrm{RTT}_{\min}}{\mathrm{seg\_size}}\right]. \quad (3)$$

The congestion window value is updated W times per round-trip time, i.e., at a rate $\frac{W}{\mathrm{RTT}}$. We can therefore construct a fluid-flow model of the congestion window behavior, which satisfies the following partial differential equation:

$$\frac{\partial W}{\partial t} = \frac{W}{\mathrm{RTT}} \cdot\left\{(1-p)\frac{W_{\max}-W}{\alpha W} + p\cdot\left[W - \frac{\widehat{B}\cdot \mathrm{RTT}_{\min}}{\mathrm{seg\_size}}\right]\right\}. \quad (4)$$

The steady-state behavior of a TCP LogWestwood+ connection is analyzed studying the fixed point of (4). The result is the following:

Proposition 1. *Under assumptions (i)–(vi), the steady state throughput of a TCP LogWestwood+ connection is given by*

$$r^{\mathrm{logwest}} = \frac{1}{2\mathrm{RTT}}\left\{\left[(1-p)\widehat{B}\cdot \mathrm{RTT}_{\min} - \frac{p}{\alpha}\right] + \sqrt{\left[\frac{p}{\alpha} - (1-p)\widehat{B}\cdot \mathrm{RTT}_{\min}\right]^2 + \frac{4pW_{\max}}{\alpha}}\right\}. \quad (5)$$

Further, such point is globally exponentially stable.

Proof. We first note that the steady-state throughput and the steady-state congestion window values are related by

$$r^{\mathrm{logwest}} = \frac{W^*}{\mathrm{RTT}}, \quad (6)$$

where W^* is the (unique) fixed point of the system, i.e., the one that satisfies:

$$\left.\frac{\partial W}{\partial t}\right|_{W=W^*} = 0.$$

Setting the right-hand side equal to zero in (4), we get, after some algebra:

$$W^2 + W\left[\frac{p}{\alpha} - (1-p)\widehat{B}\mathrm{RTT}_{\min}\right] - \frac{pW_{\max}}{\alpha}. \quad (7)$$

which admits only one positive solution. Solving and using (6) we obtain (5).

In order to show that such equilibrium point is globally exponentially stable, we consider, as in [37] the following autonomous system, obtained from (4) by setting $Q = W - W^*$:

$$\frac{\partial Q}{\partial t} = \frac{1}{\mathrm{RTT}}\cdot\left[pW_{\max} - pQ - pW^* + (1-p)\widehat{B}\cdot \mathrm{RTT}_{\min}Q + (1-p)\widehat{B}\cdot \mathrm{RTT}_{\min}W^* - (Q+W^*)^2(1-p)\right]. \quad (8)$$

After some algebra, it can be reduced to the form:

$$\frac{\partial Q}{Q^2(t) + \beta Q(t)} = -\gamma\partial t, \quad (9)$$

where:

$$\gamma = (1-p)\left[p\frac{W_{\max}}{\mathrm{RTT}} - pW^* + (1-p)\widehat{B}\cdot \mathrm{RTT}_{\min}W^* + (W^*)^2(1-p)\right] > 0; \quad (10)$$

$$\beta = 2W^* + p(1-p) - \widehat{B}\mathrm{RTT}_{\min} > 0. \quad (11)$$

Eq. (9) is satisfied by

$$Q(t) = \frac{Q(0)e^{-\gamma\beta t}}{1 + \frac{Q(0)}{\gamma(1-e^{-\gamma\beta t})}}, \quad (12)$$

which satisfies

$$Q(t) \leqslant Q(0)e^{-\gamma\beta t}, \quad (13)$$

thus, concluding the proof. □

In order to better understand the dependence of the proposed protocol performance on the round-trip time and the packet loss probability, we present an alternative model, based, again, on the results in [37]. The basic assumption is that, if the network dynamics is slow with respect to the protocol behavior, and if the loss probability p is small enough, the bandwidth estimate can be taken to be:

$$\widehat{B} = \frac{W}{\mathrm{RTT}}. \quad (14)$$

We then get the following:

Proposition 2. *Under assumptions (i)–(vi) and (14), the steady state throughput of a TCP LogWestwood+ connection is given by*

$$r^{\text{logwest}} = \frac{1-p}{2\alpha p T_q} \cdot \left(\sqrt{1 + \frac{4W_{\max} T_q \alpha p}{(1-p)\mathrm{RTT}}} - 1\right) \qquad (15)$$

where $T_q = RTT - RTT_{\min}$.

Proof. Under assumption (14), (4) becomes

$$\frac{\partial W}{\partial t} = \frac{W}{\mathrm{RTT}} \cdot \left[pW\left(\frac{\mathrm{RTT}_{\min}}{RTT} - 1\right) + (1-p)\frac{W_{\max} - W}{\alpha W}\right]. \qquad (16)$$

The fixed point can be found by setting the right hand side of (16) equal to zero. By introducing $T_q = \mathrm{RTT} - \mathrm{RTT}_{\min}$, we get

$$\alpha p T_q W^2 + (1-p)\mathrm{RTT} \cdot W - (1-p)\mathrm{RTT} \cdot W_{\max} = 0, \qquad (17)$$

which can be easily shown to present only one positive solution, given by

$$W^* = \frac{(1-p)\mathrm{RTT}}{2\alpha p T_q} \cdot \left(\sqrt{1 + \frac{4W_{\max} T_q \alpha p}{(1-p)\mathrm{RTT}}} - 1\right). \qquad (18)$$

Dividing by RTT we find (15). □

We can therefore draw the following conclusions:

- the LogWestwood+ protocol is less sensitive than standard TCP congestion control mechanism to the RTT value. Indeed, if $\frac{4W_{\max} T_q \alpha p}{(1-p)\mathrm{RTT}} \gg 1$, then $r^{\text{logwestwood}} \sim \frac{1}{\sqrt{\mathrm{RTT}}}$, which complies with TCP Westwood+ behavior [37] and shows a significant enhancement with respect to the $\frac{1}{\mathrm{RTT}}$ relationship of standard TCP NewReno [7]. However, if the packet loss probability p is low enough, we can expect such a term $\frac{4W_{\max} T_q \alpha p}{(1-p)\mathrm{RTT}}$ to be relatively small, with little impact on the connection throughput. In the limit $p \to 0$, the protocol performance turn out to be *insensitive* to the RTT value. We can therefore conclude that the LogWestwood+ protocol presents a lower level of sensitivity on RTT values with respect to Westwood+.
- under the assumption $p \ll 1$, and using an expansion of the square root term in (15), it is easy to see that the throughput of a LogWestwood+ connection scales with the packet loss probability as $r^{\text{logwest}} \sim \frac{1}{\sqrt{p}}$, thus, providing fairness with both Westwood+ [37] and legacy NewReno protocols [7].

4. Performance evaluation (i): ns2 simulations

This section presents performance evaluation results of the proposed TCP LogWestwood+ protocol using the ns-2 network simulator [10] in its current (2.31) version. The

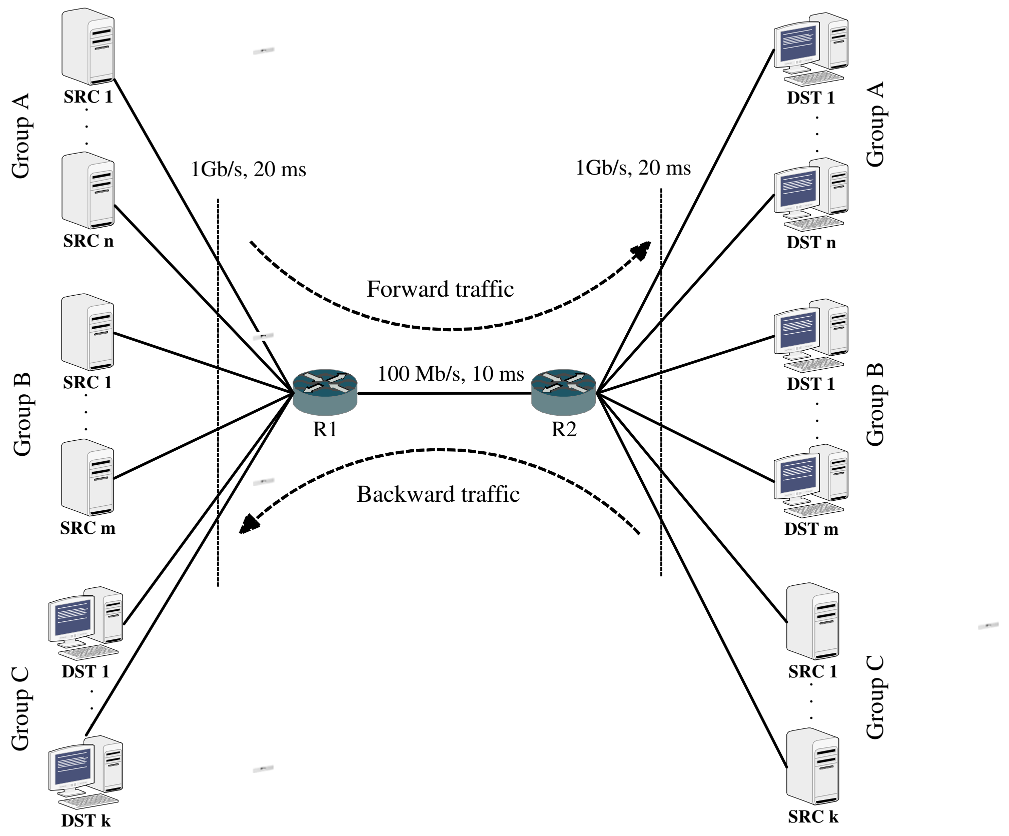


Fig. 2. The proposed simulation scenario.

simulation scenario presented in Fig. 2 is designed to evaluate protocol dynamics in single and multi-flow environments, against variable link errors as well as with cross-traffic flowing in the reverse direction.

The simulation setup is chosen to be consistent with the one presented by the authors of [18] and used for comparison of TCP Westwood+, NewReno, and Vegas. It consists of a single 100 Mb, 10 ms bottleneck between *R*1 and *R*2 routers as well as three groups of TCP source and destination nodes: *Group A* consists of *n* sources running TCP protocol version under evaluation; *Group B* is used for inter- and intra-protocol fairness measurements and consists of *m* sources running the same or concurrent with the respect to *Group A* version of TCP; *Group C* consists of *k* sources designed to produce bursty cross-traffic flowing in the reverse direction. All source and destination nodes are connected to the bottleneck buffers using 1 Gb/s, 40 ms links.

Such configuration allows 18 Mb of data to be outstanding in the network, which corresponds to a BDP value of 1500 1.5 K-byte TCP packets. The outgoing queue size of network nodes is fixed to 300 packets which corresponds to 450 Kbytes. Each simulation runs for 2000 s of simulation time, while the protocol stabilization period, i.e., slow start, is excluded from the measurement intervals.

In order to understand the performance improvements achievable with the proposed solution, we varied Packet Error Rate (PER) on the bottleneck link, bottleneck buffer size, bandwidth of the bottleneck link as well as the number of flows generated by different groups of sources.

The proposed LogWestwood+ protocol is compared with its ancestor Westwood+, Binary Increase Control (BIC), as well as with the standard TCP NewReno with selective acknowledgements (SACK) which is the most widespread TCP version currently deployed in the Internet. TCP Westwood+ modules were obtained from [38], on top of which the proposed logarithmic increase function was implemented. BIC-TCP modules were obtained from [39]. The main performance metrics used in evaluation are: bandwidth utilization, intra- and inter-protocol fairness.

4.1. Bandwidth utilization performance

Fig. 3 presents single flow congestion window dynamics observed over a 220s time interval for a scenario with a single *Group A* flow and no cross-traffic. From the graph, it is easy to see that LogWestwood+, Westwood+, and BIC keep the operational point within buffer space allocated at the bottleneck buffer, with BIC slightly overestimating available link capacity. Upon packet loss NewReno drops its window far below available end-to-end network capacity, resulting in under-utilization of network resources. LogWestwood+ protocol appears to be almost the same aggressive as BIC in the presented scenario. However, as opposed to BIC, it keeps sending rate low when approaching total link capacity.

In this scenario TCP NewReno requires almost 200 s to recover from a loss event, TCP Westwood+ takes about 100 s for window increase from bandwidth estimate to the total network capacity, BIC-TCP takes 12 s on average, while LogWestwood+ requires 16.4 s for performing the same operation.

LogWestwood+, Westwood+, and BIC keep their congestion windows in the interval between estimated capacity of the link and its total capacity (which includes buffers along the 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.

Summarizing, for high bandwidth utilization window decrease algorithm should tend not to reduce the window far below available end-to-end path capacity, while window increase algorithm should be relatively aggressive in order to fulfil the bottleneck buffer relatively fast, but try

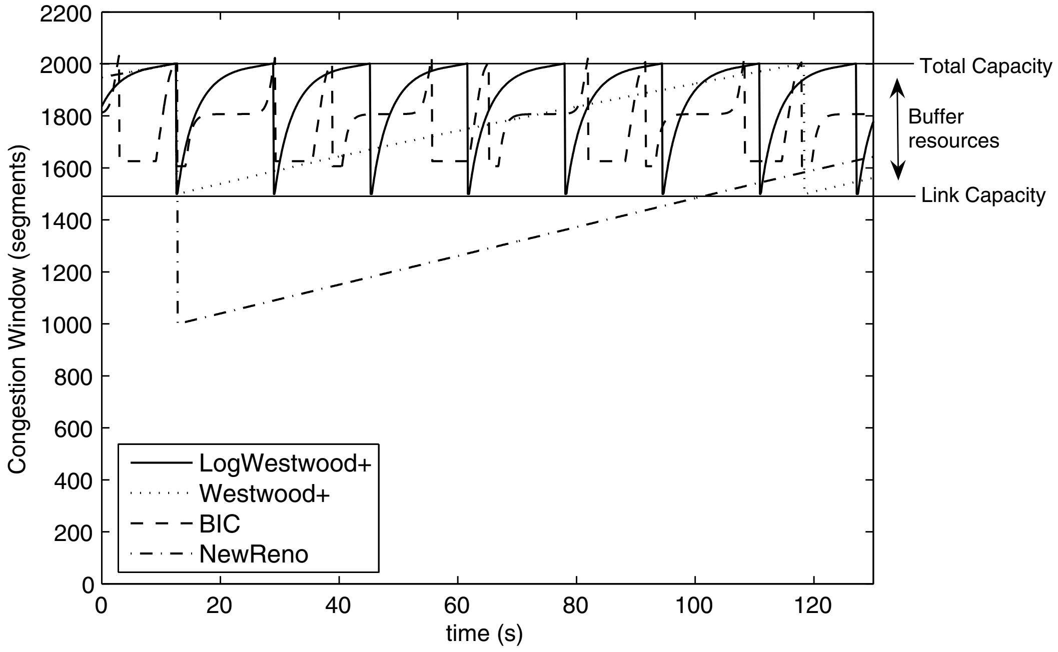


Fig. 3. Window increase of NewReno, Westwood+ and LogWestwood+.

not to overflow it as long as possible. It should also be noted that the fact of stressing the bottleneck buffer will increase average end-to-end end propagation delay would be remain true for a single flow. However, it will remain qualifiedly unchanged in multiflow network environment. In order to evaluate the performance gain obtained from the proposed logarithmic window function, an error generator is connected to the bottleneck link between routers R1 and R2 (see Fig. 2). Inserted errors are at the packet level and follow a memoryless pattern, each packet being lost independently with the probability between $\mathrm{PER} = 10^{-8}$ and $\mathrm{PER} = 10^{-1}$.

The results obtained are reported in Fig. 4 in terms of 95% confidence interval averaged over 10 runs. In the scenarios with low or no errors the confidence intervals are negligibly small, and thus, are not reported.

A single-flow goodput is measured as the number of bits successfully delivered by the connection at the receiver node. BIC and NewReno protocol start to degrade their performance at $\mathrm{PER} = 10^{-7}$ while Westwood+ and LogWestwood+ sustain $\mathrm{PER} = 10^{-6}$ and $\mathrm{PER} = 10^{-5}$ respectively. This favors LogWestwood+ deployment in high error rate networks such as wireless links.

The performance gain of LogWestwood+ increases for higher PERs as a consequence of faster window recovery from the occurred packet loss. In more details, while traditional NewReno assumes that all packet losses are caused by network congestion, 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. 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+ allows a faster congestion window recovery after a packet loss event. The smaller amount of time spent in the period of channel underutilization leads to a corresponding performance enhancement. In Fig. 5, we illustrate the performance enhancements, in terms of goodput ratio, achieved by LogWestwood+ against Westwood+, BIC, 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 $10^{-6} - 10^{-2}$). If compared with BIC and NewReno, the proposed LogWestwood+ achieves an improvement of up to 100% against BIC and up to 350% against NewReno for a PERs in the interval $10^{-5} - 10^{-2}$.

For high PERs ($> 10^{-1}$), 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. As a result, the protocols spend most of the time in the slow start leading to similar performance.

Another important observation is related TCP buffer usage. It is well known that NewReno achieves 100% of network utilization only in case it operates over a bottleneck buffer equal or greater than end-to-end BDP allocated for on per-connection basis [40]. This buffer is designed to compensate TCP window halving - the results of congestion related loss detection with three duplicate TCP ACKs.

However, an assumption of having the above mentioned buffer resources for every TCP connection in high speed networks is unrealistic. The guidelines provided by Cisco Systems suggest fixing buffer sizes of high speed routers to 500 packets [41] which is typically shared among

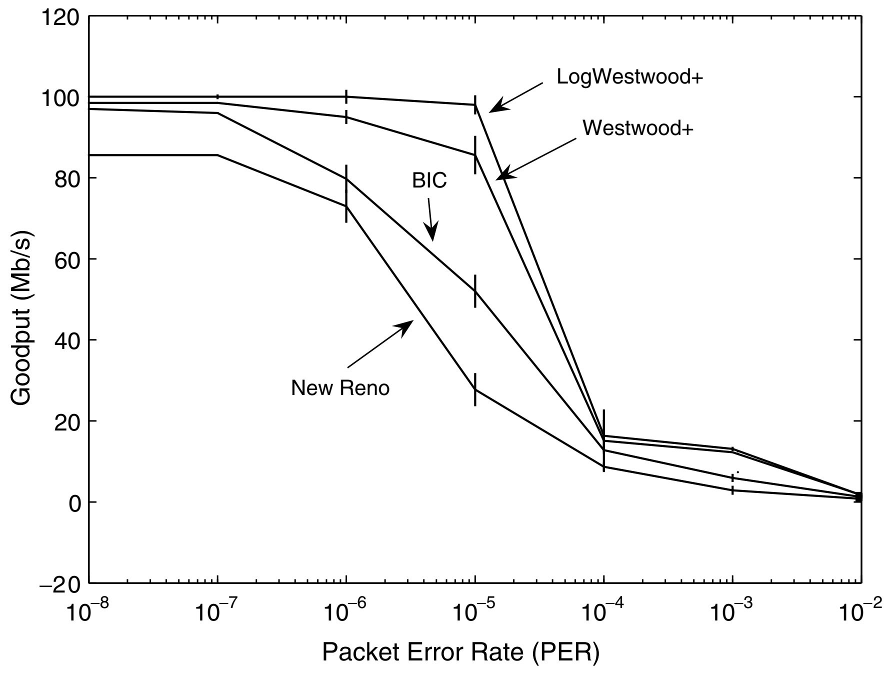


Fig. 4. Goodput performance comparison in the presence of random errors.

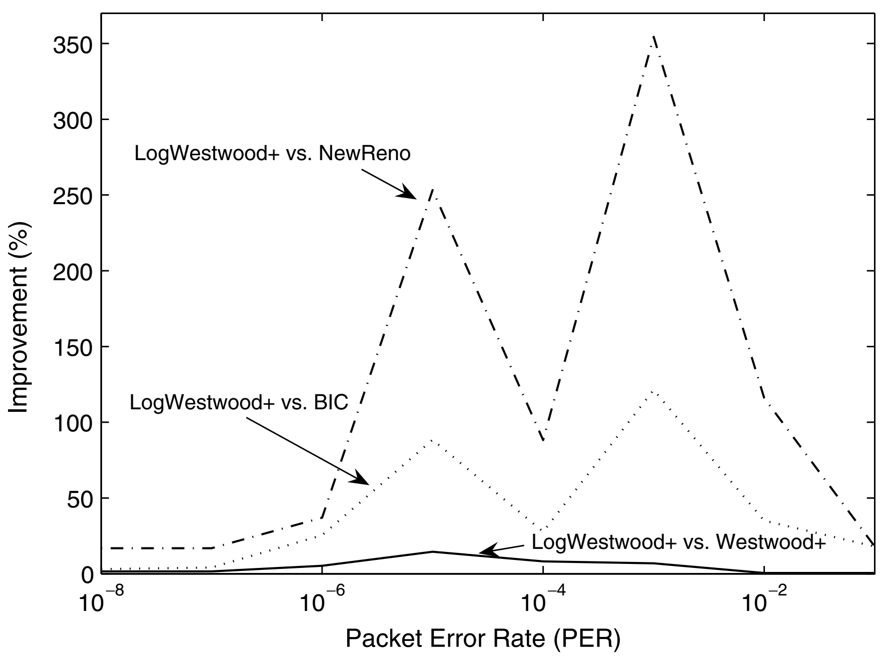


Fig. 5. LogWestwood+ performance improvement.

thousands of TCP flows with RTTs as large as 250 ms for intercontinental and 600 ms for satellite links. Insufficient buffers create the problem of TCP underutilization evidenced in Fig. 4 where even for low error rates PER $< 10^{-8}$ BIC and NewReno utilize only 97% and 85% of network resources.

Fig. 6 studies buffer requirements for evaluated TCP protocols modifications. As expected, NewReno monotonically decreases its throughput with the decrease of available buffer resources. The performance of BIC is relatively stable for buffers larger than 250 packets. However, it is dramatically reduces for small buffers sizes (less than 50 packets). Despite Westwood+ due to its adaptive decrease strategy outperforms both BIC and NewReno it is also vulnerable to small (less than 100 packets) buffer sizes. This drawback is improved by the more aggressive logarithmic increase function in LogWestwood+ which is almost insensitive to the size of the bottleneck buffer.

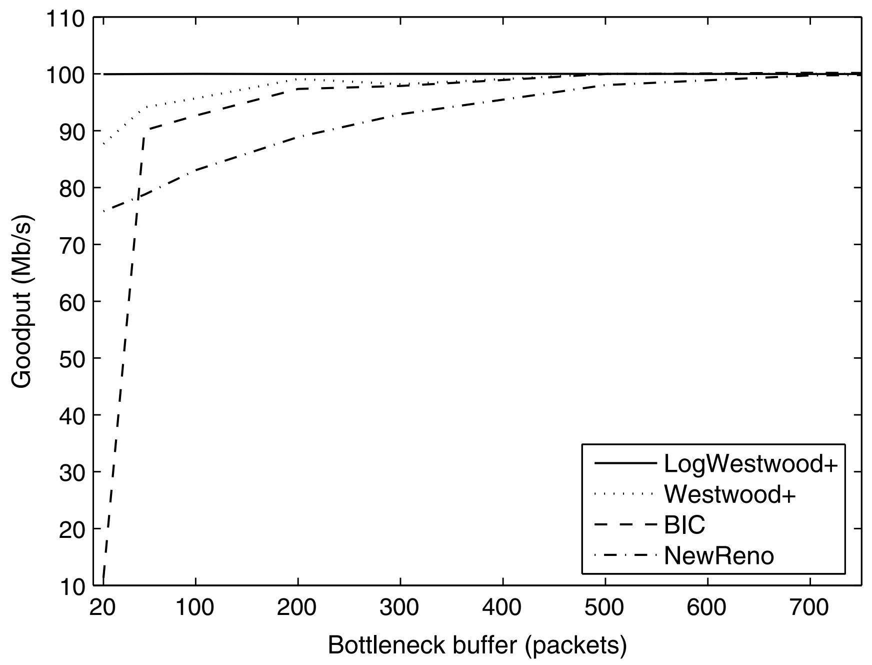


Fig. 6. TCP protocol buffer requirements.

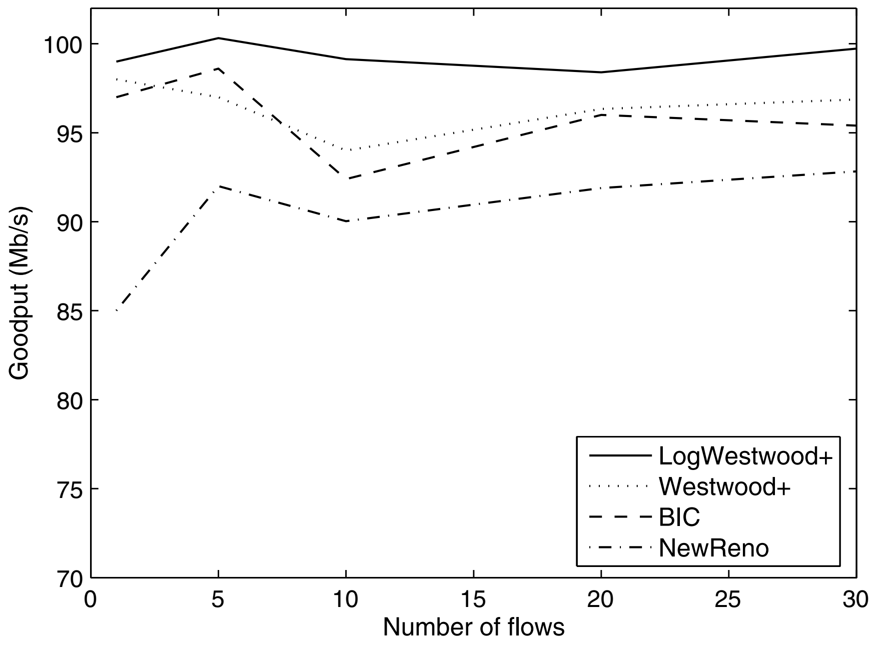


Fig. 7. Multi-flow throughput performance comparison.

Fig. 7 reports a performance comparison for the case of multiple concurrent flows. We considered the aggregated throughput as performance metric, and varied the number of *Group A* flows (started at the beginning of the simulation) 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 Westwood+, BIC and NewReno TCP implementations.

Another important metric to consider is the performance obtained in the presence of cross-traffic flowing in the direction opposite to TCP data flow. In order to simulate such cross-traffic, we established several NewReno connections between *Group C* source and destination nodes. As a result, this cross-traffic is responsible for compression of TCP ACKs generated by *Group A* flows which started to be queued in R1 buffer required to wait for *Group C* data packets transmission. Along with the compression, cross-traffic congests the reverse link leading to TCP ACKs losses. The effect of the cross-traffic at the reverse link is studied in details by the authors of [18].

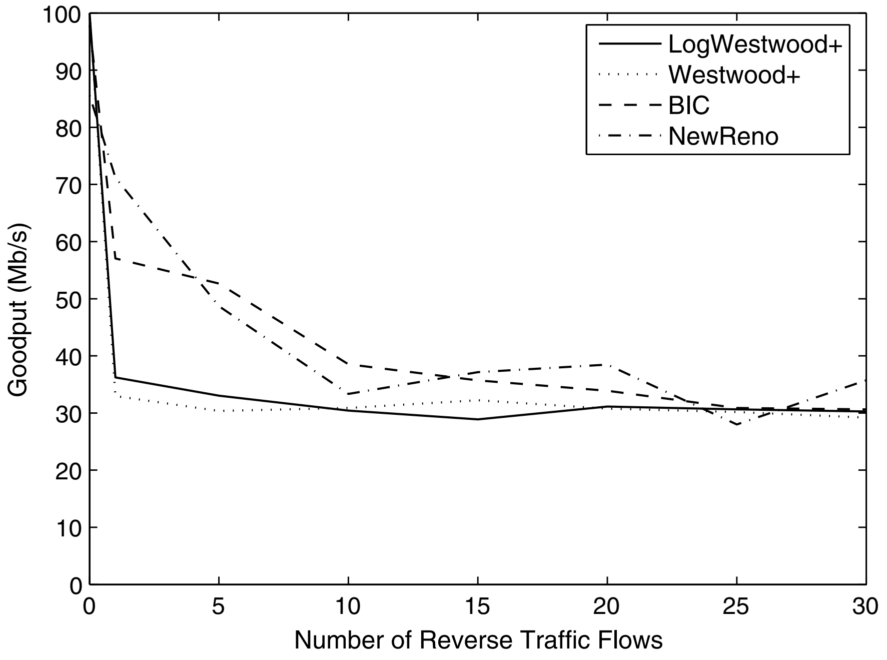


Fig. 8. Goodput performance against intensive reverse traffic.

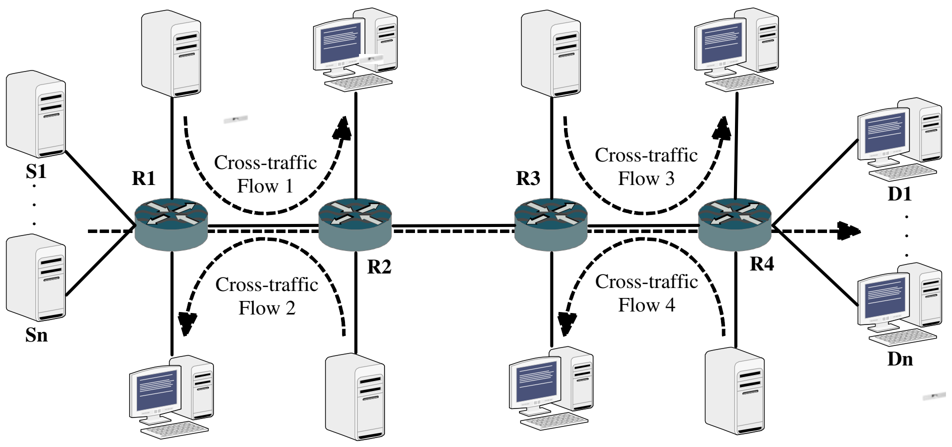


Fig. 9. Multi bottleneck topology.

Fig. 8 presents a comparison in terms of goodput performance for a single *Group A* flow against different numbers of *Group C* flows. It shows that reverse traffic degrades TCP performance severely. For low number of *Group C* flows (less than 10), NewReno and BIC outperform Westwood+ and LogWestwood+ modifications. The main reason for that is that the inserted cross-traffic disturbs Westwood+ bandwidth estimation algorithm which is based on TCP ACK flow measurements. For more than 10 reverse flows, all the evaluated protocols achieve only 30% of the available bottleneck capacity.

A more complex scenario with multiple bottlenecks and cross-traffic added in both obverse and reverse directions is presented in Fig. 9. The capacity of links between routers is 100 Mb/s, 10 ms while entry/exit links are of 1 G/s, 20 ms capacity. Similar to a scenario with a single bottleneck router buffers are limited to 500 packets. Cross-traffic is represented by NewReno flows which enter and stay the path shared with evaluated version of TCP for a single hop: cross-traffic flows 1 and 3 in forward and cross-traffic flows 2 and 4 in reverse direction. The flows running evaluated TCP version are initiated between sources *S* and destinations *D* which make them path-persistent.

The obtained results presented in Fig. 10 that LogWestwood+ and BIC are able to force the cross-traffic flows out with fewer number of flows if compared to Westwood+ and NewReno protocols. The fact that LogWestwood+ is less aggressive than BIC results in lower (on average by 2%) performance in multi bottleneck scenario but is essential for protocol fairness.

4.2. Intra- and inter-protocol fairness

In order to evaluate the fairness properties of the proposed solution, we used two groups of source and destination nodes, *Group A* and *Group B*. In this way, *Group A*

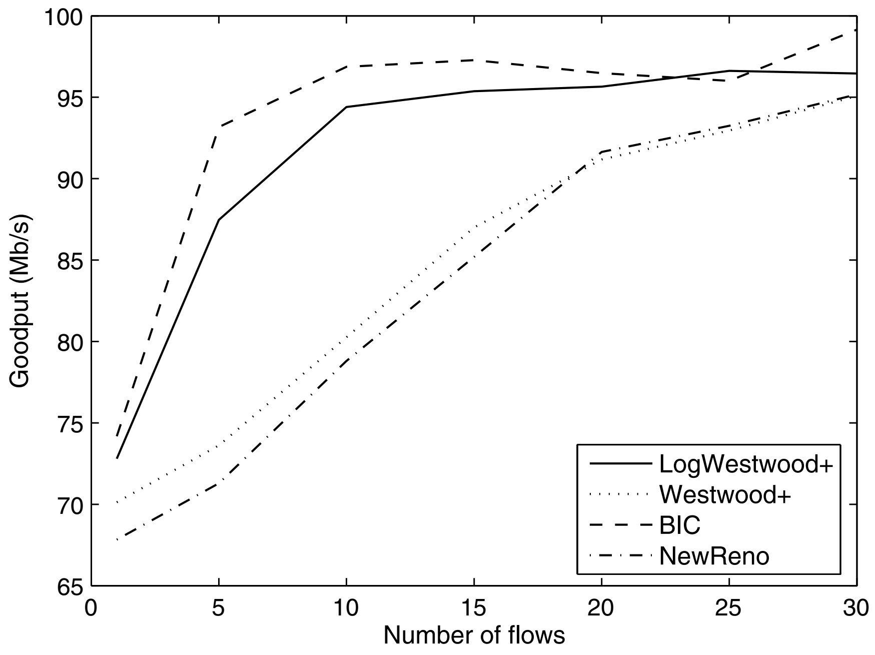


Fig. 10. Performance comparison in multi bottleneck topology with cross-traffic.

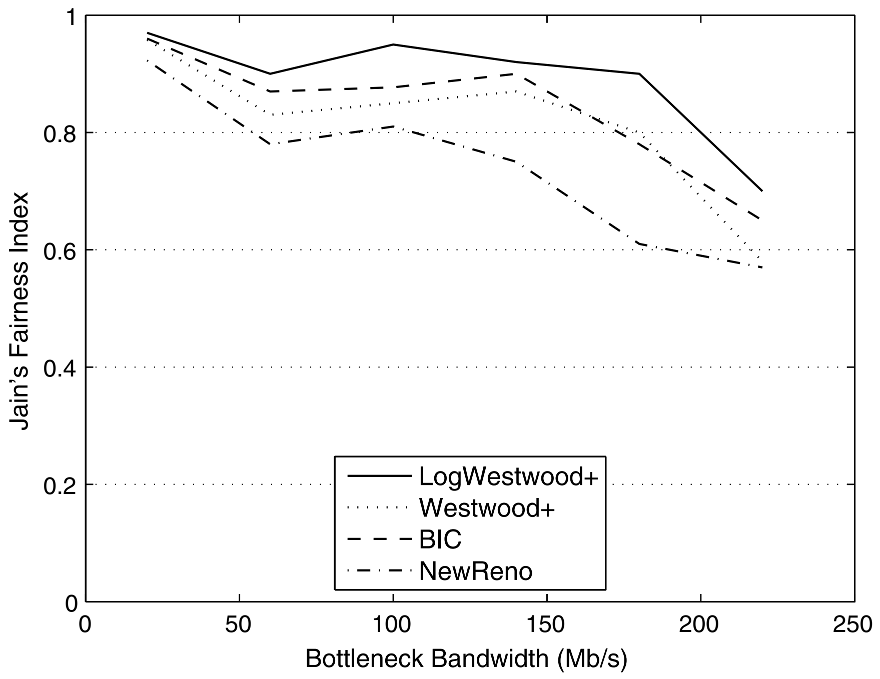


Fig. 11. Jain's protocol fairness for different bottleneck capacities.

nodes are connected using 1 Gb/s, 20 ms links, while 1 Gb/s, 80 ms links connect *Group B* nodes to the bottleneck link.

In such configuration *Group A* flows get receiver's feedback approximately 3.4 times faster than *Group B* sources. This lead to the unfairness induced by different RTTs among flows.

We used Jain's Fairness Index [42] as the main measure of intra-protocol fairness, which is defined as follows,

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)}. \tag{19}$$

The total 8 sources are equally divided between *Group A* and *Group B*. The goodput of each connection is averaged over 10 runs with different random generator seeds and bottleneck PER $= 10^{-7}$. Fig. 11 presents the computed index values for the evaluated protocols operating over a variable bottleneck capacity.

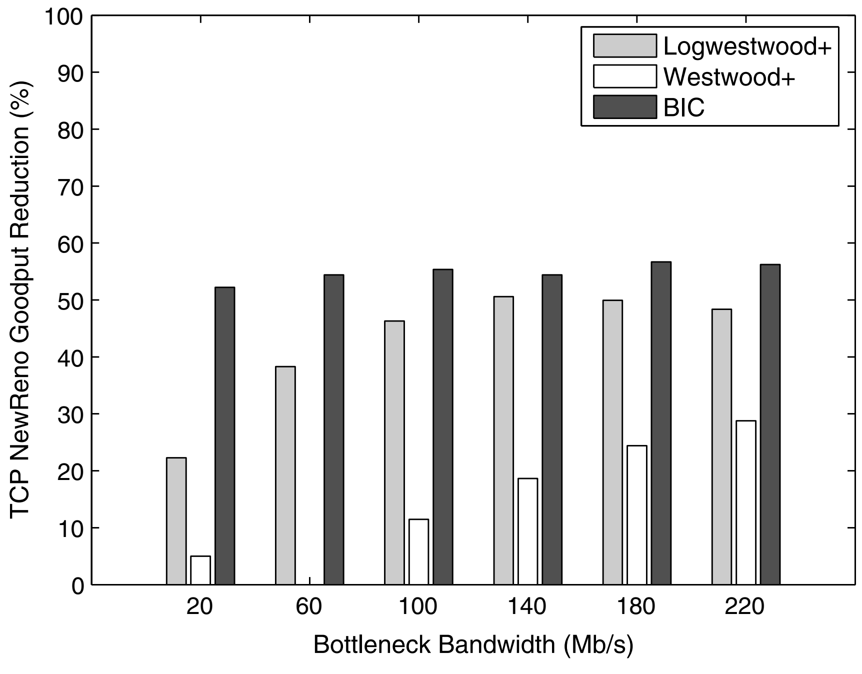


Fig. 12. Inter-protocol fairness for different bottleneck capacities.

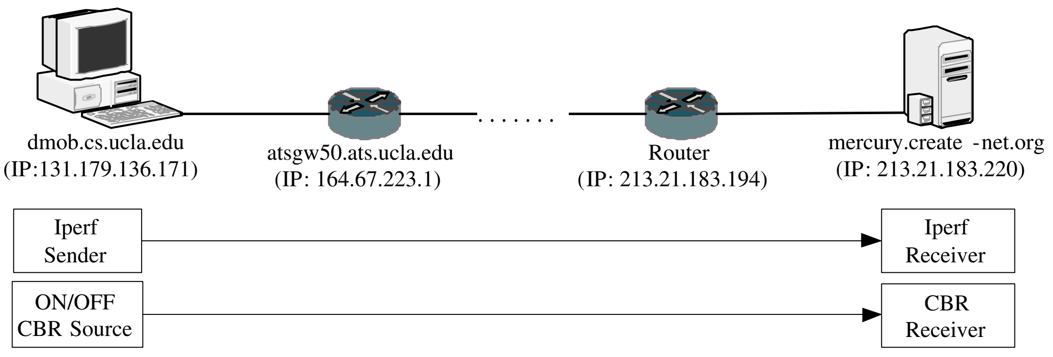
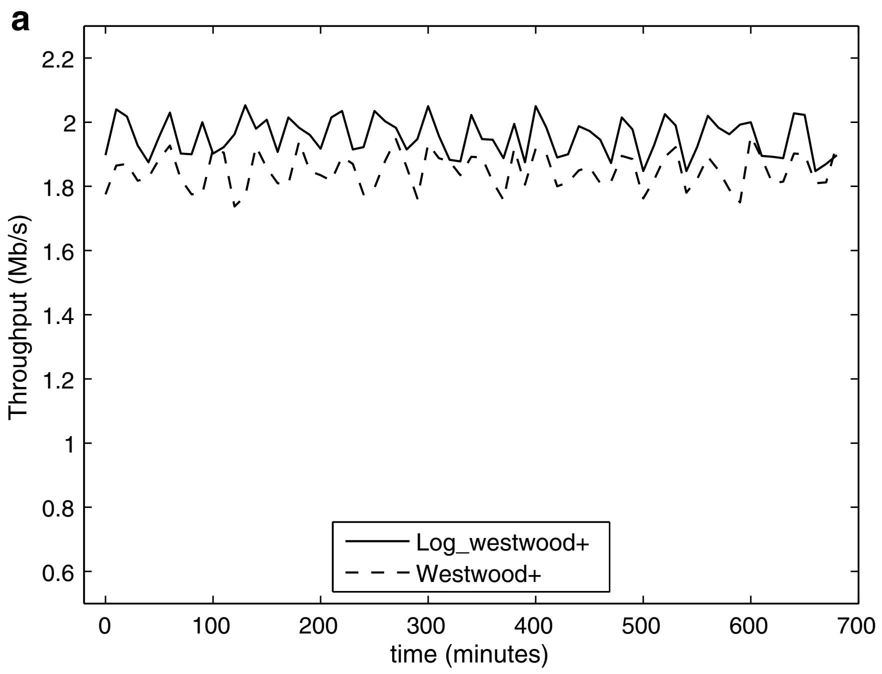


Fig. 13. Internet testbed.



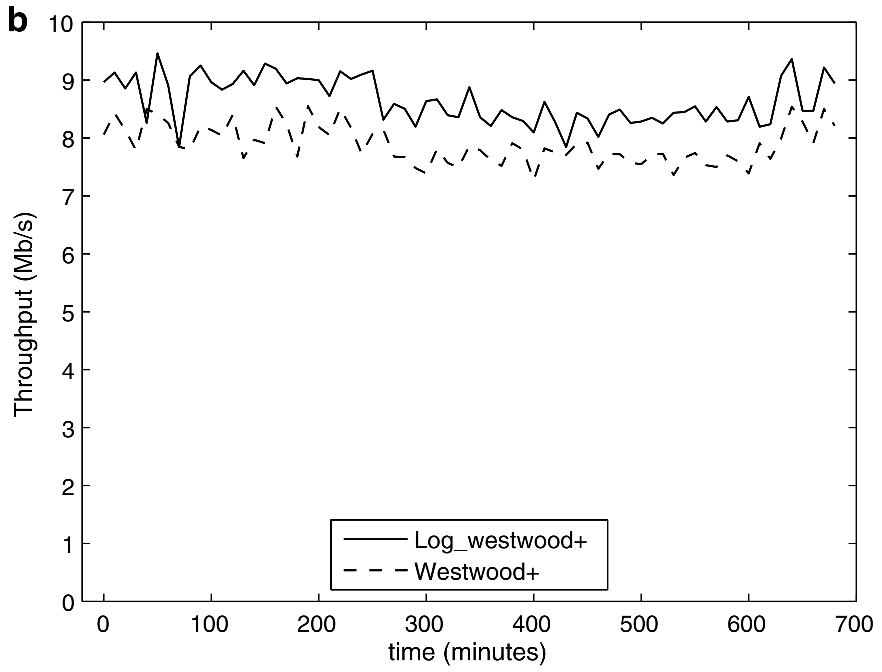


Fig. 14. (a) Single flow and (b) multi-flow throughput comparison of LogWestwood+ against Westwood+ using Internet testbed experiments.

TCP LogWestwood+ shows better fairness than other protocols in all the considered scenarios. While all the evaluated protocols are almost perfectly fair for relatively tight bottleneck links (20 Mb/s) the difference is increased with capacity increase. Generally, more aggressive TCPs demonstrate better fairness. This is mainly because they allow connections with longer RTT grab their bottleneck share faster. While Westwood+ improves NewReno's fairness, the behavior of the curve remains unchanged.

In order to evaluate inter-protocol fairness we specified *Group B* sources always to run NewReno protocol, while *Group A* sources implemented the concurrent TCP version. Additionally, unfairness due to different RTTs among flows was introduced inside each group of sources: half of each group is connected to bottleneck routers with 1 Gb/s, 20 ms links, while the other half used 1 Gb/s, 80 ms links.

In our opinion the main reason behind inter-protocol fairness evaluation is to ensure the possibility for incremental deployment of the proposed protocol. This means that the proposed protocol should not necessarily be perfectly fair with the most widely implemented TCP NewReno. However, its deployment should not degrade TCP NewReno's performance greatly.

Following this observation, we measured the difference in the goodput of *Group B* nodes running NewReno protocol with (i) *Group A* goodput running NewReno flows, and (ii) *Group A* running concurrent protocol version (Westwood+, LogWestwood+, or BIC). According to the obtained results presented in Fig. 12 Westwood+ demonstrate the best inter-protocol fairness, LogWestwood+ reduces the goodput of concurent NewReno flows by 20–50%, and BIC appears to be the most unfair one by reducing NewReno goodput by 55% on average. An additional observation is that the level of LogWestwood+ and Westwood+ unfairness depends on the bottleneck capacity.

Overall, the obtained results demonstrate moderate unfairness of LogWestwood+. This is not in the favor of incremental deployment in current Internet where most of the flows are based on NewReno protocol. However, taking into account that BIC is already a part on OS Linux kernel and its inter-protocol unfairness (also pointed in [43]) is the lowest among all the evaluated protocols LogWestwood+ remains an attractive alternative able to compete with BIC.

5. Performance evaluation (II): WAN measurements

In this section we present the outcomes of some Internet measurements we performed over an intercontinental connection (US to Italy) in order to verify the results obtained from the simulations. In order to support the desired functionality, we extended TCP Westwood+ implementation included into Linux distributions with the logarithmic window increase function. Starting from the kernel version 2.6.13 congestion control is separated from the kernel into pluggable modules which can be altered dynamically either by a `sysctl` or on a per socket basis enabling fast on-demand switching between protocols.

The testbed used for the experiments is presented in Fig. 13. Iperf version 2.0.2 [44] was used as a throughput measurement tool. The sender, located at the UCLA campus (Los Angeles, CA), communicates with the receiver node which is a part of CREATE-NET research center network (Trento, Italy) over an 18-hop link.

At the beginning of each measurement cycle, which lasts for 10 minutes, the sender starts with 1-minute long transfer using Westwood+ protocol; then it switches to LogWestwood+ for another minute followed by 8 minutes of idle time. In addition to performance tests, the sender node runs a Constant Bit Rate (CBR) imitating background traffic load. This CBR genarator is the on/off source which being in ON period it produces 1 Mb/s UDP flow. Both ON and OFF periods last for 10 s each.

Fig. 14 presents the throughput traces achieved by LogWestwood+ and Westwood+ protocols in a 10-h experiment.

The proposed LogWestwood+ outperforms Westwood+ modification for the entire duration of the experiment demonstrating approximately 4% of an improvement for single flow scenario (Fig. 14a) and up to 8% if multi-flow communication (Fig. 14b) takes place.

The experiments were performed during the weekend night when the background Internet traffic is at the minimum. However, it was not avoided completely, resulting in throughput fluctuations. For instance, a long term cross-traffic influence can be observed between 300 s and 600 s in Fig. 14b.

6. Conclusions

In this paper, we have presented a novel congestion control algorithm for performance enhancements in high speed, long distance networks. Our TCP LogWestwood+ protocol is an enhancement of TCP Westwood+ based on a logarithmic increase function. The obtained congestion control function shows low sensitivity to RTT values. Moreover, it achieves high levels of network utilization by being aggressive when the sending rate is below the available path capacity. The intra- and inter-protocol fairness is ensured by means of the adoption of a smooth window increase when approaching the available end-to-end path capacity. Additionally, being always equally or more aggressive than standard TCP, LogWestwood+ modification ensures equivalent or better performance in large BDP scenarios. A simple fluid-flow model has also been provided, in order to better understand the equilibrium and stability properties of the proposed algorithm.

However, along with advantages it should be noted that the bandwidth estimation performed by LogWestwood+ is inherited from Westwood+ modification and may be inaccurate. Specifically, this bandwidth estimation algorithm is based on filtering, at the sender side, TCP ACKs generated by the receiver. The obtained bandwidth estimate is the core of adaptive window reduction algorithm of Westwood and Westwood+. It has been reported that, in the presence of intensive cross-traffic at the reverse path, such bandwidth estimation algorithm may prove ineffective. This is a clear drawback of Westwood and Westwood+ protocols which extends its influence to LogWestwood+ protocol (demonstrated in Section 4.1).

In this framework, the logarithmic window increase proposed in the paper is considered as a possible replacement for additive increase while leaving other TCP semantics (including bandwidth estimation and adaptive window decrease) of Westwood and Westwood+ modifications unaffected.

Extensive ns2 simulations were performed to understand the performance and fairness characteristics of LogWestwood+. The proposed protocol was also implemented in a Linux system and tested running measurements over a wide area network.

Directions for future research include more extensive tests of the Linux implementation in different operating conditions, especially in the presence of dynamically-varying traffic where the performance of LogWestwood+ demonstrate high dependance on α.

Acknowledgements

The authors are grateful to Prof. Mario Gerla and Network Research Lab researchers for allowing us to perform measurements using servers located at the University of California, Los Angeles.

References

[1] D. Kliazovich, F. Granelli, D. Miorandi, TCP Westwood+ enhancements for high-speed long-distance networks, in: Proceedings of the IEEE ICC, Istambul, Turkey, 2006. Available from: <http://dit.unitn.it/~klezovic/papers/LogWestwood+.pdf>.
[2] J.B. Postel, Transmission control protocol, RFC 783 (September) (1981).
[3] G. Miller, K. Thompson, R. Wilder, Wide-area Internet traffic patterns and characteristics, IEEE Network 11 (1997) 10–23.
[4] N. Brownlee, K. Claffy, Understanding Internet traffic streams: dragonflies and tortoises, IEEE Commun. Mag. 40 (October) (2002) 110–117.
[5] V. Jacobson, M.J. Karels, Congestion avoidance and control, in: Proceedings of the ACM SIGCOMM, Stanford, CA, 1988.
[6] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Trans. Network. 7 (August) (1999) 458–472.
[7] J. Padhya, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, in: Proceedings of the IEEE SIGCOMM, 1998.
[8] S. Floyd, HighSpeed TCP for large congestion windows, RFC 3649, December, 2003.
[9] T. Kelly, Scalable TCP: improving performance in highspeed wide area networks, ACM Comput. Commun. Rev. 33 (April) (2003) 83–91.
[10] G. Marfia, C. Palazzi, G. Pau, M. Gerla, M. Sanadidi, M. Roccetti, TCP-Libra: exploring RTT fairness for TCP, in: IEEE JSAC Special Issue on Non Linear Optimization of Communication Systems, 2005. Available from: <http://www.gpau.net/papers/2005-TR050037.pdf>.
[11] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound TCP approach for high-speed and long distance networks, in Microsoft Press, MSR-TR-2005-86, July, 2005.
[12] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, K.C. Claffy, The RTT distribution of TCP flows on the Internet and its impact on TCP based flow control, in: CAIDA Technical Report, TR-2004-02, January, 2004. Available from: <http://www.ifp.uiuc.edu/~sshakkot/tr-2004-02.pdf>.
[13] J. Aikat, J. Kaur, F.D. Smith, K. Jeffay, Variability in TCP roundtrip times, in: Proceedings of the IMC, Miami Beach, FL, USA, October, 2003.
[14] A. Ghosh, D.R. Wolter, J. Andrews, R. Chen, Broadband wireless access with WiMax/802.16: current performance benchmarks and future potential, IEEE Commun. Mag. 43 (2) (2005) 129–136.
[15] H. Ekstrom, A. Furuskar, J. Karlsson, M. Meyer, S. Parkvall, Technical solutions for the 3G long-term evolution, IEEE Commun. Mag. 44 (3) (2006) 38–45.
[16] U. Varshney, R. Jain, Issues in emerging 4G wireless networks, IEEE Comput. 34 (6) (2001) 94–96.
[17] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, TCP Westwood: end-to-end bandwidth estimation of efficient transport over wired and wireless networks, in: Proceedings of the ACM Mobicom, Rome, Italy, 2001.
[18] L.A. Grieco, S. Mascolo, Performance evaluation and comparison of Westwood+, New Reno, and Vegas tcp congestion control, ACM Comput. Commun. Rev. 34 (April) (2004) 25–38.
[19] R. Ferorelli, L.A. Grieco, S. Mascolo, G. Piscitelli, P. Camarda, Live Internet measurements using Westwood+ TCP congestion control, in: Proceedings of the IEEE Globecom, Taipei, Taiwan, November, 2002.
[20] The network simulator ns2. Available from: <http://www.isi.edu/nsnam/ns>.
[21] L. Brakmo, S. O'Malley, L. Peterson, TCP Vegas: new techniques for congestion detection and avoidance, in: Proceedings of the SIGCOMM Symposium, August 1994.
[22] J. Mo, R.J. La, V. Anantharam, J. Walrand, Analysis and comparison of TCP Reno and Vegas, in: Proceedings of the INFOCOM, March, 1999.
[23] R.S. Prasad, M. Jain, C. Dovrolis, On the effectiveness of delay-based congestion avoidance, in: Proceedings of the PFLDnet, 2004.
[24] D.M.G. Giambene, Performance evaluation of scalable TCP and highspeed TCP over geostationary satellite links, in: Proceedings of the IEEE Vehicular Technology Conference, June, 2005.
[26] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast long-distance networks, in: Proceedings of the IEEE INFOCOM, Hong Kong, 2004.
[27] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Network. 1 (August) (1993) 397–413.
[28] H. Bullot, R.L. Cottrell, R. Hughes-Jones, Evaluation of advanced TCP stacks on fast long-distance networks, in: Proceedings of the UltraNet workshop, Lyon, France, November, 2003.
[29] The linux kernel archives. Available from: <http://www.kernel.org>.
[30] Performance enhancements in the next generation TCP/IP stack, in The Cable Guy, Microsoft TechNet, November, 2005. Available from: <http://www.microsoft.com/technet/community/columns/cableguy/cg1105.mspx>.
[31] L. Brakmo, L. Peterson, TCP Vegas: End to end congestion avoidance on a global Internet, IEEE J. Sel. Areas Commun. 13 (October) (1995) 1465–1480.
[32] R. Wang, G. Pau, K. Yamada, M.Y. Sanadidi, M. Gerla, TCP startup performance in large bandwidth delay networks, in: Proceedings of the IEEE INFOCOM, Hong Kong, 2004.
[33] R. Wang, K. Yamada, M.Y. Sanadidi, M. Gerla, TCP with sender-side intelligence to handle dynamic, large, leaky pipes, IEEE J. Sel. Areas Commun. 23 (2005) 235–248.
[34] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, IEEE Network 17 (November/December) (2003) 27–35.
[35] C. Dovrolis, P. Ramanathan, D. Moore, Packet dispersion techniques and capacity estimation, IEEE/ACM Trans. Network. 12 (December) (2004) 963–977.
[36] K.S.B. Miller, K. Avrachenkov, G. Miller, Flow control as stochastic optimal control problem with incomplete information, in: Proceedings of the IEEE INFOCOM, Miami, USA, March, 2005.
[37] L.A. Grieco, S. Mascolo, Mathematical analysis of Westwood+ TCP congestion control, IEEE Proc. Control Theory Appl. 15 (Jan.) (2005) 35–42.
[38] TCP Westwood+ modules for ns2. Available from: <http://193.204.59.68/mascolo/tcp%20westwood/modules.htm>.
[39] BIC-TCP - a variant for high-speed long distance networks. Available from: <http://www4.ncsu.edu/rhee/export/bitcp/>.
[40] L.L. Peterson, B.S. Davie, Computer Networks, a Systems Approach, Morgan Kaufmann, 2000.
[41] Buffer tuning for all CISCO routers. Available from: <http://www.cisco.com/warp/public/63/buffertuning.html>.
[42] R. Jain, D. Chiu, W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, in: DEC Research Report TR-301, September, 1984.
[43] M.W.K. Munir, D. Damjanovic, Linux beats windows! – or the worrying evolution of TCP in common operating systems, in: Proceedings of the PFLDnet, Marina Del Rey (Los Angeles), CA, USA, February, 2007.

[44] IPERF – The TCP/UDP bandwidth measurement tool. Available from: <http://dast.nlanr.net/Projects/Iperf/>.

Dzmitry Kliazovich received his Masters degree from Belarusian State University of Informatics and Radioelectronics in 2002, and Ph.D. degree from University of Trento, Italy, in 2006 both in Telecommunication science. In 2005/2006 he was a visiting researcher at the Computer Science Department of the University of California at Los Angeles. He is an author of more than 20 research papers and active member of Technical Program Committee for ICC, GLOBECOM, and CAMAD conferences. His main research activities are in the field of networking with a focus on cross-layer design and next-generation networks.

Fabrizio Granelli was born in Genoa in 1972. He received the "Laurea" (M.Sc.) degree in Electronic Engineering from the University of Genoa, Italy, in 1997, with a thesis on video coding, awarded with the TELECOM Italy prize, and the Ph.D. in Telecommunications from the same university, in 2001. Since 2000 he is carrying on his teaching activity as Assistant Professor in Telecommunications at the Dept. of Information and Communication Technology - University of Trento (Italy). In August 2004, he was visiting professor at the State University of Campinas (Brasil). He is author or co-author of more than 60 papers published in international journals, books and conferences. Dr. Granelli is guest-editor of ACM Journal on Mobile Networks and Applications, special issue on "WLAN Optimization at the MAC and Network Levels" and Co-Chair of 10th IEEE Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks (CAMAD'04). Dr. Granelli is General Vice-Chair of the First International Conference on Wireless Internet (WICON'05) and General Chair of the 11th IEEE Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks (CAMAD'06). He is Associate Editor of IEEE Communications Letters and IEEE Communications Surveys and Tutorials. His main research activities are in the field of networking, with particular reference to network performance modeling, medium access control, wireless networks, next-generation IP, and video transmission over packet networks. He is Senior Member of IEEE and Vice-Chair of IEEE ComSoc Technical Committee on Communications Systems Integration and Modeling (CSIM).

Daniele Miorandi (daniele.miorandi@create-net.org) received the laurea degree (summa cum laude) in Telecommunication Engineering from the University of Padova, Italy, in 2001. He received the Ph.D. degree in Communication Engineering from the University of Padova, Italy, in 2005, with a thesis entitled "Stochastic Modelling of Wireless Ad Hoc Networks". In 2003/04 he spent one year of his doctoral thesis visiting the MAESTRO project at INRIA Sophia Antipolis (France). In 2004 he had an appointment as "Incaricato di Ricerca" at IEIIT-CNR, Torino (Italy). In 2005 he joined CREATE-NET, Trento (Italy), where he is currently Head of the Pervasive Area. He is the coordinator of the EC-funded BIONETS project, focusing on bio-inspired autonomic networks and services. His research interests include design and analysis of bio-inspired communication paradigms for pervasive computing environments, analysis of TCP performance over wireless/satellite networks, scaling laws for large-scale information systems, protocols and architectures for wireless mesh networks. Dr. Miorandi has authored more than 50 papers in international journals and conferences. He serves in the Steering Committee of WiOpt, Autonomics and ValueTools.