Computer Networks 56 (2012) 462–476

Contents lists available at SciVerse ScienceDirect

Computer Networks

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

Behavior analysis of TCP Linux variants

C. Callegari *, S. Giordano, M. Pagano, T. Pepe

Dept. of Information Engineering, University of Pisa, Via Caruso 16, 56122 Pisa, Italy

ARTICLE INFO

Article history:
Received 24 September 2010
Received in revised form 20 September 2011
Accepted 8 October 2011
Available online 19 October 2011

Keywords:
Linux TCP
Throughput
Fairness
Friendliness

ABSTRACT

The Transmission Control Protocol (TCP) is used by the vast majority of Internet applications. Since its introduction in the 1970s, a lot of variants have been proposed to cope with the different network conditions we can have (e.g., wired networks, wireless networks, satellite links) and nowadays Linux OS includes 13 different TCP variants.

The aim of this paper is to offer a comparative analysis of the behavior of the different TCP Linux variants, in terms of throughput, fairness, and friendliness.

1. Introduction

The Transmission Control Protocol (TCP) [1,2] is used by the vast majority of Internet applications. The TCP is a connection-oriented transport protocol that provides a reliable byte-stream data transfer service between pairs of processes.

When two processes want to communicate, they must first establish a TCP connection (initialize the status information on each side). Since connections must be established between unreliable hosts and over the unreliable Internet communication system, a handshake mechanism with clock-based sequence numbers is used.

The TCP provides **multiplexing** facilities by using source and destination port numbers. These port numbers allow the TCP to set up virtual connections over a physical connection and multiplex the data streams through that connection.

Each connection is uniquely specified by a pair of sockets[1] identifying its two sides and by specific parameters such as sequence numbers, window sizes, and status information (necessary for reliability and flow and congestion control mechanisms – see the following for more details).

At the end of a communication, the connection is terminated or closed to free the resources for other uses.

The TCP is able to transfer a continuous stream of bytes, in each direction, by packaging some number of bytes into *segments* (TCP data unit). The size of these segments and the timing at which they are sent is generally left to the TCP module. However, an application can force delivery of segments to the output stream using a push operation provided by the TCP to the application layer. A push causes the TCP to promptly forward and deliver data to the receiver.

Apart from this *basic data transfer*, the TCP provides some more services. First of it is able to recover from data that are damaged, lost, duplicated, or delivered out of order by the Internet communication system (**reliability**).

Missing or corrupted segments are detected and retransmitted assigning a sequence number to each transmitted segment, and requiring a positive acknowledgment (ACK) from the receiver. If the ACK is not received within a timeout interval, the data are retransmitted.

At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each transmitted segment, checking it at the receiver, and discarding damaged segments.

* Corresponding author.
E-mail addresses: christian.callegari@iet.unipi.it (C. Callegari), stefano.giordano@iet.unipi.it (S. Giordano), michele.pagano@iet.unipi.it (M. Pagano), teresa.pepe@iet.unipi.it (T. Pepe).

[1] A socket is given by the concatenation of the IP addresses and port numbers.

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.10.002

The TCP also provides a means for the receiver to govern the amount of data sent by the sender (**flow control**). This is achieved by returning a "window", named advertised window, that indicates the allowed number of bytes that the sender may transmit before receiving further permission.

Moreover, TCP users may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not needed.

Finally, one of the most important features provided by the TCP is the **congestion control** that will be quickly described in the following sections.

The aim of this paper is to offer a comparative analysis of the behavior of the different TCP Linux variants, in terms of throughput, fairness, and friendliness. Indeed, at the best of our knowledge, such comparison is not reported anywhere in the literature.

The rest of the paper is organized as follows: in the next Section we provide a brief description of the related works, while in Section 3 we present a quick overview of the different TCP variants implemented in the Linux kernel 2.6.x. Then in Section 4 we present the simulation setups, while the performance evaluation of the TCP versions, in terms of throughput, fairness, and friendliness, in heterogeneous scenarios, will be presented in Section 5. Finally, Section 6 provides a discussion of the obtained results and Section 7 concludes the paper with some final remarks.

2. Related works

In this section we briefly introduce the main studies on performance evaluation of Linux TCP variants.

A number of recent works have studied TCP performance in terms of friendliness and fairness. As an example, a survey on TCP-friendly congestion control schemes is presented in [3]. In [4] the authors focus on fairness and stability of the congestion control mechanisms adopted in several versions of TCP (i.e., TCP Tahoe, TCP Reno, and TCP Vegas) through an analytic approach. Moreover, in [5,6], an evaluation of the performance of different TCP proposals for high-speed networks is presented in a fair and consistent manner.

Specifically related to Linux TCP, the paper [7] discusses how "standard" Linux TCP conforms to the IETF specifications related to TCP congestion control and it illustrates the performance effects of selected Linux-specific design solutions.

Nevertheless, to the best of our knowledge, there is no work that offers a comparative analysis of the performance of all the Linux TCP versions in terms of achieved throughput, as well as friendliness and fairness, in different scenarios.

3. TCP Linux variants

Since flow control does not take into account network status, it is not sufficient to avoid network congestions. To make the TCP sensitive to the network conditions, a mechanism, called TCP congestion control, has to be used.

The basic idea is that the rate of ACKs returned by the receiver determines the rate at which the sender can transmit data. The TCP uses several algorithms for congestion control, each of them controls the sending rate by manipulating a congestion window (*cwnd*) that limits the number of outstanding unacknowledged bytes that are allowed at any time. The *cwnd* size is increased or decreased depending on congestion level. The sender can transmit up to the transmission window, which is the minimum of the *cwnd* (flow control imposed by the sender) and the advertised window (flow control imposed by the receiver).

All the modern implementations of the TCP contain at least four intertwined algorithms: Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery [8].

The TCP version based on these "original" algorithm is the TCP Reno, also referred to as "standard" TCP.

Fig. 1 shows the typical *cwnd* behavior at the beginning of a connection. It is easy to recognize the exponential growth in the initial phase as well as the linear increase of the congestion avoidance phase.

Starting from the "standard" TCP, several variants have been proposed for responding to particular demands due to the development of new network scenarios.

The main problem with standard TCP implementations is that they basically relay on packet loss as an indicator of network congestion. Typically, in a wired network, this is true; on a wireless link, instead, noisy and fading radio channels are the more frequent causes of loss. Moreover, as link capacity grows and new Internet applications with high-bandwidth demand emerge, TCP performance is unsatisfactory, especially on high-speed and long distance networks, given the conservative behavior of TCP in adjusting its *cwnd*.

Thus, a number of TCP variants have been proposed to enhance the TCP performance in both wireless and long fat networks.

Table 1 reports a classification of all the TCP Linux variants, highlighting their main features, such as:

- target network: wired, wireless, or long fat (a network with a large bandwidth-delay product);

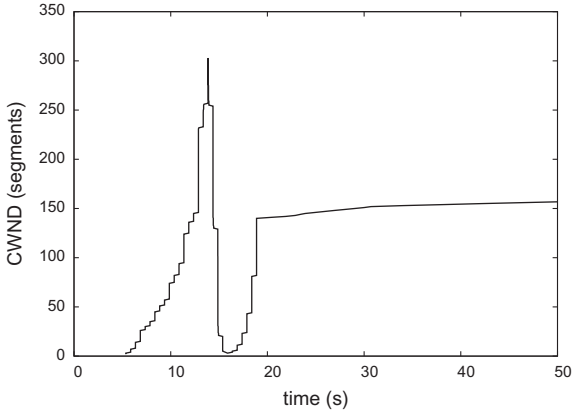


Fig. 1. *Cwnd* behavior at the beginning of a connection (TCP Reno).

Table 1
Linux TCP variants.

TCP variant	Network	Class	Main features
Reno [13]	Standard	Loss-based	• Standard TCP
Vegas [14]	Standard	Delay-based	• Proactive scheme • RTT and rate estimation
Veno [15]	Wireless	Delay-based	• Combine Reno and Vegas • Deal with random loss
Westwood [16]	Wireless	Delay-based	• Deal with "Large" dynamic channels
BIC [17]	Long fat	Loss-based	• Modification of congestion avoidance scheme
CUBIC [18]	Long fat	Loss-based	• Improved variant of BIC
HSTCP [19]	Long fat	Loss-based	• Modification of congestion avoidance scheme
Hybla [20]	Long fat	Delay-based	• Modification of congestion avoidance scheme
Scalable [21]	Long fat	Loss-based	• Modification of congestion avoidance scheme
Illinois [22]	Long fat	Loss-Delay-based	• Modification of congestion avoidance scheme
YeAH [23]	Long fat	Delay-based	• Two working modes for the congestion avoidance phase: fast and slow
HTCP [24]	Long fat	Delay-based	• Modification of congestion avoidance scheme
LP [25]	–	Delay-based	• Variant for low priority flows

- TCP class: loss based (*cwnd* is adjusted on the basis of information on the lost packets) or delay based (*cwnd* is adjusted on the basis of information on the experienced delay);
- other features typical of the single variant.

4. Simulation setup

In this section, we outline the experimental test-beds we have used to evaluate the performance offered by the different TCP variants implemented in the Linux kernel 2.6.x.

To perform our tests we have used several tools, all freely available for Linux OS. In more detail we have used:

- **Iperf:** it is a standard tool [26], commonly used for network performance measurements, and allows the creation of TCP data streams between two machines.
- **TCP probe:** it is a Linux kernel module that records the state of a TCP connection in response to each incoming packet.
- **NetEm:** it is a tool [27] that permits to emulate a generic network environment on a point-to-point connection. We have used such tool to study the protocol performance in non-optimum condition (RTT and loss probability). In this way, we have been able to emulate long distance networks in a lab environment.

In the simulations, the value of *MSS* (Maximum Segment Size) is fixed and assumes the following value: $MSS = MTU - TCP\&IP\ header$, where *MTU* is the Maximum Transmission Unit and is set to 1500 byte. Regarding the different TCP options, we have used the standard TCP Linux configuration, that is SACK (Selective ACK) [9], FACK (Forward ACK) [10], and DSACK (Delayed Selective ACK) [11] are active for all the TCP variants, while ECN (Explicit Congestion Notification) [12] is deactivated.

The different TCP variants have been evaluated in terms of achieved throughput, fairness, and friendliness. Regarding the fairness we have used the Jain's index *J* [28], defined as:

$$J = \frac{(\sum x_i)^2}{(n \cdot \sum x_i^2)}, \quad (1)$$

where n is the number of simultaneous connections (in our case $n = 2$) and x_i is the instantaneous throughput of connection i.

To be noted that the index assumes values in the range $[0;1]$ with 1 corresponding to completely fair protocols and 0 to completely unfair protocols. Finally it is important to highlight that the considered TCP variants have been tested in several network scenarios, so as to consider all the environments for which the different versions have been designed.

In the following subsection we detail the different environments.

4.1. Wired scenario

To realize the tests in the wired scenario, we have used a test-bed composed of four general purpose PCs, equipped with a 2.53 Ghz Intel CPU and 4 GB of RAM. The PCs have been interconnected with a 1 Gbps Ethernet network (as depicted in Fig. 2).

The simulations have been conducted varying the RTT, namely varying the queuing delay, d_q, as well as the loss probability, P_l. In more detail, we have considered the following settings:

- scenario 1: the P_l is fixed to 0.0001%, while d_q is variable and can assume the values $\{50, 75, 100\}$ ms, which correspond to the following values of RTT $\{100, 150, 200\}$ ms;
- scenario 2: the RTT is fixed to 100 ms, while P_l can assume the values $\{0.00005, 0.0001, 0.0005\}$%.

Moreover, we have also taken into account both long (5 min) and short (20 s) TCP connections. Regarding the bandwidth we have considered three distinct cases:

- a gigabit network;
- a gigabit network with a bottleneck link between host B and C (100 Mbps);
- a "variable bandwidth" network, to evaluate how the different variants react to bandwidth changes.

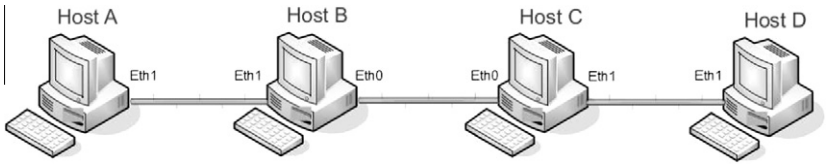


Fig. 2. Network topology (wired scenario).

4.2. Wireless scenario

Regarding the wireless scenario, we have used a standard 802.11g network of our department, so as to have a certain number of active nodes in the network. The mobile station has been put at a distance from the Access Point such that the received power was −76 dBm (non optimal conditions).

4.3. Satellite scenario

Fig. 3 shows the experimental test-bed used for the tests related to the satellite scenario. Two different LANs, located at the University of Pisa and University of Florence, are connected through the satellite network owned by the Italian National Consortium for Telecommunications (CNIT).

Note that, in this scenario, the maximum speed is 2048 kbps. It is relevant to highlight that the satellite network is based on a Hub-and-Spoke topology, so that communications between two terminals always get through to a hub located at the SkyPark (implying two round trips to the satellite).

5. Performance analysis

In the following subsections, we provide the results of the different test sessions. In the first ones we have evaluated the throughput achieved by each TCP variant when only one flow is active in the network, considering the following scenarios:

- **Scenarios 1, 2, 3, and 4:** different configurations of wired network.
- **Scenario 5:** wireless network.
- **Scenarios 6 and 7:** different configurations of satellite network.

Then, in the subsequent subsection we have evaluated the throughput when two flows, both using the same TCP variant, are active, while in the last subsection we have considered flows using different TCP variants. In all the cases we have taken into account short as well as long TCP connections.

5.1. Achieved throughput

5.1.1. Wired network

In this subsection we present the throughput achieved by each TCP variant when only one TCP flow (between host B and host C in Fig. 2) is active. For each variant we have evaluated the throughput under several conditions:

- in a gigabit network, as a function of RTT and P_l both considering long (scenario 1) and short-lived TCP connections (scenario 2)
- in a 100 Mbps network, with fixed values for RTT and P_l (scenario 3), analogous to consider a bottleneck link in our study
- in a "variable bandwidth" network, where a sudden change in the available bandwidth is present (scenario 4)

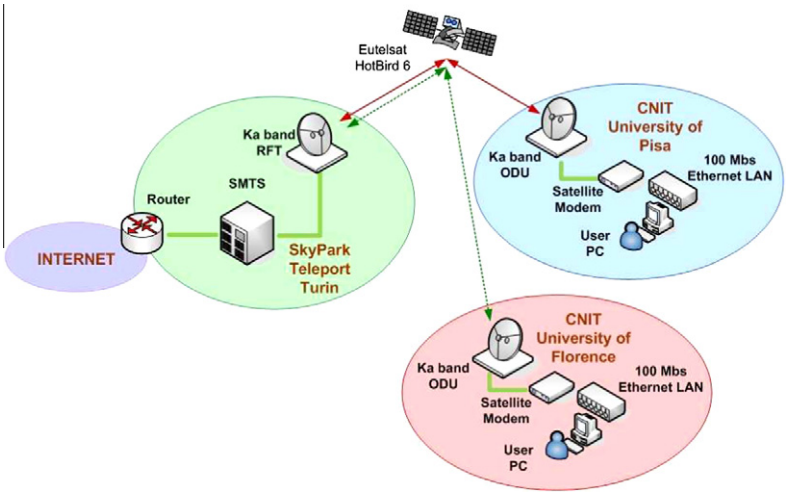


Fig. 3. Network topology (satellite scenario).

For the first scenario, we present some "sample path" graphs (obtaining varying RTT and P_l), which allow the reader to intuitively get the general behavior of the different TCP variants, while all the other cases are summarized in the tables at the end of the section.

5.1.1.1. TCP Reno. Fig. 4(a) shows that when the RTT increases the leading edge of the *cwnd* decreases. This is due to the fact that the time necessary for updating the *cwnd* increases with the RTT. As a consequence, the average throughput decreases.

In the second case (Fig. 4(b)) we observe that the throughput does not significantly vary when P_l changes from 0.00005% to 0.0001%, while it gets much worst when P_l increases to 0.0005%. Indeed, we can see that in this last case the average value is much lower that in the other cases.

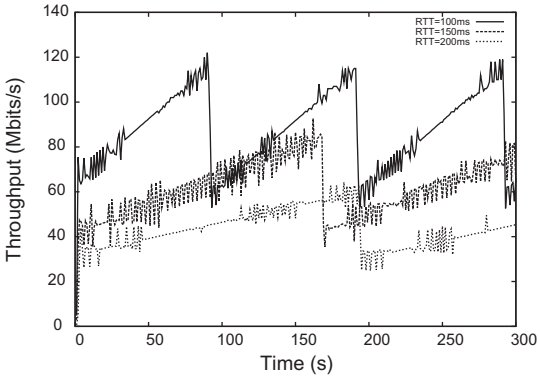
5.1.1.2. TCP Vegas. As shows in the following figures, the considerations done for TCP Reno still hold in the TCP Vegas case. This is justified by the fact that TCP Vegas should behaves differently in strong congestion situations that are not present in the considered scenarios.

Indeed, we can observe almost the same behavior of the throughput (just a bit higher than in TCP Reno) both when considering it as a function of the RTT (see Fig. 5(a)) or of P_l (see Fig. 5(b)).

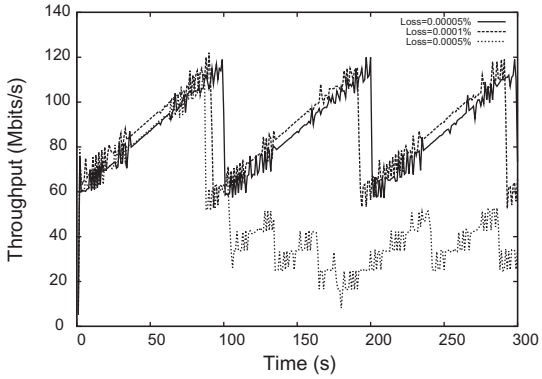
5.1.1.3. TCP Veno. As shown in the following figures, TCP Veno behaves almost equally to TCP Reno and TCP Vegas. This is reasonable since it can be seen as a "fusion" of the TCP two algorithms.

Fig. 6(a) and (b) respectively show the behavior of the throughput when varying the RTT and P_l.

5.1.1.4. TCP Westwood. The variations introduced in TCP Westwood confer better stability to the algorithm. Indeed, as shown in Fig. 7(a) and (b) the throughput presents less oscillations than in the previous cases.

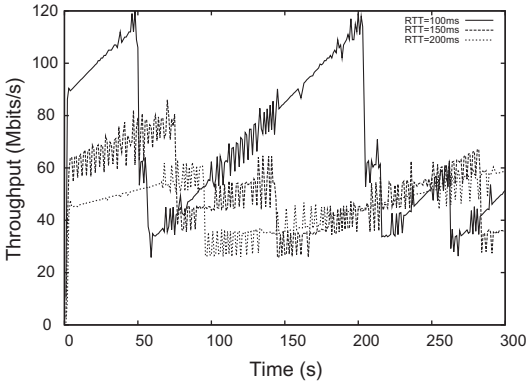


(a) Throughput behavior varying RTT

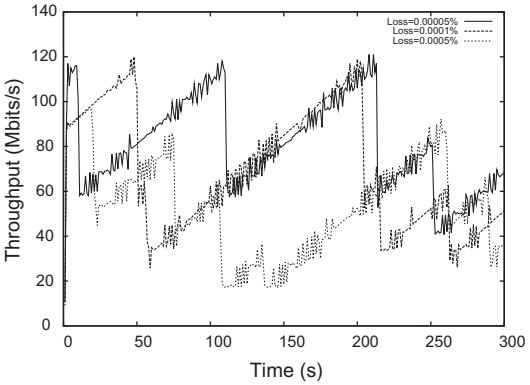


(b) Throughput behavior varying loss probability

Fig. 4. TCP Reno.

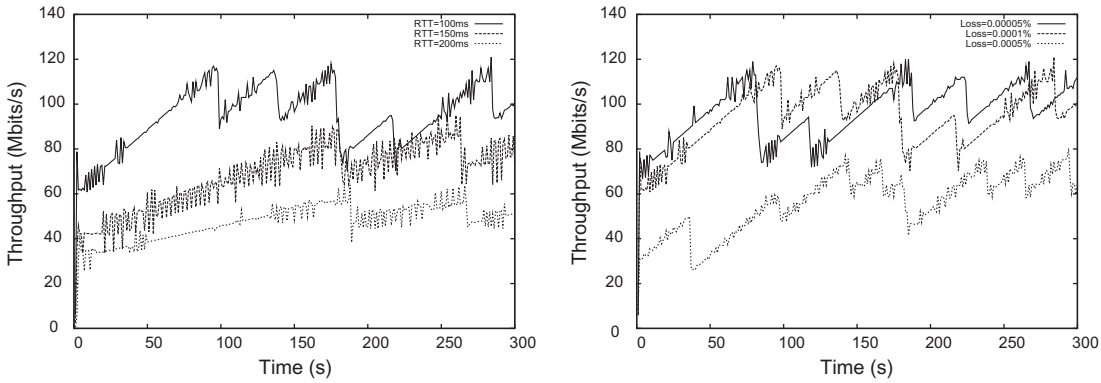


(a) Throughput behavior varying RTT



(b) Throughput behavior varying loss probability

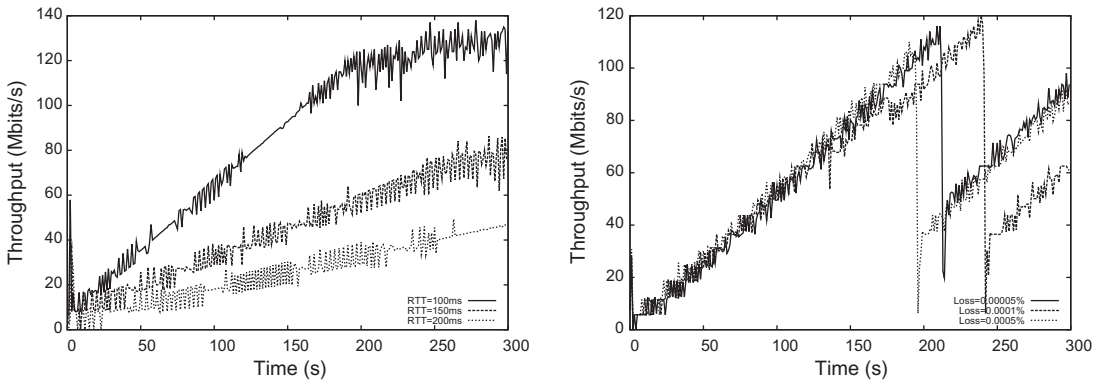
Fig. 5. TCP Vegas.



(a) Throughput behavior varying RTT

(b) Throughput behavior varying loss probability

Fig. 6. TCP Veno.



(a) Throughput behavior varying RTT

(b) Throughput behavior varying loss probability

Fig. 7. TCP Westwood.

This is mainly due to the mechanisms introduced in the protocol, so as to deal with large and dynamic channels.

In more detail Fig. 7(a) shows that the throughput strongly depends on the RTT values, while Fig. 7(b) shows that it is not much dependent on P_l.

5.1.1.5. TCP BIC. As an effect of the more "aggressive" window increase function introduced in TCP BIC, the throughput behavior is completely different from the previous cases.

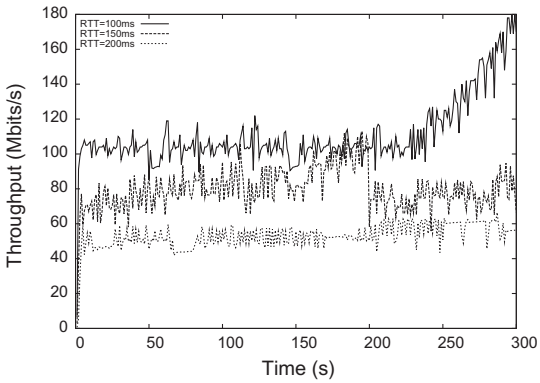
Fig. 8(a) and (b) show that, in general, TCP BIC is able to achieve a better throughput than the other TCP variants. Also in this case, the dependence on the RTT (see Fig. 8(a)) is more important than the dependence on P_l (see Fig. 8(b)).

5.1.1.6. TCP CUBIC. Coherently with the fact that TCP CUBIC is based on a "simplification" of the window increase function implemented in TCP BIC, we can observe that the throughput behavior when varying the RTT (see Fig. 9(a)) and P_l (see Fig. 9(b)) is much similar to the TCP BIC case.

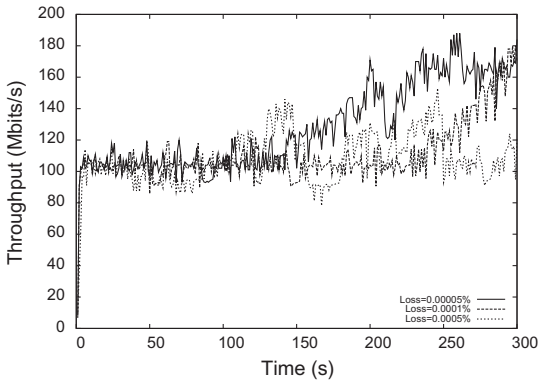
5.1.1.7. HSTCP. Since HSTCP has been implemented to overcome the limitations of the standard TCP in high-speed networks (by proposing a different congestion avoidance mechanism), we can see that the throughput behavior in the considered scenarios is completely different to the previous cases.

From Fig. 10(a) we can see that the throughput depends on the RTT, while from Fig. 10(b) we can infer that it is almost independent from P_l, as the average throughput value is almost the same when varying P_l.

5.1.1.8. TCP Hybla. Similarly to HSTCP, TCP Hybla has been introduced for high-speed networks. This justifies the fact that the behavior of the throughput is much similar to the one achieved by HSTCP.

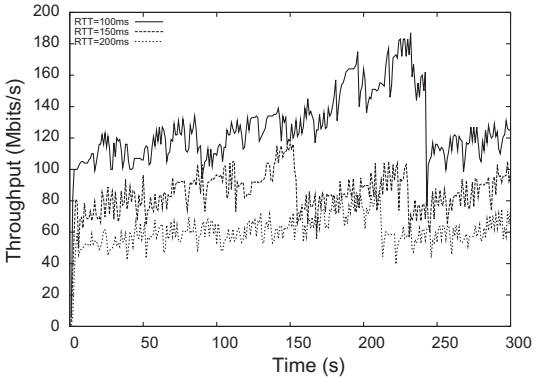


(a) Throughput behavior varying RTT

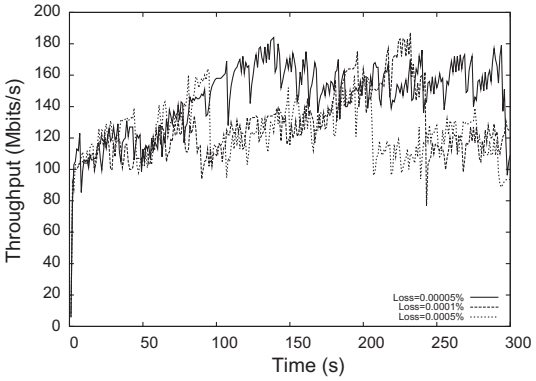


(b) Throughput behavior varying loss probability

Fig. 8. TCP BIC.

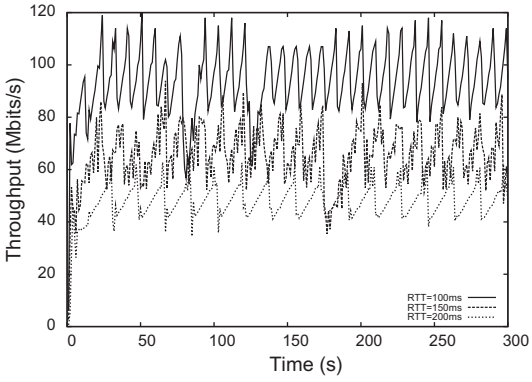


(a) Throughput behavior varying RTT

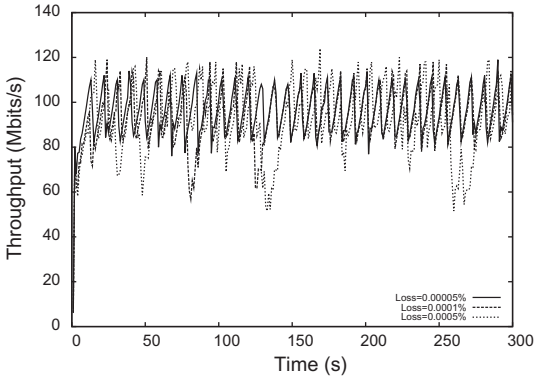


(b) Throughput behavior varying loss probability

Fig. 9. TCP CUBIC.



(a) Throughput behavior varying RTT



(b) Throughput behavior varying loss probability

Fig. 10. HSTCP.

As in the HSTCP case, the dependence on the RTT (see Fig. 11(a)) is much more important than the dependence on P_l (see Fig. 11(b)).

5.1.1.9. Scalable TCP. As the two previous cases, HSTCP and TCP Hybla, also Scalable TCP has been realized to overcome the limitations of the standard TCP in high-speed networks. Nevertheless the behavior of the throughput is quite different from the previous cases.

Fig. 12(a) and (b) respectively show that the throughput is strongly dependent on the value of both the RTT and P_l, while for the other high-speed TCPs the average throughput is quite independent of P_l.

5.1.1.10. TCP Illinois. The throughput behavior in TCP Illinois is almost the same of that shown by Scalable TCP.

From Fig. 13(a) and (b) we can see that in general the average throughput achieved by TCP Illinois is slightly higher than the one achieved by Scalable TCP.

5.1.1.11. TCP YeAH. The considerations done for Scalable TCP and TCP Illinois still hold for TCP YeAH, which is another high-speed TCP.

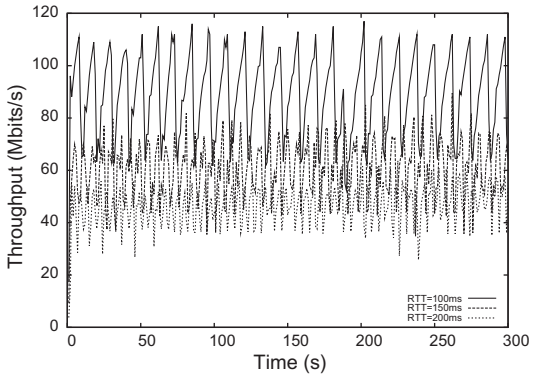
From Fig. 14(a) and (b) we can see that in general the average throughput achieved by TCP YeAH is slightly higher than the one achieved by Scalable TCP and TCP Illinois.

5.1.1.12. HTCP. The behavior of the throughput is quite similar to that achieved by TCP Hybla. Also in this case the throughput is very "irregular" (see Fig. 15(a) and (b)).

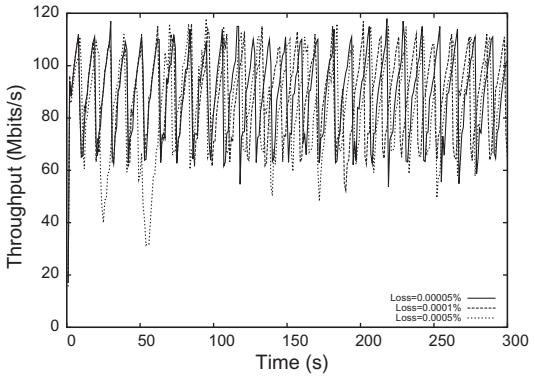
From the graphs we can see that the throughput strongly depends both on RTT and P_l.

5.1.1.13. TCP-LP. The throughput behavior shown by TCP-LP coherently reflects the fact that TCP-LP has been thought to infer congestion earlier than the standard TCP.

Indeed, from Fig. 16(a) and (b) we can clearly see that the achieved throughput is much lower than in other cases, for all the considered values of RTT and P_l.

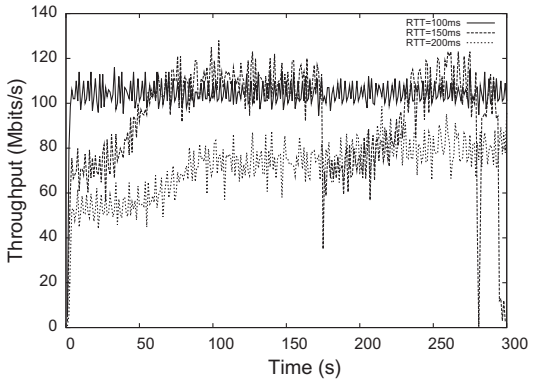


(a) Throughput behavior varying RTT

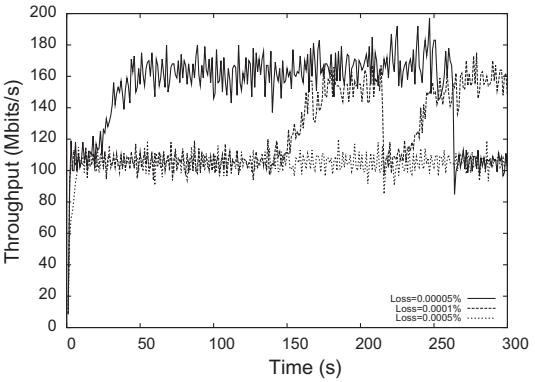


(b) Throughput behavior varying loss probability

Fig. 11. TCP Hybla.

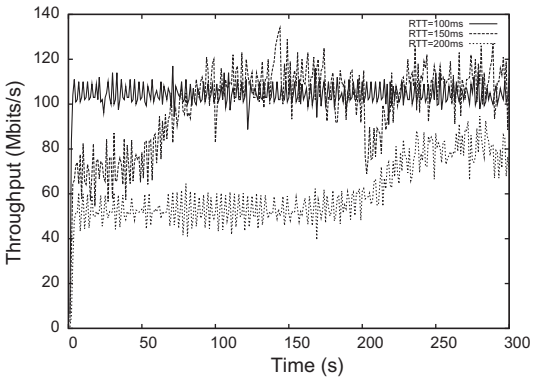


(a) Throughput behavior varying RTT

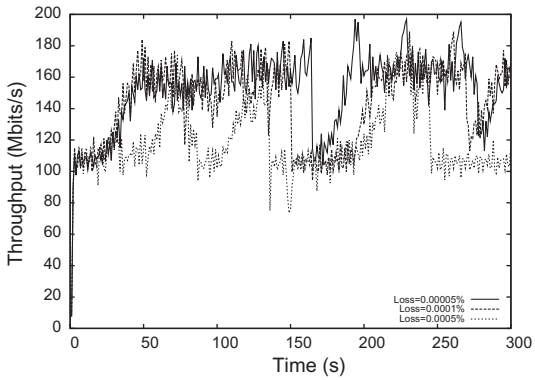


(b) Throughput behavior varying loss probability

Fig. 12. Scalable TCP.

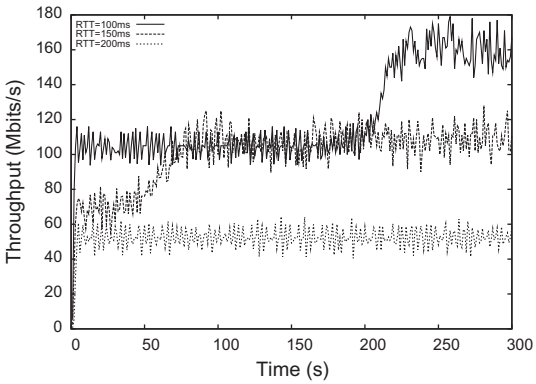


(a) Throughput behavior varying RTT

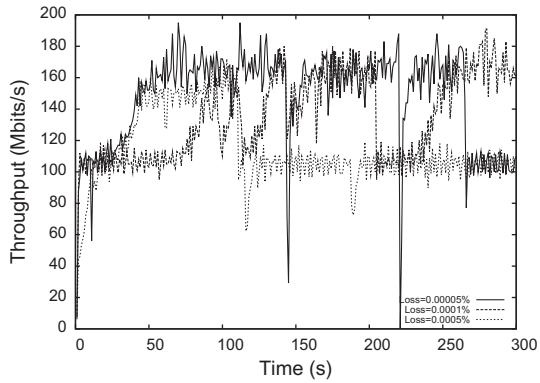


(b) Throughput behavior varying loss probability

Fig. 13. TCP Illinois.

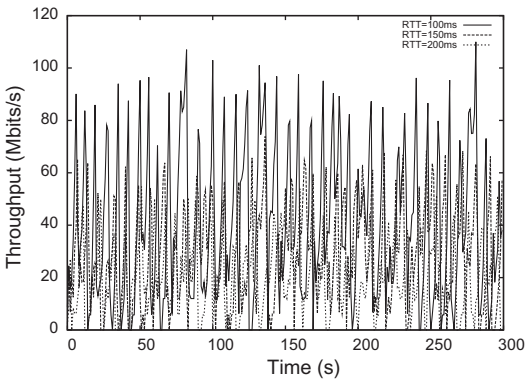


(a) Throughput behavior varying RTT

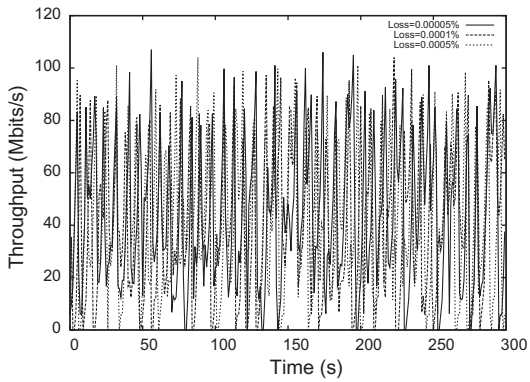


(b) Throughput behavior varying loss probability

Fig. 14. TCP YeAH.

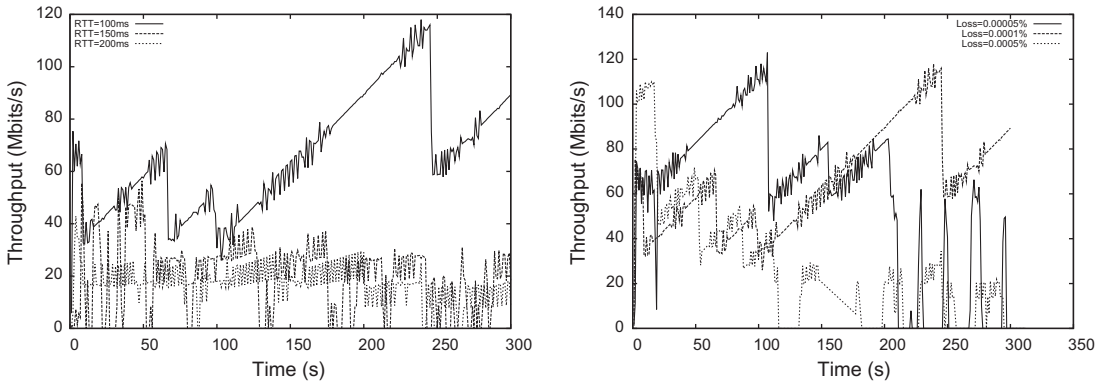


(a) Throughput behavior varying RTT



(b) Throughput behavior varying loss probability

Fig. 15. HTCP.



(a) Throughput behavior varying RTT

(b) Throughput behavior varying loss probability

Fig. 16. TCP-LP.

5.1.1.14. Global results. In Table 2 we present the average throughput achieved by the different TCP variants in the first presented scenario, considering 5 min long TCP connections, both varying the RTT and P_l. Note that the values in bold in the tables represent the best achieved results. As we can easily see from the results (that represent the mean value computed over ten independent simulations), the best performance is almost always achieved by TCP Hybla, together with some other high-speed versions (e.g., HSTCP).

Also it is important to highlight that the performance offered by the different variants do not vary that much, with the exception of TCP BIC, TCP CUBIC, TCP Scalable, and HTCP that offer a lower throughput in this scenario.

Different considerations have to be done for TCP-LP that achieves a very low throughput, obtaining the worst performance. Nevertheless, this is reasonable, given the nature of this TCP variant.

Given the fact that most of the connections in the Internet only last a very short time, in Table 3 we present the same analysis seen in the previous Table 2, taken into consideration short connections (20 s).

The considerations done for scenario 1 still hold, with the best performance always achieved by TCP Hybla that in this case behaves strongly better than all the other variants. In general we can also notice that all the high-speed variants behave better than the standard and wireless versions of the TCP. Differently from the previous case, TCP-LP is able, in this scenario, to achieve a throughput comparable to those achieved by the other TCP variants.

In Table 4 we consider a gigabit network with a 100 Mbps bottleneck link. In this case, we have only performed the analysis for a given value of RTT and P_l (i.e., RTT = 100 ms and P_l = 0.0001%). This choice is justified by the fact that from the previous tables we can see that the "relative" behavior of the different TCP variants does not depend on the values of these quantities.

In this case we can easily realize that the best performance is offered by TCP Veno and Westwood, together with TCP BIC, CUBIC, and HTCP. Regarding the other TCP variants, we can notice that they are all comparable.

The last scenario for the wired topology is represented by a "variable bandwidth" network, where the bandwidth of one link suddenly changes from 1 Gbps to 500 Mbps and then back from 500 Mbps to 1 Gbps. These tests have been carried out to evaluate how the different TCP variants react to sudden changes of the available bandwidth.

Table 2
Gigabit network – 5 min.

Wired scenario 1	Throughput (Mbps)				
Variants	RTT = 100 ms P_l = 0.0001	RTT = 100 ms P_l = 0.0005	RTT = 100 ms P_l = 0.00005	RTT = 150 ms P_l = 0.0001	RTT = 200 ms P_l = 0.0001
Reno	51.00	57.87	60.65	35.66	29.45
Vegas	57.62	59.02	62.54	39.89	27.94
Veno	55.32	60.34	59.60	39.93	26.82
Westwood	57.50	60.23	58.05	34.88	21.32
BIC	42.65	43.80	44.69	32.06	24.03
CUBIC	49.39	46.91	49.82	32.53	24.98
HSTCP	60.35	53.17	61.75	38.59	30.05
Hybla	**66.38**	**65.52**	**68.12**	**43.38**	30.65
Scalable	45.15	45.05	46.28	32.34	25.06
Illinois	60.05	59.39	59.88	40.91	**31.79**
YeAH	47.77	49.52	47.21	33.45	26.34
HTCP	42.27	42.55	43.99	28.77	21.27
LP	6.73	20.40	14.42	12.29	3.55

Table 3
Gigabit network – 20 s.

Wired scenario 2	Throughput (Mbps)				
Variants	RTT = 100 ms P_l = 0.0001	RTT = 100 ms P_l = 0.0005	RTT = 100 ms P_l = 0.00005	RTT = 150 ms P_l = 0.0001	RTT = 200 ms P_l = 0.0001
Reno	19.95	15.39	19.52	12.65	2.95
Vegas	17.75	19.24	18.64	13.89	3.34
Veno	17.13	21.09	12.72	13.38	7.53
Westwood	19.3	20.47	9.82	8.10	5.25
BIC	46.54	39.25	44.59	22.83	17.45
CUBIC	38.62	38.72	40.59	26.33	20.11
HSTCP	35.96	38.34	37.06	17.07	12.34
Hybla	**60.41**	**59.89**	**62.94**	38.75	**29.37**
Scalable	35.98	38.68	39.54	28.81	15.87
Illinois	52.47	50.17	54.42	**39.62**	26.69
YeAH	45.26	44.61	39.34	24.95	18.52
HTCP	42.61	36.23	36.55	26.81	19.95
LP	19.91	17.02	15.85	11.97	5.85

Table 4
100 Mbps network – 5 min.

Wired scenario 3 Variants	Throughput (Mbps) RTT = 100 ms P_l = 0.0001
Reno	0.193
Vegas	0.193
Veno	**0.237**
Westwood	0.230
BIC	0.236
CUBIC	**0.237**
HSTCP	0.173
Hybla	0.151
Scalable	0.180
Illinois	0.174
YeAH	0.180
HTCP	0.231
LP	0.193

Table 5
"Variable bandwidth" network.

Wired scenario 4	Throughput (Mbps) RTT = 100 ms P_l = 0.0001			
Variants	Interval 1	Interval 2	Interval 3	Mean value
Reno	52.93	1.75	13.03	21.34
Vegas	31.39	1.73	16.00	15.58
Veno	27.71	2.05	12.91	13.42
Westwood	22.25	**4.87**	16.31	13.97
BIC	46.31	1.50	**52.45**	33.27
CUBIC	49.07	2.08	62.59	**37.72**
HSTCP	63.32	1.83	48.18	35.79
Hybla	55.8	1.38	43.37	32.88
Scalable	52.98	1.34	5.64	18.45
Illinois	57.98	1.15	38.82	30.51
YeAH	56.01	1.15	38.82	30.28
HTCP	56.75	1.46	51.52	36.12
LP	14.71	1.72	12.15	9.35

In Table 5 we present the results of such tests. In more detail Interval 1 refers to the 60 s prior to the first bandwidth change and thus the available bandwidth is 1 Gbps, Interval 2 is represented by the 60 s with available bandwidth equal to 500 Mbps, and finally Interval 3 are the 60 s after the second bandwidth change, when the bandwidth is 1 Gbps.

From the results we can deduct that the variants, which react the best to the bandwidth changes, better exploiting the available bandwidth, are TCP Westwood, TCP Veno, and TCP CUBIC, while the ones that react the worst are TCP Illinois and TCP YeAH.

Nevertheless, considering the whole simulations, the best performance is achieved by TCP CUBIC, HSTCP, and HTCP, while the worst TCP variant is TCP-LP.

5.1.2. Wireless network

In this subsection we present the results, in terms of achieved throughput, obtained by the different TCP variants in a wireless scenario.

From Table 6 we can see that in this case all the TCP variants offer similar performance. Moreover we can notice that the TCP variants designed for wireless network do not outperform the other TCP versions. Indeed, in this case the best performance is offered by TCP Reno, TCP BIC, and TCP-LP, while the worst one is offered by TCP Scalable and TCP Illinois.

5.1.3. Satellite network

Regarding the satellite network, we have carried out two distinct sets of simulation:

- scenario 6: simplified version of the full topology, where only one end host is connected to the satellite network, while the other one is directly connected to the Internet. By means of the trace-route application we have seen that the route between the two end hosts is composed of 14 hops, while using the ping application we have estimated an average RTT of 626 ms.
- scenario 7: full topology, with both the end hosts connected to the satellite. In this case, the route between the two end hosts is composed of 2 satellite hops only and the average RTT is 1303 ms.

In Table 7 we present the average throughput computed over 15 min long connections, for the first and second simulation sets, respectively.

Table 6
Wireless network.

Wireless scenario Variants	Throughput (Mbps)
Reno	3.15
Vegas	2.47
Veno	2.93
Westwood	2.92
BIC	**3.17**
CUBIC	2.94
HSTCP	2.89
Hybla	3.03
Scalable	1.71
Illinois	1.41
YeAH	2.08
HTCP	2.43
LP	3.14

From the results it appears clearly that, regarding the first topology, all the variants almost achieve the same performance. Indeed, from our tests the satellite versions do not take to any significant improvement with respect to the more "standard" variants, while in the second case we can observe that there are big fluctuations in the average throughput.

Moreover in this second case we can observe that the worst performance is achieved by the TCP-LP, followed by the "standard" variants (i.e., Reno and Vegas).

On the other hand the TCP variants for high-speed networks achieve very good performance, with Scalable TCP reaching a mean throughput of about 1.5 Mbps.

5.2. Fairness

In this subsection we present an analysis of the fairness offered by the different variants of the TCP.

Fairness refers to the ability of the TCP of sharing the link capacity among the different sessions active on the link. It is assumed that all the connections are using the same version of the TCP. The maximum level of fairness is reached in the case when the N TCP sessions sharing the same bottleneck link, get $1/N$ of the link capacity each.

In these simulations, as well as in those related to the friendliness evaluation, we have only considered two competing connections, without adding any emulated delay and/or packet loss on the network.

To be noted that in this subsection (and in the subsequent one) "background" refers to a TCP connection between host B and C (already established when the other connection starts), while "main" refers to the connection between host A and D (in the scenario of Fig. 2), realized using the TCP variant we are evaluating.

To provide the reader with a quantitative analysis of the fairness, we have also evaluated the Jain's index (reported in the tables). In Table 8 we presents the results achieved over ten independent simulations, reporting for each TCP variant the mean throughput achieved by the two connections (taking into account 5 min long connections) and the Jain's index. In this scenario, we can consider a protocol to be reasonably fair when its Jain's index is equal to or greater than 0.95.

Hence, we can easily conclude that the most fair protocols are TCP Reno and TCP CUBIC that, with an index of 0.99, almost achieve the "ideal" behavior in terms of fairness. On the contrary TCP Westwood and TCP Illinois demonstrate to be the most unfair TCP variants with a Jain's index of 0.74 and 0.83, respectively.

Taking into account the short-lived nature of most of the TCP connections, we have performed another simulation set to evaluate the fairness of the TCP variants when considering 20 s long connections.

The results are reported in Table 9. We can easily deduce that short connections take to a more unfair behavior (i.e., the Jain's index assumes lower values). In this case, to discriminate between fair and unfair variants, we can set a "threshold" equal to 0.84, given which we can see that only two protocols change their "behavior", namely TCP Vegas which becomes fair and TCP CUBIC that becomes unfair.

5.3. Friendliness

In this subsection we present an analysis of the friendliness offered by the different variants of the TCP. As for the fairness, also the **friendliness** refers to the ability of the TCP of sharing the link capacity among the different ses-

Table 7
Satellite network.

Satellite scenario	Throughput (Mbps)	
Variants	Scenario 6	Scenario 7
Reno	0.394	0.816
Vegas	0.414	0.611
Veno	0.444	0.649
Westwood	**0.473**	1.057
BIC	0.450	1.145
CUBIC	0.422	1.475
HSTCP	0.305	1.120
Hybla	0.340	0.966
Scalable	0.442	**1.500**
Illinois	0.357	1.010
YeAH	0.425	1.260
HTCP	0.390	1.270
LP	0.386	0.130

Table 8
Fairness – 5 min.

Fairness	Throughput (Mbps)		
Variants	Connection 1	Connection 2	Jain's index
Reno	234.63	211.2	**0.99**
Vegas	303.44	165.40	0.92
Veno	289.12	181.00	0.95
Westwood	473.18	113.50	0.74
BIC	307.05	168.60	0.92
CUBIC	213.72	220.20	**0.99**
HSTCP	282.40	183.70	0.96
Hybla	286.49	181.90	0.95
Scalable	301.44	173.00	0.93
Illinois	373.12	139.70	0.83
YeAH	285.12	182.50	0.95
HTCP	290.18	179.70	0.95
LP	286.82	181.70	0.95

Table 9
Fairness – 20 s.

Fairness	Throughput (Mbps)		
Variants	Connection 1	Connection 2	Jain's index
Reno	330.83	160.00	0.89
Vegas	305.24	164.70	**0.92**
Veno	361.96	145.00	0.84
Westwood	462.84	98.64	0.70
BIC	452.34	100.11	0.72
CUBIC	425.91	112.90	0.75
HSTCP	318.98	165.80	0.91
Hybla	312.02	169.50	**0.92**
Scalable	377.78	134.88	0.82
Illinois	373.09	138.80	0.83
YeAH	327.39	161.60	0.90
HTCP	308.08	170.80	**0.92**
LP	364.06	142.9	0.84

sions active on the link. But in this case the connections are using different versions of the TCP.

In this analysis (simulations have been carried out as for the fairness evaluation) we have taken into account the friendliness of the different TCP variants with respect to TCP Reno, given that being the standard TCP, it is quite common that the TCP sessions will have to compete with TCP Reno for sharing the bandwidth.

As in the previous cases, in Tables 10 and 11 we report the results achieved over ten independent simulations for each case.The throughput in the first column is referred to the background connection (TCP Reno), while the second column indicates the throughput achieved by the considered TCP variant.

In more detail Table 10 reports the values of the simulations carried out considering 5 min long connections.

We can easily see that the most friendly variants are TCP YeAH and TCP LP that show a difference in the throughput of about 2% with respect to the background connection, while the most unfriendly variant is TCP Vegas, with a difference in the mean throughput of about 1300%.

Table 11 reports the values of the simulations carried out considering 20 s long connections. From the results we can see that, as expected, in this case the considered connection does not have enough time for strongly competing with the background traffic. Nevertheless we can observe that the best behavior is offered by Scalable TCP and TCP BIC.

Table 10
Friendliness vs Reno – 5 min.

Friendliness vs Reno	Throughput (Mbps)	
Variants	Reno	Connection 1
Vegas	583.38	43.42
Veno	286.03	189.40
Westwood	186.93	232.00
BIC	186.46	233.50
CUBIC	236.61	210.90
HSTCP	192.39	229.20
Hybla	225.63	216.00
Scalable	163.50	241.00
Illinois	167.83	239.30
YeAH	**216.17**	**220.10**
HTCP	193.00	229.80
LP	**216.37**	**219.60**

Table 11
Friendliness vs Reno – 20 s.

Friendliness vs Reno	Throughput (Mbps)	
Variants	Reno	Connection 1
Vegas	612.25	12.21
Veno	436.92	109.58
Westwood	331.72	159.50
BIC	**282.38**	**182.90**
CUBIC	343.85	154.3
HSTCP	294.77	177.90
Hybla	424.25	113.64
Scalable	**283.44**	**183.50**
Illinois	292.35	179.00
YeAH	312.48	169.00
HTCP	300.04	175.40
LP	348.73	150.80

6. Discussion

From these analyses we can conclude that the performance offered by the different TCP variants strongly depends on the considered scenario.

In general we can see that, in wired networks, very good performance is offered by TCP Hybla and TCP CUBIC (and some other high-speed variants), with the standard version of the TCP (TCP Reno) able to achieve good performance. As expected the worst performance is usually achieved by TCP-LP.

When moving to wireless networks, the behavior of the different variants changes, with the best performance offered by three variants belonging to three distinct categories: standard TCP (TCP Reno), high-speed TCP variant (TCP BIC) and TCP-LP. In this case we can see that the variants designed for wireless networks are not able to achieve the best performance.

Finally in the satellite networks we can conclude that the best performance is obtained by the variants specifically designed for high-speed long distance network, with Scalable TCP that strongly outperforms all the other variants.In general we can conclude that a "good choice" could be represented by TCP CUBIC which is able to never degrade its performance.

Finally regarding fairness and friendliness we can conclude that the TCP variant that shows the most fair/friendly behavior in the different scenarios is TCP Reno. Note that this is a reasonable result, given that the different variants have been proposed, in most cases, to present a "more aggressive" behavior with respect to the "standard" TCP.

7. Conclusions

The TCP is for sure the most used transport layer protocol in the Internet. In the years several variants have been proposed to overcome the limitations that the standard TCP has in particular scenarios, such as wireless or satellite networks.

As a results, the current kernel of the Linux OS (version 2.6.x) includes 13 different versions, going from the standard TCP (TCP Reno) and its improved version (TCP Vegas), to the variants for wireless networks (TCP Veno and TCP Westwood), high-speed networks (TCP BIC, TCP CUBIC, HSTCP, TCP Hybla, TCP Illinois, Scalable TCP, TCP YeAH, and HTCP), and also a low-priority version (TCP-LP).

In this paper we have offered a complete comparison, by means of several experimental tests, of their behavior in terms of achieved throughput, fairness, and friendliness.

Acknowledgement

The authors thank Francesco "Ardo" Silei for his contribution in support to this work.

References

[1] J. Postel, RFC 793: Transmission Control Protocol, September 1981. <ftp://ftp.internic.net/rfc/rfc793.txt>
[2] W.R. Stevens, TCP/IP Illustrated, The Protocols, us ed., vol. 1, Addison-Wesley Professional, 1994.
[3] J. Widmer, R. Denda, M. Mauve, A survey on TCP-friendly congestion control, IEEE Network 15 (2001) 28–37.
[4] G. Hasegawa, M. Murata, H. Miyahara, Fairness and stability of congestion control mechanisms of TCP, in: INFOCOM, 1999, pp. 1329–1336.
[5] B. Even, Y. Li, D.J. Leith, Evaluating the performance of TCP stacks for high-speed networks (2006).
[6] A. Hamlili, Performance evaluation for competing high-speed TCP protocols, in: GIIS'09: Proceedings of the Second International Conference on Global Information Infrastructure Symposium, IEEE Press, Piscataway, NJ, USA, 2009, pp. 286–289.
[7] P. Sarolahti, A. Kuznetsov, Congestion control in linux TCP, in: Proceedings of USENIX, 2002, pp. 49–62.
[8] M. Allman, V. Paxson, E. Blanton, TCP Congestion Control, RFC 5681 (Draft Standard), September 2009. <http://www.ietf.org/rfc/rfc5681.txt>
[9] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, RFC 2018: TCP selective acknowledgment options, status: PROPOSED STANDARD, October 1996. <ftp://ftp.internic.net/rfc/rfc2018.txt>
[10] M. Mathis, J. Mahdavi, Forward acknowledgement: refining TCP congestion control, in: Proceedings of the SIGCOMM 1996, ACM, New York, NY, USA, 1996.
[11] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, An Extension to the Selective Acknowledgement (SACK) Option for TCP, RFC 2883 (Proposed Standard), July 2000. <http://www.ietf.org/rfc/rfc2883.txt>
[12] A. Kuzmanovic, A. Mondal, S. Floyd, K. Ramakrishnan, Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets, RFC 5562 (Experimental), June 2009. <http://www.ietf.org/rfc/rfc5562.txt>
[13] S. Floyd, T. Henderson, A. Gurtov, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 3782 (Proposed Standard), April 2004. <http://www.ietf.org/rfc/rfc3782.txt>
[14] L.S. Brakmo, S.W. O'Malley, L.L. Peterson, TCP Vegas: New Techniques for Congestion Detection and Avoidance, in: SIGCOMM, 1994, pp. 24–35.
[15] C.P. Fu, S. Liew, TCP Veno: TCP enhancement for transmission over wireless access networks, selected areas in communications, IEEE J. (2003) 216–228.
[16] S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, R. Wang, TCP Westwood: Bandwidth estimation for enhanced transport over wireless links, in: MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2001, pp. 287–297. <http://doi.acm.org/10.1145/381677.381704>.
[17] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast long-distance networks, in: INFOCOM, 2004.
[18] I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, SIGOPS Oper. Syst. Rev. 42 (5) (2008) 64–74. <http://doi.acm.org/10.1145/1400097.1400105>.
[19] S. Floyd, HighSpeed TCP for Large Congestion Windows, 2002.
[20] C. Caini, R. Firrincieli, TCP Hybla: a TCP enhancement for heterogeneous networks, Int. J. Satellite Commun. Network. 22 (2004).
[21] T. Kelly, Scalable TCP: improving performance in highspeed wide area networks, SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 83–91. <http://doi.acm.org/10.1145/956981.956989>.
[22] S. Liu, T. Başar, R. Srikant, TCP-Illinois: a loss and delay-based congestion control algorithm for high-speed networks, in: Valuetools '06: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools, ACM, New York, NY, USA, 2006, p. 55. <http://doi.acm.org/10.1145/1190095.1190166>.
[23] A. Baiocchi, A.P. Castellani, F. Vacirca, YeAH-TCP: yet another highspeed TCP, in: Proceedings of PFLDnet, 2007.
[24] R. Shorten, D. Leith, H-TCP: TCP for high-speed and long-distance networks, in: Proceedings of the PFLDnet, 2004.
[25] A. Kuzmanovic, E.W. Knightly, A. Service, TCP-LP: a distributed algorithm for low priority data transfer, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, vol. 3, IEEE Societies, 2003, pp. 1691–1701.
[26] NLANR/DAST: Iperf – The TCP/UDP Bandwidth Measurement Tool. <http://dast.nlanr.net/Projects/Iperf/>.
[27] S. Hemminger, Network emulation with NetEm, in: Linux Conf Au, 2005. <http://linux-net.osdl.org/index.php/Netem>
[28] R.K. Jain, D.-M.W. Chiu, W.R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, Tech. Rep., September 1984. <http://arxiv.org/abs/cs.NI/9809099>.

C. Callegari received his Laurea degree in Telecommunication Engineering "cum laude" on October 2004 from the University of Pisa, discussing a thesis titled "Simulative analysis of RSVP-TE, and evaluation of end-to-end rerouting techniques in MPLS networks". In 2005 he obtained the qualification to practice the profession of Engineer and he is a member of the IEEE Communication Society. On January 2005 he joined the department of Information Engineering at the University of Pisa as a Ph.D. student, and in 2008 he obtained his Ph.D. in Information Engineering. He is actually a Postdoctoral fellow in the Telecommunication Network research group at the department of Information Engineering of the University of Pisa. His research and professional areas of interest are Network Security, Traffic Classification, Traffic Engineering, MPLS architecture and Network Simulation.

S. Giordano received the Laurea degree "cum laude" in Electronics Engineering and the Ph.D. degree in Information Engineering from the University of Pisa in 1990 and 1994, respectively.
He worked with Consorzio Pisa Ricerche since 1990 in the field of Telecommunication Networks participating and coordinating several research activities. Since the end of 2001 he is associate professor at the Department of Information Engineering of the University of Pisa (Telecommunication Networks Group) where he give lectures on "Telecommunication Networks", and "Design and Simulation of Telecommunication Networks". His research and professional areas of interest are Broadband Communications, Telecommunication Networks Analysis and Design, Simulation of Communication Networks and Systems, Multimedia communications.

M. Pagano received Laurea (cum laude) in Electronics Engineering in 1994 and a Ph.D. in Electronics Engineering in 1998, both from the University of Pisa. Since 2007, he is an associate professor at the Dipartimento di Ingegneria dell'Informazione of the University of Pisa, where he is the official instructor of the courses of ''Telematics'', ''Performance of Multimedia Networks'' and ''Network Security''.
His research interests are related to statistical characterization of traffic flows and to network performance analysis, mainly in the framework of architectures able to support Quality of Service. Performance evaluation has been carried through analytical approaches as well as by means of discrete event simulation. Finally, a new research field is represented by network security issues, mainly in the framework of Intrusion Detection Systems. He has co-authored around 100 papers published in international journals and presented in leading international conferences.

T. Pepe received her Laurea degree in Telecommunication Engineering ''cum laude'' on September 2008 from the University of Pisa, discussing a thesis titled ''Design and simulative analysis of a new algorithm for cost and performance optimization of Service Overlay Network ''. In 2009 she obtained the qualification to practice the profession of Engineer.
On November 2008 she won a grant funded by the MIUR for a Ph.D. position in Information engineering at the Department of Information Engineering of the University of Pisa.
She is actually a Ph.D. Student in the Telecommunication Network research group of the Department of Information Engineering at the University of Pisa.
Her research and professional areas of interest are Network Security and Traffic Classification.