TCP Performance over Satellite in case of Multiple Sessions per Links using Efficient Flow Control and Real OS

M. Luglio[1], C. Roseti[2] and M. Gerla[3]
[1, 2] Dipartimento di Ingegneria Elettronica, Università di Roma Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy
[3] Computer Science Department, University of California Los Angeles
Boelter Hall, Los Angeles CA, 90095 USA

Abstract

Transmission Control Protocol is the layer 4 protocol over which all the Internet applications are based. The high latency severely limits performance, especially if high bandwidth is available. Due to the high Bandwidth Delay Product a long time to fill the pipe is needed.
To counteract such impairments many solutions have been proposed and some of them are actually implemented in real systems. These solutions are based either on the modification of the mechanism, mainly the flow control, or on the architecture, mainly on splitting the connection, which may even include the use of proprietary protocols. In both cases, the features of the Operative Systems of the end machines play a very important role because the dimension of the available buffer determines the initial slow start threshold and in general limits the performance of the flow control mechanism.
Moreover, real scenarios may include both a single TCP session and multiple session per link. In the former case performance are only limited by the latency and by the error rate while in the latter case the simultaneous sessions can further degrade overall performance if fairness and friendliness of different schemes are not so excellent.
In this paper, after describing the main solutions proposed or adopted to improve TCP performance over satellite links, we will present the main results of the optimization of the buffer dimension. Then, we will analyze through simulations the performance of the main TCP schemes with respect to real OS in case of single flow per link. Finally, we approach a scenario with many TCP connections in competition for the available bandwidth in a satellite link. In fact, a fair sharing of resources is suitable in a best-effort Internet environment where there are competing TCP flows.

1 Introduction

TCP is the layer 4 protocol which ensures reliable end-to-end communication implementing the concept of the acknowledgement of the received data [1]. When the end-to-end delay is high, as when a satellite link is part of the path, TCP performance rapidly decreases because the window takes a very long time to increase as well as the pipe to be filled. In order to improve TCP mechanism efficiency over the satellite links, many solutions can be adopted. Some of them are specifically proposed for satellite while others more general [2].
In any case, both efficient TCP algorithms and a powerful OS are needed to achieve good performance. In particular, the finite dimension of the receiving buffer (peculiar of the OS) impacts TCP performance. This feature is even more important if a very efficient TCP algorithm is implemented. In fact, the flow control mechanism aims at regulating the dimension of the sliding window but even though an efficient TCP algorithm were able to reach large windows the buffer length can strongly limit performance.

[1] luglio@uniroma2.it (contact author)
[2] cesare.roseti@uniroma2.it
[3] gerla@cs.ucla.edu

In this paper we investigate how the two aspects (flow control and OS features) impact TCP performance in a scenario including a satellite link as part of the end-to-end path. The optimization of the buffer size using different TCP algorithm has been carried out through simulations performed using ns2 tool [3]. Performance of TCP over satellite links using different flow control mechanism and real OS have been carried out too and shown.

2 TCP performance over satellite links

Satellite systems characteristics have an impact on TCP performance well documented in [4][5][6][7]. In fact, the higher latency with respect to terrestrial networks implies that it takes longer to reach the optimum window size while a higher packet loss can be experienced as a consequence of the greater BER in particular channel conditions. Furthermore, when the satellite provide wide band access the bandwidth x delay product becomes very large impacting the ramping time.
To mitigate such impairments several countermeasures can be implemented both at physical level (mainly to reduce losses) and at network (including layer 4) level [2]. In the next section the main solutions proposed or implemented to improve TCP performance will be described.

3 Solutions to improve TCP performance

The proposed or adopted solutions can be classified as follows:
1. Lower level approach
2. use of TCP implementing modified flow control mechanisms and options.
3. modification of architecture introducing *intermediate nodes* in the network and implementing between the nodes standard or proprietary solutions.

3.1 Lower Level mitigations

To improve TCP performance on a satellite channel, it is recommended to use the following two non-TCP mechanisms: the “Path MTU Discovery” and the “Forward Error Correction” (FEC) [2][4]. The first mechanism determines the maximum packet size a connection can use on a given link without being subject to IP fragmentation. On the other hand, to improve Bit Error Rate (BER) the forward error correction coding (FEC) can be used.

3.2 Modified flow control solutions

3.2.1 Congestion Control: Fast Retransmit and Fast Recovery.

The fast retransmit and fast recovery (FR-FR) is a congestion control algorithm that allows to rapidly recover lost packets [8][9]. In TCP the default mechanism to detect dropped packets is the timeout. During the timeout period, new or duplicate packets cannot be sent. With FR-FR, if the a packet is received out of sequence, a duplicate acknowledgement is sent to the source. If the sender receives three duplicate acknowledgements, it assumes that the corresponding packet is lost and: 1) sets *ssthresh*=0,5**cwnd*, retransmits the missing packet, sets *cwnd*=*ssthresh*+(3*packet size); 2) each time another duplicate acknowledgement arrives, set *cwnd*=*cwnd*+1 packets and transmits a packet if allowed; 3) when an acknowledgement arrives for new packets: *cwnd*=*ssthresh*. FR-FR works very efficiently when there are many isolated packet losses.

3.2.2 Window scaling

The TCP header uses a 16-bit field to report the receiver window size to the sender. The standard maximum TCP window size is 65 kbytes and the TCP throughput is limited by the following formula:

$$Throughput = window\ size\ /\ RTT$$

For example, considering a geosynchronous satellite channel with an RTT of 500 ms the maximum throughput is limited to:

Throughput = 65535 bytes / 500 ms ~ 1 Mbit/s

Therefore, a single standard TCP connection cannot fully utilize the bandwidth available on some satellite channels. To circumvent the problem, the "Window Scale" option has been defined [10]. Such an option expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32-bit value in the 16-bit window field in the TCP header.

3.2.3 Selective Acknowledgements (SACK)

TCP uses a cumulative acknowledgement scheme in which received segments that are out of sequence are not acknowledged and the TCP sender can only learn about a single lost packet per round trip time. This forces the sender to either wait a RTT to realize if packets are lost, or to avoid to retransmit segments which have been correctly received. SACK is a strategy which allows TCP receivers to inform TCP senders exactly which packets arrived, and then to recover more quickly from lost packets avoiding needless retransmissions [10].

3.2.4 TCP Westwood

To improve the performance of the congestion control mechanism, a modification to the Fast Recovery algorithm has been proposed, named TCP Westwood (TCPW), requiring modifications only to the TCP sender [12][13]. TCPW exploits two basic concepts: 1) the end-to-end estimation of the available bandwidth, and 2) the use of such estimate to set the slow start threshold and the congestion window after a packet loss. Using estimated bandwidth instead of packet loss feedback allows Westwood to more efficiently utilize network capacity especially in case of high-bandwidth and huge-delay.

3.2.5 TCP Hybla

In heterogeneous networks, TCP connections characterized by large RTT (i.e. including a satellite segment) present poor performance compared to the wired connections with short RTT. In fact, since the congestion window growth depends by the reception of feedback information (acknowledgements), a large RTT affects the throughput and the channel utilization. TCP Hybla [14] proposes a modification of congestion control policy in order to accelerate the increase of congestion window (in both slow start and congestion avoidance phases) in the long RTT connections.

3.3 Modified architecture solutions

3.3.1 PEPs Performance Enhancing Proxies

To improve the TCP performance on links characterized by high latency and high bandwidth-delay product, instead of or in addition to protocol enhancements, several architectural enhancements, or *middleware* techniques have been proposed. This approach has been generalized to the so-called "Performance Enhancing Proxy" or PEP [14][16]. Many PEPs techniques use a split connection TCP implementation in order to address a mismatch in TCP capabilities between two end-systems. For example, a "splitting" scheme involves segmenting the end-to-end connection into satellite and non satellite segments. This is useful for filling the satellite link which has a high bandwidth-delay product by turning on, for example, appropriated TCP option (i.e. "window scaling option" [10]). Another PEP technique is based on "capturing" a TCP connection and intelligently processing data packets and acknowledgements. This approach is commonly referred to as "spoofing".

3.3.2 Snooping

The snooping technique aims at improving the TCP performance in the links including wireless segments, when many of the assumptions made by TCP are violated [17]. The snoop functions are based only on network-layer software modifications (snoop layer) at the basestation/gateway without breaking the TCP end-to-end semantic. In fact, the main idea is to cache packets and to monitor the acknowledgements flow at the basestation/gateway in order to carry out locally retransmissions.

3.3.3 Early Bandwidth Notification (EBN)

In order to improve the TCP performance in the today's heterogeneous network with variable bandwidth, a modified network architecture has been proposed [18]. The main idea is that an intermediate router measures the current bandwidth available to a TCP flow and feedback such information to the TCP sender. The TCP sender will use this information to adjust his congestion window size. To estimate the bandwidth there are several ways. The most common method is to keep track of all the input and output interfaces.

3.3.4 Xpress Transport Protocol (XTP)

The SkyX system replaces TCP over the satellite link with the Xpress Transport Protocol (XTP) designed for the long latency, high loss, asymmetric bandwidth, typical of satellite communications [19][20]. The XTP is based on orthogonal protocol functions for separating communication paradigm (datagram, virtual circuit, transaction, etc.) from the error control policy employed, separation of rate and flow control, explicit first class for reliable multicast, and data delivery service independence. XTP regulates the data flow by end-to-end windowing flow control mechanism based on 64-bit sequence numbers and a 64-bit sliding window. XTP also provides rate control whereby an end-system or intermediate system can specify the maximum bandwidth and the burst size (maximum number of bytes to be sent in a burst of packets) that it will accept on a connection. Finally, error control incorporates positive and, when appropriate, negative acknowledgements to retransmit missing or damaged data packets. Retransmission may be either go-back-N or selective retransmission.

3.3.5 Space Communication Protocol Standard (SCPS)

The "Space Communication Protocol Standards" (SCPS) are a set of layered protocols optimized for the space segments [21]. The SCPS can be used as an integrated protocol stack, or the individual protocols can be used in combination with Internet protocols. These protocols include: 1) A file handling protocol (SCPS-FP); 2) A retransmission control protocol (SCPS-TP); 3) A data protection mechanism (SCPS-SP); 4) A scalable networking protocol (SCPS-NP). In particular, the SCPS-TP is based on the Internet Transmission Control Protocol (TCP) [22] and User Datagram Protocol (UDP) [23], the Internet Host Requirements Document [24], and the "TCP Extensions for High Performance" [10]. SCPS-TP has developed three capabilities to manage data loss due to transmission errors:

1) *Explicit Corruption Response*. When a isolated data loss occurs, SCPS-TP maintains the transmission rate (controlled by the congestion window) and the retransmission timeout value unchanged.
2) *Selective Negative Acknowledgement (SNACK)*. The SNACK capability has been developed to identify specific data that requires retransmission, and to request immediate retransmission of that data.
3) *Header Compression*. SCPS-TP defines a header compression capability to reduce the size of transmitted packets. This scheme is loss-tolerant, meaning that the loss of one packet does not render subsequent packets unintelligible.

4 Simulation scenario

Simulations have been performed using the well known and reliable NS-2 simulator. NS-2 is an object-oriented simulator widely adopted to reproduce network scenarios [3][25], especially in new transport protocols investigation. In order to run our tests we used the version 2.1b8a. Moreover, we have added the code to simulate the behavior of TCP Westwood ABSE (Adaptive Bandwidth Share Estimation) [25]. The main simulation parameters are the following:

- MinRTT = 508 ms;
- Packet Error Rate (PER) $\in \{0; 10^{-4}; 10^{-3}\}$;
- TCP packet size = 1500 bytes;
- Bandwidth = 4 Mbit/s;
- Transport protocol: TCP New Reno vs. TCP Westwood ABSE.

Moreover, the payload architecture is considered transparent. Simulations have run using different combinations of transport protocol schemes, various PER, and the default receiver buffer size considering the characteristics of real operating systems (Windows, Linux, BSD).

5 Impact of receiver buffer size on the TCP Performance

In the connections with large product bandwidth*delay a limited receiver buffer can affect the TCP performance. In fact, the "Advertised Receive Window" mechanism [1] limits the maximum number of bytes that can be transmitted on the free space in the receiver buffer. To evaluate these effects the following parameters are considered:

– A single TCP connection (TCP New Reno vs. TCP Westwood ABSE);
– Low error rate (PER = 10^{-4});
– typical default values of "maximum socket buffer size" enabled by the commercial OS:
  64 kByte (i.e. Window NT, Linux 2.4);
  128 kByte (i.e. Digital Unix 4.0);
  256 kByte (i.e. BSD/OS).

The outcomes show that TCP Westwood performs overall better than TCP New Reno. In fact, since Westwood resets the congestion window (cwnd) after a packet lost at the bandwidth estimate value, it results more reactive to increase the transmission rate, and then to fill the pipe. Moreover, as shown in Figure 1, in every case Westwood reaches the maximum throughput allowed from the receiver buffer size, whereas New Reno is strongly limited by the transmission errors.

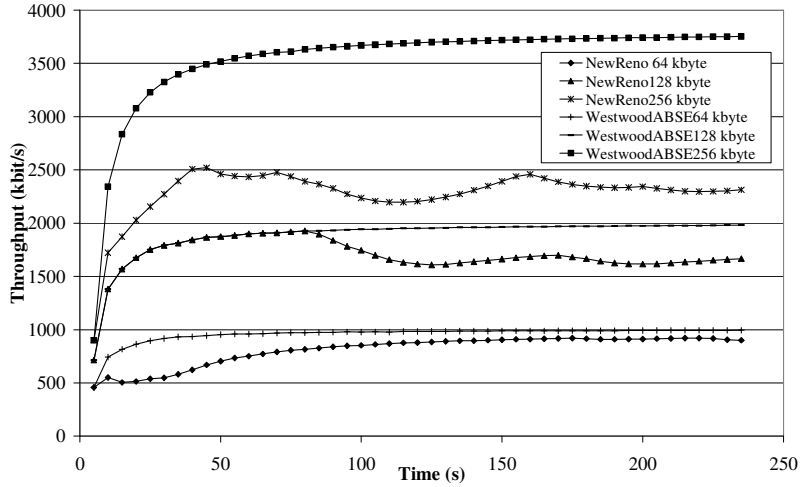

Figure 1: Receiver buffer optimization

6 Multiple TCP connections: fairness and channel utilization

The channel capability can be shared by several TCP flows. Basically, there are two parameters that must be monitored to evaluate the good utilization of the available resources:

1. the fairness index;
2. the total utilization of the channel.

By considering a relevant packet error rate ($5*10^{-5}$), we have analyzed how all the TCP connections share the total bandwidth through the Jain's fairness index [27]:

$$\text{Fairness\_Index} = \frac{\left(\sum x_i\right)^2}{n\sum\left(x_i\right)^2}$$

where $x_i = \frac{T_i}{O_i}$ is the normalized average throughput for the connection i (T_i = measured average throughput, and O_i = fair average throughput), and n is the number of connections.

On the other hand, we have calculated the total utilization of the channel as:

$$\text{Total\_utilitazion} = \frac{\sum T_i}{B}$$

where B is the channel bandwidth.

In our simulations we have considered 5 TCP flows. As shown in Figure 2, both New Reno and Westwood share fairly the bandwidth between the different flows. More specifically, the Westwood flows reach an optimal average utilization of the channel, whereas the NewReno flows waste more than 60% of available bandwidth.

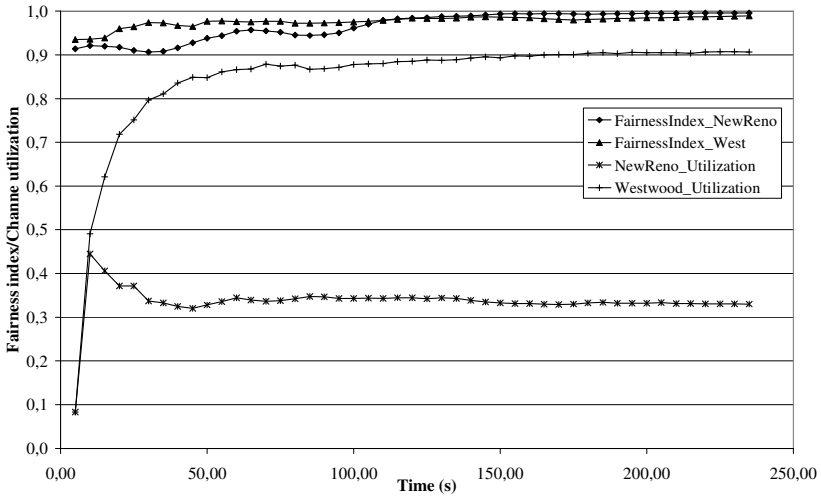

Figure 2: Fairness index and channel utilization

7 TCP friendliness

We have also considered a hybrid scenario where the satellite channel is shared by flows using different TCP schemes. Then, we have analyzed the bandwidth sharing among 5 New Reno flows and 5 Westwood ABSE flows, considering two cases:

1. The 5 New Reno and 5 Westwood flows start simultaneously. We compared the bandwidth sharing with the case in which all the flows support TCP New Reno (Figure 3).
2. Initially, 10 New Reno flows start simultaneously. After 500 seconds, 5 Westwood flows replace 5 New Reno flows. We analyzed the change of the average throughput of New Reno flows after the introduction of the Westwood flows (Figure 4).

The outcomes in Figure 3 and Figure 4 show that Westwood flows are friendly with NewReno flows. In fact, as shown in Figure 2, in the case of 10 New Reno flows the average throughput for each flow is about 280 kbit/s and the total utilized bandwidth is 2.8 Mbit/s (1.2 Mbit/s wasted). Instead, in the case of 5 Westwood flows and 5 NewReno flows, the average throughput for each NewReno flow decreases just of 90 kbit/s, whereas the Westwood flows utilize all the remaining bandwidth. Figure 3 emphasize this behavior when the transmission starts with 10 New Reno flows and after 500 seconds 5 Westwood replace 5 New Reno.

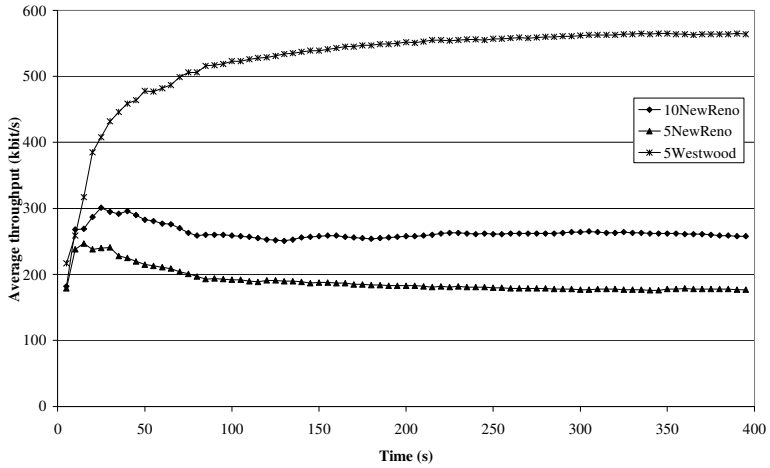

Figure 3: Friendliness with simultaneous flow start

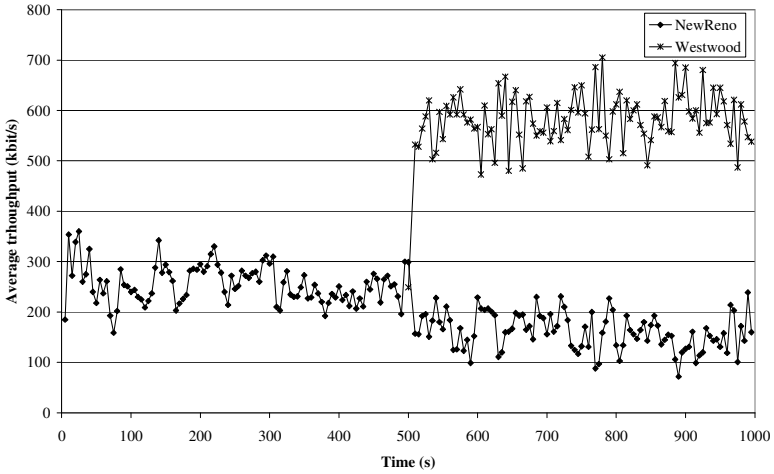

Figure 4: Friendliness with differentiated flow start (WW delayed)

Conclusions

The paper describes and analyze many techniques proposed to improve performance of TCP over satellite links. In addition, significant evaluation of how different flows and different schemes work together over real OS have been carried out.

References

[1] R Stevens. "TCP/IP Illustrated, vol.1", *Addison Wesley, Reading*, MA, USA,1994.

[2] M. Allman, D. Glover, L. Sanchez, "Enhancing TCP over Satellite Channels using Standard Mechanism", *RFC 2488*, January 1999.

[3] NS-2 Network Simulator (Ver.2) LBL, URL: http://www.mash.cs.berkley.edu/ns/

[4] C. Partridge, T. J. Shepard, "TCP/IP Performance over Satellite Links", *IEEE Network*, September-October 1997, pp. 44-49.

[5] M. Allman, C. Hayes, H. Kruse, S. Osterman, "TCP Performance over Satellite Links", *5th Int. Conference on Telecommunication Systems Modelling and Design*, 1997, pp.1-13.
[6] M. Gerla, M. Luglio, R. Kapoor, J. Stepanek, F. Vatalaro, M. A. Vázquez-Castro, "TCP via Satellite Constellations"**,** *Fourth European Workshop on Mobile/Personal Satcoms (EMPS* 2000), London, September 2000, pp. 82-91.
[7] M. Gerla, R. Kapoor, J.Stepanek, P. Loreti, M. Luglio, F. Vatalaro, M. A. Vazquez-Castro, "Satellite Systems Performance with TCP-IP Applications", *Tyrrhenian International Workshop on Digital Communications, IWDC 2001,* September 17-20, 2001, Taormina, Italy, pp. 76-90.
[8] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, Jan. 1997.
[9] S. Floyd and T. Henderson. "The NewReno Modification to TCP's Fast Recovery Algorithm", *Internet RFC 2582*, 1999.
[10] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", *RFC 1323*, May 1992.
[11] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options", *RFC 2018*, October 1996.
[12] M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo. "TCP Westwood: Congestion Window Control using Bandwidth Estimation", IEEE Global Telecommunications Conference, *Globecom*, San Antonio, Texas, USA, 25-29 November 2001, vol. 3, pp. 1698-1702
[13] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", *In Proceedings of ACM Mobicom 2001, pp 287-297, Rome, Italy*, July 16-21.
[14] C. Caini, R Firrincieli, C. Raffaelli, "An RTT Invariant Congestion Control Scheme for Heterogeneous IP Networks", Proc. 12th IST Summit on Mobile and Wireless Communications, Aveiro, Portugal, June 2003, pp. 802-806.
[15] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies", *RFC 3135*, June 2001.
[16] Luglio, M.Y. Sanadidi, M. Gerla, and J. Stepanek, "On-Board Satellite 'Split TCP' Proxy", IEEE Journal on Selected Areas in Communications, JSAC, special issue on Broadband IP Networks via Satellites, February 2004, Vol. 22, n. 2, pp. 362-370.
[17] E. Amir, H. Balakrishnan, S. Seshan, R. H. Katz, "Efficient TCP over networks with wireless links", Proceedings of Fifth Workshop on Hot Topics in Operating Systems, 1995, 4-5 May 1995, pp 35-40.
[18] D. Dutta, Y. Zhang, "An early bandwidth notification (EBN) architecture for dynamic bandwidth environment", IEEE International Conference on Communications, 2002, ICC 2002,Volume 4, 28 April-2 May 2002 pp. 2602-2606.
[19] http://www.ca.sandia.gov/xtp/
[20] http://www.mentat.com/skyx/skyx-whitepaper.html
[21] http://www.scps.org
[22] J. Postel. "Transmission Control Protocol". *Internet RFC 2140*, 1981.
[23] J. Postel. "User Datagram Protocol", *RFC 768*, August 1980.
[24] R, Braden. "Requirements for Internet Hosts", *RFC 1122*, October 1989.
[25] TCP Westwood modules for ns-2: *http://www1.tcl.polito.it/casetti/tcp-westwood*
[26] R. Wang, M. Valla, M. Y. Sanadidi, M. Gerla, "Adaptive bandwidth share estimation in TCP Westwood", IEEE Global Telecommunications Conference, 2002, GLOBECOM '02,Volume 3, 17-21 Nov. 2002, pp. 2604 –2608.
[27] R. Jain, W. Hawe, D. Chiu, "A Quantitative measure of fairness and discrimination for resource allocation in Shared Computer Systems," DEC-TR-301, September 26, 1984.