TCP PERFORMANCE FOR VEHICULAR USERS THROUGH SATELLITE LINKS

M. Casoni and E. Luppi
Department of Information Engineering
University of Modena and Reggio Emilia - Italy
Via Vignolese, 905 - 41100 - Modena - Italy
Tel.: +39-059-2056167; Fax.: +39-059-2056129
email: {casoni.maurizio@unimore.it, luppi.enrica@unimo.it}

***Abstract*— In this paper the reference system is given by wireless users aboard of fast moving vehicles, such as trains, connected to the Internet by means of satellite links. Performance of several TCP flavors are investigated in this scenario using a Performance Enhancing Proxy architecture. TCP performance are reported also in presence of UDP flows and some enhancements, such as large initial window and window scale, and are evaluated in terms of throughput and fairness. Results show that TCP Vegas is the best performing when large initial window and window scale options are implemented.**

I. INTRODUCTION

The ever increasing popularity of hybrid networks, i.e. networks that have both terrestrial and wireless links, reveals that this kind of networks will play a crucial role in future infrastructures for multimedia applications. In particular, in this article we face the problem of evaluating end-to-end performance for customers located inside fast moving vehicles, such as a trains. The network inside the vehicle, e.g. a wireless local area network, is connected to the terrestrial network via one or more satellite links. In this scenario broadband satellite links play a significant role because of their many advantages, such as global coverage, bandwidth flexibility, reliability and multicast capability.

Internet and the related TCP/IP family of protocols has definitely become the leading solution for wide area data internetworking so it is crucial for the performance study of a network combining wired and wireless links. As is widely known, TCP/IP protocols [1] provide a universal network-level internetworking and are the leading communication protocols.

TCP aims at providing reliable connection-oriented communications between hosts in the Internet and it must also cope with the different transmission media which today compose the Internet: in particular, wireless links are one of these and they must be carefully taken into account.

TCP is a reliable window-based acknowledgment-clocked flow control protocol, designed to avoid to overload the network and to react to a possible congestion at network level. This is performed by means of congestion-avoidance algorithms which have expanded to include slow start, fast retransmit and fast recovery. Another window is also defined, the advertised window, as the maximum amount of data the receiver is capable of managing without loss. The congestion window is flow control imposed by the sender whereas the advertised window is flow control imposed by the receiver. The former is based on sender's perception of network congestion while the latter is related to the amount of buffer space at the receiver allocated to a given connection. The TCP sender never sends more than the minimum between congestion and advertised window.

However, TCP performance degrade in networks with long propagation delay and high link error rates such as satellite networks [2] [3]. As a matter of fact, TCP assumes congestion in the network as the main reason for packet losses and delays and if these are caused by something else, as in the wireless case, its performance dramatically decreases [4] [5].

To avoid these problems many solutions have been proposed so far and some of them are actually implemented. These solutions are based either on the modification of the TCP algorithm or on the system architecture.

In this preliminary work, we have evaluated the performance of different TCP flavors, in term of throughput and fairness, in a hybrid network that makes use of a geosynchronous satellite and employs a Performance Enhancing Proxy (PEP) architecture. This PEP solution is suitable to enhance the TCP performance using an appropriate TCP flavor on the satellite link. The overall system, local area aboard the train and satellite network, are investigated through simulation by means of the Network Simulator (ns2) platform.

This paper is organized as follows. Section 2 describes the system architecture. Section 3 details the TCP flavors considered and the enhancements for performance improvement. Numerical results are in Section 4 and in Section 5 conclusions are drawn.

0-7803-9746-0/06/$20.00©2006 IEEE

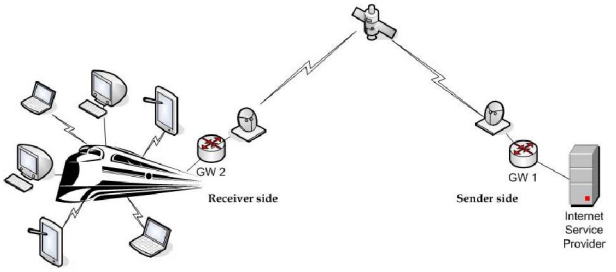

Fig. 1. Reference scenario.

II. System Architecture

The reference network is composed of two parts: a local area network (LAN), wired or wireless inside a train coach, and wireless links, that connect the customers to a Internet Service provider via a satellite (Figure 1).

This is a very challenging scenario since many issues are still open and under investigation. For instance, service provisioning during link outage, e.g. in tunnels, and multimedia applications distribution.

In this early work, the reference architecture employs a PEP at the transport layer [6]. It is a symmetric PEP which means that it has identical behavior in both directions, i.e. the direction of data flow and the one of ACK flow. It implements the TCP spoofing strategy. There are two gateways, one at the sender and the other at the receiver side of the satellite link. The connection between sender and receiver is split into three segments, in order to separate the satellite segment from both the customer and the ISP network. The satellite segment can then make use of an optimized TCP. In the other two segments, a standard TCP is used between senders, or receivers, and respective gateway. The gateway converts TCP to a dedicated protocol for satellite links.

The spoofing gateway sends back ACKs for the received TCP segments to give the illusion of a short delay path and thus can speed up the sender's data transmission. Moreover, it suppresses the true acknowledgment stream from the satellite host and takes responsibility for sending any missing data.

The main advantage of this method is that TCP enhancement is achieved without any modification to the end users avoiding thus to install new software.

III. Transport Protocols for Satellite Links

In this section three TCP flavors are shortly described, focusing basically on their different congestion control algorithms and will be evaluated when applied over the satellite link: Sack, Westwood, Vegas and FAST.

TCP Sack [7] is a TCP flavor that include a strategy which modify the TCP behaviour in case of multiple dropped segment. Using SACK option the receiver informs the sender about the segment that are successfully received, in this way the sender need to retransmit only lost segment.

The main idea of TCP Westwood [8] is to proper estimate the available bandwidth. The TCP sender continuously monitors ACKs from the receiver and computes its current *Eligible Rate Estimate* (ERE). ERE is based on the rate of ACKs and on their payload. After a packet loss, the sender modifies the values of *ssthresh* and *cwnd* as a function of ERE.

TCP Vegas [9] does not wait for a loss to reduce the congestion window. The *expected throughput* of a connection is estimated to be the number of segments in the pipe, i.e. the number of bytes travelling from the sender to the receiver. If congestion exists in the network, the *actual throughput* is less than the expected throughput. The evaluation of expected and the actual throughput is made every RTT. Vegas uses this idea to decide if it should increase or decrease the window. In particular:

$$Expected\_Thr = \frac{cwnd}{RTT_{min}}$$

where RTT_{min} is the minimum RTT experienced so far. The *actual throughput* is computed my measuring the RTT for a given segment and the bytes sent during it. Depending on the value of $Diff = Expected\_Thr - Actual\_Thr$ a linear increase or decrease of *cwnd* occurs. If $Diff$ is less than a parameter α , a linear increase of *cwnd* takes place in the next RTT, else if $Diff$ exceeds parameter β, $cwnd$ is linearly decreased in the next RTT.

TCP FAST [10] employs a congestion control algorithm based on queueing delay as measure of congestion. It periodically updates the congestion window *cwnd* according to

$$cwnd \leftarrow min\Big\{2cwnd, (1-\gamma) + \\ +\gamma\left(\tfrac{baseRTT}{RTT}cwnd + \alpha(cwnd, qdelay)\right)\Big\} \qquad (1)$$

where γ is a constant between 0 and 1, *RTT* is the current average RTT, *baseRTT* is the minimum RTT observed so far, α is a parameter that controls fairness and the number of packets each flow buffered in the network and $qdelay$ is the end-to-end (average) queueing delay.

This algorithm aims at fully utilizing a link with high bandwidth-delay product. The dynamic of FAST and Vegas are quite similar. While TCP Vegas adjusts its window up or down by one packet per RTT, FAST TCP adjust its window by a large amount, up and down, when the number of buffered packets is less or greater than two target values related to α and β, and by a small amount when is close to them.

TCP FAST is supposed to be a high-speed version of Vegas. TCP FAST has been developed for high bandwidth-delay product links and the experimental results have

been so far related to networks with a great amount of bandwidth, such as OC-192 (10 Gbit/s) [10] or 800 Mbit/s [11], and terrestrial propagation delays, as 100, 150 and 200 ms. TCP FAST performance are here evaluated in a different scenario, where a satellite link is included, with higher propagation delays but much lower amount of bandwidth.

TCP performance have been studied by evaluating two metrics, the throughput and the fairness. The throughput is a measure of the variability of the bandwidth usage over a given time scale.

A parameter closely related to throughput is the *channel utilization* U defined as:

$$U = \frac{\sum_{i=1}^{n} B_{P_i}}{Bandwidth} \tag{2}$$

where B_{P_i} is the throughput of the *i*-th flow for a generic transport protocol P.

A widely used metric to evaluate the fairness is the Chiu and Jain's Fairness Index [12]. When it is measured among flows of the same TCP flavor, it is referred to as *intra-fairness*. Consider *n* flows employing the same protocol type *P*, and define $B_{P,min} = minB_{Pi}, i = 1, .., n$ and $B_{P,max} = maxB_{Pi}, i = 1, .., n$, the intra-fairness ratio for the considered streams is defined as [13].

$$F_{intra} = \frac{B_{P,min}}{B_{P,max}} \tag{3}$$

The best intra-fair behavior implies $F_{intra} = 1$.

In order to tackle some specific issues of the satellite link, such as high transmission error rates and long propagation delays, all TCP flavors utilized in the satellite segment have been also extended with some useful options, as *window scale option* and *large initial window*.

A. Window Scale Option.

The *maximum window size* (MWS) of standard TCP is 64 KB, which is a rather small window that may degrade TCP performance over satellite link. When the Window Scale Option [14] is used, in the TCP header a scale factor is set to increase the receiver window. The scale factor is a power of two. This option might become useful in satellite links if we remind that TCP throughput is limited by RTT according to:

$$Throughput = window\_size/RTT \tag{4}$$

that, in the case of geosynchrounous satellite and in a connection that uses the standard value of MWS, is equal to

$$Throughput = 64KB/550ms \simeq 1Mbit/s \tag{5}$$

which makes hard to fully utilize the satellite bandwidth.

B. Large Initial Window

The value of *cwnd* determines the amount of data transmitted. During the slow start phase, that is the initial phase of a connection, the increase of *cwnd* is such so as to probe the network bandwidth, but with the high RTT values typical of satellite links it may become a limiting factor to TCP performance.

Usually, the initial value of congestion window is equal to 1 at the sender side and it increases every incoming ACK. Large Initial Window [15] proposes to start with a larger value of *cwnd*, allowing more segments to be transmitted during the first RTT of the connection. This enhancement speeds up the number of ACKs received at the sender side allowing *cwnd* to increase at a faster rate. The initial value of *cwnd* is defined as follows:

$$IW = min\{4MSS, max\{2MSS, 4380bytes\}\} \tag{6}$$

where MSS defines the maximum segment size. Equivalently, the upper bound for IW is based on the MSS, as follows:

If (MSS $\leq$ 1095 bytes)
*then IW $\leq$ 4 * MSS;*
If (1095 bytes $\leq$ MSS $\leq$ 2190 bytes)
then IW $\leq$ 4380 bytes;
If (2190 bytes $\leq$ MSS)
*then IW $\leq$ 2 * MSS;*

The IW only applies to the first data transmission following the TCP three way handshake. With this option the duration of the Slow Start phase can be reduced by up to three RTTs.

IV. Numerical Results

In this section, numerical results on TCP performance regarding the reference scenario are reported. They have been obtained by means of simulations performed using the *ns-2 simulator* vers. 2.28 [16], which is an object-oriented tool widely adopted to evaluate network performance.

The network topology is shown in Fig. 1, where there are six TCP agents which represent the customers (receiver side).

GW1 and GW2 are the gateways in which PEPs have been implemented; they receive data and ACKs from a connection and resend them into a new TCP connection on the opposite side. A PEP actually acts as forwarding agent. When it receives a data packet from a connection, it immediately acknowledges it and resend it to the new TCP connection. All the new TCP connections are multiplexed into the outgoing link. The gateway output queue has a finite buffer capacity managed by a drop tail queue policy.

By using the forwarding agent in GW1 and GW2 the end-to-end connection is then split into three segments. In the first segment there are two TCP New Reno with Sack

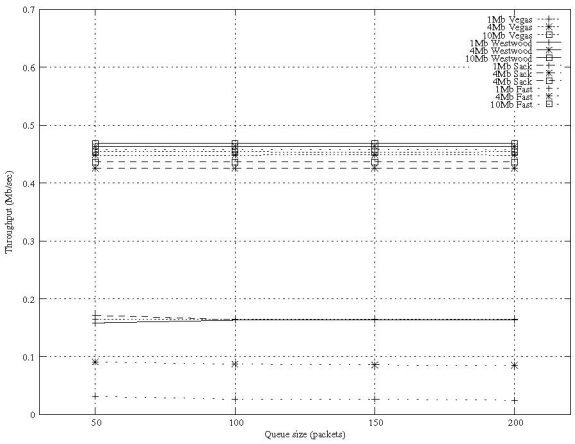

Fig. 2. Average throughput as a function of the output buffer size for six TCP flows.

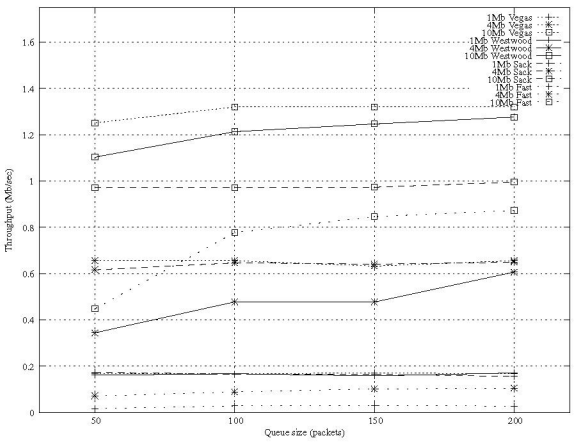

Fig. 3. Average throughput as a function of the output buffer size for six TCP with large initial window and window scale option.

agents, two TCP Vegas agents and two TCP Westwood agents. The basic idea is to consider a general scenario in which customers on the train may have different TCP flavors on their laptops. It is also assumed that the same customer TCP flavor get reused in the third segment, where GW1 is connected to an ISP. Both the segments have 100 Mb/s of bandwidth and a 1 ms delay.

The second segment, the one in the middle, represents the GW1-GW2 communication channel through the satellite. It is also the bottleneck link and is the target of our investigations. This segment is composed of two links: the former between GW1 and the satellite and the latter between the satellite and GW2. Every link has 140 ms delay, that leads to a minimum RTT of 560 ms, and a Packet Error Rate of 10^{-3}. Performance in the second segment are evaluated when all customers TCP flows are converted to either TCP Vegas or TCP Westwood or TCP Sack or TCP Fast. Three values, 1, 4 and 10 Mbit/s, for the satellite link bandwidth, the bottleneck, are taken into account and three different queue lengths, 50, 100 and 200 packets, for the output port of the gateways are considered.

The values of α and β for Vegas are 2 and 4, respectively. The FAST TCP here evaluated [17] assumes the function $\alpha(w, qdelay)$ constant, $\alpha = \beta = 100$.

File Transfer Protocol (FTP) is the application over the TCP agents on the customer side. In the other segments there is not additional competing traffic, since satellite links are dedicated communication channel between GW1 and GW2 and the satellite is used to redirect the incoming traffic only.

Figure 2 reports the average TCP throughput in the second segment as a function of the GW1 output queue size for three values of satellite links, 1, 4 and 10 Mbit/s. It is worth reminding that all six customer TCP flows (two SACK, two Westwood and two Vegas) are mapped into either six SACK or six Westwood or six Vegas or six FAST flows. As expected, the throughput increases when the bandwidth increases from 1 to 4 but no remarkable improvement is obtained with a further increase to 10 Mbit/s. TCP Fast has a singular behavior since it shows very low performance in case of low bandwidth, i.e., 1 and 4 Mbit/s. On the other hand, TCP Westwood seems to provide the better throughput, which, in addition, does not seem to be influenced by the output queue size. This means that, in this case, is useless to employ large buffer, i.e. greater than 80 packets, since just for a 1 Mbit/s bandwidth there is a slight dependance.

Table I reports the intra-fairness index.

	1 Mb	4 Mb	10 Mb
Reno+Sack	0.89	0.92	0.94
Vegas	0.82	0.97	0.96
Westwood	0.91	0.98	0.98
Fast TCP	0.87	0.88	0.93

TABLE I. Intra-fairness for six flows with different bandwidth values and 200 packets queue length.

All TCP flavors show high intra-fairness degree and especially TCP Westwood seems the most fair.

In order to improve the throughput *window scale option* and *large initial window* are introduced in the second segment.

Figure 3 shows again the average throughput as a function of the GW1 output queue size for the same values of satellite links as Figure 2. This Figure clearly states the effectiveness of the two additional mechanisms just introduced and the greater the bottleneck bandwidth the higher the throughput. In case of hard bottleneck, i.e. 1 Mbit/s, it is not possible to get any benefit, with 4 Mbit/s the throughput gain is roughly 50% (from 450 to 610 kbit/s) except for Fast TCP which remains low, and when the bandwidth is equal to 10 Mbit/s the improvement

ranges from twice to three times and also for Fast the gain is remarkable. It is worth noting that the queue size now impacts on the throughput and a 200 packet buffer can now be usefully employed. In this case, TCP Vegas is the best at exploiting *window scale option* and *large initial window*. On the other hand, the low performance of Fast TCP is due to the fact the bandwidth-delay product is not so high for making it effective.

	1 Mb	4 Mb	10 Mb
Reno+Sack	0.71	0.69	0.78
Vegas	0.70	0.70	0.77
Westwood	0.57	0.36	0.37
Fast TCP	0.81	0.87	0.93

TABLE II. Intra-fairness for six TCP flows with different bandwidth values and queue length of 200 packets, with large initial window and window scale option.

Again, intra-fairness has been evaluated and it is reported in Table II. In general the fairness is decreased with respect to table I, Fast TCP shows the best and Westwood the worst intra-fairness behavior.

	1 Mb	4 Mb	10 Mb
Reno+Sack	0.79	0.84	0.95
Vegas	0.37	0.71	0.96
Westwood	0.72	0.81	0.92
Fast TCP	0.16	0.33	0.65

TABLE III. Intra-fairness values for six TCP and two UDP flows with different bandwidth values and 200 packets queue length.

Considering the reference scenario (Figure 1), it is very important to evaluate the TCP performance in presence of multimedia real-time flows which typically do not make use of TCP as transport protocol. Thus, two continuous bit rate (CBR) over UDP flows running at 250 Kb/s are introduced and start working after 50 seconds.

By comparing the results of Figure 2 and Figure 4, the throughput drastically decreases for all TCP flavors when the bottleneck bandwidth is equal to 1 and 4 Mb/s, revealing the strong impact of the bandwidth grabbed by UDP flows. On the other hand, when the bottleneck is not so hard the TCP flows do not get penalized at all. Also in this case, TCP flows show good fairness, especially for 10 Mbit/s bandwidth, as shown in table III.

The last issue to investigate is to evaluate the TCP performance in presence of UDP flows when *window scale option* and *large initial window* are introduced in the second segment.

By comparing Figure 5 with Figure 4 and Figure 3 it is clear both the impact of the UDP flows and the benefits from *window scale option* and *large initial window*. The main result is that TCP Vegas exhibits the best performance being the least sensitive to UDP flows and the one which

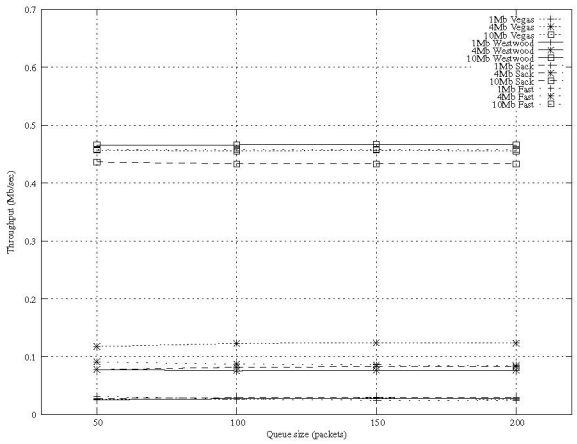

Fig. 4. Average throughput as a function of the output buffer size for six TCP and two UDP flows.

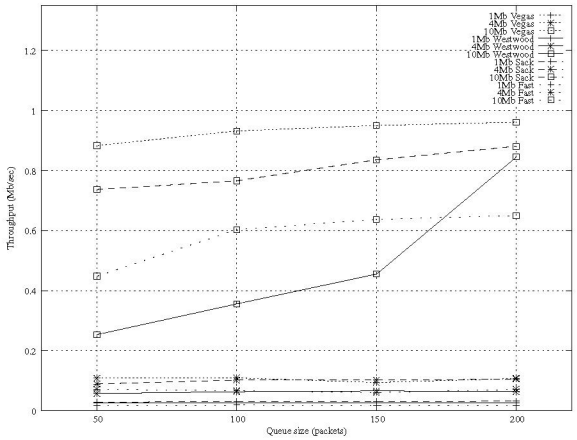

Fig. 5. Average throughput as a function of the output buffer size for six TCP and two UDP flows with large initial window and window scale option.

exploits at the best the *window scale option* and *large initial window*.

As regards the intra-fairness, reported in table IV, TCP Vegas and Fast show the best values for the 10 Mbit/s case.

	1 Mb	4 Mb	10 Mb
Reno+Sack	0.82	0.70	0.74
Vegas	0.37	0.41	0.83
Westwood	0.81	0.51	0.46
Fast TCP	0.26	0.32	0.85

TABLE IV. Intra-fairness values for six TCP flows and two UDP flows and 200 packet queue length with large initial window and window scale option.

Finally, Figure 6 reports the channel utilization U as a function of the queue size, for six TCP and two UDP flows and with a bottleneck bandwidth equal to 10 Mbit/s The most interesting outcome is the different behavior of the TCP flavors with large initial window and window scale

option. TCP Vegas is the best at exploiting the channel bandwidth and is also the best in the output port buffer usage. This means that Vegas with a 50 packets buffer utilizes the channel better than Sack and Westwood with a 200 packets. Of course, this can implies savings in the system design. This confirms the way the different algorithms work: TCP Vegas is aimed at keeping the queue as short as possible while the bandwidth estimation of Westwood is effective only in presence of proper output queue lengths.

It is almost straightforward now concluding that for an effective system design TCP Vegas is the best solution for both metrics, throughput and fairness, providing the highest throughput and intra-fairness degree. Also, this shows that TCP Vegas is quite robust, meaning that it does not get penalized too much with respect to UDP sources.

V. Conclusions

In this paper the reference system investigated is given by wireless users aboard of fast moving vehicles, such as trains, connected to the Internet by means of satellite links.

Performance of several TCP flavors (Sack, Westwood, Vegas and Fast) have been analyzed by using a Performance Enhancing Proxy architecture. TCP performance have been evaluated in terms of throughput, fairness and channel utilization. The effects of multimedia real-time flows, which use UDP as transport protocol, on the TCP performance have been shown with and without the exploitation of additional mechanisms such as large initial window and window scale.

Results have revealed that Vegas can provide remarkable improvements for the throughput in presence of the UDP flows with large initial window and window scale options. Vegas is the best as regards the channel utilization as well, and it also shows a high degree of intra-fairness. Lastly, Vegas is also the best for what concern the buffer usage of the gateway output ports allowing for component savings.

This work represents the first basic step for more advanced investigations of the reference system at network and application levels. Current and next works deal with hand-over support in case of satellite link outage and support for multicasting.

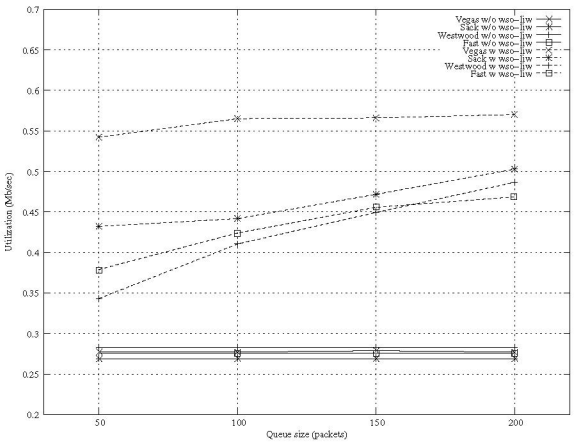

Fig. 6. U as a function of the queue size for six TCP and two UDP flows with large initial window and window scale option.

References

[1] W.R.Stevens, "TCP/IP Illustrated," vol.1, Addison Wesley, 1996.

[2] C. Partridge and T.J. Shepard, *"TCP/IP performance over satellite links"*, IEEE Network Magazine, September/October 1977, pages 44-49.

[3] I. Akyildiz, G. Morabito and S. Palazzo, *"Research issues for transport protocols in satellite IP networks"*, IEEE Personal Communications, June 2001, pages 44-48.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Transactions on Networking*, vol.5, no.6, pp.756-769, December 1997.

[5] G. Xylomenos, G. Polyzos, P. Mahonen, M. Saaranen, "TCP Performance Issues over Wireless Links", *IEEE Communications Magazine*, vol.39, no.4, pp. 52-58, April 2001.

[6] J. Border et al. *"Performance enhancing proxies intended to mitigate link-related degradations"*, RFC 3135 IETF, June 2001.

[7] S. Floyd, J.Mahdavi, M. Mathis, M. Podolsky, *An Extension to the Selective Acknowledgement (SACK) Option for TCP"*, RFC 2883, IETF, July 2000.

[8] C. Casetti, M. Gerla, S. Mascolo M. Y. Sanadidi, and R. Wang. *"TCP Westwood: Bandwidth estimation for enhanced transport over wireless links"*, Wireless Networks 2002.

[9] L. Bramko, S. O'Malley and L. Peterson, *"TCP Vegas:new tecniques for congestion detection and avoidance"*, Proceedings of the ACM SIGCOMM, pp. 24-35, August 1994.

[10] Jin Cheng, D. Wei, S.H. Low, J. Bunn, H.D. Choe, J.C. Doylle, H. Newman, S. Ravot, S. Singh, F. Paganini, G. Buhrmaster, L. Cottrell, O. Martin, Wu-chun Feng, *FAST TCP: from theory to experiments*, IEEE Network Volume 19, Issue 1, Jan.-Feb. 2005 Page(s):4 - 11

[11] Jin Cheng, D.X. Wei, S.H. Low, *"FAST TCP: motivation, architecture, algorithms, performance"*, IEEE INFOCOM 2004, Vol.4, 7-11 March 2004 Page(s):2490 - 2501.

[12] D. M. Chiu, R. Jain, *"Analysis of the increase and decrease algorithms for congestion avoidance in computer networks"*, Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14.

[13] D. Bertsekas, R. Gallager, *"Data Networks,"* Prentice-Hall, 1987.

[14] V. Jacobson, R. Braden and D. Borman, *"TCP extensions for high perormance"*, RFC 1323, IETF, May 1992.

[15] M. Allman, S. Floyd, and C. Partridge, *"Increasing TCP's initial window"*, RFC 3390, IETF, October 2002.

[16] NS-2 Network Simulator (Ver. 2) http://www.mash.cs.berkeley.edu/ns/

[17] http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/