

The Impact of Efficient Flow Control and OS Features on TCP Performance over Satellite Links

M. Luglio¹, C. Roseti² and M. Gerla³

^{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 over 90% of the Internet applications are based. The high latency peculiar of satellite links causes severe degradation of performance, especially if high bandwidth is available. In fact, due to the high Bandwidth Delay Product the pipe requires a long time to fill.

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 TCP algorithm, mainly the flow control and the recovery, or on the architecture, mainly on splitting the connection (eg, PEP – Performance Enhancing Proxies), which may even include the implementation of proprietary solutions. In both cases, the features of the Operating Systems in the end machines play a very important role because the size of the available buffer pool determines the initial slow start threshold and in general limits the performance of the flow control mechanism.

In this paper we describe the main solutions proposed or adopted to improve TCP performance over satellite links and analyze through simulations their performance with respect to existing OSs. The optimization of the buffer dimension has been performed too and the main results are presented.

Keywords

Flow control, Operative Systems, Satellite, TCP

Acronyms

ABSE	Adaptive Bandwidth Share Estimation	RTT	Round Trip Time
BDP	Bandwidth Delay Product	SACK	Selective Acknowledgement
BER	Bit Error Rate	SNACK	Selective Negative Acknowledgement
FEC	Forward Error Correction	SCPS	Space Communication Protocol Standard
FR-FR	Fast Retransmit-Fast Recovery	SCPS-FP	SCPS-File Protocol
GEO	Geosynchronous Earth Orbit	SCPS-NP	SCPS-Network Protocol
ICMP	Internet Control Message Protocol	SCPS-SP	SCPS-Security Protocol
IP	Internet Protocol	SCPS-TP	SCPS-Transport Protocol
LEO	Low Earth Orbit	TCP	Transmission Control Protocol
MTU	Maximum Transmission Unit	VoIP	Voice over IP
OS	Operative System	XTP	Xpress Transport Protocol
PEP	Performance Enhancing Proxy		

¹ luglio@uniroma2.it (contact author)

² cesare.roseti@uniroma2.it

³ gerla@cs.ucla.edu

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 decrease because the window takes a very long time to increase as well as the pipe to be filled. Nevertheless, the satellite assumes a meaningful role to allow the access to the telecommunication networks in very remote location and is irreplaceable where terrestrial infrastructures are scarce or absent, where it represents the only means to communicate (long range mobile services, maritime, aeronautical environment) or when terrestrial infrastructures may be damaged or out of order due to disasters.

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 for more general environment [2]. Many proposals are based on the modification of the flow control (e.g. enlarging the dimension of the initial window, reducing the number of acknowledgments), of the recovery mechanism (dynamic estimation the available bandwidth, etc.); others are based on the modification of the architecture (e.g. splitting the path and terminating connection at each step acknowledging packet reception). Among the latter typology of solutions some adopt standard algorithms or a modified version while there are solutions based on proprietary standard implemented in real systems.

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 size is the characteristic of the OS which 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.

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.

In particular, in section 2 performance of TCP in satellite environment will be analyzed, in section 3 the main solution to improve performance will be described and analyzed, in section 4 the features of OS impacting TCP performance will be highlighted and some real OS will be shown as examples, in section 5 the simulation scenario will be described, in section 6 the results of the OS buffer

optimization will be shown and discussed, in section 7 the performance of TCP with different OS using two different flow control mechanisms will be shown and finally in the last section conclusions will be drawn.

2 TCP performance over satellite links

Satellite systems can use the geostationary orbit (GEO) or Low Earth Orbit (LEO). Different satellite architectures show different performance. In fact, GEO constellation are characterized by very high delay (RTT = 550 ms) which can vary in the order of about 20 ms as a function of the earth station location in the coverage area. In addition, the great distance implies also high free space losses which can increase BER. LEO constellation are characterized by shorter distance with respect to GEO (RTT = 20 -50 ms) but the variability, depending on satellite movement, can be greater and more dynamic. The free space losses are reduced too. In addition, frequent handover represents a critical aspect that must be carefully taken into account.

In both cases, the two implications have an impact on TCP performance [4][4][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 impairments experienced at TCP level on a satellite link can be mitigated using different approaches. The proposed or adopted solutions can be classified as follows:

1. Lower level approach
2. use of TCP over the satellite link, implementing modified flow control mechanisms and options.
3. break TCP semantics 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. In fact, if the packet is too large to be forwarded, the gateway fragments the packet and forwards the fragments. Instead, with the Path MTU discovery mechanism, an ICMP message is returned to the sender to indicate that the original packet could not be transmitted without being fragmented and also to report the largest packet size that can be forwarded by the gateway. In this way, the packet overhead is reduced and TCP senders can increase the congestion window more rapidly (in terms of bytes). On the other hand, the most important problem for developers of TCP in satellite network is high Bit Error Rate (BER). TCP's strategy considers all packet drops as signals of network congestion and reduces its window size in an attempt to alleviate the congestion. Therefore, packets dropped due to corruption cause TCP to reduce the size of its sliding window, even though these packet drops not for network congestion. This reaction is unwanted because it results in throughput loss. Then, the TCP operates efficiently when nearly all loss is due to network congestion. To such purpose, the forward error correction coding (FEC) can be used to improve the bit-error rate in satellite environments.

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 most efficiently when there are many isolated packet losses.

3.2.2 Large TCP Windows

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:

$$Throughput = 65535\ bytes / 500\ ms \sim 1\ Mbit/s$$

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. 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 have arrived, and then to recover more quickly from lost packets avoiding needless retransmissions [10].

3.2.4 TCP Westwood

The TCP end-to-end control is based on implicit feedback, such as timeouts, duplicate acknowledgements, round trip measurements. But, when losses occur, TCP cannot distinguish between congestion losses and transmission losses and, interpreting this event always as a network congestion, activates algorithms which reduce outgoing data stream. To improve the performance of the congestion control mechanism, an improvement to the Fast Recovery algorithm has been proposed, named TCP Westwood (TCPW), requiring modifications only to the TCP sender [11][12]. 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.

3.3 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 [13][14][15][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" [17]). 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”.

In the frame of PEP implementation, if a link is composed by both a terrestrial and a satellite segment, it’s possible to consider a particular architecture in which a gateway intercepts the TCP connection from the sender and converts the data to the optimized protocol for transmission over the satellite, and another gateway on the opposite side of the satellite link translates the data back to TCP for communication with the receiver. Moreover, this architecture offers vastly improved performance while remaining entirely transparent to the end users and fully compatible with the Internet infrastructure. In other words, the gateway splits the single TCP connection into three separate components:

- 1) A TCP connection between the sender and the gateway;
- 2) A optimized transport protocol connection over the satellite between the two gateways;
- 3) A TCP connection between the opposite gateway and the receiver.

In this way, the TCP congestion avoidance mechanisms persist over the terrestrial connections to protect the stability of the routed network and an optimized transport protocol allows to reach the maximum bandwidth utilization in the satellite segment. Two implementations of this concept are described in the following subsections.

3.3.2 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 conditions typical of satellite communications [18][19]. 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) it will accept on a connection. Finally, error control incorporates positive and, when appropriate, negative acknowledgement to make retransmission of missing or damaged data packets. Retransmission may be either go-back-N or selective retransmission.

3.3.3 Space Communication Protocol Standard (SCPS)

The “Space Communication Protocol Standards” (SCPS) are a set of layered protocols optimized for the space segments [20]. 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) [21] and User Datagram Protocol (UDP) [22], the Internet Host Requirements Document [23], and the “TCP Extensions for High Performance” [17]. Then, SCPS extensions to these base standards improve performance in the space communications. For example, SCPS-TP has developed three capabilities to address the possibility of 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 OS features impacting TCP performance

Under ideal conditions, the optimum performances are reached when data pipe between a sender and a receiver is kept full. Moreover, the maximum amount of data that can be in transit in the network are achieved by the bandwidth-delay product, that is, the product of the bottleneck link bandwidth and Round Trip Time (RTT). But, in order to prevent packet loss due buffer overflow, the receiver is required to have enough buffer space to store all incoming data. For this purpose, TCP implements a mechanism called “Advertised Receive Window” [1]. Under this scheme, the maximum number of bytes, that the sender can transmit, is equal to the free space in the receiver buffer. Therefore, the maximum transmission rate possible is regulated by the following formula:

$$\text{Max Transmission Rate} = \text{Advertised Receive Window} / \text{Round Trip Time}$$

Typically, operative systems limit the amount of memory that can be used by an application for buffering network data. Without the support of the “Window Scale” option [17], the maximum buffer size that can be used by the application is limited to 64 kbytes. Then, in order to take full advantage of the high speed networks, the host system must be configured to support large enough socket buffers for reading and writing data in the network. Today, typical Unix systems consider a default maximum buffer size for the socket between 128 kbytes

Table 1: TCP features under various operating systems

Operating System	RFC1191 Path MTU Discovery	RFC1323 Support	Default maximum socket buffer size	Default TCP socket buffer size	Default UDP socket buffer size	Applications (if any) which are user tunable
BSD/OS	Yes	Yes	256kbyte	8kbyte	9216 snd 41600 rcv	None
(Compaq) Digital Unix 4.0	*	Yes Winscale, No Timestamp	128kbyte	32kbyte	9216 snd 41600 rcv	None
FreeBSD 2.1.5	Yes	Yes	256 kbyte	16kbyte	40kbyte	None
HPUX 9.{00,01,10,20,30}	Yes	Yes	256kbyte	32kbyte	9216	FTP
IBM AIX 3.2 & 4.1	No	Yes	64kbyte	16kbyte	9216 snd 41600 rcv	None
Linux 2.4.00 or later	Yes	Yes	64kbyte	32kbyte	*	None
MacOS (Open Transport)	Yes	Yes	Limited only by available system RAM	32kbyte	64kbyte	Fetch (ftp client)
FTP Software (Netmanage) OnNet Kernel 4.0 for Win95/98	Yes	Yes	963,75MB	8kbyte [146kbyte for satellite tuning]	8kbyte snd 48kbyte rcv	FTP server
Sun Solaris 7	Yes	Yes	1MB TCP, 256kbyte UDP	8kbyte	8kbyte	None
Microsoft Windows NT 3.5/4.0	Yes	No	64kbyte	Max(~8kbyte, min(4*MS S, 64kbyte))	*	*

and 1 Mbyte, but these values may not be sufficient to fully utilize the available bandwidth. Thus, to maximize the TCP performance on an high speed link, it's desirable to make sure that the host has set appropriately the socket buffer. Moreover, it's worth to enable some TCP options, such as "SACK" [10] and "Large Window" [17]. Table 1 shows the characteristics of the main OS [24].

5 Simulation scenario

Simulations have been performed using ns-2 simulator [3] which is an object-oriented tool widely adopted to reproduce network scenarios. We selected this platform due to the reliability of its outcomes and the validity of its models, especially in new transport protocols investigation. In addition, the open source code allows great flexibility. In order to run our tests we used the version 2.1b8a. Moreover, we added the code to simulate the behavior of TCP Westwood ABSE (Adaptive Bandwidth Share Estimation) [25][26][27]. Figure 1 represents the scenario utilized for the simulations. 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 protocols: TCP New Reno, TCP Westwood ABSE.

Moreover, the payload configuration is assumed non-regenerative. Simulations were performed using different combinations of transport protocol and various PER.

6 OS parameters optimization

In Figure 2 we show the average throughput achieved for different combinations of transport protocol, various PER in the satellite link and different values for the buffer size in the receiver host.

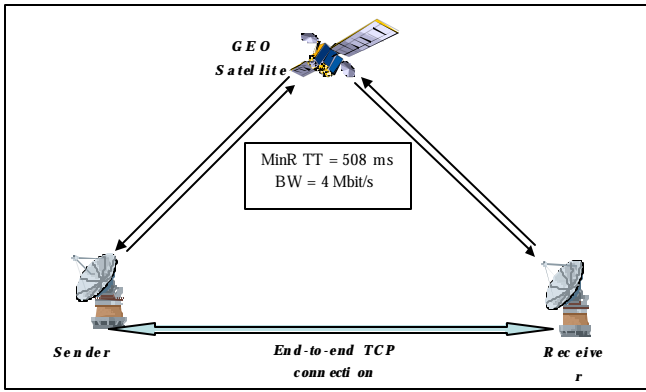


Figure 1: Simulation scenario

In the error free case, TCP Westwood ABSE and TCP New Reno have the same trend. In fact, in both cases the average throughput grows linearly up to the buffer size catches up the data pipe value (~170 packets), and remains constant for greater values. Instead, if errors are present in the channel, TCP Westwood outperforms New Reno. In fact, after the saturation point (buffer size > data pipe value) the bandwidth utilization goes from 88,45% (with $PER = 10^{-4}$) to 75,5% (with $PER = 10^{-3}$) for TCP Westwood. On the other hand, New Reno catches up only a bandwidth utilization of 29,79% if $PER = 10^{-4}$, and 16,24% if $PER = 10^{-3}$. Finally, Figure 2 shows as TCP Westwood trend is more stable for values lower than the saturation point.

7 Performance of TCP with different OS

Simulations results presented in Figure 3 show the time required to transmit a 50 Mbytes file and a 100 Mbytes file from a server to a client via a GEO satellite considering no error rate. We have considered typical default values of “maximum socket buffer size” enabled by the commercial operating system:

- o 64 kbyte (i.e. Window NT, Linux 2.4);
- o 128 kbyte (i.e. Digital Unix 4.0);
- o 256 kbyte (i.e. BSD/OS).

The outcomes show that both New Reno and Westwood halve the transfer time when the receive buffer size doubles.

In Figure 4 and Figure 5 we compare the transfer time for different combinations of transport protocol, different PER (10^{-4} and 10^{-3} respectively) and various values of receiver buffer size. In presence of errors, Westwood and NewReno have different behavior. In fact, since the transmission errors affect strongly the New Reno performance [9], a greater receive buffer bring a smaller reduction of the transfer time. On the other hand, Westwood, due to its congestion control algorithm [11], performs better in the link with errors, and then halves the transfer time when the buffer size doubles.

Moreover, in both larger receive window and higher error rate case, Westwood allows to transmit 100 Mbytes more rapidly than NewReno 50 100 Mbytes.

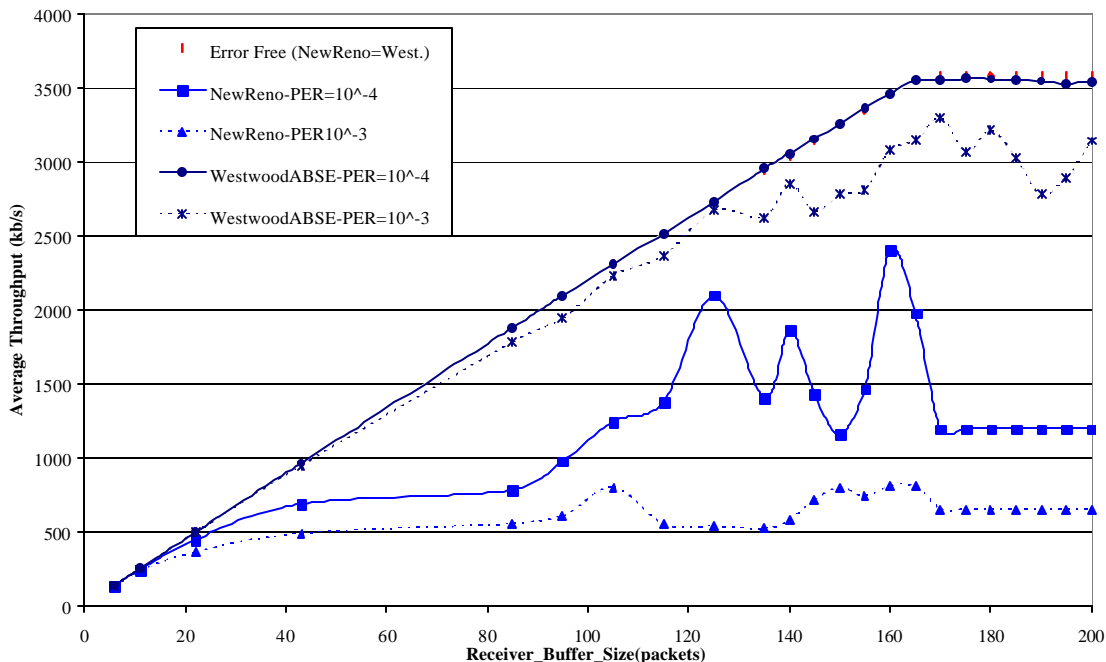


Figure 2: Receiver buffer optimization

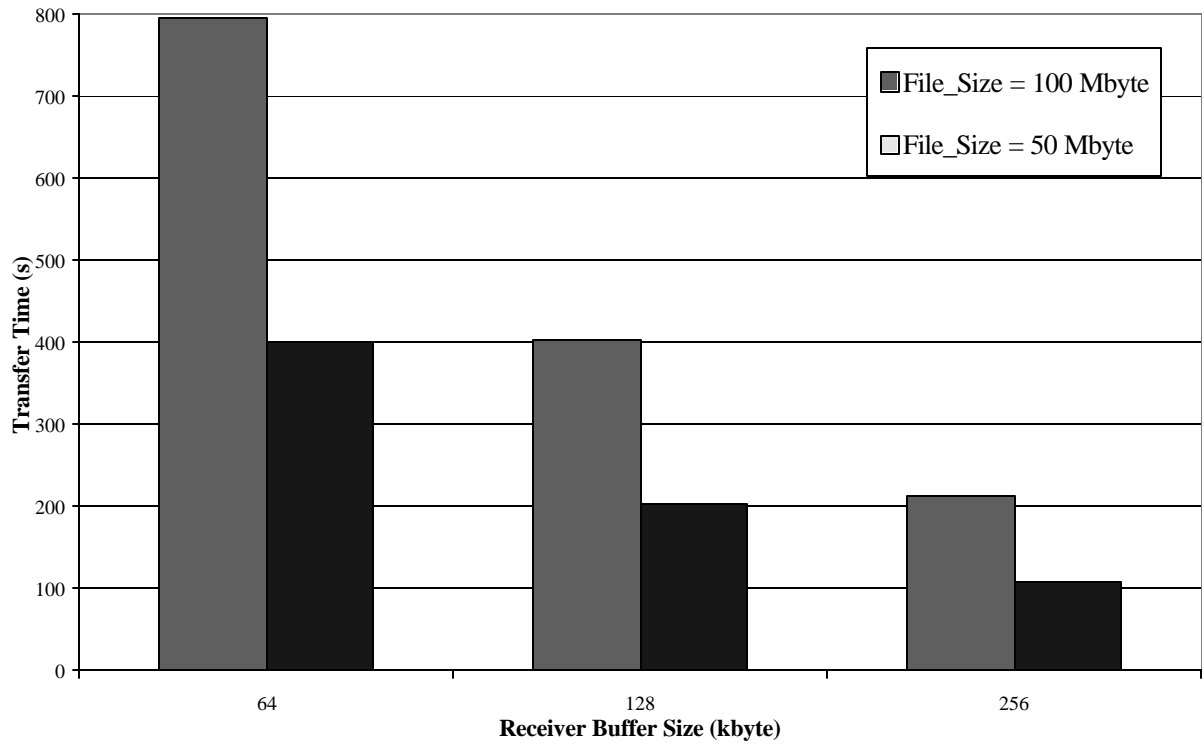


Figure 3: Transfer delay as a function of the buffer size with no PER

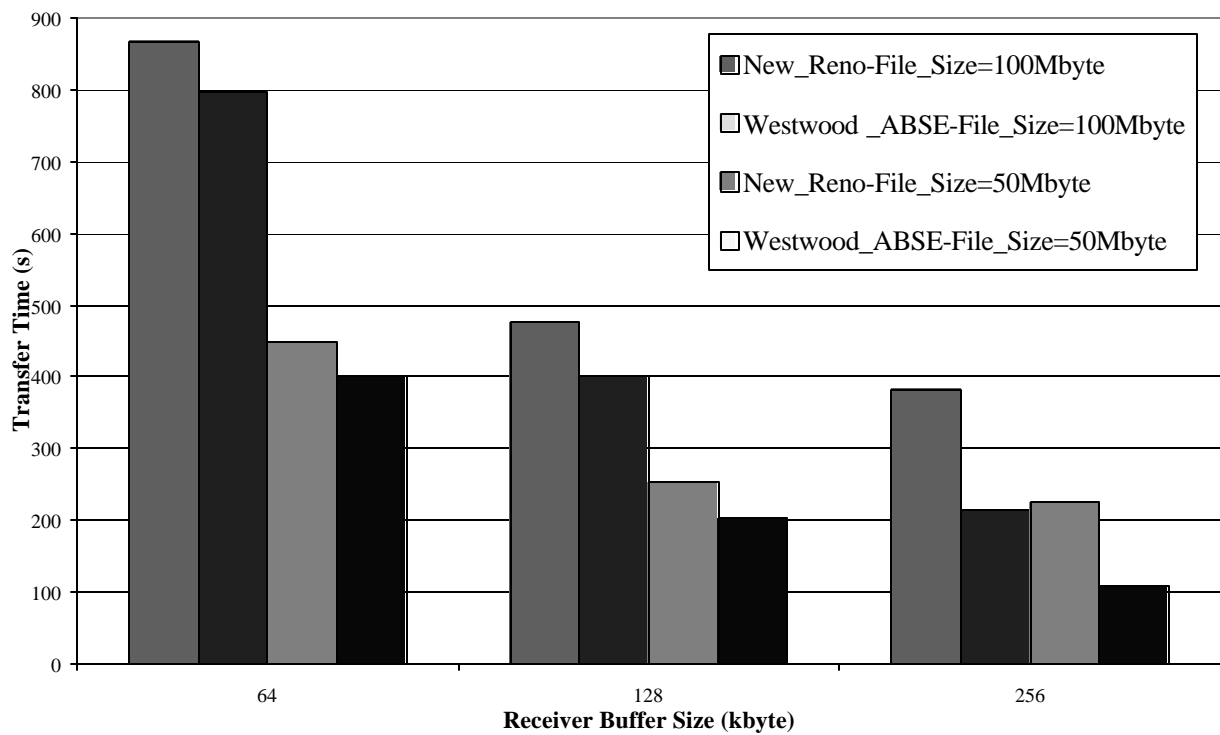


Figure 4: Transfer delay as a function of the buffer size with PER 10⁻⁴

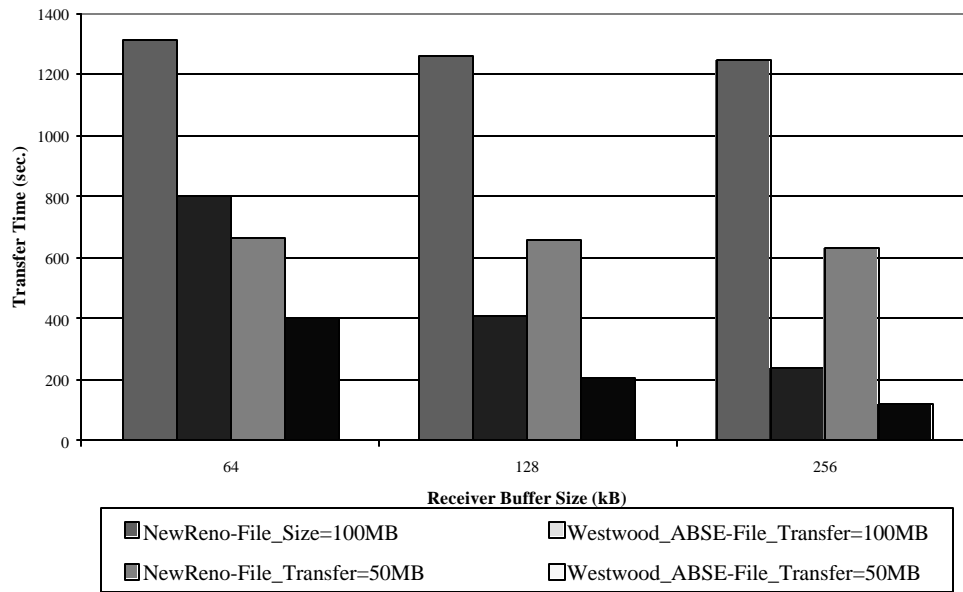


Figure 5: Transfer delay as a function of the buffer size with PER 10⁻³

Conclusions

Performance of TCP based applications are limited in satellite environment due to the high latency especially when large bandwidth is allowed. The paper describes and analyze many techniques proposed to improve performance. Among the techniques based on flow control modification, TCP Westwood shows very good performance. Nevertheless, utilizing Westwood with real OS the improved performance in some cases can be limited by the finite receive buffer size.

The results show how the increased buffer size offered by some OS is exploited to significantly improve performance.

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.berkeley.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.
- [7] M. Gerla, R. Kapoor, J. Stepanek, P. Loreti, M. Luglio, F. Vatalaro, M. A. Vázquez-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] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options", *RFC 2018*, October 1996.
- [11] M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo. "TCP Westwood: Congestion Window Control using Bandwidth Estimation", *Globecom*, San Antonio, Texas, USA, November 2001.
- [12] 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 2001.
- [13] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies", *RFC 3135*, June 2001.

- [14] M. Luglio, J. Stepanek and M. Gerla, "TCP Performance using Splitting over Satellite Link", *8th Ka Band Utilization Conference*, September 25-27, 2002, Baveno, Italy, pp 45-52.
- [15] J. Stepanek, A. Razdan, A. Nandan, M. Gerla, M. Luglio, "The use of a Proxy on Board the Satellite to Improve TCP Performance", *In Proceeding of IEEE Global Telecommunications Conference (GLOBECOM'02)*.
- [16] M. Luglio, M.Y. Sanadidi, M. Gerla, and J. Stepanek, "On-Board Satellite 'Split TCP' Proxy".
- [17] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", *RFC 1323*, May 1992.
- [18] <http://www.ca.sandia.gov/xtp/>
- [19] <http://www.mentat.com/skyx/skyx-whitepaper.html>
- [20] <http://www.scps.org>
- [21] J. Postel. "Transmission Control Protocol". *Internet RFC 2140*, 1981.
- [22] J. Postel. "User Datagram Protocol", *RFC 768*, August 1980.
- [23] R. Braden. "Requirements for Internet Hosts", *RFC 1122*, October 1989.
- [24] http://www.psc.edu/networking/perf_tune.html
- [25] M. Gerla, M.Y. Sanadidi, and R. Wang, "Adaptive Bandwidth Share Estimation in TCP Westwood", *to appear in Globecom 2002, Taipei, Taiwan*.
- [26] M. Gerla, M.Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation".
- [27] TCP Westwood modules for ns-2:
<http://www1.tcl.polito.it/casetti/tcp-westwood>
- [28] V. Bharadwaj, J. Baras and N. Butts, "An architecture for Internet service via broadband satellite networks", *Int. J. Satell. Commun. vol. 19, 2001, pp. 29-50*.
- [29] M. Allman, S. Floyd and C. Partridge, "Increasing TCP's initial window," *IETF Internet Draft: draft-ietf-tsvwg-initwin-00.txt*, May 2001.
- [30] M. Allman (editor), "Ongoing TCP Research Related to Satellites", *RFC 2760*, Feb. 2000.
- [31] M. Allman and D. Glover. "Enhancing TCP Over Satellite Channels using Standard Mechanisms", *IETF draft*, January 1998.


From August to December 1992 he worked, as visiting Staff Engineering at Microwave Technology and Systems Division of Comsat Laboratories (Clarksburg, Maryland, USA).

He received the Young Scientist Award from ISSSE '95. Since October 1995 he is research and teaching assistant at University of Rome "Tor Vergata" where he work on designing satellite system for multimedia services both mobile and fixed, in the frame of projects funded by EC and ESA.

He taught Signal Theory and collaborated in teaching Digital Signal Processing and Elements of Telecommunications.

In 2001 and 2002 he taught Satellite Network class at University of California Los Angeles (UCLA) Computer Science Department. Now he teaches Signals & Transmission and Satellite Telecommunications.

"Cesare Roseti is a Ph.D student in the Electronic Engineering department at the University of Rome "Tor Vergata". He received his graduate degree (1st level), cum laude, in Telecommunication Engineering in 2002 and his graduate degree (2nd level), cum laude, in Telecommunication Engineering in 2003, both from University of Rome "Tor Vergata". His current research interests are in the analysis of transport protocols in satellite environments."

Mario Gerla received a graduate degree in engineering from the Politecnico di Milano in 1966, and the M.S. and Ph.D. degrees in engineering from UCLA in 1970 and 1973, respectively. After working for Network Analysis Corporation from 1973 to 1976, he joined the Faculty of the Computer Science Department at UCLA where he is now Professor. His research interests cover the performance evaluation, design and control of distributed computer communication systems; high speed computer networks; wireless LANs, and; ad hoc wireless networks. He has worked on the design, implementation and testing of various wireless ad hoc network protocols (channel access, clustering, routing and transport) within the DARPA WAMIS, GloMo projects. Currently he is leading the ONR MINUTEMAN project at UCLA, and is designing a robust, scalable wireless ad hoc network architecture for unmanned intelligent agents in defense and homeland security scenarios. He is also conducting research on QoS routing, multicasting protocols and TCP transport for the Next Generation Internet (see www.cs.ucla.edu/NRL for recent publications). He became IEEE Fellow in 2002. 

Author Biography

Luglio Michele received the Laurea degree in Electronic Engineering at University of Rome "Tor Vergata". He received the Ph.D. degree in telecommunications in 1994 at University of Rome "Tor Vergata".