# Evaluating TCP Mechanisms for Real-Time Streaming over Satellite Links

Panagiotis Papadimitriou and Vassilis Tsaoussidis

Demokritos University, Electrical & Computer Engineering Department,
Xanthi, 67100, Greece
{ppapadim, vtsaousi}@ee.duth.gr

**Abstract.** Real-time streaming over satellite IP networks is challenging, since satellite links commonly exhibit long propagation delays and increased error rates, which impair TCP performance. In this context, we quantify the effects of satellite links on TCP efficiency and streaming video delivery. We investigate a solution-framework composed by TCP implementations which are expected to perform adequately in such environments. Furthermore, we study the supportive role of Selective Acknowledgments (SACK). Along with protocol performance, we also evaluate the impact of delayed acknowledgments. Our simulation results illustrate that most existing end-to-end solutions do not comply with the stringent QoS provisions of time-sensitive applications, resulting in inefficient bandwidth utilization and increased delays in data delivery. Finally, with the absence of a satellite-optimized TCP implementation for real-time streaming, we identify TCP Real as the most prominent solution, since it manages to alleviate most of the impairments induced by satellite links, sustaining a relatively smooth transmission rate.

## 1 Introduction

Satellite systems evolve towards the delivery of broadband IP services and are candidates to integrate the wireless data networks, due to their wide coverage and broadcast capabilities. Geostationary (*GEO*) and low-altitude earth orbit (*LEO*) satellites enable the delivery of time-sensitive data, such as audio and video content, over large coverage areas. Unfortunately, satellite networks demonstrate several drawbacks. Firstly, in order to provide services at a reasonable cost, satellite links exhibit bandwidth asymmetry, since they comprise a high-capacity forward space link and a low-bandwidth reverse (space or terrestrial) path. Some satellite networks are inherently bandwidth asymmetric, such as those based on a direct broadcast satellite (*DBS*) downlink and a return via a dial-up modem line. For purely GEO or LEO systems, many proposed systems offer the capability to download at tens of Mb/s, but they do not provide uplinks at rates faster than several hundred Kb/s or a few Mb/s, due to uplink carrier sizing. Furthermore, satellite networks demonstrate relatively increased propagation delays which dramatically affect the bandwidth-delay product (*BDP*). Long transmission distances result in fading channels and eventually in bit error rates (*BER*) which remain higher ($10^{-6}$ or worse) than in terrestrial networks. Reception of corrupted data may trigger requests for retransmission resulting in possible congestion and increasing end-to-end delays.

T. Braun et al. (Eds.): WWIC 2006, LNCS 3970, pp. 62 – 74, 2006.

Most Internet transport protocols exhibit limited efficiency under these awkward conditions. *Transmission Control Protocol* (*TCP*), based on the principles of congestion management [9], *Slow-Start* [16], and *Additive Increase Multiplicative Decrease* (*AIMD*) [4], was designed to provide a reliable data delivery service for wired IP networks. As a result, it demonstrates inadequate performance in heterogeneous wired/wireless environments, such as satellite networks. Authors in [18] outline three major shortfalls of TCP: (i) ineffective bandwidth utilization, (ii) unnecessary congestion-oriented responses to wireless link errors (e.g. fading channels) and operations (e.g. handoffs), and (iii) wasteful window adjustments over asymmetric, low-bandwidth reverse paths. More precisely, TCP commonly sets the initial slow-start threshold (*ssthress*) to an arbitrary value independently of BDP. If *ssthress* is adjusted too high relatively to the network BDP, the exponential increase of congestion window (*cwnd*) may cause multiple packet drops and coarse timeouts. Inversely, in the situation of a relatively low value of ssthress the slow-start phase is concluded prematurely resulting in poor startup utilization. Furthermore, standard TCP is not able to detect the nature of the errors that cause packet drops and consequently determine the appropriate error-recovery strategy. Hence, TCP invokes congestion-oriented responses to all wireless errors, which are common in satellite links, resulting in unnecessary throughput degradation. Bandwidth asymmetry also impacts TCP performance [2]. Despite the small size of acknowledgment (*ACK*) packets, the reverse channel is often unable to carry the high rate of ACKs. The congestion in the reverse path inevitably increases *Round Trip Time* (*RTT*) diminishing the protocol efficiency.

Apart from the particular characteristics of satellite links, TCP should comply with the stringent requirements and constraints of time-sensitive traffic. Real-time applications are comparatively intolerant to delay and variations of throughput and delay. Furthermore, reliability parameters, such as packet drops and bit errors, usually compose an impairment factor, since they cause a perceptible degradation in media quality. Standard TCP usually induces oscillations in the achievable transmission rate and occasionally introduces arbitrary delays, since it enforces reliability and in-order delivery. In this context, several TCP protocol extensions [5, 22] have emerged to overcome the standard TCP limitations providing more efficient bandwidth utilization in order to achieve a smooth transmission and playback rate.

Along these lines, the constraints of transmission over satellite links, as well as streaming media requirements call for effective and robust transport protocol services. Although numerous research proposals have emerged towards improving transport services over wireless/satellite links, the converged domain of time-sensitive data delivery over satellite IP networks has not attracted the required attention from the research community. Realizing the issues and parameters that affect TCP performance over satellite links, our objective is to exploit TCP's potential for efficient streaming media delivery over such environments. In this context, we investigate a solution-framework based on the most prominent end-to-end solutions. Furthermore, we assess the impact of delayed ACKs and link asymmetry on TCP performance, as well as the associated impact of diverse link error rates. In this study, we do not include *User Datagram Protocol* (*UDP*) in our evaluation experiments; the protocol lacks all basic mechanisms for error recovery and flow/congestion control, and thus provides a

different type of service. In [14] we have shown that UDP may perform worse than TCP in several occasions. In addition, the absence of congestion control poses a threat to network stability.

The rest of the paper is organized as follows. Section 2 summarizes related work, while in Section 3 we formulate a transmission gap model for asymmetric satellite links. Section 4 includes our evaluation methodology followed by Section 5, where we analyze the results of the experiments we performed. Finally, in Section 6 we highlight our conclusions and refer to future work.

## 2 Related Work

TCP's efficiency for streaming media delivery over satellite links has not been studied in depth. Proposed mechanisms range from minor tweaks, such as issuing delayed ACKs in order to reduce the network load in the reverse path, to sophisticated solutions, such as *TCP Spoofing* and split-connection protocols [3]. With the absence of a dedicated and efficient end-to-end solution, most research approaches commonly choose to tune an existing protocol in order to achieve the desired performance. Most related research efforts focus on bulk-data transmission over satellite IP networks and study the associated TCP performance [8, 15, 21, 2]. *TCP-Peach* [1] is a proposed congestion control scheme that explicitly addresses satellite IP networks. TCP-Peach incorporates two new algorithms, namely *Sudden Start* and *Rapid Recovery*, instead of the typical Slow-Start and Fast Recovery. Inline with the *Probing mechanism* and *Immediate Recovery* proposed in [17], these algorithms are based on the concept of using *dummy segments* to probe the availability of network resources without carrying any new information to the sender. The protocol achieves improved throughput performance; however, it does not account for the *Quality of Service* (*QoS*) provisions required by time-sensitive traffic.

TCP selective acknowledgments (*SACK*) options [11] were proposed in order to alleviate TCP's inefficiency in handling multiple drops in a single window. TCP SACK enables the receiver to inform the sender about segments that were received out of order. Hence, the sender avoids retransmitting segments whose successful delivery at the other end is not evident from the duplicate ACKs received. TCP SACK yields improved performance for a relatively large sending window. Furthermore, by reducing the rate of ACKs, remarkable gains can be attained in asymmetric links.

Several TCP protocol extensions have emerged to overcome the standard TCP limitations providing more efficient bandwidth utilization and sophisticated mechanisms for congestion control, which preserve the fundamental QoS guarantees for time-sensitive traffic. Authors in [5, 22] proposed a family of TCP compatible protocols, called *TCP-friendly*. TCP-friendly protocols achieve smooth window adjustments, while they manage to compete fairly with TCP flows. *TCP-friendly Rate Control* (*TFRC*) [5] is a representative TCP-friendly protocol, where its transmission rate is adjusted in response to the level of congestion, as indicated by the loss rate. Multiple packet losses in the same RTT are considered as a single loss event by TFRC and hence, the protocol follows a more gentle congestion control strategy. The

protocol eventually achieves the smoothing of the transmission gaps and therefore, is suitable for applications requiring a smooth sending rate, such as streaming media. However, this smoothness has a negative impact, as the protocol becomes less responsive to bandwidth availability [20].

TCP Westwood [10] is a TCP-friendly protocol that emerged as a sender-side-only modification of TCP Reno congestion control. TCP Westwood exploits end-to-end bandwidth estimation in order to adjust the values of ssthresh and cwnd after a congestion episode. The protocol incorporates a recovery mechanism which avoids the blind halving of the sending rate of TCP Reno after packet drops and enables TCP Westwood to achieve a high link-utilization in the presence of wireless errors. However, in [13] we showed that TCP Westwood tends to overestimate the available bandwidth, due to ACKs clustering. *TCP Westwood+* is a recent extension of TCP Westwood, based on the *Additive Increase/Adaptive Decrease* (*AIAD*) mechanism. TCP Westwood+ obtains more accurate estimates of the available bandwidth [7].

*TCP Real* is a high-throughput transport protocol that incorporates congestion avoidance mechanism in order to minimize transmission-rate gaps. As a result, this protocol is suited for real-time applications, since it enables better performance and reasonable playback timers. *TCP Real* [19] employs a receiver-oriented and measurement-based congestion control mechanism that significantly improves TCP performance over heterogeneous networks and asymmetric paths. The protocol approximates a receiver-oriented approach beyond the balancing trade of the parameters of additive increase and multiplicative decrease. In this context, TCP Real introduces another parameter, namely $\gamma$, which determines the window adjustments during congestion avoidance. More precisely, the receiver measures the data-receiving rate and attaches the result to its *ACKs*, directing the transmission rate of the sender. When new data is acknowledged and the congestion window is adjusted, the current data-receiving rate is compared against the previous one. If there is no receiving rate decrease, the congestion window is increased by 1 *Maximum Segment Size* (*MSS*) every RTT ($\alpha = 1$). If the magnitude of the decrease is small, the congestion window remains temporarily unaffected; otherwise, the sender reduces the congestion window multiplicatively by $\gamma$. In [19] a default value of $\gamma = 1/8$ is suggested. However, this parameter can be adaptive to the detected conditions. Generally, TCP Real can be viewed as a TCP ($\alpha$, $\beta$, $\gamma$) protocol, where $\gamma$ captures the protocol's behavior prior to congestion, when congestion boosts up.

Besides transport layer modifications, there are several techniques operating on the link layer, which attempt to ameliorate the impact of wireless errors [3]. *Forward Error Correction* (*FEC*) introduces added overhead to data bits in order to cope with data corruption. Corrupted packets may be directly corrected, without retransmission, which is critical for lossy links exhibiting long delays. *Automatic Repeat Request* (*ARQ*) mechanisms are invoked when packets containing bit errors can not be corrected. In this case, the erroneous packets are discarded and a retransmission is directly triggered within TCP's timeout.

## 3 Transmission Gap Analysis of Asymmetric Satellite Links

In this section, we formulate a model for transmission gaps that explicitly addresses asymmetric satellite links. The proposed model applies to bi-directional satellite systems which exhibit bandwidth asymmetry; hence, both forward and reverse path have the same propagation delay *P*. Based on [12], we define the transmission period *t(n)*, as the period between two consecutive transmissions (with individual window sizes). In this context, we model t(n) as a function of transmission number *n*. We also define *W(n)* as the number of data packets sent at the $n^{th}$ transmission. We assume that W(0) = 1 and W(n) inflates up to the maximum window size advertised by the receiver. The transmission time *T(n)* required for sending W(n) data packets is:

$$T(n) = \frac{S \cdot W(n)}{BW_{Dn}} \tag{1}$$

where *S* and $BW_{Dn}$ denote the fixed packet size (including TCP and IP headers) and the bandwidth of the downlink, respectively. After $n^{th}$ transmission, W(n) ACK packets are expected to reach the sender. Hence, the transmission time *T′(n)* required for sending W(n) ACKs is denoted by:

$$T'(n) = \frac{S' \cdot W(n)}{BW_{Up}} \tag{2}$$

where *S′* and $BW_{Up}$ are the ACK packet size and the bandwidth of the uplink, respectively. ACKs transmission time is not negligible despite their small packet size S′, since $BW_{Up}$ is constrained. Ignoring any processing and queuing delays and with respect to equations (1) and (2), we can approximate RTT from the 1st transmission period where W(0) = 1:

$$RTT_{init} = 2 \cdot P + T(0) + T'(0) \tag{3}$$

$$RTT_{init} = 2 \cdot P + \frac{S}{BW_{Dn}} + \frac{S'}{BW_{Up}} \tag{4}$$

Equation (4) reveals that in a GEO satellite system with a propagation delay P typically exceeding 200ms, retransmitting a lost video packet is unfruitful either by TCP or link layer mechanisms, such as ARQ.

Transmission period t(n) is eventually determined by the maximum value of $RTT_{init}$, transmission rate T(n) and ACK transmission rate T′(n):

$$t(n) = \max(RTT_{init},\ T(n),\ T'(n)) \tag{5}$$

In the case of a relatively small window size W(n), the system throughput does not reach the bandwidth of the downlink $BW_{Dn}$ and hence, t(n) is determined by the value of $RTT_{init}$. As a result, only minimal variations may be induced in the

transmission periods, since $RTT_{init}$ is basically defined by the link propagation delay P. In this case, transmission gaps are minimized achieving a smooth sending and playback rate. On the other hand, whenever throughput instantly approximates $BW_{Dn}$, transmission time T(n) is maximized and eventually designates the transmission period t(n). Let a transmission number k, where all the available network resources are allocated, and consequently packet drops occur. According to standard TCP, the window of the next transmission W(k+1)[1] will be halved and the associated transmission time T(k+1) is expressed as:

$$T(k+1) = \frac{S \cdot \frac{W(k)}{2}}{BW_{Dn}} \tag{6}$$

A similar outcome is reached in the situation of a link error, since TCP commonly invokes congestion-oriented responses and reduces its window. Under these conditions, apart from the impairments due to lost packets, the significant variations in the transmission periods induce gaps which further degrade media quality.

We additionally consider the implication where the sender does not receive a number of ACK packets, due to a constrained uplink bandwidth $BW_{Up}$ or heavy back traffic. In this case, ACK transmission time T´(n) exceeds both T(n) and $RTT_{init}$, and consequently defines the value of t(n). Similarly, transmission delay variations in the reverse path impact the associated transmission periods and diminish real-time application performance. However, although TCP manages to relinquish the resources allocated, when it detects congestion according to (6), it is not able to relieve the congestion in the reverse path. Along these lines, reaching $BW_{Up}$ capacity poses the highest threat on asymmetric links.

If we adopt the approach of delaying ACKs by a certain period *d*, we derive from (4) a modified $RTT_{init}$ formula:

$$RTT'_{init} = 2 \cdot P + \frac{S}{BW_{Dn}} + \frac{S'}{BW_{Up}} + d \tag{7}$$

Furthermore, let the receiver send *L* delayed ACK packets which correspond to the transmission of W(n) data packets. Consequently, ACK transmission time is now expressed as:

$$T''(n) = \frac{S' \cdot L}{BW_{Up}} \tag{8}$$

A considerable ACK delay d is translated to a minimal number of L ACK packets, which may render ACK transmission time T´´(n) negligible. From equations (7), (8), it is obvious that by issuing delayed ACKs, we effectively reduce back traffic in the expense of increasing RTT. With respect to (5), we reach the conclusion that delayed ACKs may degrade TCP performance in the case where t(n) = $RTT'_{init}$. That is, no

[1] We ignore the retransmission window. Consequently, after loss detection the TCP sender halves the congestion window at transmission number (k+1).

data/ACK congestion has occurred in the forward/reverse path. Inversely, gains are expected from delayed ACKs in the situation of congestion in the downlink channel (t(n) = T(n)), and especially during heavy back traffic, where t(n) = T´´(n) < T´(n).

## 4 Evaluation Methodology

The evaluation plan was implemented on the *NS-2* network simulator. LEO systems with RTTs in the range of 40-200ms cause slight degradation in TCP performance, despite the large RTT variations [8]. However, due to large RTTs (approximately 530ms), maintaining efficient TCP performance over GEO latencies is challenging. Along these lines, we focus on quantifying the effects of GEO systems on TCP efficiency and streaming video delivery. We simulated the system in Fig. 1, where *N* senders transmit an MPEG-4 video stream to *N* receivers through a bi-directional GEO satellite link with 5 Mbps downlink and 256 Kbps uplink channel. We consider the modeled satellite system, as a retransmitter of data traffic (received from a terrestrial gateway) to ground gateways and user terminals. The transmitted video streams are multiplexed in Station 1, before traversing the satellite link. In accordance with the lossy nature of satellite links, we simulated an error model for both forward and reverse satellite channels with configurable bit error rate (*BER*). BER is adjusted at $10^{-4}$, unless otherwise explicitly stated.

We assume a window scale option which overcomes the limitation of the maximum window size (i.e. 64 KB) allowed by standard TCP. Hence, we adjusted the maximum window size at 240 KB. Segment size is set to 1000 bytes and consequently, a window may accommodate at the most 240 segments approximately. Since the simulated network exhibits an average RTT of 550ms, simulation running time was fixed to 200 seconds, an appropriate time-period for all the protocols to demonstrate their potential. We performed the experiments over standard TCP Reno, the modified TCP Reno variant [6], known as NewReno, augmented with the SACK [11] option, and the protocols TCP Westwood+ and TCP-Real. Concerning the relaxed packet loss requirements of time-sensitive applications, as well as the implications that may be induced by FEC/ARQ [3] in order to maximize reliability, we chose not to include such mechanisms in our experiments.

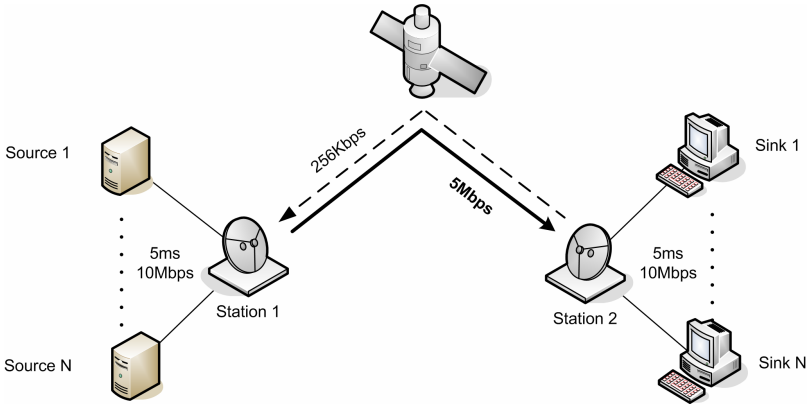


**Fig. 1.** Simulation topology

In order to simulate real-time traffic, we developed an *MPEG-4 Traffic Generator*. The traffic generated closely matches the statistical characteristics of an original video trace. We used three separate *Transform Expand Sample* (*TES*) models for modeling I, P and B frames, respectively. The resulting MPEG-4 stream is generated by interleaving data obtained by the three models. The MPEG traffic generator was integrated into NS-2 and provides the adjustment of the data rate of the MPEG stream, as well as useful statistical data (e.g. average bit-rate, bit-rate variance).

We hereby refer to the performance metrics supported by our simulation model. System goodput is used to measure the overall system efficiency in bandwidth utilization. In [14] we proposed a new metric for the performance evaluation of time-sensitive traffic, called *Real-Time Performance*. The metric monitors packet inter-arrival times and distinguishes the packets that can be effectively used by the client application from delayed packets (according to a configurable inter-arrival threshold). The proportion of the delayed packets is reflected in *Delayed Packets Rate*. Hence, *Real-Time Performance* index is defined as the ratio of the number of *timely received packets* over the total number of packets sent by the application:

$$\text{Real}-\text{Time Performanœ Index} = \frac{\#\text{timely received packets}}{\#\text{sent packets}} \leq 1$$

In accordance with video streaming requirements, we adjusted the inter-arrival threshold at 200ms. Since MPEG traffic is sensitive to packet drops, we additionally define *Packet Drop Rate*, as the ratio of the number of lost packets over the number of packets sent by the application. Most of our experiments were performed on several flows, so we present the average of the real-time performance of each MPEG flow.

## 5 Results and Discussion

In the sequel, we demonstrate and comment on the most prominent results from the experiments we performed based on three distinct scenarios. The basic parameters of each simulation scenario are as described in the previous section.

### 5.1 TCP Performance

Initially we performed a series of experiments in order to evaluate the video performance delivered by the selected TCP variants. We simulated a wide range of MPEG flows (1-50) adjusting the contention accordingly. We measured *goodput* and *real-time performance*, and we additionally selected statistics from delayed and lost packets, since both are influencing factors which impact video quality. We hereby demonstrate the associated results of TCP Reno, TCP NewReno with SACK, TCP Westwood+ and TCP Real (Figs. 2-5).

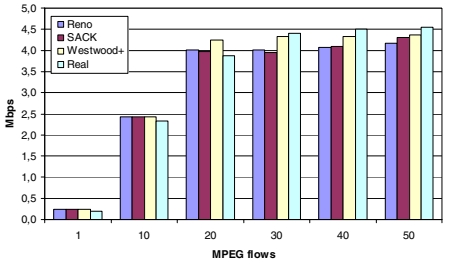


**Fig. 2.** System goodput

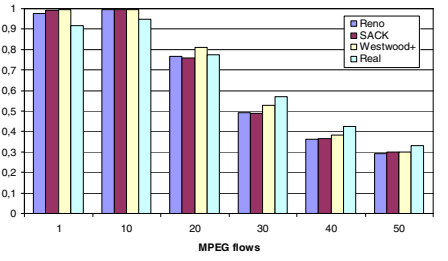


**Fig. 3.** Average Real-Time Performance

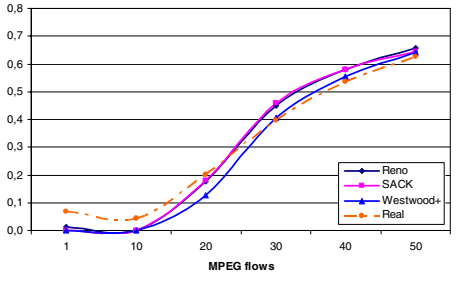


**Fig. 4.** Packet Drop Rate

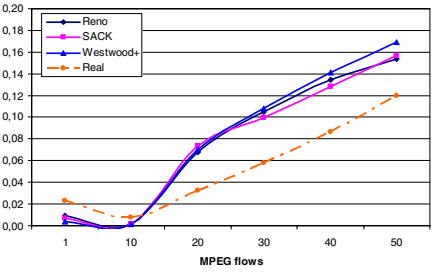


**Fig. 5.** Delayed Packets Rate

With the exception of TCP Real (and in part of Westwood+), the protocols are unable to sustain goodput rates close to the bottleneck link rate (Fig. 2), despite the relatively large window (i.e. 240 KB). Therefore, the available bandwidth is not fully exploited, mainly due to link asymmetry and long latency. Inline with our analysis in Section 3, heavy ACK traffic across the constrained uplink channel extends the transmission periods ($t(n) = T'(n)$) and inevitably induces variable transmission gaps, which impair the performance of video delivery (Fig. 3).

A comparison between standard TCP Reno and TCP NewReno (with SACK) reveals that SACK alone is not sufficient to enable high performance (Figs. 2, 3). However, slight gains (especially for high contention) are eventually attained, since NewReno prevents coarse timeouts and multiple window reductions, while SACK accelerates the loss recovery phase. Both TCP Reno and TCP NewReno are based on "blind" increase/decrease window mechanisms that dynamically exploit bandwidth availability, without relying on precise measurements of current conditions. Furthermore, they invoke unnecessary congestion-oriented responses to the increased bit errors along the satellite link. Along these lines, they exhibit limited efficiency in the context of real-time application performance (Fig. 3), primarily due to the significant delays in video-data delivery. Fig. 5 illustrates that a notable proportion of packets reach the recipient exceeding the delay requirements of streaming video both for Reno and NewReno.

On the other hand, TCP Westwood+ and TCP Real rely on bandwidth estimation schemes and are intended to sustain a smooth sending rate minimizing the transmission gaps. TCP Westwood+, in contradiction to the initial version of

Westwood, computes one sample of available bandwidth every RTT using all data acked in the specific RTT, therefore obtaining more accurate estimates (Fig. 2). However, from the perspective of real-time delivery, Westwood+ efficiency is not so profound (Fig. 3), since it delivers a considerable amount of delayed packets (Fig. 4). Inline with our analysis, reaching the downlink capacity (i.e. flows 30-50) maximizes transmission time T(n) and generates variable transmission periods which impact video delivery.

Unlike Westwood, TCP Real yields satisfactory performance on video delivery for a wider range of flows (Fig. 3), which is the combined result of high goodput rates (Fig. 2) and a gentle proportion of delayed packets (Fig. 5). TCP Real effectively manages to amortize the low throughput of the initial window built across a longer period of high throughput. Furthermore, the protocol exploits the integrated error detection mechanism, as well as the additional parameter $\gamma$. In this context, the desired smoothness is counterbalanced with responsiveness, which is critical during congestion episodes.

### 5.2 TCP Performance vs. Error Rates

In this scenario, we performed our experiments using diverse bit error rate adjustments (BER: $10^{-6}$ - $10^{-3}$). In satellite networks, BER scarcely exceeds $10^{-4}$. However, we simulated a satellite link with BER as high as $10^{-3}$ in order to study protocol efficiency under error-prone connections. We hereby demonstrate results from 20 MPEG flows in order to investigate the associated impact on the performance of video delivery (Figs. 6-9).

TCP Westwood+ is the less sensitive protocol to the diverse bit error rates (Fig. 8), although it does not incorporate an inherent mechanism for error detection. Furthermore, it maintains an acceptable delayed packets rate for intensely error-prone satellite links (Fig. 9). On the contrary, TCP-Real demonstrates limited efficiency at high bit error rates, despite the incorporated error classification mechanism. Apparently, the mechanism operates inadequately for increased link errors. SACK's supportive role results in perceptible performance gains, since the loss recovery phase is accelerated and the packet drop rate is sustained to slightly lower levels (Fig. 8). However, inline with Reno, NewReno with SACK is still inefficient for excessively lossy links.

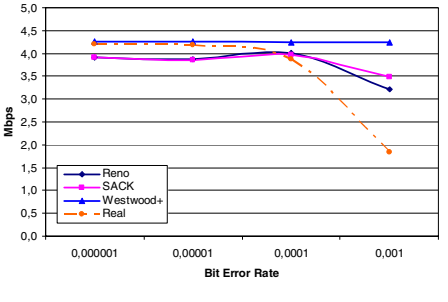


**Fig. 6.** System Goodput (20 flows)

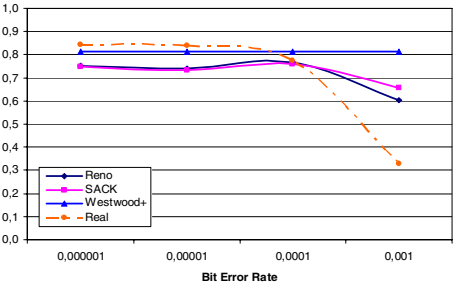


**Fig. 7.** Average Real-Time Performance (20 flows)

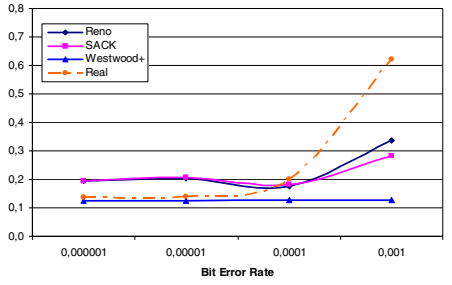


**Fig. 8.** Packet Drop Rate (20 flows)

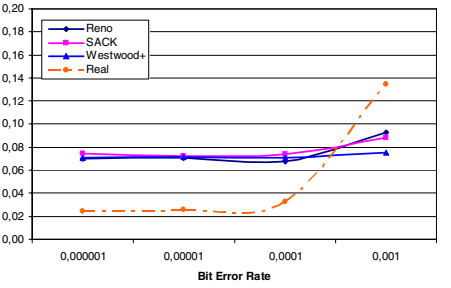


**Fig. 9.** Delayed Packets Rate (20 flows)

### 5.3 Impact of Delayed ACKs

We conclude our evaluation scenarios by studying the impact of delayed ACKs on protocol efficiency in satellite environments. We also investigate whether reducing back traffic induces implications which affect the performance of video delivery. We performed our experiments issuing ACKs with no delay and with delays of 100ms, 200ms and 500ms, successively. In the sequel, we discuss the behavior of TCP NewReno with SACK which produced the most conclusive results (Figs. 10-13).

Although delayed ACKs tend to slow down the initial slow-start phase (due to the decreased number of ACKs sent by a delayed-ACK receiver), our results illustrate noticeable performance gains both for TCP efficiency (Fig. 10) and video quality (Fig. 11). The reported gains are attained in the situation where goodput reaches the capacity of downlink channel (Fig. 10: flows 20-50), and video transmission time is maximized according to our analysis. The delayed ACKs effectively reduce the traffic in the reverse path, which is a critical factor in asymmetric links. The beneficial role of delayed ACKs is illustrated in Fig. 13, where the number of delayed packets is slightly decreased (100ms and 500ms delack). However, we observe that issuing ACKs with delays more than 100ms does not result in perceptible performance gains (Figs. 10, 11), since the increased RTTs (as derived from equation (7)) counterbalance the benefits from the reduced ACK transmission time T´´(n) (equation (8)).

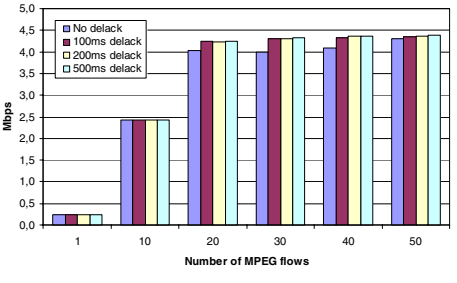


**Fig. 10**. System Goodput (NewReno-SACK)

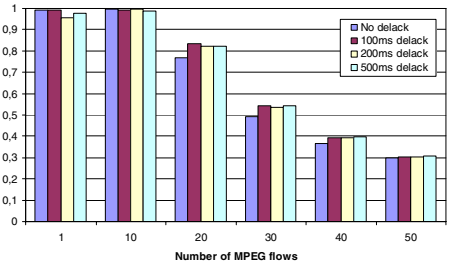


**Fig. 11**. Average Real-Time Performance (NewReno-SACK)

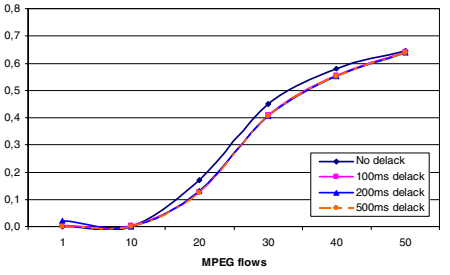


**Fig. 12.** Packet Drop Rate (NewReno-SACK)

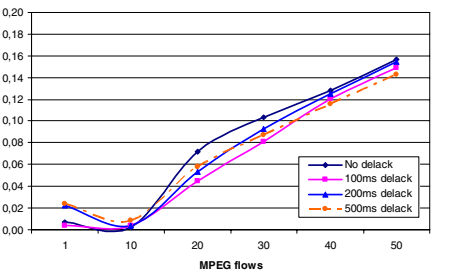


**Fig. 13.** Delayed Packets Rate (NewReno-SACK)

## 6 Conclusions and Future Work

We demonstrated the challenges and limitations of TCP from the perspective of real-time performance over asymmetric satellite links. We identified that protocol efficiency over such environments is strictly related with effective bandwidth utilization and minimized transmission gaps. Focusing on the study of GEO systems where RTTs exhibit insignificant variations, transmission gaps are primarily induced by reaching downlink capacity, and especially by congested back traffic across a constrained reverse path. We also showed that issuing delayed ACKs occasionally results in performance gains.

The algorithms of TCP Westwood+ and TCP-Real do not always obtain accurate estimates, occasionally failing to achieve full utilization of asymmetric link capacities. However, they are more effective than "blind" increase/decrease window mechanisms (e.g. TCP Reno), which rely on specific events triggered by violated thresholds. TCP-Real, in particular, yields satisfactory performance regardless of link multiplexing; only link errors with BER in excess of $10^{-4}$ degrade its performance and the perceived video quality. However, error rates of this magnitude are uncommon in modern satellite systems. The investigation of additional protocols efficiency (e.g. SCTP), as well as alternative satellite systems, such as *Demand Assigned Multiple Access* (*DAMA*) satellite services, is under way.

## References

1. I. F. Akyildiz, G. Morabito and S. Palazzo, TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks, IEEE Transactions on Networking, 9(3), pp. 307-321, June 2001
2. H. Balakrishnan, V. Padmanabhan, G. Fairhurst and M. Sooriyabandara, TCP Performance Implications of Network Path Asymmetry, RFC 3449, December 2002
3. H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, A Comparison of Mechanisms for Improving TCP Performance over Wireless Links, ACM/IEEE Transactions on Networking, 5(6), pp. 756-769, 1997
4. D. Chiu and R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks, Journal of Computer Networks, 17(1), pp. 1-14, 1989

5. S. Floyd, M. Handley, J. Padhye, and J. Widmer, Equation-Based Congestion Control for Unicast Applications, In Proc. of ACM SIGCOMM 2000, Stockholm, Sweden, August 2000
6. S. Floyd and T. Henderson, The NewReno Modification to TCP's Fast Recovery Algorithm, Internet RFC 2582, 1999
7. L. Grieco and S. Mascolo, Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control, ACM Computer Communication Review, 34(2), pp. 25-38, April 2004
8. T. R. Henderson and R. H. Katz, Transport protocols for Internet-compatible satellite networks, IEEE Journal of Selected Areas in Communications (JSAC), Vol. 17, pp. 326-344, Feb. 1999
9. V. Jacobson, Congestion avoidance and control, In Proc. of ACM SIGCOMM '88, Stanford, USA, August 1988
10. S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links, In Proc. of MobiCom '01, Rome, Italy, July 2001
11. M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, TCP Selective Acknowledgment Options, RFC 2018, October 1996
12. H. Obata, K. Ishida, J. Funasaka and K. Amano, TCP Performance Analysis on Asymmetric Networks Composed of Satellite and Terrestrial Links, In Proc. of 8th Int/nal Conference on Network Protocols (ICNP), Osaka, Japan, November 2000
13. P. Papadimitriou and V. Tsaoussidis, Assessment of Internet Voice Transport with TCP, To appear in Int/nal Journal of Communication Systems (IJCS), Wiley Academics
14. P. Papadimitriou and V. Tsaoussidis, On Transport Layer Mechanisms for Real-Time QoS, Journal of Mobile Multimedia (JMM), 1(4), pp. 342-363, January 2006
15. C. Partridge and T. J. Shepard, TCP/IP Performance over Satellite Links, IEEE Network, 11(5), pp. 44-49, September-October 1997
16. W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, January 1997
17. V. Tsaoussidis, H. Badr, TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains, In Proc. of 8th Int/nal Conference on Network Protocols (ICNP), Osaka, Japan, November 2000
18. V. Tsaoussidis and I. Matta, Open issues on TCP for Mobile Computing, Journal of Wireless Communications and Mobile Computing, 2(1), pp. 3-20, February 2002
19. V. Tsaoussidis and C. Zhang, TCP Real: Receiver-oriented congestion control, Computer Networks, 40(4), pp. 477-497, November 2002
20. V. Tsaoussidis and C. Zhang, The dynamics of responsiveness and smoothness in heterogeneous networks, IEEE Journal on Selected Areas in Communications, 23(6), pp. 1178-1189, June 2005
21. L. Wood, G. Pavlou and B. Evans, Effects on TCP of Routing Strategies in Satellite Constellations, IEEE Communications Magazine, 39(3), pp. 172-181, March 2001
22. Y. R. Yang and S. S. Lam, General AIMD Congestion Control, In Proc. of 8th Int/nal Conference on Network Protocols (ICNP), Osaka, Japan, November 2000