Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 1451–1465
computer communications
www.elsevier.com/locate/comcom

On TCP performance over asymmetric satellite links with real-time constraints

Panagiotis Papadimitriou *, Vassilis Tsaoussidis

Demokritos University, Electrical & Computer Engineering Department, 12 Vas. Sofias Street, Xanthi 67100, Greece

Available online 20 January 2007

Abstract

Real-time transmission over asymmetric satellite IP links is challenging, since satellite systems commonly exhibit long propagation delays, while bandwidth asymmetry often enforces a variable and infrequent rate of acknowledgment packets (*ACKs*) across the upstream channel with several undesirable implications. In this context, we formulate an analytical model in order to quantify the impact of satellite systems and link asymmetry on TCP performance and real-time delivery. We emphasize on the effects of asymmetric links, and especially on the implications that cause interruptions in the sending rate, and eventually disturb smooth delivery. Since TCP performance is in part throttled by the rate of arriving ACKs, we additionally investigate the impact of delayed ACKs. Although delayed ACKs occasionally diminish the transmission rate, we uncover notable gains in terms of smoothness and real-time delivery. Furthermore, we demonstrate conclusive performance studies tackling the supportive role of selective acknowledgments (*SACK*) and the effect of varying bit error rates. Our simulation results illustrate that most existing end-to-end solutions do not comply with the stringent *Quality of Service* (*QoS*) provisions of time-sensitive applications, resulting in ineffective bandwidth utilization and varying delays in data delivery. Finally, with the absence of a satellite-optimized TCP implementation for real-time transmission, we identify the most prominent end-to-end solutions that manage to alleviate most of the impairments induced by asymmetric satellite links, sustaining a relatively smooth transmission rate.

Keywords: TCP; QoS; Satellite networks; Performance evaluation

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. However, satellite networks demonstrate certain limitations. First, 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 higher 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 increasing the overhead, as well as end-to-end delays.

Most Internet transport protocols exhibit limited efficiency under these awkward conditions. *Transmission*

* Corresponding author. Tel.: +30 25410 84382.
E-mail addresses: ppapadim@ee.duth.gr (P. Papadimitriou), vtsaousi@ee.duth.gr (V. Tsaoussidis).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.12.030

Control Protocol (*TCP*), based on the principles of congestion management [1], *Slow-Start* [2], and *Additive Increase Multiplicative Decrease* (*AIMD*) [3], 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 IP networks. Authors in [4] 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.

TCP performance is also affected by bandwidth asymmetry [5]. Despite the small size of acknowledgment packets (*ACKs*), the reverse channel is often unable to carry their high rate. When ACKs arrive at the upstream bottleneck link at a higher rate than the link can support, a queue is built up. In the presence of a large sending window, if buffering in the uplink is not enough to accommodate incoming ACKs, some of them are dropped due to buffer overflow. Since most TCP implementations inflate their cwnd in response to the number of ACKs they receive, a possible infrequency of ACKs degrades the sending rate. In addition, the loss of ACKs renders TCP's *Fast Retransmit* and *Fast Recovery* [6] algorithms less effective, since the sender may not receive the threshold number (commonly 3) of duplicate ACKs (*DACKs*). Therefore, congestion in the reverse path may result in coarse timeouts and multiple window reductions, while *Round Trip Times* (*RTTs*) are often increased, diminishing the protocol efficiency.

A part 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 [7–9] have emerged to overcome the standard TCP limitations providing more effective 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 real-time applications 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 timely delivery over such environments. Our study builds on and extends the analysis and the experimental results of [10]. Based on a comprehensive analytical approach, as well as extensive simulations, we investigate the performance issues that arise in real-time streaming with window-based congestion control mechanisms. We emphasize on link asymmetry and the potential implications on TCP performance, due to the imperfection and variability in the ACK feedback. In this context, we uncover undesirable effects on standard TCP, which degrade the performance of real-time delivery in several occasions. Quantifying the efficiency of a solution-framework based on selected TCP implementations, we eventually identify the most prominent end-to-end mechanisms that manage to alleviate most of the impairments induced by satellite links and bandwidth asymmetry. 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 [11] we have shown that UDP may perform worse than TCP in several occasions. In addition, the absence of congestion control for UDP poses a threat to network stability.

The rest of the paper is organized as follows. Section 2 summarizes related work and provides an overview of TCP performance issues and selected proposals for transmission over satellite links. In Section 3 we formulate an analytical model in order to quantify the effects of satellite systems and link asymmetry on TCP performance and real-time delivery. Section 4 includes the parameters of our evaluation methodology, followed by Section 5, where we demonstrate conclusive performance studies based on extensive simulations. Finally, in Section 6 we highlight our conclusions.

2. An overview of related work and TCP performance issues

TCP's efficiency for real-time delivery over satellite links has not been studied in depth. Most related research efforts focus on bulk-data transmission over satellite IP networks and study the associated TCP performance [5,12–14]. On the other hand, the literature includes several TCP enhancements, which exhibit improved real-time performance [7–9]. However, such end-to-end solutions do not address the particular characteristics of satellite links. In

the sequel, we provide a taxonomy of the most prominent approaches, discussing separately TCP issues in satellite links, enhanced TCP versions with real-time capabilities, and selected mechanisms operating on the link-layer.

2.1. TCP issues in satellite networks

Departing from TCP performance issues during Slow-Start, the associated algorithm may take a long time to allocate a considerable amount of network resources. More precisely, according to [13] the time t required by Slow-Start in order to reach a data rate D is:

$$t = \mathrm{RTT}(1 + \log_2(D * \mathrm{RTT}/S))$$

where S is the average packet size. Concerning a GEO satellite link with a typical RTT of 550 ms, several RTTs are needed to finish startup. Numerous proposals address the limitation of utilizing inadequate resources during the Slow-Start phase. According to a technique, namely TCP Spoofing [15], a router located near the source transmits ACK packets to the sender. Consequently, the sending window is growing rapidly and performance is improved, since the duration of Slow-Start is significantly reduced. However, TCP Spoofing operates efficiently only under certain provisions, which may not be satisfied within the context of "real Internet". That is, ACKs are required to traverse the same path with data. Furthermore, dynamic routing changes or crashing routers may lead to loss of data.

An alternative approach is split connection protocols, such as *Indirect-TCP* (*I-TCP*) [16]. A split connection protocol virtually splits a TCP connection into multiple separate connections. However, these protocols do not handle handoff operations efficiently [15], since handoff procedures tend to be slow and complicated. Furthermore, due to splitting, end-to-end semantics of TCP is violated. Additional proposals include an increased initial congestion window and *Fast Start*. According to the former approach, the congestion window (*cwnd*) is initially adjusted: $1 < \mathrm{cwnd} < 4$. On the other hand, Fast Start [17] provides an alternative to Slow-Start by exploiting the most recent transmission rate and reusing it for a consequent transfer. However, such an assumption is more suitable for short flows (e.g. Web transfers) and may eventually lead to congestion, if the last recorded rate is too high for current network conditions.

TCP-Peach [18] 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 conventional Slow-Start and Fast Recovery. Inline with the *Probing mechanism* and *Immediate Recovery* proposed in [19], 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. Dummy segments are treated as low-priority segments (priority assignment at the IP layer is therefore required) in order to avoid implications with actual data traffic. The main advantage of TCP-Peach is the compatibility with traditional TCP implementations, since it merely includes modifications in the end-user behaviors. The protocol achieves improved goodput performance; however, it is basically intended for bulk-data transfers. Therefore, it does not account for the *Quality of Service* (*QoS*) provisions required by time-sensitive traffic.

TCP selective acknowledgments (*SACK*) options [20] were proposed in order to alleviate TCP's inefficiency in handling multiple drops in a single window. The use of selective acknowledgements is optional, and during the connection setup is determined whether SACK would be supported. 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 DACKs received. TCP SACK yields improved performance for a relatively large sending window. Furthermore, by reducing the rate of ACKs, notable gains may be attained in asymmetric links.

2.2. Transport layer enhancements with real-time capabilities

Several TCP protocol extensions have emerged in order to overcome the standard TCP limitations providing more effective bandwidth utilization and sophisticated mechanisms for congestion control, which preserve the fundamental QoS guarantees for time-sensitive traffic. Authors in [7–9] proposed a family of TCP compatible protocols, called *TCP-friendly*. Generally, we consider TCP-friendly any protocol whose long-term arrival rate does not exceed the one of any conformant TCP in the same circumstances [21]. TCP-friendly congestion control has the ability to maintain network stability by promptly responding to congestion and to be cooperative with other flows, while it commonly provides more efficient QoS, (e.g. a smoothed sending rate and bounded latency for playback multimedia applications). The differences between standard TCP and TCP-friendly congestion control lie mainly in the specific values of additive increase rate α and multiplicative decrease ratio β, while their similarities in their AIMD-based congestion control (a characteristic that enables us to include them both in the family of TCP (α, β) protocols). Standard TCP is therefore viewed as a specific case of TCP (α, β) with $\alpha = 1$ and $\beta = 0.5$. On the other hand, TCP-friendly protocols are designed to satisfy the requirements of time-sensitive applications. However, they may exhibit further weaknesses, when bandwidth becomes available rapidly [22]. *GAIMD* [9] is a TCP-friendly protocol that generalizes AIMD congestion control by parameterizing α and β. For the family of TCP protocols, authors in [9] derives a simple relationship between α and β in order to be friendly to standard TCP. Based on experiments, they propose an adjustment of $\beta = 0.875$ as an appropriate smooth decrease ratio, and a moderated increase value $\alpha = 0.31$ to achieve TCP friendliness.

```
if an ACK is received
    sample_BWE[n] = (acked * pkt_size * 8) / (now - last_ACK_time)
    BWE[n] = (1 - beta) * (sample_BWE[n] + sample_BWE[n - 1]) / 2 + beta * BWE[n - 1]
end if
```

Fig. 1. TCP Westwood bandwidth estimation algorithm.

TCP Westwood [23] is a sender-side-only modification of TCP Reno congestion control, which exploits end-to-end bandwidth estimation to properly set the values of slow-start threshold and congestion window after a congestion episode. TCP Westwood significantly improves fair sharing of high-speed networks capacity. The protocol incorporates a recovery mechanism which avoids the blind halving of the sending rate of TCP Reno after packet losses, and enables TCP Westwood to achieve a high link-utilization in the presence of wireless errors. The specific mechanism considers the sequence of bandwidth samples *sample_BWE[n]* obtained using the ACK arrivals and evaluates a smoothed value, *BWE[n]*, by low-pass filtering the sequence of samples, as described by the pseudocode in Fig. 1, where *acked* is the number of segments acknowledged by the last *ACK*; *pkt_size* is the segment size in bytes; *now* is the current time; *last_ACK_time* is the time the previous *ACK* was received; *beta* is the pole used for the filtering (a value of 19/21 is suggested). However, in [24] 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. Unlike the initial version of Westwood, TCP Westwood+ computes one sample of available bandwidth every RTT using all data acknowledged in the specific RTT, therefore obtaining more accurate estimates [25].

TCP Real [26,27] is a high-throughput transport protocol that incorporates congestion avoidance mechanism in order to minimize transmission-rate gaps. As a result, the protocol is suited for real-time applications, since it enables better performance and reasonable playback timers. TCP Real 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 γ, 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 γ. In [26] 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 (α, β, γ) protocol, where γ captures the protocol's behavior prior to congestion, when congestion boosts up.

Although we explicitly study and analyze the behavior of window-based mechanisms over asymmetric satellite links, we briefly refer to selected rate-based protocols, which compose an elegant framework for multimedia applications. *TCP-friendly Rate Control* (*TFRC*) [7] is a representative TCP-friendly, rate-based congestion control protocol. According to TFRC, the transmission rate is adjusted in response to the level of congestion, as it is indicated by the loss rate. Unlike standard TCP, the instantaneous throughput of TFRC has a much lower variation over time, and consequently, only smooth adjustments are needed. Furthermore, 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. TFRC 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 [22]. *Scalable Streaming Video Protocol* (*SSVP*) [28] is a new congestion control scheme, which operates on top of UDP and is optimized for unicast video streaming applications. The transmission rate is controlled in a TCP-friendly fashion by properly adjusting the inter-packet-gap, spacing outgoing packets evenly to produce a smoothed flow. SSVP eventually adapts to the vagaries of the network and provides efficient QoS provisioning for video streaming applications.

2.3. Link-layer enhancements

Besides transport-layer modifications, there are several techniques operating on the link-layer, which attempt to ameliorate the impact of wireless errors [15]. The most remarkable implementations, which provide error-correction, are *Forward Error Correction* (*FEC*) and *Automatic Repeat Request* (*ARQ*) [29]. FEC introduces added overhead to data bits in order to cope with data corruption. Corrupted packets are directly corrected, without retransmission, which is critical for lossy links exhibiting long delays. In addition, FEC does not interfere with TCP mechanisms. However, the redundant information is not exploited in the absence of link errors resulting in a waste of bandwidth. Furthermore, FEC requires additional

resources in CPU processing time, memory and power consumption.

On the other hand, ARQ mechanisms are invoked when packets that contain bit errors cannot be corrected. In this case, the erroneous packets are discarded and a retransmission is directly triggered. Unlike FEC, ARQ allocates additional network resources only when a packet is retransmitted. The mechanism generally operates more efficiently for low bit rates. An undesirable effect of ARQ is that it may interfere with TCP [15]. Concerning the relaxed packet loss requirements of time-sensitive applications, as well as the implications that may be induced by FEC/ARQ in order to maximize reliability, we chose not to include these mechanisms in our performance studies.

3. TCP performance analysis over asymmetric satellite links with real-time constraints

Departing from the analytical approach found in [10], we formulate a model in order to quantify the effects of satellite systems and link asymmetry on TCP performance, and especially on real-time delivery. The proposed model applies to bi-directional satellite systems that exhibit bandwidth asymmetry; hence, both forward and reverse paths have the same propagation delay P. 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 nth transmission, as shown in Fig. 2. We assume that $W(0) = 1$ and $W(n)$ inflates up to the maximum window size (MWS) (provided that the downstream link is not saturated). Thus

$$1 \leqslant W(n) \leqslant \frac{\mathrm{MWS}}{S}$$

where S is the fixed packet size (including TCP and IP headers). The window transmission time $T(n)$ required for sending $W(n)$ data packets is

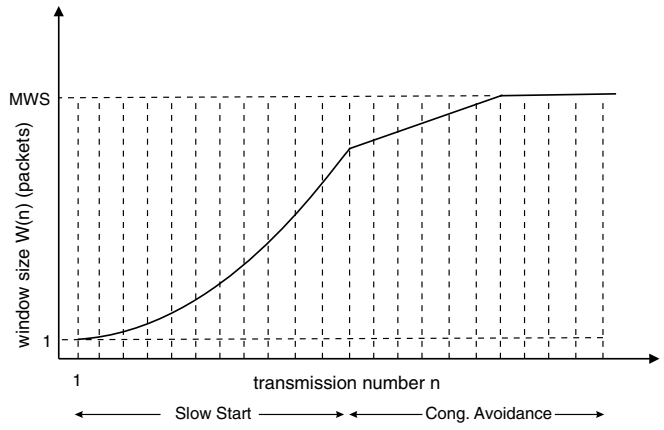

Fig. 2. The evolution of window size vs. number of transmissions.

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

where $\mathrm{BW_{Dn}}$ denotes the bandwidth of the downlink channel.

TCP receivers typically generate an ACK for each incoming data packet, and consequently after nth transmission, $W(n)$ ACK packets are expected to reach the sender. Hence, the ACK transmission time $T(n)$ required for sending $W(n)$ ACKs is denoted by

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

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

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

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

Considering real-time traffic where data packets bear information with a limited useful lifetime, retransmissions are often a wasted effort. Eq. (4) reveals that in a GEO satellite system with a propagation delay P commonly exceeding 400 ms, retransmitting a lost video packet is unfruitful either with TCP or link-layer mechanisms, such as ARQ.

Transmission period $t(n)$ is eventually determined by the maximum value of RTT, window transmission time $T'(n)$ and ACK transmission time $T'(n)$

$$t(n) = \max(\mathrm{RTT}, T(n), T'(n)) \tag{5}$$

In the case of a relatively small window size $W(n)$, the system throughput does not reach the bandwidth of the downlink $\mathrm{BW_{Dn}}$ and hence, $t(n)$ is determined by the value of RTT (Fig. 3a). As a result, only minimal variations may be induced in the transmission periods, since RTT 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 window size approaches MWS, throughput may instantly approximate $\mathrm{BW_{Dn}}$. In this case, window transmission time $T(n)$ is maximized and eventually designates the transmission period $t(n)$ (Fig. 3b). Let a transmission number k, where all the available network resources are allocated, and consequently packet drops occur. If we assume that both upstream and downstream links are not heavily congested and the sender is able to receive 3 DACKs in response to the packet loss in the forward path, *Fast Retransmit* and *Fast Recovery* are triggered. Hence, the window of the next transmission

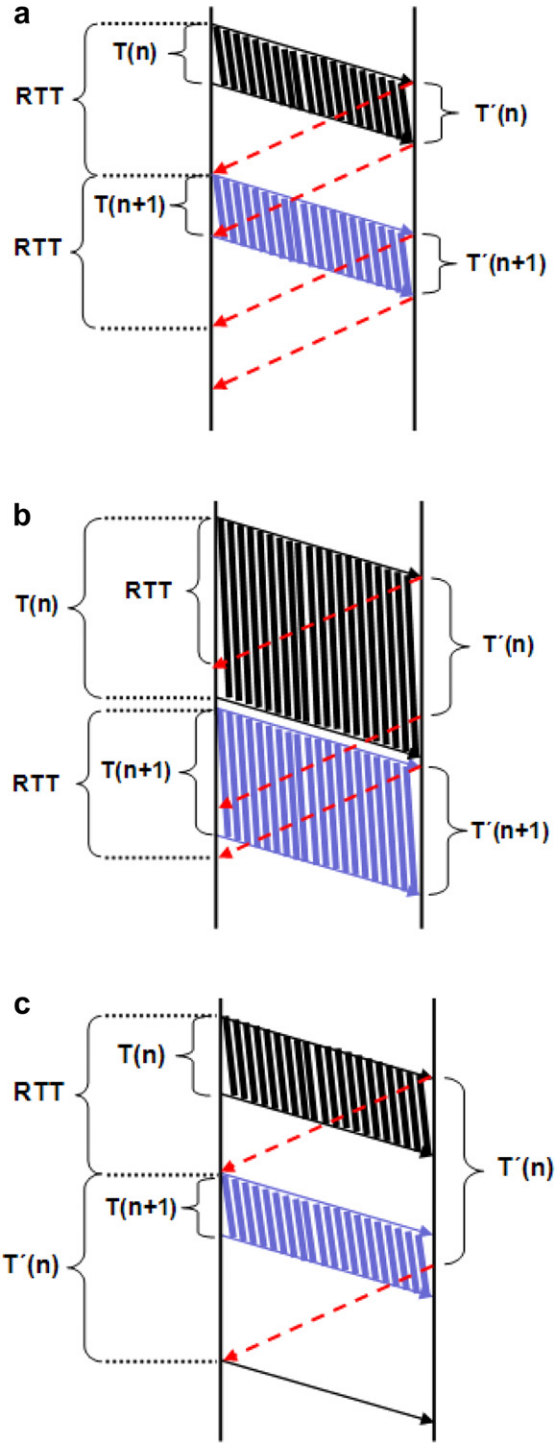

Fig. 3. Transmission period model. (a) RTT is maximized. (b) $T(n)$ is maximized. (c) $T'(n)$ is maximized.

$W(k+1)$ will be $\beta^* W(k)$ and the associated transmission time $T(k+1)$ is expressed as

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

where β is the protocol decrease ratio. A similar outcome is reached in the situation of a link error, since TCP commonly invokes congestion-oriented responses and reduces its window, even if the downstream link is not saturated. Under these conditions, apart from the impairments due to lost packets, the significant variations in the transmission periods induce gaps which further degrade the perceived quality (since $t(n) = T(n)$).

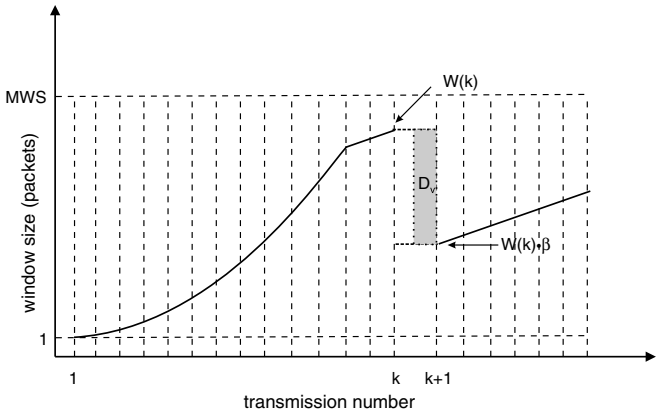

Fig. 4. The effect of delay due to coding rate reduction.

If the sender has not received 3 DACKs, a timeout event is triggered, followed by an abrupt window reduction which diminishes the sending rate and may cause a noticeable interruption in the transmitted stream. In the presence of a scalable source coding scheme, such interruptions may be more evident and frustrate the end user. More precisely, assuming a packet drop during transmission number k, the sending rate is decreased, and immediately the video coder is notified to reduce the video coding rate. This process (i.e. coding rate reduction) inevitably incurs an additional delay D_V. Therefore, the transition from transmission number k to $k+1$ will include an idle time-period resulting in a perceptible transmission gap (Fig. 4).

We additionally consider the implication where the sender does not receive a number of ACK packets, due to a constrained uplink bandwidth $\mathrm{BW_{Up}}$ or heavy back traffic. In this case, ACK transmission time $T'(n)$ exceeds both $T(n)$ and RTT, and consequently, defines the value of $t(n)$ (Fig. 3c). 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 Eq. (6), it is not able to relieve the congestion in the reverse path.[1] Even if the upstream link has deep queues, the reverse channel will get saturated before the downstream link, degrading throughput performance in the forward direction. More precisely, the ACKs generated in response to receiving data packets reflect the temporal spacing of these data packets all the way back to the sender, enabling it to transmit new packets that maintain the same spacing [1].

[1] *ACK Congestion Control* (*ACC*) is an experimental technique that requires modifications both in the network (extension to ECN) and the TCP receiver. Current TCP versions do not employ ACC.

However, the limited upstream capacity and queuing at the upstream bottleneck router alters the inter-ACK spacing of the reverse path, and hence that observed at the sender. When ACKs arrive at the upstream bottleneck link at a higher rate than the link can support, the spacing between them, when they emerge from the link, is dilated enforcing the TCP sender to clock out new data packets at a slower rate. Therefore, the performance of the TCP connection is no longer dependent on the downstream bottleneck link alone; instead, it is throttled by the rate of arriving ACKs. As a result, the rate of *cwnd* growth slows down, while certain TCP variants that dynamically exploit bandwidth availability by measuring the rate of incoming ACKs (i.e. TCP Westwood) may achieve inadequate bandwidth utilization. In summary, reaching BW_{Up} capacity poses the highest threat on asymmetric links.

When ACKs are delayed, the receiver must generate an ACK within a certain period d (usually $d \leqslant 500$ ms). If we adopt the approach of delaying ACKs by a certain period d, we derive from Eq. (4) a modified RTT formula

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

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

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

A considerable ACK delay d is translated into a minimal number of L ACK packets, which may render ACK transmission time $T''(n)$ negligible. In the presence of delayed ACKs, the corresponding transmission period $t'(n)$ is determined by

$$t'(n) = \max(\mathrm{RTT}', T(n), T''(n)) \tag{9}$$

From Eqs. (7), (8), it is obvious that by issuing delayed ACKs, we effectively reduce reverse traffic in the expense of increasing RTT. With respect to Eqs. (5), (9), we reach the conclusion that delayed ACKs may degrade TCP performance in the case where $t'(n) = \mathrm{RTT}$ (i.e. no data/ACK congestion has occurred in the forward/reverse path)

$$t'(n) = \mathrm{RTT}', \mathrm{RTT}' > \mathrm{RTT} \Rightarrow t'(n) > t(n)$$

Inversely, gains are expected from delayed ACKs in the situation of congestion in the downlink channel (i.e. $t'(n) = T(n)$), and especially during heavy back traffic

$$t'(n) = T''(n), T''(n) < T'(n) \Rightarrow t'(n) < t(n)$$

Despite these gains, reducing the number of ACKs per received data segment may result in undesirable effects. ACK clocking with delayed ACKs reflects the spacing between data packets that actually trigger ACKs. Since the TCP sender clocks out new data packets in response to the inter-ACK spacing of the reverse path, TCP may need more time in order to open the sending window. This may also increase the TCP sender burst size.

4. Experimental environment

4.1. Experimental settings

The evaluation plan was implemented on the *NS-2* network simulator. LEO systems with RTTs in the range of 40–200 ms cause slight degradation in TCP performance, despite the considerable RTT variations [12]. However, due to large RTTs (approximately 530 ms), maintaining efficient TCP performance over GEO latencies is more challenging. Along these lines, we focus on quantifying the effects of GEO systems on TCP efficiency and real-time delivery. We simulated the system in Fig. 5, where N send-

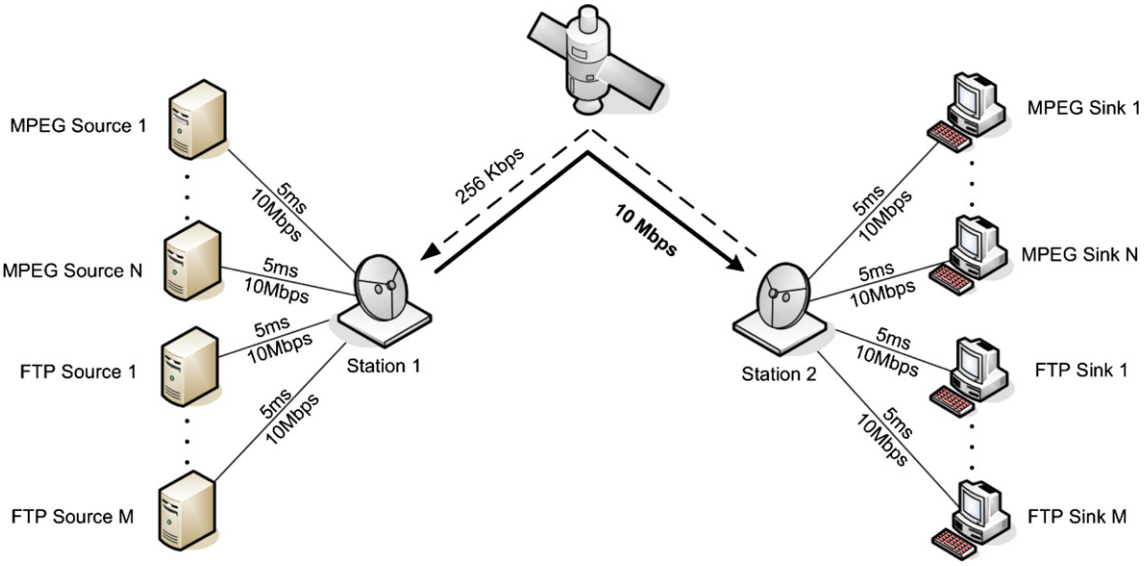

Fig. 5. Simulation topology.

ers transmit an MPEG-4 video stream to N receivers through a bi-directional GEO satellite link with 10 Mbps downlink and 256 Kbps uplink channel. We have also attached M FTP sources transmitting to M FTP receivers via the satellite link. 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 data 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 is adjusted at 10^{-5}, 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. Therefore, we adjusted the maximum window size at 240 KB. Packet size is set to 1000 bytes and consequently, a window may accommodate at the most 240 packets approximately. Since the simulated network exhibits an average RTT of 550 ms, simulation running time was fixed to 200 s, an appropriate time-period for all the protocols to demonstrate their potential. We simulated MPEG flows over standard TCP Reno, the modified TCP Reno variant [30], known as NewReno, augmented with the SACK option, and the protocols TCP Westwood+ (*TCPW*) and GAIMD (0.31, 0.875). All the FTP connections run over TCP Reno. Concerning the relaxed packet loss requirements of time-sensitive applications, as well as the implications that may be induced by FEC/ARQ [15] in order to maximize reliability, we chose not to include such mechanisms in our experiments.

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).

4.2. Measuring performance

We hereby refer to the performance metrics supported by our simulation model. Since the simulated topology includes MPEG flows competing with corporate FTP flows, the performance metrics are applied separately to the MPEG and FTP traffic. Goodput was used to measure the overall system efficiency in bandwidth utilization, and is defined as

$$\text{Goodput} = \frac{\text{Original Data}}{\text{Connection Time}}$$

where *Original Data* is the number of bytes delivered to the high-level protocol at the receiver (i.e. excluding retransmitted packets and overhead) and *Connection Time* is the amount of time required for data delivery. Long-term fairness is measured by the *Fairness Index*, derived from the formula given in [3], and defined as

$$\text{Fairness Index} = \frac{\left(\sum_{i=1}^{n} Throughput_i\right)^2}{n\left(\sum_{i=1}^{n} Throughput_i^2\right)}$$

where $Throughput_i$ is the throughput of the ith flow and n is the total number of flows. Inter-protocol fairness measurements were conducted based on normalized throughput, which is the ratio of the average throughput received by each flow over the bandwidth fair share on each case.

In order to quantify the performance on video delivery, we monitor packet inter-arrival times and eventually distinguish the packets that can be effectively used by the client application from delayed packets (according to a configurable inter-arrival threshold). The proportion of delayed packets is reflected in *Delayed Packets Rate*. Along these lines, each recipient, receiving packets from the MPEG streaming application, calculates the number of delayed packets based on the following algorithm:

Algorithm 1: Delayed packets
For each packet received with sequence number *i*,
determine whether it is delayed

```
if threshold > 0 then
   set packetTime[i] = currentTime
   increase packetsReceived
if   i > 0   and   packetTime[i] − PacketTime[i − 1] >
threshold then
      increase delayedPackets
   end if
end if
```

Several notations used in the pseudocode algorithms are as follows:

1. *threshold:* packet inter-arrival time threshold
2. *delayedPackets:* number of packets with inter-arrival times exceeding the threshold
3. *packetTime:* packet arrival time.

In accordance with video streaming requirements, we adjusted the inter-arrival threshold at 100 ms. Since real-time traffic is sensitive to packet losses, we additionally define *Packet Drop Rate*, as the ratio of the number of lost packets over the number of packets sent by the application.

5. Performance evaluation

In the sequel, we demonstrate and comment on the most prominent results from the experiments we performed based on four 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) competing with light FTP traffic (5 connections). The target sending rate for the MPEG flows is adjusted at 320 Kbps in order to enforce strong contention with the interfering FTP flows. We measured *goodput* and *protocol fairness*, and we additionally selected statistics from delayed and lost packets (only for the MPEG flows), since both are influencing factors that impact video quality. We hereby demonstrate the corresponding results of TCP Reno, TCP NewReno with SACK, TCPW and GAIMD (Fig. 6).

Apparently, all TCP protocols are unable to sustain goodput rates close to the bottleneck link rate (Fig. 6a), despite the relatively large maximum window (i.e. 240 KB). The MPEG connections in each case are affected by link asymmetry, while they are also sensitive to the disturbances caused by interfering FTP traffic. More precisely, in the situation of high link-multiplexing, the resulting infrequency of ACKs diminishes the sending rate, since *cnwd* is adjusted in response to the incoming rate of ACKs. Furthermore, *Fast Retransmit* and *Fast Recovery* are not triggered, when the upstream link is heavily congested and the TCP sender does not receive 3 DACKs.

Despite these undesirable implications, we note that GAIMD and especially TCPW achieve higher bandwidth utilization, outperforming TCP Reno and TCP NewReno with SACK. Both protocols invoke gentle responses in the face of packet loss, maintaining a higher sending rate. TCP Westwood+, in contradiction to the initial version of Westwood, computes one sample of available bandwidth every RTT using all data acknowledged in the specific RTT, therefore obtaining more accurate estimates (Fig. 6a). However, from the perspective of real-time delivery, Westwood+'s efficiency is not evident, since it delivers a considerable amount of delayed packets (Fig. 6d). Inline with our analytical approach, reaching the downlink capacity (i.e. flows 30–50) maximizes transmission time $T(n)$ and results in variable transmission periods which impact video delivery. The protocol responds inappropriately to the variation in the rate of arriving ACKs, since the disturbed inter-ACK spacing reflects the fluctuations in the receiving rate, due to congestion incidents.

Unlike TCPW, GAIMD yields satisfactory performance on video delivery for a wider range of flows, which is the joint effect of relatively low packet losses (Fig. 6c) and a

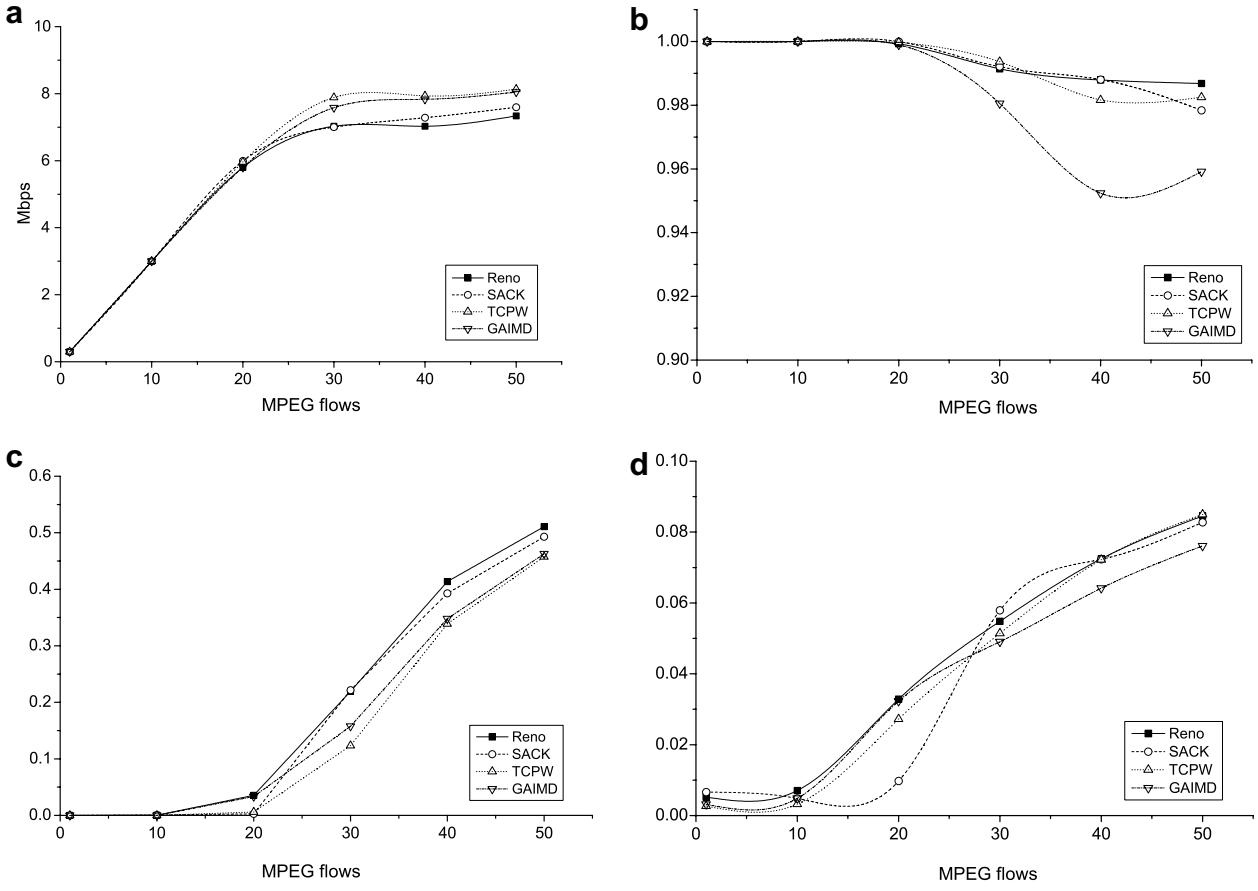

Fig. 6. TCP performance. (a) Goodput of MPEG flows. (b) Fairness index. (c) Packet drop rate. (d) Delayed packets rate.

gentle proportion of delayed packets (Fig. 6d). The protocol avoids the "blind" halving of *cwnd* by employing a large decrease ratio (0.875), therefore achieving the desired smoothness in the expense of being less responsive than standard TCP (1.0, 0.5). A perceptible lack of responsiveness is reflected in Fig. 6c, where GAIMD exhibits more packet drops than TCPW. However, counterbalancing the tradeoff between smoothness with responsiveness is hard to achieve, and in the case of time-sensitive traffic smoothness composes the primary target.

A comparison between standard TCP Reno and TCP NewReno (with SACK) reveals that SACK alone is not sufficient to enable high performance (Fig. 6a). 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 bit errors along the satellite link. Along these lines, they exhibit limited efficiency in the context of real-time delivery. Fig. 6d illustrates that a notable proportion of packets reach the recipient exceeding the delay requirements of streaming video both for Reno and NewReno.

Finally, Fig. 6b depicts some interesting results from the perspective of bandwidths sharing over asymmetric satellite links. GAIMD is the only protocol which exhibits a slight inefficiency in long-term fairness. The protocol employs a low increase rate of 0.31, which inevitably impacts convergence speed. In a system with multiple flows, competing connections may perceive a different amount of packet loss. Considering a low convergence speed in conjunction with very large RTTs, GAIMD flows eventually converge to different *cwnd* parameter values and achieve diverse throughput rates.

5.2. TCP performance vs. error rates

In this scenario, we performed our experiments using diverse bit error rate adjustments (BER: $10^{-7} - 10^{-4}$). In satellite networks, BER can be as high as 10^{-4}, which corresponds to a packet loss probability of 10^{-2} or higher. We hereby demonstrate results from 10 (Fig. 7) to 20 MPEG flows (Fig. 8) in order to investigate the effect of error-prone transmission on protocol efficiency and the performance of video delivery. Apart from MPEG traffic, the system includes 5 FTP connections in both cases.

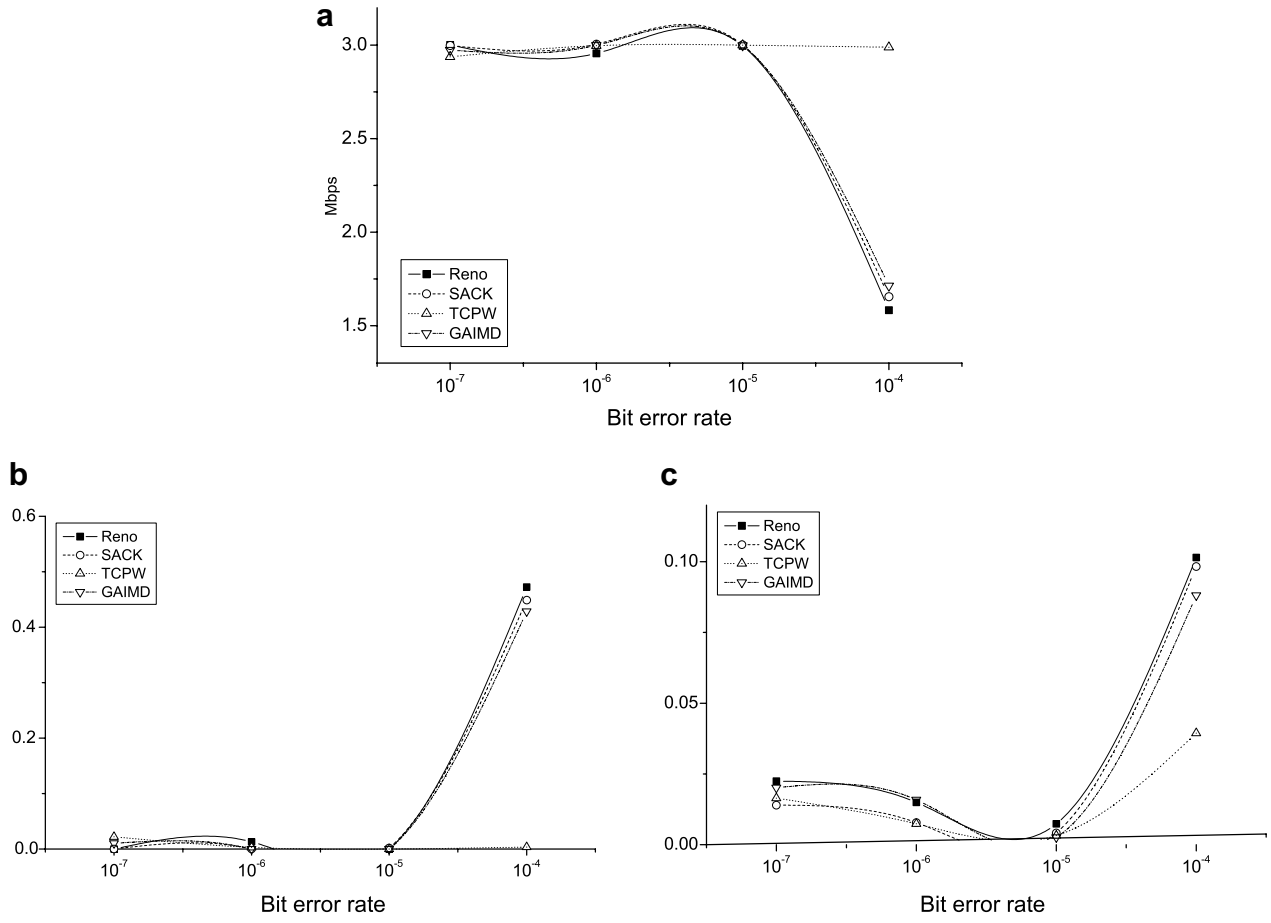

Fig. 7. TCP performance vs. bit errors (10 MPEG and 5 FTP flows). (a) Goodput of MPEG flows. (b) Packet drop rate. (c) Delayed packets rate.

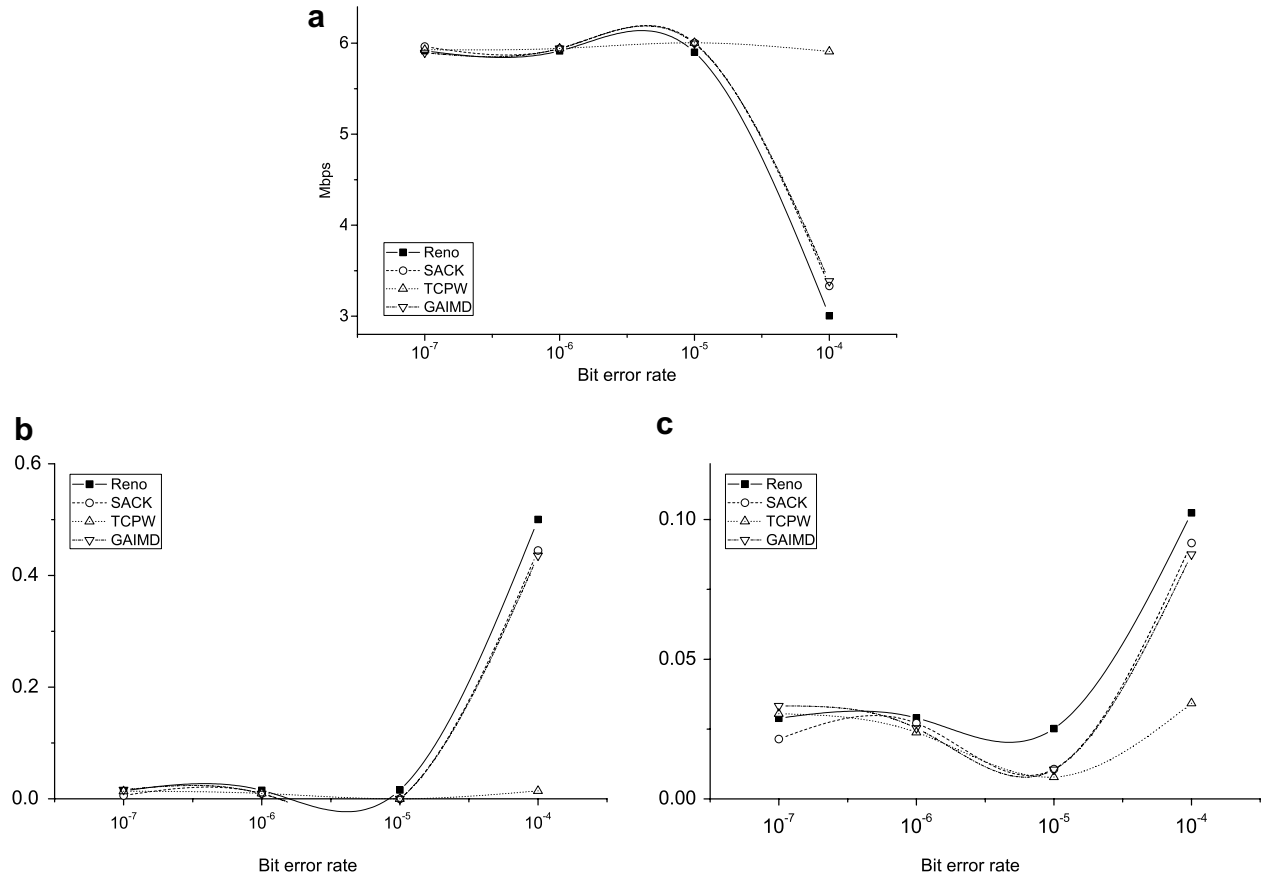

Fig. 8. TCP performance vs. bit errors (20 MPEG and 5 FTP flows). (a) Goodput of MPEG flows. (b) Packet drop rate. (c) Delayed packets rate.

Figs. 7 and 8 illustrate that TCP performance degrades significantly for BER higher than 10^{-5}. TCPW composes the only exception, since it is by far the less sensitive protocol to the increased number of wireless errors. TCPW relies on a bandwidth estimation mechanism which responds remarkably well to a wide range of bit error rates, rendering TCPW the protocol of choice for BER exceeding 10^{-5}. Furthermore, it maintains an acceptable delayed packets rate for intensely error-prone satellite links (Figs. 7c and 8c). On the contrary, TCP Reno, NewReno and GAIMD compose "blind" increase/decrease windows mechanisms, which rely on specific events triggered by violated thresholds. Therefore, their congestion-oriented responses to link errors result in wasteful backward window adjustments that dramatically affect bandwidth utilization. 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 (Figs. 7b and 8b). However, inline with Reno, NewReno with SACK is still inefficient for excessively lossy links. The performance on video delivery is significantly degraded, as the effect of low goodput rates (Figs. 7c and 8c).

According to Figs. 7 and 8, these undesirable effects are evident only for BER $\geqslant 10^{-5}$, which is not common in satellite systems. However, TCP's limited efficiency at high bit error rates calls for a next level of enhancement, where TCP would enable a sophisticated error-recovery strategy adjusted to the error characteristics of the underlying network and possible performance tradeoffs. Based on such an approach, the TCP sender would not be obliged to reduce its transmission rate in the event of a wireless error.

5.3. Impact of delayed ACKs

We assess the impact of delayed ACKs on protocol efficiency in satellite environments. We primarily investigate whether reducing ACK traffic induces implications, which affect the performance of video delivery. We simulated diverse MPEG flows (1–50) competing with 5 FTP connections. We performed our experiments issuing ACKs with no delay and with delays of 100, 200 and 500 ms, successively. In the sequel, we discuss the behavior of TCP NewReno with SACK (Fig. 9) and TCP Westwood+ (Fig. 10), which produced the most conclusive results.

According to Figs. 9a and 10a, reducing the number of ACKs results in minor (NewReno) or no gains (TCPW) in terms of bandwidth utilization. Delayed ACKs tend to slow down the initial slow-start phase (due to the decreased number of ACKs sent by the TCP receiver), while they increase the time needed for the TCP sender to open the

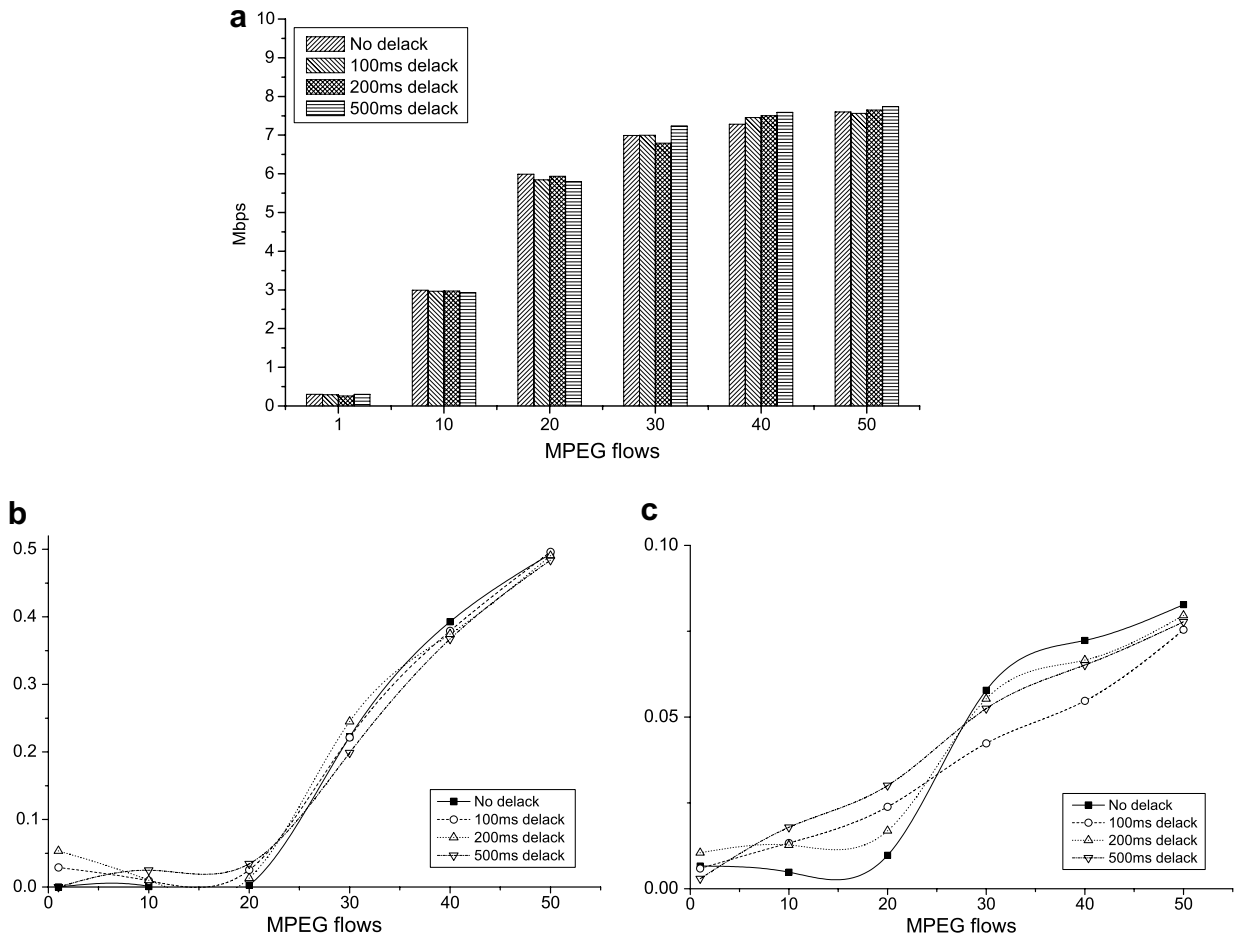

Fig. 9. TCP performance with delayed ACKs (NewReno-SACK). (a) Goodput of MPEG flows. (b) Packet drop rate. (c) Delayed packets rate.

sending window. Furthermore, the increased RTTs diminish the sending rate. Fig. 10a also validates TCPW's behavior in response to delayed ACKs, as analyzed in Section 3. More precisely, TCP's flow throughput is throttled by the rate of arriving ACKs with undesirable effects, especially when the congestion control mechanism relies on measuring the specific rate in order to obtain bandwidth estimates (i.e. TCP Westwood). Therefore, delayed ACKs occasionally degrade TCPW efficiency, since the protocol achieves inadequate bandwidth utilization for high link-multiplexing.

On the other hand, we uncover notable gains in terms of video delivery when issuing delayed ACKs. ACK clocking with delayed ACKs reflects the spacing between data packets that actually trigger ACKs. Since the sending rate depends on the rate of incoming ACKs, reducing and eventually smoothing this rate enables a smoothed transmission rate, inline with streaming video provisions. The beneficial role of delayed ACKs is illustrated in both Figs. 9c and 10c, where the number of delayed packets is noticeably decreased. However, we note that issuing ACKs with delays more than 100 ms does not result in perceptible performance gains, since the increased RTTs (as derived from Eq. (7)) counterbalance the benefits from the reduced ACK transmission time $T''(n)$ (Eq. (8)).

5.4. Inter-protocol fairness

We conclude our performance studies by investigating the interactions of a variety of MPEG flows with interfering FTP traffic over asymmetric satellite links. We demonstrate results only from TCP Westwood+ and GAIMD, which adequately enable real-time delivery in such environments. Along these lines, we simulated a wide range of MPEG flows (1–50) competing with moderate (20 flows) and heavy (40 flows) FTP traffic over TCP Reno. We repeated the experiment for TCPW and GAIMD, and measured the normalized throughput of the MPEG and the FTP connections for each protocol separately (Figs. 11 and 12).

A comparative overview of Figs. 11 and 12 reveals that GAIMD coexists fairly with TCP, while TCPW allocates increased network resources exceeding the link fair share. When competing with 40 FTP/Reno flows (Fig. 11b), TCPW connections apparently "steal" bandwidth from the FTP flows, enforcing the Reno senders to invoke sharp backward window adjustments under these awkward

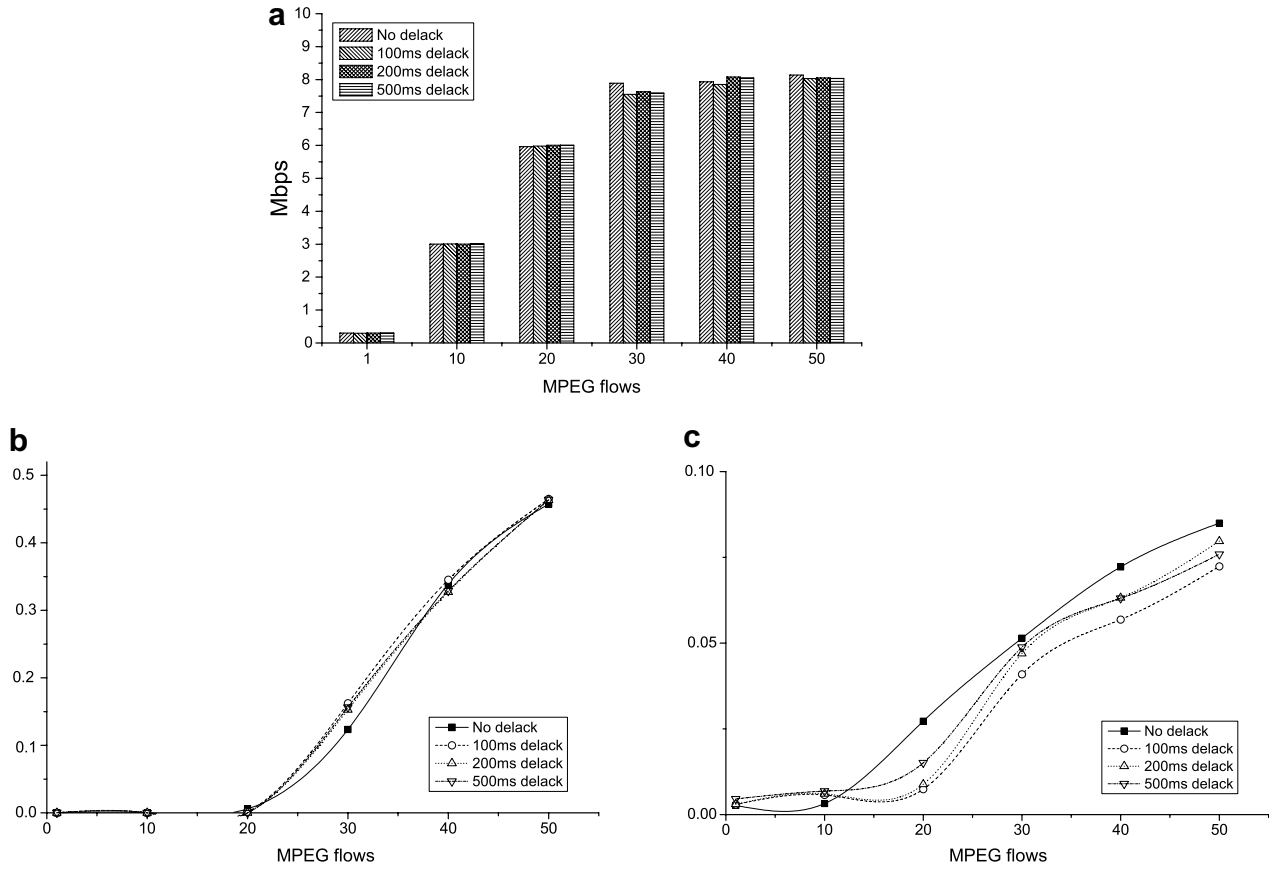

Fig. 10. TCP performance with delayed ACKs (TCPW). (a) Goodput of MPEG flows. (b) Packet drop rate. (c) Delayed packets rate.

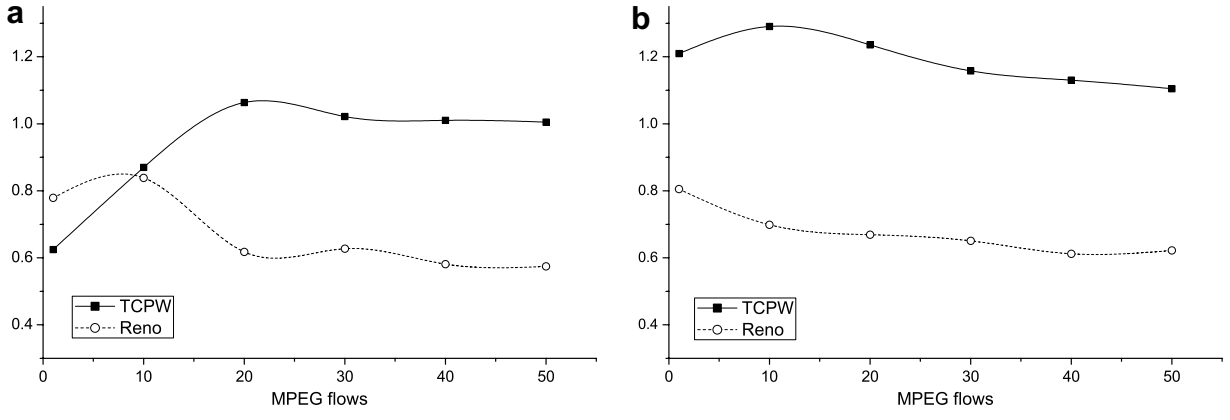

Fig. 11. Normalized throughput (TCPW). (a) 1–50 MPEG flows vs. 20 TCP Reno flows. (b) 1–50 MPEG flows vs. 40 TCP Reno flows.

conditions. As we reported in Subsection 2.2, the congestion control mechanism of TCPW tends to overestimate the available bandwidth. Thus, the protocol confines the transmission rate of all interfering connections, which undesirably obtain resources consistently below of the fair share of the link.

On the other hand, GAIMD maintains friendliness with corporate connections, since the employed α, β parameters satisfy the TCP-friendly equation obtained in [9]

$$\alpha = \frac{4(1-\beta^2)}{3}$$

Therefore, as depicted in Fig. 12, GAIMD allows interfering TCP flows to obtain network resources closer to the fair share of the link (i.e. normalized throughput of 1) in comparison with TCPW, which is critical in systems with multiple flows and different protocols.

We note that the effects of link asymmetry, such as increased sender burst size and longer time to open *cwnd*,

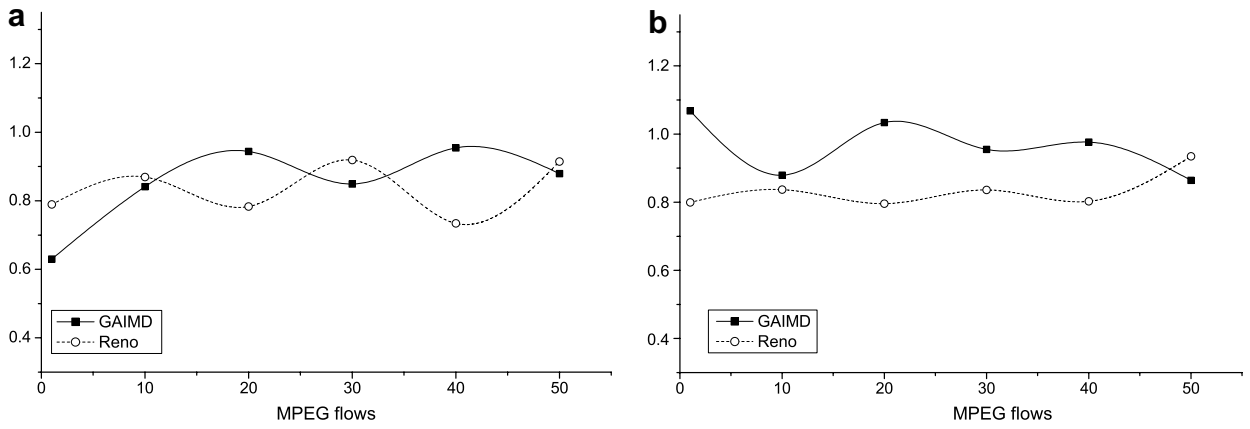

Fig. 12. Normalized throughput (GAIMD). (a) 1–50 MPEG flows vs. 20 TCP Reno flows. (b) 1–50 MPEG flows vs. 40 TCP Reno flows.

give an impulse to contention conditions among competing connections, and consequently, compromise inter-protocol fairness.

6. Conclusions

Focusing on the study of GEO systems where RTTs experience insignificant variations, we investigated how satellite links and bandwidth asymmetry impact TCP performance and real-time delivery. Based on an analytical approach, as well as extensive simulations, we demonstrated that in such environments, TCP performance is not explicitly dependent on the downlink capacity, but it is throttled by the rate of incoming ACKs across the reverse path. Apart from inadequate bandwidth utilization, a variable and infrequent rate of ACKs deteriorates the performance of real-time delivery, causing transmission gaps which are eventually perceived as playback interruptions at the receiver. During heavy traffic in the uplink, *Fast Retransmit* and *Fast Recovery* algorithms may not be triggered resulting in coarse timeouts and multiple window reductions. We also showed that delayed ACKs do not enable perceptible gains in terms of bandwidth utilization. On the contrary, they occasionally degrade the performance of protocols that dynamically exploit bandwidth availability by measuring the rate of arriving ACKs, such as TCP Westwood. Despite this implication, we uncovered notable gains in terms of real-time delivery, since a reduced and regular rate of ACKs enables a smoother transmission rate.

From the perspective of individual protocol performance, we showed that the algorithm of TCP Westwood+ does not always obtain accurate estimates; yet, it is more effective than ''blind'' increase/decrease window mechanisms (i.e. TCP Reno), which rely on specific events triggered by violated thresholds. Even by enabling SACK, the benefits for TCP are minimal in such environments. We also identified a significant discrepancy between the throughput rates achieved by competing Westwood+ and TCP connections in several occasions. On the other hand, GAIMD yields satisfactory performance in terms of real-time delivery in a wide range of network and session dynamics. Employing a high decrease ratio, the protocol responds to congestive and wireless losses by reducing the congestion window more gently than standard TCP. GAIMD's performance (along with standard TCP) deteriorates only over highly error-prone links (i.e. BER $> 10^{-5}$), where flow characteristics do not follow a prescribed and static behavior. However, error rates of this magnitude are not common in modern satellite systems.

References

[1] V. Jacobson, Congestion avoidance and control, in: Proceedings of ACM SIGCOMM '88, Stanford, USA, August 1988.
[2] W. Stevens, TCP Slow-Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, January 1997.
[3] D. Chiu, R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks, Journal of Computer Networks 17 (1) (1989) 1–14.
[4] V. Tsaoussidis, I. Matta, Open issues on TCP for mobile computing, Journal of Wireless Communications and Mobile Computing 2 (1) (2002) 3–20.
[5] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, M. Sooriyabandara, TCP Performance Implications of Network Path Asymmetry, RFC 3449, December 2002.
[6] M. Allman, V. Paxson, W. Stevens, TCP Congestion Control, RFC 2581, April 1999.
[7] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based Congestion control for unicast applications, in: Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
[8] Y.R. Yang, M.S. Kim, S.S. Lam, Transient behaviors of TCP-friendly congestion control protocols, in: Proceedings of IEEE INFOCOM 2001, Anchorage, Alaska, USA, April 2001.
[9] Y.R. Yang, S.S. Lam, General AIMD congestion control, in: Proceedings of 8th Int/Nal Conference on Network Protocols (ICNP), Osaka, Japan, November 2000.
[10] P. Papadimitriou, V. Tsaoussidis, Evaluating TCP mechanisms for real-time streaming over satellite links, in: Proceedings of 4th IEEE Int/Nal Conference on Wired/Wireless Internet Communications (WWIC 2006), Bern, Switzerland, May 2006.
[11] P. Papadimitriou, V. Tsaoussidis, On transport layer mechanisms for real-time QoS, Journal of Mobile Multimedia 1 (4) (2006) 342–363.
[12] T.R. Henderson, R.H. Katz, Transport protocols for Internet-compatible satellite networks, IEEE Journal of Selected Areas in Communications (JSAC) 17 (2) (1999) 326–344.

[13] C. Partridge, T.J. Shepard, TCP/IP performance over satellite links, IEEE Network 11 (5) (1997) 44–49.
[14] L. Wood, G. Pavlou, B. Evans, Effects on TCP of routing strategies in satellite constellations, IEEE Communications Magazine 39 (3) (2001) 172–181.
[15] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, A comparison of mechanisms for improving TCP performance over wireless links, ACM/IEEE Transactions on Networking 5 (6) (1997) 756–769.
[16] A. Bakre, B.R. Badrinath, I-TCP: indirect TCP for mobile hosts, in: Proceedings of 15th Int/Nal Conference on Distributed Computing Systems (IDCS), Vancouver, Canada, 1995.
[17] V.N. Padmanabhan, R. Katz, TCP fast start: a technique for speeding up web transfers, in: Proceedings of IEEE Globecom, Sydney, Australia, November 1998.
[18] F. Akyildiz, G. Morabito, S. Palazzo, TCP-peach: a new congestion control scheme for satellite IP networks, IEEE Transactions on Networking 9 (3) (2001) 307–321.
[19] V. Tsaoussidis, H. Badr, TCP-Probing: towards an error control schema with energy and throughput performance gains, in: Proceedings of 8th Int/Nal Conference on Network Protocols (ICNP), Osaka, Japan, November 2000.
[20] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP Selective Acknowledgment Options, RFC 2018, October 1996.
[21] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.
[22] V. Tsaoussidis, C. Zhang, The dynamics of responsiveness and smoothness in heterogeneous networks, IEEE Journal on Selected Areas in Communications 23 (6) (2005) 1178–1189.
[23] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, TCP Westwood: bandwidth estimation for enhanced transport over wireless links, in: Proceedings of MobiCom '01, Rome, Italy, July 2001.
[24] P. Papadimitriou, V. Tsaoussidis, Assessment of Internet voice transport with TCP, Int/Nal Journal of Communication Systems (IJCS) 19 (4) (2006) 381–405.
[25] L. Grieco, S. Mascolo, Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control, ACM Computer Communication Review 34 (2) (2004) 25–38.
[26] V. Tsaoussidis, C. Zhang, TCP real: receiver-oriented congestion control, Computer Networks 40 (4) (2002) 477–497.
[27] C. Zhang, V. Tsaoussidis, TCP real: improving real-time capabilities of TCP over heterogeneous networks, in: Proceedings of 11th IEEE/ACM NOSSDAV, New York, USA, June 2001.
[28] P. Papadimitriou, V. Tsaoussidis, End-to-end congestion management for real-time streaming video over the Internet, in: Proceedings of 49th IEEE GLOBECOM, San Francisco, USA, November 2006.
[29] A. Chockalingam, M. Zorzi, V. Tralli, Wireless TCP performance with link layer FEC/ARQ, in: Proceedings of IEEE ICC 1999, Vancouver, Canada, June 1999.
[30] S. Floyd, T. Henderson, The NewReno Modification to TCP's Fast Recovery Algorithm, Internet RFC 2582, 1999.

Panagiotis Papadimitriou obtained a B.Sc. in Computer Science from University of Crete, Greece; and an M.Sc. in Information Technology from University of Nottingham, UK. He is currently a Ph.D. Candidate in Electrical and Computer Engineering in Demokritos University, Greece. Panagiotis is a Visiting Lecturer in the Information Management Department of Technological Educational Institute (TEI) of Kavala, Greece. His research interests include multimedia QoS in the Internet, transport protocols, satellite IP networks and streaming media.

Vassilis Tsaoussidis received a B.Sc. in Applied Mathematics from Aristotle University, Greece; a Diploma in Statistics -and Computer Science from the Hellenic Institute of Statistics; and a Ph.D. in Computer Networks from Humboldt University, Berlin, Germany (1995). Vassilis held faculty positions in Rutgers University, New Brunswick, SUNY Stony Brook and Northeastern University, Boston. In May 2003, Vassilis joined the Department of Electrical and Computer Engineering of Demokritos University, Greece. His research interests lie in the area of transport/network protocols, i.e. their design aspects and performance evaluation. Vassilis is an editor for IEEE Transactions in Mobile Computing, the Journal of Computer Networks, the Journal of Wireless Communications and Mobile Computing, the Journal of Mobile Multimedia, and the Journal of Parallel Emergent and Distributed Systems. He is in the steering committee of WWIC; he chaired several conferences, and participated in several Technical Program Committees in his area of expertise, such as INFOCOM, Networking, Globecom, and several others.