INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS AND NETWORKING
Int. J. Satell. Commun. Network. (in press)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/sat.855

Transport layer protocols and architectures for satellite networks

Carlo Caini[1,*,†], Rosario Firrincieli[1], Mario Marchese[2], Tomaso de Cola[2], Michele Luglio[3], Cesare Roseti[3], Nedo Celandroni[4] and Francesco Potortì[4]

[1] *DEIS/ARCES, University of Bologna, Viale Risorgimento 2, 40136, Bologna, Italy*
[2] *CNIT, University of Genoa Research Unit, Via Opera Pia 13, 16145, Genoa, Italy*
[3] *DIE, University of Rome Tor Vergata, via del Politecnico 100133, Roma, Italy*
[4] *ISTI, via Moruzzi 1, 56124, Pisa, Italy*

SUMMARY

Designing efficient transmission mechanisms for advanced satellite networks is a demanding task, requiring the definition and the implementation of protocols and architectures well suited to this challenging environment. In particular, transport protocols performance over satellite networks is impaired by the characteristics of the satellite radio link, specifically by the long propagation delay and the possible presence of segment losses due to physical channel errors. The level of impact on performance depends upon the link design (type of constellation, link margin, coding and modulation) and operational conditions (link obstructions, terminal mobility, weather conditions, etc.). To address these critical aspects a number of possible solutions have been presented in the literature, ranging from limited modifications of standard protocols (e.g. TCP, transmission control protocol) to completely alternative protocol and network architectures. However, despite the great number of different proposals (or perhaps also because of it), the general framework appears quite fragmented and there is a compelling need of an integration of the research competences and efforts. This is actually the intent of the transport protocols research line within the European SatNEx (Satellite Network of Excellence) project. Stemming from the authors' work on this project, this paper aims to provide the reader with an updated overview of all the possible approaches that can be pursued to overcome the limitations of current transport protocols and architectures, when applied to satellite communications. In the paper the possible solutions are classified in the following categories: optimization of TCP interactions with lower layers, TCP enhancements, performance enhancement proxies (PEP) and delay tolerant networks (DTN). Advantages and disadvantages of the different approaches, as well as their interactions, are investigated and discussed, taking into account performance improvement, complexity, and compliance to the standard semantics. From this analysis, it emerges that DTN architectures could integrate some of the most efficient solutions from the other categories, by inserting them in a new rigorous framework. These innovative architectures therefore may represent a promising solution for solving some of the important problems posed at the transport layer by satellite networks, at least in a medium-to-long-term perspective. Copyright © 2006 John Wiley & Sons, Ltd.

Received July 2005; Revised 1 October 2005; Accepted 1 July 2006

KEY WORDS: transport protocols; TCP; satellite networks; PEP; DTN

*Correspondence to: C. Caini, DEIS/ARCES, University of Bologna, Viale Risorgimento 2, 40136, Bologna, Italy.
† E-mail: ccaini@deis.unibo.it

Contract/grant sponsor: SatNEx Network of Excellence; contract/grant number: IST-507052

1. INTRODUCTION

Internet satellite communications face some specific challenges deriving from the characteristics of satellite radio links. First, the round trip time (RTT) is greatly increased by the long propagation time on the satellite radio channel, especially when MEO and GEO constellations are considered. For example, in a hybrid GEO system (forward link via satellite and return link via a wired terrestrial network) the RTT of the satellite segment is $\cong$ 300 ms, including routing and processing delays; this value is increased to $\cong$ 600 ms in a bidirectional GEO system. Second, the presence of random losses not resulting from congestion cannot always be considered negligible, depending upon the link design (link margin, coding and modulation) and operational conditions (link obstructions, terminal mobility, weather conditions, etc.). Both these aspects may put significant limitations on the transport layer performance when the upper layers call for a reliable service over a satellite channel [i.e. in the present Internet the use of transmission control protocol (TCP) instead of User Datagram Protocol (UDP)] [1–3]. To this regard, it should be reminded that the TCP protocol design considered the main case of terrestrial wired networks. In this environment it proved very successful, by supporting the development of computer communications for more than twenty years with only a limited number of modifications. However, because the performance of common variants of TCP (Reno, NewReno and SACK) largely depends on network delays, the long propagation delay of satellite connections (especially for MEO and GEO) may result in significant throughput degradation. Moreover, long RTTs amplify the consequences of spurious calls to the congestion control algorithm triggered by random losses. These calls derive from a conservative choice made in the TCP design. In fact, as it is very difficult (when not impossible) to disambiguate the origin of the losses (i.e. if they are caused by congestion or by channel errors), TCP treats any loss as an indication of congestion. This explicit choice is also supported by the consideration that channel errors are rare in terrestrial wired networks. Unfortunately, dealing with satellite radio channel, random losses may occasionally be more frequent. For them, the congestion window (cwnd) reduction is clearly a harmful response, especially because long RTTs significantly slow down the cwnd re-opening. In fact, even a 1% packet error rate (PER) may have significant consequences for GEO satellites, resulting in a throughput degradation of more than an order of magnitude. One way to reduce this problem is to act at the physical and/or link layer, in order to reduce the PER, by making use of FEC (forward error correction) and automatic repeat request (ARQ) techniques: the impact on TCP of the former will be treated in the paper, for the latter see Reference [4]. However, adverse operational conditions can still result in a residual PER on the satellite channel, and in this case their recovery is still in charge of TCP.

Although a large number of different solutions have been proposed in the literature in order to overcome the severe problems posed by satellite links at transport level, we are still far from a definitive solution. As most contributions focus on the proposals conceived and promoted by the respective authors, the general framework appears quite fragmented. Therefore, since the problems posed by satellite Internet are both severe and urgent, there is a compelling need for integration of the research competences and efforts. These are the objectives of the transport protocols research line within the European SatNEx (Satellite Network of Excellence) project [5], which more generally aims at dissemination of knowledge and integrating research on satellite communications. This paper stems from the competence and research integration performed by the authors within the framework of the SatNEx project. It aims at providing the reader with a broad but unified (and hopefully clear) view of all the possible approaches that can be pursued to overcome the limitations of current transport protocols and architectures, when applied to satellite communications. Consequently, the paper must encompass the description and discussion of quite different countermeasures, classified here according to the following categories: TCP interactions with lower layers, TCP enhancements, performance enhancement proxies (PEP) [6], delay tolerant networks (DTN) [7].

The first category considers mechanisms that, although external to the transport layer, can improve the transport performance by exploiting layer interactions. Two examples are reported in this paper, namely the 'path MTU discovery', and the use of 'forward error correction' (FEC). The second category deals with TCP enhancements that revise and improve the basic TCP procedures. Of the many proposals appearing in the literature, this paper focuses in particular on TCP Peach [8], Westwood [9], Hybla [10], SCTP [11] and CK-STP [12], because they seem well suited to coping with satellite characteristics, i.e. long RTTs and random losses. A different approach is pursued by PEPs, the third category, which may imply a modification of the TCP/IP semantics, as in TCP spoofing [13], or the introduction of significant modifications to the network architecture, as in TCP splitting [14]. Note that in the last case, optimized transport protocols other than TCP can be used on the satellite segment, such as the Xpress Transport Protocol (XTP) [15] and the Space Communication Protocol Standard (SCPS) [16]. PEPs may offer good performance, but sometimes at the expense of violating the basic end-to-end TCP semantics. This issue is tackled by the concept of DTN, which specifically addresses communications in the so-called 'challenged networks' (long, or extremely long, RTTs, random losses and discontinuous channel availability), of which satellite communications are a significant example. As shown in this paper, the DTN innovative architecture seems potentially able to retain the advantages of many of the aforementioned approaches. Therefore, the DTN concept actually appears to be a promising solution to some of the satellite networking problems, at least in medium-to-long term. At present, however, the research on DTN architecture is still ongoing, and no definitive performance assessment can be done.

The paper is organized as follows. Section 2 summarizes the congestion control and the loss recovery algorithms of standard protocol; the rationale is to provide the unfamiliar reader with the basic elements of the most widely adopted transport protocol, essential to understand the subsequent sections which focus on the presentation of specific countermeasures; in Section 3 the trade-offs implicit in the use of low layer mitigations to optimize TCP performance are discussed and in Section 4 a survey of the most promising TCP enhancements is given, showing how the penalization suffered by satellite connections can be effectively addressed. Other widespread techniques, based on the introduction of PEP at the transport layer, are presented in Section 5; DTN, and related protocols, are presented in Section 6; finally, conclusions are drawn in Section 7.

2. OVERVIEW ON CURRENT TCP

Most of this section is taken from Reference [17], which specifies the 'slow start' (SS), 'congestion avoidance' (CA), 'fast retransmit' and 'fast recovery' algorithms, and from Reference [18] which deals with the 'retransmission timeout' (RTO) policy. In the following these basic procedures are briefly summarized using the definitions contained in Table I.

Table I. Definition of main TCP parameters.

Parameter	Definition
Congestion window (cwnd)	A TCP state variable that limits the amount of data a TCP can send. It is a limit on the amount of data the sender can transmit before receiving an acknowledgment (ACK). Some implementations maintain cwnd in units of bytes, while others in units of full-sized segments
Flight size	The amount of data that has been sent but not yet acknowledged. Actually, it identifies the segments still 'in flight', still inside the network
Full-sized segment	A segment containing the maximum number of data bytes permitted (i.e. SMSS bytes of data)
Initial window (IW)	Size of the sender's congestion window after the three-way handshake is completed
Loss window (LW)	Size of the congestion window after a TCP sender detects loss using its retransmission timer (see below)
Receiver maximum segment size (RMSS)	Size of the largest segment the receiver can accept
Receiver window (rwnd)	The most recently advertised receiver window: it is a receiver-side limit on the amount of outstanding data.
Restart window (RW)	Size of the congestion window after a TCP restarts transmission after an idle period
RTT	The Round Trip Time is the time elapsed from the transmission of a segment and the reception of the corresponding ACK
Segment	Any TCP/IP data or acknowledgment packet (or both)
Sender maximum segment size (SMSS)	Size of the largest segment that the sender can transmit. It depends on the type of network used and other factors
ssthresh	The SS threshold is the switching point between the SS and CA phases

2.1. Slow start

The aim of slow start algorithm is to probe the network to check the available capacity by gradually increasing the cwnd, instead of starting with a high fixed value that may cause congestion. Initial Window (IW), the initial value of cwnd, must be less than or equal to 2SMSS bytes. In order to speed up the TCP start-up phase, it is possible to use an increased initial window [19].

$$\mathrm{IW} = \min(4\ \mathrm{SMSS}, \max(2\ \mathrm{SMSS},\ 4380\ \mathrm{bytes})) \quad (1)$$

The initial value of ssthresh may be arbitrarily high, but is immediately reduced in response to the first congestion episode. The slow start algorithm is used when cwnd $<$ ssthresh.

During slow start, a TCP increases cwnd by SMSS bytes for each ACK received that acknowledges new data

$$\mathrm{cwnd} = \mathrm{cwnd} + \mathrm{SMSS} \quad (2)$$

The slow start phase terminates when the cwnd exceeds the ssthresh or when a packet is lost.

2.2. Congestion avoidance

The congestion avoidance algorithm is used when cwnd is greater than ssthresh and continues until congestion is detected. During congestion avoidance, for every incoming non-duplicate

ACK, the cwnd in byte is updated according to the following rule

$$\text{cwnd} = \text{cwnd} + (\text{SMSS} \cdot \text{SMSS})/\text{cwnd} \tag{3}$$

The above formula is approximated by the implementations that maintain cwnd in units of full-sized segments, instead of bytes, which find it more convenient to increment the cwnd by 1 full-sized segment per RTT. When a TCP sender detects segment loss using the retransmission timer, the value of ssthresh must be set to no more than the value given in the following formula, where *FlightSize* is the amount of not yet acknowledged data in the network

$$\text{sstresh} = \max(\text{FlightSize}/2,\ 2\ \text{SMSS}) \tag{4}$$

Furthermore, upon expiration of an RTO (as explained in the following), cwnd must be set to no more than the loss window, LW, which equals 1 full-sized segment (regardless of the value of IW). Therefore, in this case, after retransmitting the dropped segment the TCP sender uses the slow start algorithm until the new value of ssthresh is reached, at which point congestion avoidance takes over again.

2.3. *Fast retransmit and fast recovery*

A TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives indicating which sequence number is actually expected. From the sender's perspective, duplicate ACKs can be caused by a number of network problems (e.g. dropped segments, re-ordering of data, and replication of data). Obviously, a TCP receiver will send an immediate and cumulative ACK when the incoming segment fills a gap in the sequence space. The TCP sender uses the Fast Retransmit algorithm to detect and repair loss, triggered by the arrival of three duplicate ACKs. After entering fast retransmit, TCP performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire. Then, the fast recovery algorithm governs data recovery until the last transmitted segment at the fast retransmit entering is acknowledged. This aspect mainly differentiates TCP Reno and NewReno with respect to TCP Tahoe and Old Tahoe [17].

The fast retransmit and fast recovery algorithms are usually implemented together (TCP Reno) as follows.

1. When the third duplicate ACK is received, the ssthresh value is set to the value given in (4).
2. The lost segment is retransmitted and cwnd is set to ssthresh plus 3SMSS.
3. For each additional duplicate ACK received, cwnd is increased by SMSS.
4. A segment is transmitted, if allowed by the new value of cwnd and the receiver's advertised window.
5. When the next ACK arrives that acknowledges new data, cwnd is set to ssthresh (the value set in step 1).

A further differentiation between Reno and NewReno specifications is present. Specifically, TCP Reno concludes the fast recovery phase once an ACK acknowledging new data is received and then enters the congestion avoidance phase. On the contrary, TCP NewReno distinguishes between partial and full acknowledgements. In the former case, the received ACK does not cover all the sent data, and consequently indicates further packet losses. In this case, the fast recovery phase is not exited and TCP NewReno reacts by retransmitting the first unacknowledged segment and deflating the congestion window by the acknowledged data. In the latter case, when a full acknowledgement is received, TCP NewReno behaves like TCP Reno. The aim of the NewReno modification is to avoid unnecessary multiple cwnd reductions in the presence of a burst of losses. Note however that, as in Reno, no more than one segment per RTT can be recovered, with consequent poor performance on satellite channels. To overcome this limitation, the TCP selective acknowledgment (SACK) option [20] can be adopted, which allows the recovery of multiple losses in the same RTT.

2.4. Retransmission timeout

TCP uses a retransmission timer to ensure data delivery in the absence of any feedback from the remote data receiver. The duration of this timer is referred to as RTO (retransmission timeout).

To compute the current RTO, a TCP sender makes use of two state variables, SRTT (smoothed RTT) and RTTVAR (RTT variation). Another important role is also played by the timer granularity, indicated in the following as G.

The rules governing the computation of SRTT, RTTVAR, and RTO are as follows [18].

1. Until a RTT measurement has been made for a segment sent between the sender and receiver, the sender sets RTO to 3 s, though the 'backing off' on repeated retransmission still applies.
2. When the first RTT measurement R is made, the host sets $SRTT$ equal to R, RTTVAR to R/2 and RTO to SRTT + max (G, $K \cdot$ RTTVAR), where $K = 4$.
3. When the next RTT measurement R' is available, a host sets
   a. $\mathrm{RTTVAR}_{\mathrm{new}} = (1-\beta)\cdot \mathrm{RTTVAR}_{\mathrm{old}} + \beta \cdot |\mathrm{SRTT}_{\mathrm{old}} - R'|$
   b. $\mathrm{SRTT}_{\mathrm{new}} = (1-\alpha)\cdot \mathrm{SRTT}_{\mathrm{old}} + \alpha \cdot R'$, where the subscripts 'new' and 'old' indicate the new updated value and the old one, respectively. Moreover, we have $\alpha = 1/8$ and $\beta = 1/4$. After the computation, a host updates RTO following the relation RTO = SRTT + max (G, $K \cdot$ RTTVAR).

On the basis of the computation of the RTO value, the following procedure manages the retransmission timer.

1. For each data transmission (or retransmission), if the timer is not running, it starts so that it will expire after RTO seconds.
2. When all outstanding data has been acknowledged, the retransmission timer is turned off.
3. When a non-duplicated ACK is received, the timer is restarted so that it will expire after RTO seconds.

When the retransmission timer expires:

1. The earliest segment that has not been acknowledged by the TCP receiver is retransmitted.
2. The host performs the back-off operation by setting RTO to RTO $\cdot$ 2 ('back off the timer'). This RTO value is maintained until either a fresh RTT estimate is available or a new RTO expiration occurs.

3. TCP INTERACTIONS WITH LOWER LAYERS

TCP performance on satellite channels can be improved by optimizing interactions with the lower layers. Two significant examples are considered here: the 'path maximum transfer unit (MTU) discovery' and the 'modulation and FEC coding' optimization.

3.1. *Path MTU discovery*

The path MTU discovery procedure aims at finding the maximum packet size that a connection can use on a given link without being subject to IP fragmentation. In fact, if the packet is too large to be forwarded, the gateway fragments the packet and forwards the fragments. Instead, with the Path MTU discovery mechanism, an ICMP message is returned to the sender to indicate that the original packet could not be transmitted without being fragmented and also to report the largest packet size that can be forwarded by the gateway. In this way, the segment overhead is reduced and the TCP sender can increase the congestion window more rapidly (in terms of bytes) [21]. It is worth noting that the Path MTU Discovery procedure, although usually applied in the presence of TCP connections, can be carried out independently of TCP. In contrast, the modulation and FEC coding optimization described next will require the knowledge of the TCP performance to be carried out. In other words, a much more complex joint optimization, involving both the physical and the transport layer, must be performed, as pointed out in the following.

3.2. *Modulation and FEC coding optimization*

As explained in the introduction, the PER due to wireless channel errors has an impact on TCP connections that is much more severe than it is on stream data, due to its interaction with the basic TCP congestion control mechanism. Due to this high sensitivity, it is paramount to reduce the PER, by using one or more of the following techniques: packet-level forward erasure coding [22], link-level ARQ, FEC codes and adaptive modulation. Here we illustrate the concept by only considering modulation and FEC coding. As far as ARQ is concerned, we refer the reader to Reference [4], which provides a comprehensive discussion on potential benefits and possible interaction of ARQ with TCP, as well as a useful guidance on ARQ use on different types of links.

3.2.1. Interaction between TCP performance and FEC and modulation schemes. The residual bit error rate (BER) at the physical layer depends on various parameters, most of which cannot easily be changed once the satellite system is already in service. In particular, the bandwidth available for the radio signal is often fixed, as well as the maximum transmitted power. What can be easily changed at present is the FEC scheme which is usually employed at the physical level, as well as (in some cases) the modulation technique employed (in particular, the number of levels in an M-PSK or in an M-QAM scheme). In general, for a given FEC technique the resilience to errors increases with the amount of redundancy introduced, i.e. by reducing the code rate R_c. Analogously, for the M-PSK and M-QAM modulations, performance in terms of bit error probability is generally improved by decreasing the cardinality of the signal constellation M, and consequently the number of bits per modulation symbol, given by $\log_2 M$. In both cases, the price to pay for a lower BER is reduced efficiency in the use of the available spectrum [23].

3.2.2. Trading available information bit rate for a lower TCP packet error rate. For a satellite system limited in the physical channel bandwidth (in Hz) B and in the maximum transmitted power C, but with some level of flexibility in the choice of the FEC and modulation scheme, it is possible to find a trade-off between the information bit rate available to TCP and the PER. Although intuitive, the terms of this trade-off need to be rigorously defined, proceeding step-by-step. First, dealing with M-PSK or M-QAM, the channel bandwidth B roughly numerically coincides with the signal symbol rate B_s. Therefore, the modulation efficiency η, defined as the ratio between the bit rate and the signal occupied bandwidth B, coincides with the number of bit per symbol. The available information bit rate B_r can then be obtained as a function of the symbol rate B_s, by the following,

$$B_r = R_c \eta B_s \tag{5}$$

As far as the constraint on the received signal power C is concerned, it can be directly translated into a constraint on the received energy per information bit, as $E_b = C/B_r$. Then, by dividing both members by the spectral density noise N_o, this formula can be conveniently rewritten in terms of signal-to-noise ratios,

$$E_b/N_o = (C/N_o)/R_c \eta B_s \tag{6}$$

The final step consists in considering the link between the BER and the E_b/N_o ratio, which depends on the modulation and the FEC scheme adopted. In particular, we can observe that the well-known performance graphs (BER vs E_b/N_o) reported in the literature show a decrement of the BER, for the same E_b/N_o ratio, if η (inside the same modulation scheme) or R_c (inside the same family of codes) are decreased, because the modulation and coding scheme become more robust against noise [24].

Now, we are able to draw the desired conclusion. By making use of (5) we can observe how, once the symbol rate B_s is fixed, by reducing the modulation efficiency or the code rate we correspondingly reduce the available information bit rate B_r. This parameter represents an upper limit on the maximum transmission rate at the TCP level and its decrease is the price paid for the wanted BER and PER reduction. Note that if C/N_o is high enough, the PER is negligible, and it is disadvantageous to reduce the available bit rate by decreasing R_c or η. Conversely, if C/N_o is low, the PER is high, TCP is unable to reach the available bit rate, and therefore performance could be effectively improved by increasing R_c or η. In other words, finding the optimum modulation and coding scheme, in terms of TCP throughput, means finding the optimum balance between robustness to noise and available information bit rate.

As an example of how it is possible to trade the information bit rate available to TCP for PER, thus optimizing TCP throughput, Figure 1 shows the throughput achievable by considering two modulation schemes, namely BPSK ($M = 2$, $\eta = 1$) and QPSK ($M = 4$, $\eta = 2$) and four coding schemes, all based on convolutional coding, with code rates 1/2 (base code), 3/4 (punctured base code), 7/8 (punctured base code), 1/1 (no coding). The base code is the standard NASA $R_c = 1/2$, constraint length 7, convolutional code [24]. The optimization subspace consequently has seven possible values (note that BPSK uncoded is never used, because it provides lower performance than QPSK with the $R_c = 1/2$ base code). Four curves are depicted, each for a different C/N_o. Each curve represents how the TCP throughput changes as a function of the information bit rate available to TCP when the redundancy varies on the parameters subspace. For each channel condition, an optimum combination of parameters exists which gives the maximum TCP throughput. In particular, we can observe that for the highest C/N_o

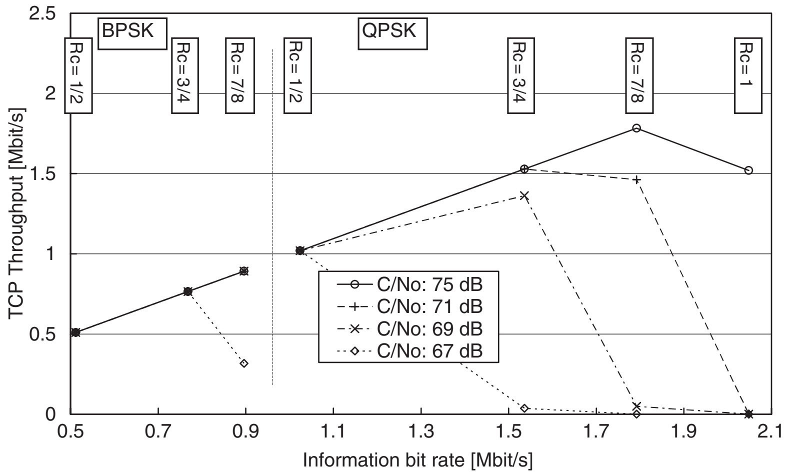

Figure 1. Throughput of a single TCP connection vs the available information bit rate for different C/N_o values; RTT = 520 ms, B_s = 1.024 Msymbol/s, TCP segment size = 1000 bytes, gateway buffer size = bandwidth-delay product.

considered, the maximum is achieved by making use of QPSK with $R_c = 7/8$ coding. By making use of a higher redundancy, the consequent PER improvement would not compensate the lowering of the available information bit rate; *vice versa* for a lower redundancy. Decreasing the quality of the channel, the optimum is achieved for lower information bit rates, as expected. Note that while the general trend holds for any TCP version, the exact optimization point will depend on the sensitivity to PER of the specific TCP version considered. Results reported in Figure 1 refer to NewReno (in order not to be limited by the advertised window, the window scale option [25] is activated in the receiver, as in all the other cases presented in the paper).

We conclude this section with some considerations regarding the applicability of the modulation and coding optimization, just described, to the recent DVB-S2 standard [26]. DVB-S2 considers very powerful error-correcting codes, such that an optimization based on channel coding would be useless because the curves that give the PER vs E_b/N_o are so steep that we have a sort of on–off behaviour of the physical channel: either the PER is negligible, or it is so high that it collapses TCP performance. However, when a satellite return channel is used [27], optimization on the return channel could be really useful. An additional possibility would be to insert a link-layer erasure code [28, 29] just above the MAC layer, which could be an all-software solution, independent of the underlying hardware characteristics.

4. TCP ENHANCEMENTS

In recent years, several solutions have been proposed for improving TCP performance over both wired and wireless links, including satellite connections. Some suggestions focus on limited modifications of standard procedures and/or on tuning TCP parameters [30, 31], while others envisage the introduction of alternative TCP procedures that replace or add to the standard algorithms. With reference to the former category, in Reference [19], the advantages of a larger slow start initial window (SS-IW) are pointed out, while in Reference [32] modifications of the slow start threshold and packet spacing are envisaged to prevent early buffer overflow. Along the same lines, in Reference [33] two minor modifications to the fast retransmit algorithm are suggested, as in Reference [20] the adoption of the SACK (selective ACK) option, to quickly recover from multiple losses in a window of data. As a general comment, although all these refinement are certainly useful, they are inadequate for really overcoming the problems posed by wireless and especially satellite links, which require more significant enhancements. The number of proposals characterized by the introduction of major TCP modifications is quite large, including among the others TCP Vegas, TCP Peach, TCP Westwood, and TCP Hybla. However, regarding TCP Vegas [34], it should be noted that it does not specifically address the main problems posed by satellite connections; therefore, its adoption in a satellite environment does not seem really promising. In contrast, the other TCP enhancements mentioned attempt to solve, or at least mitigate, the performance penalization which derives from long RTT and/or random errors; therefore, their specific aims and characteristics will be accurately discussed in the following subsections. Finally, some attention will be devoted to the Stream Control Transmission Protocol (SCTP) [11], which was initially designed for the transport of signalling packets for Voice over IP applications, but came to be progressively implemented as a general transport protocol for reliable data communication over challenging channels. Although it cannot be strictly considered to be a TCP enhancement, being alternative to TCP, it is included in this section because most of its procedures are closely related to TCP basic algorithms.

4.1. TCP Peach

TCP Peach is composed of two new algorithms, namely sudden start and rapid recovery, as well as the two traditional TCP algorithms, congestion avoidance and fast retransmit [8]. Sudden start and rapid recovery are designed to replace, respectively, the slow start and fast recovery algorithms of standard TCP. The new algorithms are based on the novel concept of using dummy segments to probe the availability of network resources without carrying any new information to the sender. Dummy segments are low-priority segments generated by the sender as a copy of the last transmitted data segment, i.e. they do not carry any new information to the receiver. The sender uses the dummy segments to probe the availability of network resources. If a router on the connection path is congested, then it discards the IP packets carrying dummy segments first. Consequently, the transmission of dummy segments does not cause a throughput decrease of actual data segments, i.e. the traditional segments. If the routers are not congested, then the dummy segments can reach the receiver. The sender interprets the ACKs for dummy segments as evidence that there are unused resources in the network, and can increase its transmission rate accordingly. By exploiting dummy segments, sudden start provides a fast opening of the congestion window at the beginning of the connections independently of the actual RTT, while rapid recovery can counteract the effect of random errors due to a noisy channel. The main requirement of TCP Peach is that the network routers implement some forms of class-based queuing (CBQ) policy, to differentiate between high- and low-priority traffic, i.e. between data packets and dummy segments.

The advantages offered by TCP Peach in a satellite environment experiencing strong losses are depicted in Figure 2, where TCP Peach and TCP NewReno behaviours are compared by highlighting the dynamics of transmission of the TCP Peach segments. We can notice that at the beginning of the transmission, TCP Peach operates in the sudden start phase, which exploits the

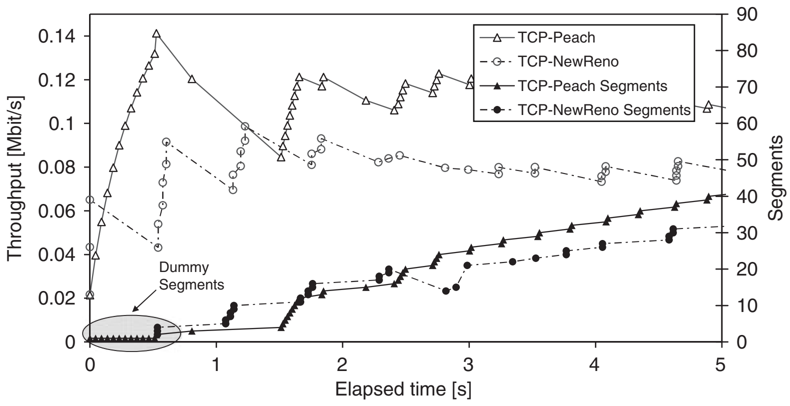

Figure 2. TCP Peach behaviour; RTT = 520 ms, PER = 1%, channel capacity = 2 Mbit/s, TCP segment size = 1448 bytes, gateway buffer size = bandwidth-delay product.

transmission of dummy segments in order to probe the available network resource, and thus to guarantee the maximum data rate achievable on the satellite channel. When a packet loss is detected by means of three duplicated ACKs, TCP Peach performs the fast retransmit algorithm, as TCP Reno and TCP NewReno would. Once the retransmission has completed, the rapid recovery phase takes place: in this case, unlike standard TCP, for each received ACK two dummy segments are sent in order to evaluate the network available bandwidth. This difference is the key element in the improvement of the performance with respect to standard TCP. The advantage is that when the regular transmission restarts, ruled by the congestion avoidance algorithm, the arrival of the ACKs acknowledging the dummy segments transmitted during the rapid recovery phase cause a fast increase of the congestion window, hence exploiting the channel bandwidth more effectively than performed by standard TCP.

4.2. TCP Westwood

TCP Westwood [9, 35] was introduced for the purpose of limiting the consequences of the losses introduced by a wireless channel. To this end, TCP Westwood introduces a modification of the fast recovery algorithm called faster recovery. Instead of halving the congestion window after three duplicate ACKs, and fixing the slow start threshold to this value, TCP Westwood sets the ssthresh as a function of the estimated available bandwidth. In this way, channel losses do not cause the dramatic slow-down of the transmission rate typical of the standard versions. The bandwidth is estimated by measuring and averaging the rate of returning ACKs. In particular, the reception of an ACK at the time t_k implies that an amount of data d_k has been received by the TCP receiver. Therefore, the kth sample of bandwidth used by a given connection is measured as

$$b_k = \frac{d_k}{t_k - t_{k-1}} \tag{7}$$

where t_{k-1} is the arrival time of the previous ACK. Finally, TCP Westwood makes use of a discrete-time smoothing filter to perform the estimate of the available bandwidth; the value of the filtered available bandwidth b_k at time $t = t_k$, is given by

$$\hat{b}_k = a_k \cdot \hat{b}_{k-1} + (1 - a_k)\left(\frac{b_k + b_{k-1}}{2}\right) \tag{8}$$

where $a_k = (2\tau - \Delta t_k)/(2\tau + \Delta t_k)$ depends on the inter-arrival time ($\Delta t_k = t_k - t_{k-1}$) which is variable, and $1/\tau$ is the cut-off frequency of the filter. Then, after loss detection, TCP Westwood sets the ssthresh and cwnd as follows:

$$\text{ssthresh} = \hat{b} * \text{RTT}_{\min}$$
$$\text{cwnd} = \begin{cases} \text{ssthresh} & \text{if cwnd} > \text{ssthresh} \\ 1 & \text{after a timeout} \end{cases} \tag{9}$$

Nevertheless, in the case of 'drop-tail' routers, where 'bursty' traffic can be observed, such a bandwidth estimation method tends to over-estimate the connection fair share. To achieve high link utilization and, at the same time, guarantee friendliness, TCP Westwood implements a novel congestion control mechanism based on eligible rate estimation (RE) rather than bandwidth estimation [36]. Briefly, an alternative available bandwidth sample is defined as the amount of data delivered in the last T time interval, divided by T. Furthermore, an updated release of Westwood [37] implements one more mechanism, called 'adaptive start' (ASTART), to adjust the ssthresh in the start-up phase, according to the bandwidth estimation. In fact, by setting the initial ssthresh to an arbitrary value, TCP may suffer from two different problems:

1. If the initial ssthresh is too low, compared to bandwidth-delay product, TCP prematurely switches to the Congestion Avoidance phase, leading to inefficient bandwidth utilization.
2. If the initial ssthresh is too high, the exponential cwnd increase of the slow start phase may cause multiple losses at the bottleneck router, with a series of long recovery phases often followed by timeouts.

The behaviour of TCP Westwood plus the ASTART mechanism over an end-to-end satellite link with frequent channel losses (PER = 1%) is shown in Figure 3. The graph aims to highlight

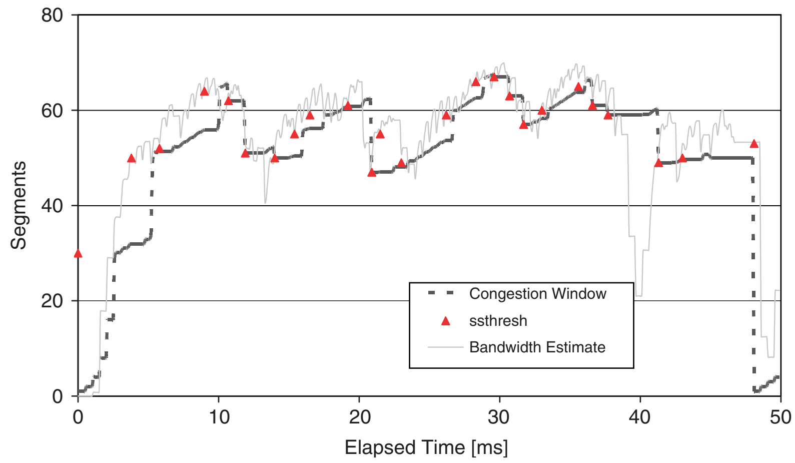

Figure 3. TCP Westwood plus ASTART behaviour; RTT = 520 ms, PER = 1%, channel capacity = 2 Mbit/s, TCP segment size = 1448 bytes, gateway buffer size = bandwidth-delay product.

the mechanism of dynamic setting of the congestion window and of the slow start threshold in both the start-up phase (ASTART algorithm) and after a loss event (faster recovery algorithm). Specifically, whenever a packet loss occurs, ssthresh (markers) and cwnd are set equal to the current bandwidth estimation. In this way, TCP Westwood avoids unnecessary cwnd reductions (in the example provided all the losses are due to adverse channel conditions).

4.3. TCP Hybla

TCP Hybla [10, 38] has been conceived with the primary aim of counteracting the performance deterioration originated by the long RTTs typical of satellite connections. It consists of a set of procedures that includes an enhancement of the standard congestion control algorithms for both the slow start and the congestion avoidance phases, the mandatory adoption of the SACK policy, the adoption of Hoe's channel bandwidth estimation, the use of timestamps and the implementation of packet spacing techniques.

The modification of the standard congestion control rules is dictated by the TCP Hybla ideal aim of obtaining for long RTT connections the same instantaneous segment transmission rate of a comparatively fast reference TCP connection (e.g. a wired one). To derive the new laws, the adoption of a continuous model to describe the congestion window evolution in time, $W(t)$ (expressed in SMSS units), proved essential, making it possible to rigorously define the segment transmission rate (i.e. the amount of segments transmitted per second) as

$$B(t) = W(t)/\mathrm{RTT} \tag{10}$$

Equation (10) shows that in order to achieve the aforementioned goal of a transmission rate independent of the actual RTT, a *faster* $W(t)$ increase rate for long RTT connections is required, by contrast with what actually happens in standard TCP, where long RTTs in fact result in a slower increase rate. Before presenting the new congestion control rules, it is necessary to introduce a new parameter, the normalized RTT, ρ, defined as the ratio between the actual RTT and the RTT of the reference connection to which TCP Hybla aims to equalize the performance, denoted by RTT_0 (see Reference [10]),

$$\rho = \mathrm{RTT}/\mathrm{RTT}_0 \tag{11}$$

In the references it is shown that the same $B(t)$ of the reference connection can be achieved by longer RTT connections provided that the standard congestion control algorithms are replaced by the following, which represent the TCP Hybla congestion control rules

$$W_{i+1} = \begin{cases} W_i + 2^{\rho} - 1 & \mathrm{SS} \\ W_i + \rho^2/W_i & \mathrm{CA} \end{cases} \tag{12}$$

Note that the congestion avoidance update rule is similar to the constant-rate algorithm [38], which, on the other hand, did not consider the slow start phase. Another important difference is that in the actual implementation of (12), TCP Hybla sets the minimum value for ρ to 1, in order to avoid slowing down the connections that are already 'fast' (i.e. connections with $\mathrm{RTT} < \mathrm{RTT}_0$). Bandwidth estimation is exploited in order to appropriately set the initial *ssthresh*.

As far as the other features are concerned, we note that, since the congestion control algorithm of TCP Hybla is much more efficient than standard TCP in guaranteeing a satisfactory transmission rate for long RTT connections, a larger average cwnd has to be

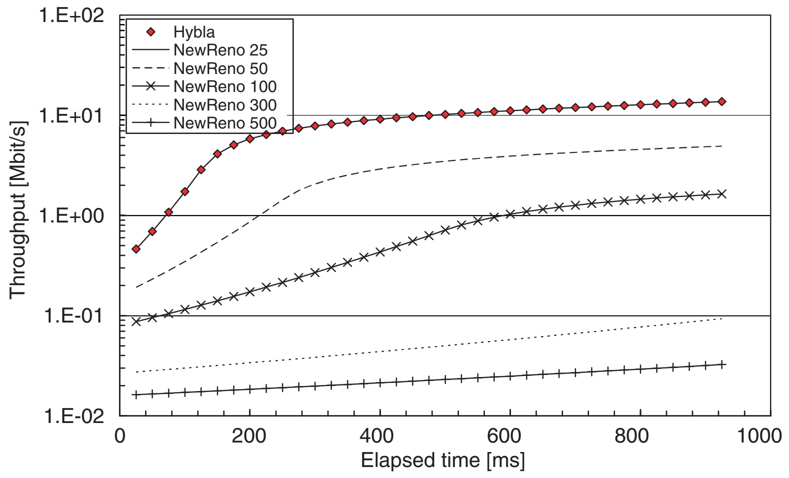

Figure 4. TCP Hybla vs standard TCP: independence of TCP Hybla performance of the actual RTT value; initial ssthresh = 32 kB, $RTT_0 = 25$ ms, PER = 0, TCP segment size = 1448 bytes; analytical data referring to an ideal channel.

expected. As a result, multiple losses in the same window will be more frequent, which suggests considering the SACK policy mandatory. Large cwnds also frequently cause severe inefficiencies of the 'exponential back-off' RTO policy. These can be avoided by exploiting timestamps [18], at present largely implemented in the most widespread operating systems. Finally, the adoption of packet spacing [39] has been envisaged to counteract the burstiness due to large congestion windows, with the consequent possible large losses at intermediate router queues.

In Figure 4 the different behaviour of TCP Hybla and TCP NewReno in the presence of different RTTs is depicted, considering the amount of transmitted data from the connection start, on an ideal channel (analytical results). In the example, which considers the initial slow start phase followed by the congestion avoidance phase (ssthresh = 32 kbytes), RTT_0 has been set to 25 ms for comparison purposes, to show that TCP Hybla ideal performance equals that of a NewReno connection with RTT = 25 ms, independently of its actual RTT. This theoretical independence of the TCP Hybla performance of the RTT is shown in the figure by the overlapping of TCP Hybla markers (the same for every RTT) with the NewReno 25 continuous line. The other TCP NewReno curves are reported to show its performance dependence on RTT. Of course, the choice of the 'right' value for RTT_0 for TCP Hybla may be an issue, at least in open networks, where path delays of terrestrial connections vary significantly (see Reference [10] for a detailed discussion on the implications and suggested criteria for the RTT_0 choice). Finally, we refer the reader to Reference [40] for TCP Hybla comparative performance on real channels.

4.4. SCTP

The stream control transmission protocol (SCTP) [11, 41] was designed to accomplish signalling transport over IP networks for voice over IP (VoIP) applications. However, it was soon noticed that SCTP could be successfully employed in a wider range of applications than just for signalling transport. In particular, some features could be useful for overcoming the limitations of TCP in wireless and satellite environments. The design of SCTP retains some basic standard features of TCP, such as window-based congestion control, error detection, and retransmission, which proved robust in the Internet environment. In addition to introducing some refinements to them, as shown below, SCTP also adds several new features that are not available in TCP, in particular multi-homing and multi-streaming. Multi-homing allows two endpoints to set up an association with multiple IP addresses for each endpoint (in SCTP, 'association' is the name for the communication relationship between endpoints). In this way, it is possible to introduce a sort of space diversity in the network, which may be useful for coping with long link failures without interrupting data transfer. For example, a satellite channel can be used as secondary path to backup a terrestrial link (primary path). Nevertheless, the management of the handover process is quite challenging, because the time necessary to detect a link failure results into large handover delay [41] (for instance, in Reference [42], the average handover delay has been estimated in 15 s). Multi-streaming is used to alleviate some problems that stem from the TCP's strict byte-order delivery policy. In SCTP each stream is a sub flow within the overall data flow, and the delivery of each sub flow is independent of the others. Other improvements introduced in SCTP are related to security issues. Before concluding, it might be useful to describe in detail the main refinements introduced by SCTP in the standard mechanisms, as follows:

- SCTP incorporates a fast retransmit algorithm based on SACK gap reports similar to that of TCP SACK. In contrast to standard TCP, the use of SACK is mandatory (as it is in TCP Hybla) allowing better performance in the case of multiple losses from a single window of data.
- During slow start or congestion avoidance of SCTP, the congestion window (cwnd) is increased by considering the number of acknowledged bytes and not the number of ACKs received, improving the accuracy of the congestion control.
- During congestion avoidance of SCTP, cwnd can only be increased when the full cwnd is utilized; this restriction does not exist in standard TCP. However, it should be noted that a similar mechanism is present in the Linux implementations of TCP and TCP Hybla.

5. PERFORMANCE ENHANCEMENT PROXIES

The shortcomings arising from the application of TCP-based services over satellite environments have been discussed in detail in the previous sections, together with some countermeasures that can really enhance TCP performance in such environments. However, in most scenarios there is the need to keep the protocol stack on the end terminals unchanged, at least in a short-to-medium-term perspective. It should be noted, for instance, that in the most common commercial operating systems, the user has basically no control over the TCP variant to be used. In this case, performance optimization can only be carried out by acting on the satellite segment of the network. For this purpose, a great deal of research was performed in the late nineties, resulting in a set of possible architectures and schemes able to balance the TCP drawbacks and assure performance optimizations at the cost of increased hardware complexity and (at least in some cases) also of severe semantics violations. These solutions are generally referred to as PEPs [6]. In this section, the most promising PEP techniques (namely spoofing and splitting) are discussed, by presenting for the latter a protocol architecture that makes use of specialized transport protocol solutions for satellite links. Finally, this section is completed by a

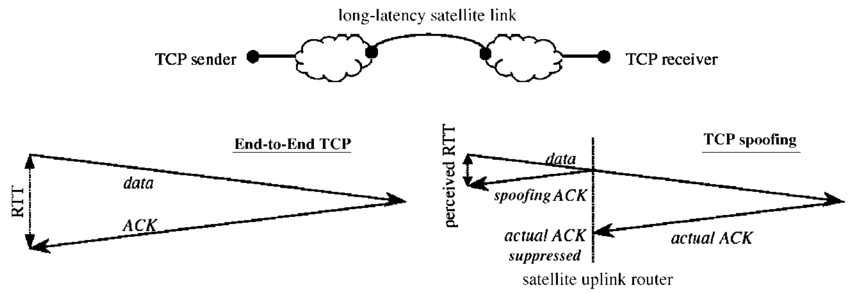

Figure 5. TCP Spoofing application over satellite [14].

brief review of other basic techniques usually employed in TCP PEP architectures. Many of them are also described in Reference [43], which deals with asymmetric paths.

5.1. *Spoofing*

TCP spoofing [14] is a viable solution for mitigating the performance penalization that stems from long RTTs. It consists in accelerating the growth of the TCP transmission window by masking the long delay experienced on the satellite link (Figure 5). More specifically, the satellite gateway, transparently to the end hosts, is responsible for generating automatically fake ACKs in correspondence to the incoming TCP segments. In this way, the latency needed for the hosts to transmit new segments is the sole delay experienced by the terrestrial link, excluding the satellite portion. The main drawback of this approach is that the end-to-end semantics is no longer respected, creating problems and inefficiencies for the applications, which on the contrary rely upon this characteristic. Moreover, the adoption of this scheme implies that the 'true' ACKs generated by the destination host have to be intercepted by the satellite gateway and then suppressed in order to avoid spurious duplicate ACKs arriving at the source side; in the case of lost segments, the gateway will be also be responsible for retransmitting the missed segments. For this task, it is also necessary that a symmetric routing be performed; in other words, the TCP data segments and the related ACKs have to pass through the same paths. If this is not the case, the satellite gateway, responsible for performing spoofing operations, will not be able to effectively manage the TCP connections on the satellite link.

5.2. *Splitting*

The rationale behind the splitting approach [14] is to separate the satellite portion from the rest of the network in order to have dedicated and distinct connections. In practice, this approach can be realized by considering splitting functionalities acting on the satellite gateways. In this way, the TCP connection established between the hosts is split into three separate connections, of which two are over the paths linking the terminal hosts with the satellite gateways; and the third between the gateways over the satellite portion. From the network point of view, a new architecture is defined and its added value consists in the possibility of hiding the satellite link to the end users, which continue to use the TCP stack for connecting to satellite gateways, while between them it is possible to take advantage of a specifically designed transport protocol. In the following, an example of such architectures, namely the satellite protocol stack (SPS) architecture is given, and the role of the agents involved in the communication is also highlighted.

5.2.1. The satellite protocol stack (SPS) architecture. The general architecture is reported in Figure 6. The network is composed of terrestrial components, represented by the Internet in the figure, and of a satellite component (a backbone, in this case). The latter is isolated from the rest of the network by using relay entities. The transport layer of this new SPS is called satellite transport layer (STL) and it implements a satellite transport protocol (STP), suited for the specific environment.

The architecture proposed may be a valid alternative also in the case that the satellite component represents an access network, as depicted in Figure 7. The relay entity is a simple tool, directly attached to the application PC. It may also be a hardware card inside the application PC, as a network or a video card. It is straightforward that the relay entity module can be embedded into a software module in order to load it directly into the devices.

From the protocol layering point of view, the key point is represented by the two relay entities, which are two gateways towards the satellite portion of the network. The SPS acts on the satellite links by using the necessary information because it has the knowledge and the control of all the parameters. The relay layer guarantees the communication between the satellite transport layer and the protocol used in the cable part (i.e. TCP).

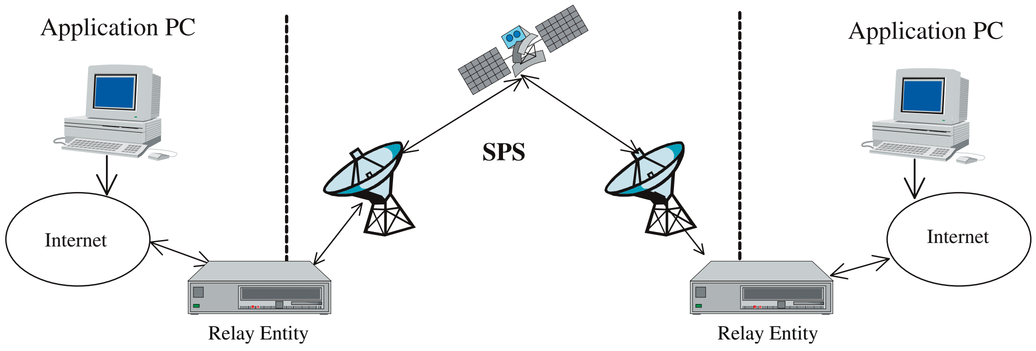

Figure 6. SPS architecture; backbone case.

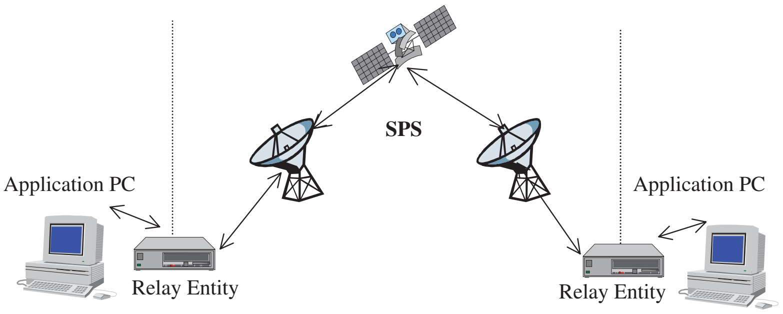

Figure 7. SPS architecture; access network case.

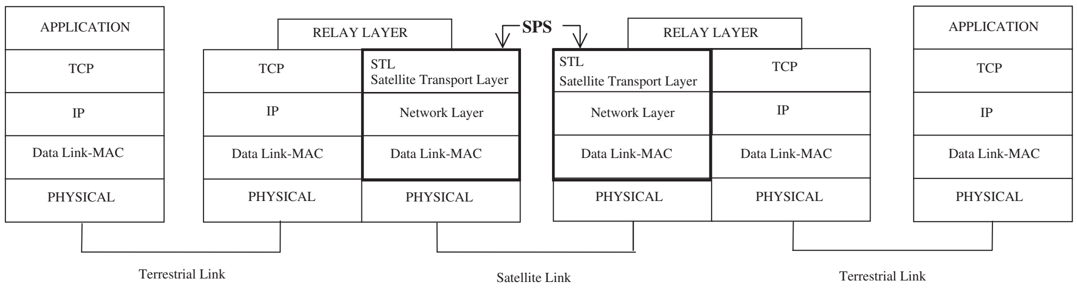

Figure 8. SPS protocol architecture.

On one hand, the proposed architecture (shown in Figure 8) exploits the capability of adopting on the non-wired network portion a transport protocol designed specifically for the satellite environment. On the other hand, it breaks the end-to-end semantics, delivering to the intermediate gateways (namely the relay entities as indicated in Figures 6 and 7) the responsibility for checking the overall communication reliability and maintaining the connection state. The lack of end-to-end semantics implies that, even if a TCP connection is established between the peers at the beginning of the communication, actually three separate connections are in place, namely between first application PC and first relay entity, between first and second relay entities, between second relay entity and second application PC.

In more detail, taking Figure 8 as reference, the connection at the transport layer is divided into three parts, dedicated to the two wired links and the satellite, respectively. The source receives the ACK from the first relay entity, which opens another connection, with different parameters based on the current status of the satellite portion, and allocates the resources available. The relay entity on the other side of the satellite link operates similarly towards the destination, while the transport layer of the cable portions is untouched.

5.2.2. Optimized satellite transport protocols. In the following, three transport protocols that can be used to connect the two relays entities, namely the ‘Xpress Transport Protocol’ (XTP), the ‘Space Communications Protocol Standards–Transport Protocol’ (SCPS-TP), and the Complete Knowledge Satellite Transport Protocol (CK-STP) will be described as significant examples of optimized transport protocols. Note, however, that they can also be applied alone, whenever an end-to-end satellite only connection is considered.

5.2.2.1. Xpress transport protocol. The XTP is a transport protocol designed for the long-latency, high-loss, and asymmetric bandwidth conditions that are typical of satellite communications [15]. This protocol was designed to allow the application to easily select the type of service requested. For instance, by choosing the corresponding configuration, XTP is able to provide the application with the same service usually provided by TCP or UDP (but with a better efficiency). For example, in its TCP-like virtual-circuit mode, XTP applies the Selective Retransmission algorithm for loss recovery. Basically, when the receiver detects gaps in the received packets sequence, it transmits to the sender a list of the missing packets, allowing the sender to quickly and efficiently carry out all the retransmission required. Furthermore, XTP makes available additional services and features, such as a reliable multicast protocol, a rate and burst control and in general the flexibility to match the service to the specific application needs.

Table II. SCPS layered protocols.

SCPS file protocol (SCPS-FP)
SCPS transport protocol (SCPS-TP)
SCPS security protocol (SCPS-SP)
SCPS network protocol (SCPS-NP)

XTP regulates the data flow by means of an end-to-end windowing flow control mechanism, based on 64-bit sequence numbers and a 64-bit sliding window. XTP also provides rate control whereby an end-system or an intermediate system can specify the maximum bandwidth and burst size it can accept. Finally, the proposed error control scheme is based on both positive and negative ACKs, to effect retransmission of missing or damaged data packets. Retransmissions can be performed either by making use of 'go-back-*N*' or 'selective repeat' policies.

5.2.2.2. Space communication protocol standard. The SCPS project aims to provide a suite of standard protocols optimized for space communications [16, 43, 44]. The SCPS protocol stack is shown in Table II.

Each protocol is compatible with standard Internet protocols. The SCPS Transport Protocol is based on TCP and UDP plus some extensions addressed to space requirements, as:

1. 'TCP extensions for high performance' [25];
2. 'TCP for transactions' [45];
3. 'Selective negative acknowledgements' [46];
4. 'Header compression' [47].

In order to cope with losses due to data corruption, link outages and congestion, SCPS-TP implements different error recovery strategies. For instance, when an isolated data loss occurs, the transmission rate is not decreased and the RTO value is left unchanged ('explicit corruption response'). Regarding congestion control, it can be performed by applying either standard mechanisms (i.e. slow start, congestion avoidance, RTO exponential back-off) or the TCP Vegas congestion control mechanism. Optionally, SCPS-TP congestion control can be disabled.

5.2.2.3. CK-STP. CK-STP stands for Complete Knowledge-Satellite Transport Protocol, where the complete knowledge philosophy indicates the possibility of completely knowing the main characteristics of the networks in which the data communication is performed (i.e. available bandwidth and propagation delay) [12]. Moreover, assuming that in satellite networks (more generally in wireless environments) congestion events may be relatively infrequent (otherwise explicit countermeasures, such as explicit congestion notification techniques can be taken), the loss of packets is mainly due to channel errors.

In order to ensure an effective exploitation of the network available resources in terms of bandwidth capabilities, the initial window is set to the bandwidth-delay product and the congestion window value is no longer increased when the acknowledgments arrive. Moreover, the TCP buffers on the receiver and sender sides must be properly tuned in order to make the abovementioned modifications effective. Also, the TCP recovery mechanism has been modified by entering fast retransmit once a duplicated acknowledgment arrives. This proposal's main

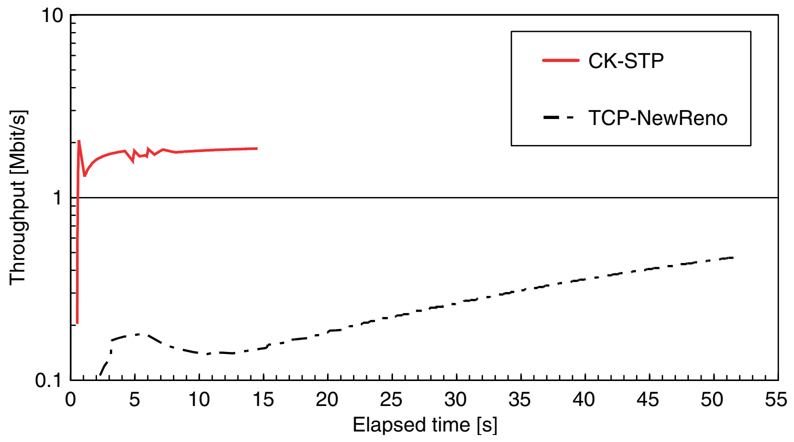

Figure 9. CK-STP vs TCP NewReno: transfer of a 3 MB file; RTT = 520 ms, PER = 1% in the first 10 s only; channel capacity = 2 Mbit/s, TCP segment size = 1448 bytes, gateway buffer size = bandwidth-delay product.

improvement is that at the end of the recovery phase, the congestion window value is not reduced, and as a consequence, the transmission rate is maintained at the value set at the opening of the TCP connection in order to fill the bandwidth pipe. Timeout management has been slightly modified, too. In particular, when a timer expires the congestion window is not reduced to one segment, but once the retransmission phase is terminated the data communication continues with the maximum possible rate as in the previous case.

The advantages of such an approach are immediate if compared with the TCP behaviour. In fact, while TCP shrinks the congestion window once a lost packet is detected, CK-STP leaves the cwnd unmodified, thus assuring high-performance results. In Figure 9, CK-STP is compared with TCP-NewReno in a satellite environment experiencing losses in the temporal window 0–10 s. The figure shows that independently of the losses, CK-STP is able to react effectively and to fully take advantage of the available channel bandwidth, equal to 2 Mbit/s.

5.3. *Other TCP PEP techniques*

Spoofing/splitting techniques are often considered synonymous for TCP PEP. However, although they are the most common ones, other PEP mechanisms have been defined [6, 48]. In fact, some aspects related to satellite links (i.e. the effects of large segment bursts, or of network path asymmetry) require the adoption of further enhancements. This section lists some of the most common mechanisms applied in TCP PEP:

- *ACK spacing*. In environments with a large bandwidth-delay product, the ACKs tend to reach the TCP source bunched, causing undesirable segment bursts. In fact, the transmission of large bursts may lead to the loss of many segments in the gateway buffer, triggering very long and underperforming error recovery phases. To avoid this problem, the ACK spacing technique smoothes out the flow of the TCP ACKs.
- *ACK filtering and reconstruction*. In the case of links with very high bandwidth asymmetry, the ACK flow in the low-speed direction may experience loss due to congestion events, limiting the transmission speed in the broadband forward link. The ACK filtering and

reconstruction mechanism allows reducing the number of ACK traversing the return link, by filtering them at one side of the link. Then, the correct ACK sequence will be reconstructed at the opposite side of the link, before reaching the TCP source.

- *Tunnelling*. Tunnelling is a PEP technique that encapsulates the messages in order to force them to traverse a particular link. A PEP agent installed at the end of the 'tunnel' will remove the encapsulation wrappers before delivering the message to the final end-system. Such a technique might be very useful in the case of split connections to guarantee the transit of the packets across the distributed PEP agents.
- *Compression*. The compression technique aims to reduce the number of bytes to send across a link. Then, in the case of bandwidth-limited links, such a technique leads basically to the following benefits: improvement of the link efficiency, latency reduction and PER reduction.
- *Handling periods of link disconnection with TCP*. Wireless links can be affected by outage periods, which have dramatic effects on the TCP performance. In fact, when the TCP sender does not receive the expected ACKs, a retransmission timeout occurs and the congestion window is consequently shut down to one segment. To mitigate this problem, a PEP TCP agent may monitor the traffic coming from the sender towards the receiver, in order to always keep the last ACK and eventually use it to set the TCP advertised window to zero, if a disconnection period is detected. In this way, the TCP sender will go into persist mode (in practice the connection is 'frozen'), which prevents the RTO timer from expiring.
- *Priority-based multiplexing*. Typically, a link is shared by connections supporting applications with different characteristics. The goal of the priority-based multiplexing is to privilege the 'urgent' data transfers (i.e. interactive applications), by delaying simultaneous low-priority bulk-transfers.

6. THE DELAY TOLERANT NETWORKS

The shortcomings deriving from the application of TCP-based transport protocols over satellite networks are more pronounced and critical in the case of environments where larger delays and higher BERs are experienced. In general, the transportation of data over the so-called challenged networks requires the adoption of *ad hoc* protocol architectures aimed at reducing the impact of the transmission channel impairments. In this view, it relates to protocol solutions conceived for the higher layers, namely over the transport layer. In particular, the advent of DTN may open up new pathways for communication in hazardous environments and extends the concept of networking.

6.1. The DTN concept

The DTN architecture addresses the issue of transporting data in so-called challenged environments. Indeed, experience has shown that legacy TCP is defeated when networks include links/nodes that are intermittent, error-prone or have a large propagation delay. The DTN architecture proposes a transport service based on store/forward relay of 'bundles' (application data) in order to cope with these conditions. The concept of DTN was first conceived within the Inter Planetary Network Research Group (IPNRG) of the Internet Research Task Force

(IRTF), to specifically address the problems of space communications. Later, its field of applicability was enlarged to encompass all the 'challenged networks', either spatial or terrestrial. To this end, in 2002 the IRTF Delay Tolerant Networks Research Group (DTNRG) was established. Among the various documents published by this group, worth mentioning are the bundle protocol specification, which has now reached the third version of the draft standard, and the DTN API implementation, whose latest release is the 2.1.1, as well as several independent projects demonstrating typical usages of DTN technology. The main reference is the DTN Research Group web page [49].

The basic idea behind DTN is the subdivision of a large and heterogeneous network in homogeneous regions and in the introduction of a new layer, namely the bundle layer, between application and transport layers. Long end-to-end connections at the bundle layer, which have to go through many homogeneous sub-networks, are then split into many segments, each of which is carried out on a homogeneous sub-network as a common end-to-end connection at the transport layer. At first glance this idea resembles the splitting architecture, described in the previous section, and actually it can be seen as a powerful extension of it. However, there are many differences. First, the large heterogeneous network can be split into more than two regions. Second, there is no implicit (and problem-prone) violation of the end-to-end semantics of the transport protocol. In DTN, real end-to-end control is assigned to the new bundle layer. The transport protocol is now explicitly assigned the task of end-to-end control over the same sub-network. This clear subdivision leads, in a very natural and convenient way, to the use of TCP versions optimized for the specific characteristics of (usually intermediate) sub-networks, or the use of a completely different stack. In this way, the advantages of the different approaches seen in the previous sections (i.e. TCP enhancements, splitting architectures, optimized satellite transport protocols) can be retained and augmented, while most of the disadvantages are actually dropped. More details regarding the DTN concept and architecture are provided in the following.

6.2. The DTN architecture

The DTN core is represented by the bundle protocol, which runs on top of a traditional protocol stack. A convergence layer ensures a smooth interface between the bundle layer and the underlying stack, which may be, or may not be, the usual TCP/IP stack. Bundles are sent, forwarded and received by bundle agents. Each agent is assigned an identifier, called tuple, used (among other things) for routing. It is composed of two parts: a region ID and an agent ID. The agent ID is unique within one region and is only used when the tuple reaches a node of the destination region (late binding). Bundle transport is performed in store and forward mode. Each agent participating in the DTN has the possibility of storing the bundle if the need arises (custodial service).

The delay tolerant networking architecture is shown in Figure 10. In particular, we can point out that the DTN architecture allows partitioning the whole network in separated regions, which adopt their own protocol stack (indicated in figure as region-specific layers). In this view, we can say that the DTN approach creates a 'network of networks' exchanging data between the application hosts based on the following paradigm. On one hand, the DTN nodes (edge gateways) assure the end-to-end reliability of the communication by means of the custody transfer and of the bundle retransmission functions implemented in the bundle layer. On the other hand, the communication achieved within the 'inner' network is ruled by the

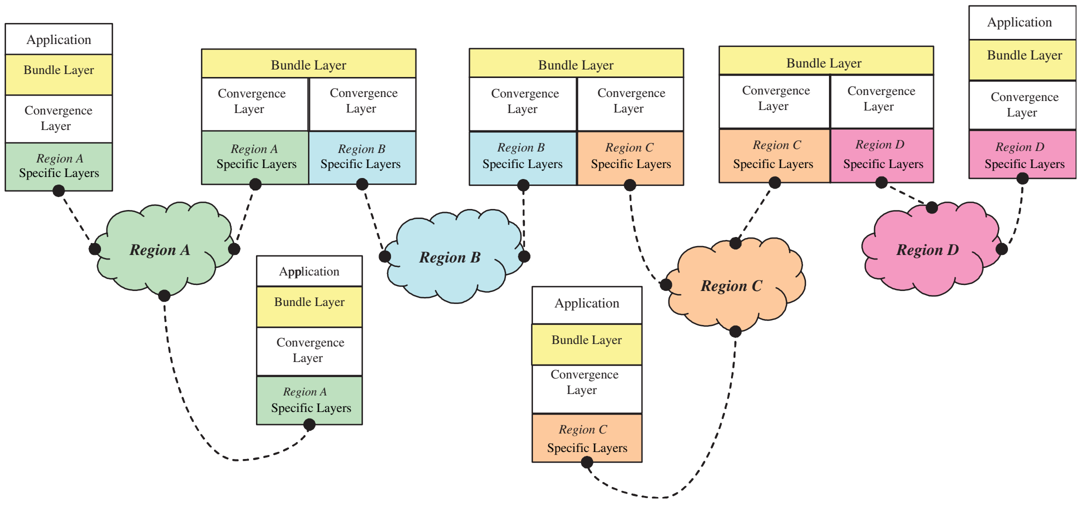

Figure 10. DTN protocol architecture.

region-specific layers. It is straightforward that in general also TCP/IP protocol suite may be adopted in the regions. In this case, it is important to state some analogies with the splitting architecture presented in the previous section. Actually, we can point out that both architectures are able to partition the network into separate portions, in order to adopt an '*ad hoc*' protocol stack within each. Some differences are also present. In the case, of the splitting architecture, the application hosts are not aware of the splitting/spoofing operations performed in the middle; moreover, the intermediate agents are responsible for intercepting the TCP connections, and consequently violate the end-to-end semantics. On the other hand, in the DTN architecture the application hosts implement the bundle layer and a converge layer: hence, they are 'aware' that the TCP connection will be established with the first DTN gateway present on the path. Furthermore, in this perspective each region delimited by DTN gateways acts as a separate administrative domain, in which naming, routing and security issues are managed independently of the other regions.

The main advantage of the described architecture is that it copes well with environments where end-to-end connectivity is not always ensured. However, delayed communication does not mean unreliable communication. The bundle protocol provides for delivery and forward notifications. Other transport features include urgent data delivery, bundle fragmentation and security services (authentication). Typical scenarios of application, which are strictly correlated with satellite environments, are represented by interplanetary networks and more in general by deep-space communications. In this view, the need for a protocol technology able to store the information and to propagate it towards the next hops only when the link is assumed to be reliable is of fundamental importance. This necessity is justified by the fact that interplanetary links experience very high BERs, and hence the transmission of data towards the destination is advisable only when the transmission medium is available. Moreover, the storage into local units avoids a great number of retransmissions otherwise required whenever the channel quality is very low; in this way, issues regarding power consumption are addressed as well. Actually, the possibility of delaying the transmission, and hence of reducing considerably the number of necessary retransmissions leads to a significant power saving. On the other hand, the adoption of DTN architecture, and specifically the custodial service, may introduce the problem of congestion events on the bundle layer buffer. This problem is not unusual and may be experienced once a transmission link is not available and the DTN gateway has to store data until the communication link is available again. Given the long duration of link outage in the deep-space links, the DTN buffer will likely suffer overflow events and hence will drop several bundles. Actually, the design of congestion control schemes suitable for the DTN is being studied. Nevertheless, its investigation is not trivial because of the peculiarities of the analysed scenario. In particular, the long delays experienced make the application of closed loop schemes unfeasible. Moreover, the sole monitoring of the buffer occupation is not sufficient to cope with this problem. To this end, the additional monitoring of the variation in time of the buffer queue occupation and of the incoming traffic load could bring significant advantages.

Another important issue to be addressed is the reaction to bundle loss events. In this case, if the DTN service is properly configured, the bundle layer itself will be responsible for retransmitting the missed information. A possible alternative is represented by recovery mechanisms implemented in the underlying layers. A possible candidate for performing this task is represented by the Licklider Transmission Protocol (LTP), which is essentially a point-to-point protocol and inherits its basic procedures from CCSDS file delivery protocol (CFDP) specification [50]; it is briefly described in the following.

6.3. *Licklider Transmission Protocol*

The (LTP) is designed to provide retransmission-based reliability over links in challenged Internet environments exhibiting extremely long RTT, frequent interruptions in connectivity, and high BERs [51]. Since communication across interplanetary space is the most prominent example of this sort of environment, LTP is principally aimed at supporting 'long-haul' reliable transmission in interplanetary space, although it may have applications in other environments as well.

In an interplanetary Internet, LTP is intended to serve as a reliable convergence layer over single hop deep-space RF links, working actually as a data link layer protocol directly over the physical interface. LTP implements ARQ of data transmissions by soliciting selective-acknowledgment reception reports.

From the architecture point of view, the LTP protocol has been designed to work under the bundle protocol in order to assure reliable communications whereas the higher layers might not support recovery or retransmission functions. In particular, the application of such a solution may be envisaged in all the cases where the bundle protocol is set to only perform store and forward operations, neglecting possible operations of bundle retransmission, and hence operating in unacknowledged mode. According to this view, the LTP protocol will be responsible for guaranteeing the communication robustness on a point-to-point basis, exploiting the procedures of 'retransmission' and 'accelerated retransmission'.

7. CONCLUSIONS

The study and analysis of transmission mechanisms for satellite communication are demanding tasks, since they require the definition and the implementation of innovative protocol and architecture solutions well suited to the specific environment. In this paper, after showing how

TCP can benefit from lower-layer optimization, particular attention was focused on the transport protocol solutions whose specification is strictly inherited from TCP, such as TCP Peach, TCP Westwood, TCP Hybla and CK-STP, and SCTP. They address, with different aims and different solutions, the main problems posed by satellite links. Then, different architecture technologies were investigated, considering two significant PEP examples, such as TCP spoofing and TCP splitting. Their pros and cons were highlighted, taking into account performance improvement, complexity, and compliance with the standard semantics. Finally, emergent technologies, such as the DTN architectures, which can guarantee a high degree of reliability in network scenarios affected by a high unpredictability and unavailability of resources, were explored. Thanks to them, the different approaches investigated in the paper can be integrated and the advantages increased accordingly. In a medium-to-long-term perspective, they may represent a definitive solution to the transport-level challenges posed by satellite communications.

REFERENCES

1. Hu Y, Li V. Satellite-based internet: a tutorial. *IEEE Communication Magazine* 2001; **39**(3):154–162.
2. Xylomenos G, Polyzos GC, Mahonen P, Saaranen M. TCP performance issues over wireless links. *IEEE Communications Magazine* 2001; **39(**4):52–58.
3. Barakat C, Altman E, Dabbous W. On TCP performance in a heterogeneous network: a survey. *IEEE Communication Magazine* 2000; **38**(1):40–46.
4. Fairhurst G, Wood L. Advice to link designers on link Automatic Repeat reQuest (ARQ). *IETF RFC 3366 (BCP62)*, August 2002.
5. Satellite Network of Excellence, web site: http://www.satnex.org/.
6. Border J, Kojo M, Griner J, Montenegro G, Shelby Z. Performance enhancing proxies intended to mitigate link-related degradations. *IETF RFC 3135*, June 2001.
7. Cerf V *et al*. Delay Tolerant Network Architecture. draft-irtf-dtnrg-arch-04.txt, December 2005.
8. Akyldiz IF, Morabito G, Palazzo S. TCP-Peach: a new congestion control scheme for satellite IP networks. *IEEE/ACM Transactions on Networking* 2001; **9**(3):307–321.
9. Casetti C, Gerla M, Mascolo S, Sanadidi MY, Wang R. TCP Westwood: end-to-end congestion control for wired/wireless networks. *Wireless Networks Journal* 2002; **8**(4):467–479.
10. Caini C, Firrincieli R. TCP Hybla: a TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking* 2004; **22**(5):547–566.
11. Fu S, Atiquzzaman M. SCTP: state of the art in research, products, and technical challenger. *IEEE Communication Magazine* 2004; **43**(4):64–76.
12. Marchese M. TCP/IP-based protocols over satellite systems: a telecommunication issue. In *Reliability, Survivability and Quality of Large Scale Telecommunication Systems*, Stavroulakis P (ed.). Wiley: Chichester, 2003; 167–198.
13. Ishac J, Allman M. On the performance of TCP spoofing in satellite networks. *Proceedings of IEEE MILCOM'01*, McLean, VA, U.S.A., 2001; 700–704.
14. Henderson TR, Katz RH. Transport protocols for Internet-compatible satellite networks. *IEEE Journal on Selected Areas Communications* 1999; **17**(2):326–334.
15. PIE. XTP Protocol Definition Revision 3.6. *PEI 92–10, Protocol Engines Incorporated*, Mountain View, CA, 11 January 1992.
16. Dube R, Rais CD, Kuang-Yeh W, Tripathi SK. Signal stability-based adaptive (SSA) routing for ad hoc mobile networks. *IEEE Personal Communication Magazine* 1997; **4**(1):36–45.
17. Allman M, Stevens W. TCP congestion control. *IETF RFC 2581*, April 1999.
18. Paxon V, Allman M. Computing TCP's retransmission timer. *IETF RFC 2988*, November 2000.
19. Allman M, Floyd S, Partridge C. Increasing TCPs initial window. *IETF RFC 2414*, September 1998.
20. Mathis M, Mahdavi J. TCP selective acknowledgment options. *IETF RFC 2018*, October 1996.
21. Mogul J, Deering S. Path MTU discovery. *IETF RFC 1191*, November 1990.
22. Rizzo L. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review* 1997; **27**(2):24–36.
23. Sklar B. *Digital Communications* (2nd edn). Prentice-Hall: Englewood cliffs, NJ, 2001.
24. Larsen KJ. Short convolutional codes with maximal free distance for rates 1/2, 1/3 and $\frac{3}{4}$. *IEEE Transaction on Information Theory* 1973; **IT-19**(3):371–372.
25. Jacobson V, Braden R, Borman D. TCP extensions for high performance. *IETF RFC 1323*, May 1992.

26. ETSI. Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications. *ETSI EN 302 307 V 1.1.1*. ETSI, June 2004.
27. ETSI. Digital video broadcasting (DVB); interaction channel for satellite distribution systems. *ETSI EN 301 790 V1.4.1*. ETSI, September 2005.
28. Celandroni N, Potortí F. Maximizing single connection TCP goodput by trading bandwidth for BER. *International Journal of Communication Systems* 2003; **16**:63–79.
29. Barakat C, Altman E. Bandwidth trade-off between TCP and link-level FEC. *Computer Networks* 2002; **39**(2): 133–150.
30. Allman M, Glover D, Sanchez L. Enhancing TCP over satellite channels using standard mechanisms. *IETF RFC 2488 (BCP: 28)*, January 1999.
31. Allman M, Dawkins S, Glover D, Griner J, Tran D, Henderson T, Heidemann J, Touch J, Kruse H, Ostermann S, Scott K, Semke J. Ongoing TCP research related to satellites. *IETF RFC 2760*, February 2000.
32. Barakat C, Chaher N, Dabbous W, Altman E. Improving TCP/IP over geostationary satellite links. *Proceedings of IEEE GLOBECOM '99*, Rio de Janeiro, Brazil, 1999; 781–785.
33. Hoe JC. Improving the start-up behavior of a congestion control scheme for TCP. *Proceedings of ACM SIGCOMM '96*, Palo Alto, California, U.S.A., 1996; 270–280.
34. Brakmo LS, Peterson LL. TCP Vegas: end-to-end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications* 1995; **13**(8):1465–1480.
35. Wang R, Valla M, Sanadidi MY, Gerla M. Adaptive bandwidth share estimation in TCP Westwood. *Proceedings of GLOBECOM'02*, 2002; 2604–2608.
36. Gerla M, Ng BKF, Sanadidi MY, Valla M, Wang R. TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs. *Journal of Computer Communications* 2004; **27**(1).
37. Wang R, Pau G, Yamada K, Sanadidi MY, Gerla M. TCP startup performance in large bandwidth delay networks. *INFOCOM 2004*, Hong Kong, March 2004; 796–805.
38. Floyd S. Connections with multiple congested gateways in packet-switched networks, part I: one-way traffic. *ACM Computer Communications Review* 1991; **21**(5):30–47.
39. Caini C, Firrincieli R. Packet spreading techniques to avoid bursty traffic in satellite TCP connections. *Proceedings of IEEE 59th Vehicular Technology Conference*, VTC2004-Spring, Milan, Italy, May 2004; 2906–2910.
40. Caini C, Firrincieli R. End-to-end TCP enhancements performance on satellite links. *Proceedings of IEEE Symposium on Computers and Communications (ISCC 2006)*, Cagliari, Italy, June 2006; 1031–1036.
41. Stewart R, Xie Q, Morneault K, Sharp C, Schwarzbauer H, Taylor T, Rytina I, Kalla M, Zhang L, Paxson V, Paxon, Allman M. Stream control transmission protocol. *IETF RFC 2960*, October 2000.
42. Kelly A, Perry P, Murphy J. A modified SCTP handover scheme for real time traffic. *HETNETs Working Conference*, Ilkley, England, July 2003.
43. Space Communications Protocol specification-Transport Protocol (SCPS-TP). CCSDS 714.0-B-1, http://www.scps.org.
44. Zhang Y. *Interworking and Computing over Satellite Networks*. Kluwer Academic Publisher: Dordrecht, 131–154, 2003.
45. Braden R. T/TCP - TCP extensions for transactions functional specification. *IETF RFC 1644*, July 1994.
46. Fox R. TCP big window and NAK options. *IETF RFC 1106*, June 1989.
47. Jacobson V. Compressing TCP/IP headers for low-speed serial links. *IETF RFC 1144,* February 1990.
48. Balakrishnan H, Padmanabhan VN, Fairhurst G, Sooriyabandara M. TCP performance implications of network path asymmetry. *IETF RFC 3449 (BCP: 69)*, December 2002.
49. DTN Research Group web page: http://www.dtnrg.org.
50. Ioannidis J, Maguire G. The coherent file distribution protocol. *IETF RFC 1235*, June 1991.
51. Ramadas M, Burleigh S, Farrell S. Licklider transmission protocol-specification. draft-irtf-dtnrg-ltp-03.txt, July 2005.