PETRA: Performance Enhancing Transport Architecture for Satellite Communications

Mario Marchese, *Member, IEEE*, Michele Rossi, *Student Member, IEEE*, and Giacomo Morabito, *Member, IEEE*

***Abstract*—This paper presents a performance enhancing transport architecture for the satellite environment. This solution improves the network transport performance by overcoming the limits imposed by a transmission control protocol/Internet protocol (TCP/IP)-based stack suite, while maintaining the interfaces offered by it. This is an important issue since TCP/IP is widely used and most of the applications are based on it. The work starts from the state-of-the-art about the transport layer over satellite by distinguing two alternative frameworks: the black box (BB) and the complete knowledge (CK) approaches. In the former, the network is considered as a "black box" and only modifications in the terminal tools are permitted. In the latter, the complete control of any network element is allowed so as a performance optimization procedure is possible. The proposed architecture [called Performance Enhancing Transport Architecture (PETRA)] is designed in all details using the second approach. PETRA uses network elements, called relay entities, to isolate the satellite portions in case of heterogeneous networks, while a transport layer protocol stack is used to optimize the transport of information over satellite links. A special satellite transport protocol, that is part of the transport layer protocol stack, is used over such links to perform error recovery. Simulation results show that the proposed framework significantly enhances throughput performance.**

***Index Terms*—Delay effects, satellite communications, transport protocols.**

I. Introduction

THE transmission control protocol (TCP) is the connection-oriented, end-to-end reliable transport protocol utilized in the Internet. The motivations, the philosophy, and the functional specification of the protocol are contained in [1]. TCP assumes that the service offered by lower layer protocols is unreliable; various features such as timeout timers, packet reordering, and retransmissions are used in TCP with the aim of providing a reliable channel to higher layer applications. Unfortunately, these features have been designed to be effective over wired networks, but it is well known that they often fail when the underlying channel is characterized by both a large *bandwidth delay* product and *high error rates*. This is the case of heterogeneous networks containing satellite links, where special countermeasures have to be taken to correct the inefficiencies of the TCP protocol.

Satellite networks are attractive since they offer clear advantages with respect to cable networks [2]: their architecture is more scalable, the diffusion throughout the land is wide, and the multicast service is very simple. Given the inefficiencies of standard TCP over such networks, one could design a new protocol, specifically tailored for these environments. However, given the widespread diffusion of TCP/IP applications, it is very difficult to think of a protocol different from TCP, that is still transparent to the user, to be used over satellite links. For these reasons, it is more appealing to provide new solutions, where the TCP/IP can still be used at the end terminals and its inefficiencies are accounted by algorithms and protocols running at the end terminals and in the satellite portion of the network. The aim of this paper is to propose and validate a transport architecture to deal with this problem.

In the sequel, a transport layer architecture that allows transporting TCP/IP flows efficiently and transparently through satellite networks to the final user is presented. The proposed architecture, which we call performance enhancing transport architecture (PETRA), divides the end-to-end connection into different segments. The bridging between different network segments is performed by elements named relay entities (REs). The objective of PETRA is the optimization of both the throughput performance for the satellite network environment and the efficient utilization of network resources. This is achieved without redesigning the protocol interfaces, so that they maintain the same characteristics of the interfaces currently used. As a consequence, system performance is optimized and at the same time, utilization of standard applications is still possible, thus reaching a high degree of portability. Moreover, the proposed solution preserves the end-to-end reliability and semantic of the transport layer by introducing, at the transport layer, a new sublayer named *upper transport layer*. The transport layer is divided into two sublayers.

- *Lower Transport Layer* (LTL): A different LTL instance is utilized in each segment of the end-to-end path. The LTL is responsible for the transport in each of these segments. The LTL utilized in the satellite segment is called *satellite transport protocol plus* (STPP). It is similar to the STP proposed in [3] with the following differences.
  - — STPP introduces a mechanism that avoids deadlock situation.
  - — STPP introduces a mechanism which stops the flow over the satellite link to avoid buffer overflows in the REs.

Manuscript received December 15, 2002; revised July 1, 2003 and September 20, 2003. This work was supported in part by the European Space Agency (ESA) under Contract 14956/00/NL/ND.

M. Marchese is with the CNIT-Italian National Consortium for Telecommunications, DIST-University of Genoa Research Unit, University of Genoa, Genoa 16145, Italy (e-mail: mario.marchese@cnit.it).

M. Rossi is with the Department of Engineering, University of Ferrara, Ferrara 44100, Italy (e-mail: mrossi@ing.unife.it).

G. Morabito is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, University of Catania, Catania 95125, Italy (e-mail: giacomo.morabito@diit.unict.it).

Digital Object Identifier 10.1109/JSAC.2003.819981

- *Upper Transport Layer* (UTL): Its algorithms are executed end-to-end and are responsible for maintaining the end-to-end reliability and semantics of the transport layer.

PETRA is targeted toward a geostationary orbit (GEO) satellite environments, where the long round-trip delay heavily affects the system performance, but radio-mobile and low earth orbit (LEO) satellite systems, characterized by channels with fading and high bit-error rate, are not excluded. PETRA has a high degree of flexibility that allows an efficient adaptation to these peculiarities.

The idea of splitting the connection into different segments has been already proposed in the past [4]. The major contributions of the paper are the following.

- A new transport layer protocol architecture is proposed where two sublayers (LTL and UTL) coexist as explained above.
- Algorithms and their interactions are considered in all details. Moreover, the optimal values of several algorithm parameters are calculated analytically or empirically.
- New flow control algorithms are proposed for enabling effective integration of satellite systems in the Internet.

The rest of this paper is organized as follows. Section II presents the state-of-the-art regarding transport layer issues in satellite networks. Section III contains the description of the new protocol architecture (PETRA) and a list of requirements oriented to the implementation of the architecture. Section IV concerns the parameters setting within the PETRA architecture. Performance evaluation is provided in Section V, whereas our conclusions are drawn in Section VI.

II. State-of-the-Art

The problem of improving the performance of TCP over satellite has been investigated in the literature for some years. Many national and international programs and projects (extensively listed in [5]) in Europe, Japan, and U.S. have been devoted to satellite networks and applications. In particular, some of them, or part of them, are aimed at improving performance at the transport level. NASA ACTS [6], [7], ESA ARTES-3 [8], and CNIT-ASI [9] deserve a particular attention, among many others.

Also, International Standardization Groups as the Consultative Committee for Space Data Systems (CCSDS), which has already emitted a recommendation (see [10]) and the European Telecommunications Standards Institute (ETSI) [11], which is running its activity within the framework of the Satellite Earth Station–Broadband Satellite Multimedia (SES–BSM) working group, are active on the subject.

Accordingly, a large literature exists on this topic, see [12] for a first overview and [13] for a more specific study in TCP/IP networks with high delay per bandwidth product and random loss. More recently, [14] provides summaries of improved TCP versions, as well as issues and challenges in satellite TCP; [3] highlights the ways in which latency and asymmetry impair TCP performance; [15] lists the main limitations of the TCP over satellite and proposes many possible methods to act. References [16] and [17] represent, at the best of the authors' knowledge, the more recent tutorial papers on the topic: various possible improvements both at the transport level and at the application and network level are summarized and referenced.

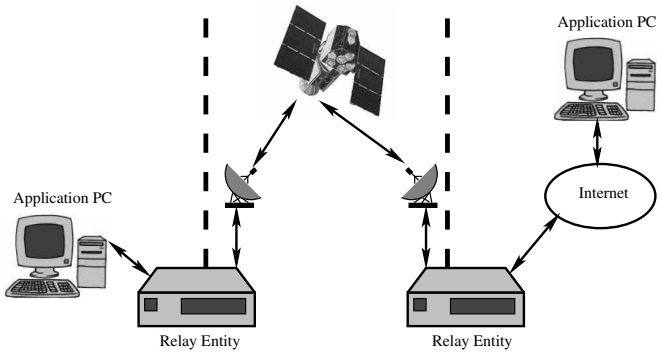

Fig. 1. PETRA architecture.

A recent issue of International Journal of Satellite Communications is entirely dedicated to IP over satellite [18]. In more details, [19] proposes a TCP splitting architecture for hybrid environments (see also [20]); [21] analyzes the performance of web retrievals over satellite and [5], [22], and [23] show an extensive analysis of the TCP behavior by varying parameters as the buffer size and the initial congestion window. Reference [24] focuses also on the buffer management but in an ATM environment.

The proposed approaches can be divided into two classes[1] [25].

- **Black Box (BB) Approaches**: Only the end terminals are modified. The rest of the network is considered nonaccessible (i.e., a black box).
- **Complete Knowledge (CK) Approaches**: Tuning of the parameters and algorithms of all the network components is allowed.

Many solutions based on the BB approach have been proposed so far, e.g., [26]–[30]. These solutions are general and do not violate the end-to-end semantic of TCP. Also, considerable work has been carried out according to the CK approach. Such methodologies, e.g., TCP splitting [12], [14], [19], [20] and TCP spoofing [12], [20], bypass the concept of end-to-end service by either dividing the TCP connection into segments or introducing intermediate gateways, with the aim of isolating the satellite link. The recent RFC 3135 [4] is dedicated to extend this concept by introducing performance enhancing proxies (PEPs) intended to mitigate link-related degradations. RFC 3135 is a survey of PEP techniques, not specifically dedicated to the transport layer, even if the emphasis is put on it. Motivations for their development are described, as well as consequences and drawbacks. The solutions based on the CK approach provide higher throughput performance with respect to BB but violate the end-to-end characteristics of the transport layer [12], [17].

III. PETRA

A. Operative Environment and Proposed Approach

The operative environment is shown in Fig. 1. The two application PCs are the end systems and are connected through *terrestrial segments* to the REs. Observe that the terrestrial segments can be the access local area network (LAN), as well as

[1]The classification proposed is not the only possible one and, probably, it is not exhaustive (i.e., not all the algorithms and methods in the literature can be classified within one of the two classes).

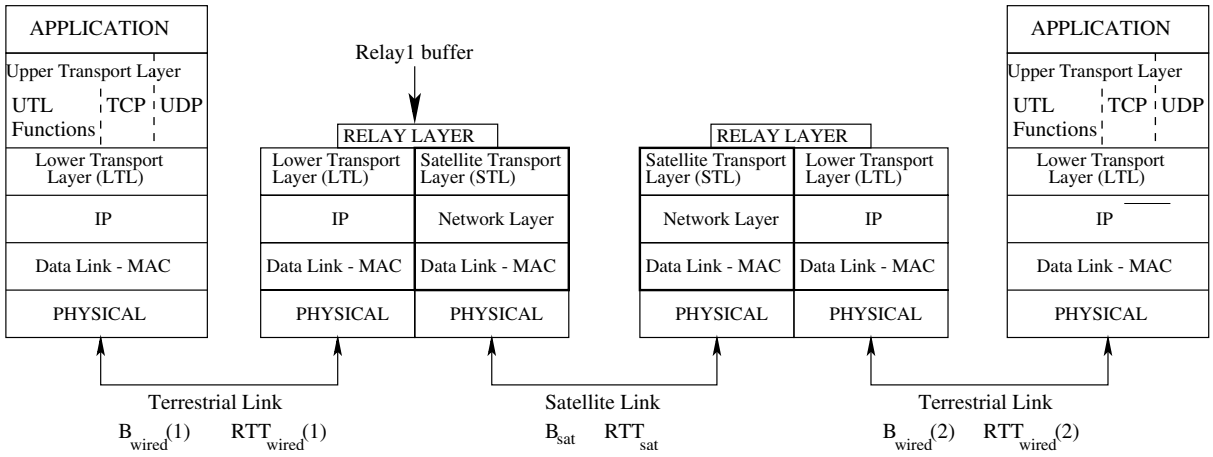

Fig. 2. End-to-end PETRA architecture.

the Internet. The REs are connected with each other by means of the satellite communication system. In Fig. 1, only one satellite segment is considered, however, the extension to the case with multiple satellite segments is straightforward.

Also, note that in Fig. 1 the satellite network is within the backbone. If the satellite network is, instead, the access network, then the RE is a software module directly connected to the application PC or a hardware card inside the application PC. PETRA algorithms are deployed in the application PCs and REs. A different transport protocol is run in each network segment. This allows to adapt the protocol to the specific characteristics of the segment. Moreover, an end-to-end algorithm is used between the two application PCs in order to maintain the end-to-end semantics and reliability of the transport protocol.

B. Transport Layer Protocol Stack

The proposed protocol architecture is called PETRA and is shown in Fig. 2. The transport layer is divided into two sublayers.

- UTL which guarantees the end-to-end semantics.
- LTL which is responsible for the transport within a single communication segment. The LTL in the satellite segment is denoted as *satellite transport layer* (STL).

A *relay module* is defined in the REs, which bridges the different LTLs. Let us stress that the relay module is a part of the transport layer.

C. Upper Transport Layer (UTL)

The UTL is responsible of preserving the end-to-end semantic of the transport layer.

UTL segments data coming from the user application into ULT data units (called *batches*) and transmits them according to a window-based flow control characterized by a window size W_{UTL}. The value of W_{UTL} must be large enough to ensure a continuous transmission flow over the satellite link for an entire end-to-end round-trip time (RTT).[2] Optimal choice of the W_{UTL} is given in Section IV-A. UTL is also responsible for failure recovery. To this purpose a timeout $T_0(\mathrm{UTL})$ is defined. If an UTL batch is not acknowledged within a time period equal to $T_0(\mathrm{UTL})$, then the connection is terminated. The timeout value depends on the end-to-end RTT, whose accurate estimate is, therefore, critical.

[2]It is the time elapsed between the departure of the first bit of a UTL batch and the instant in which the relative (end-to-end) UTL ACK is received.

D. Lower Transport Layer (LTL)

The LTL is responsible of the data integrity over the wired portions of the path. It segments the batches coming from the UTL layer into segments of (usually) smaller size (equal to the LTL MSS). In the terrestrial segments, LTL runs a TCP-like algorithm, with the only exception that the LTL ACK policy is slightly different from the one adopted in a standard TCP receiver. In particular, the LTL ACK flow can be stopped to realize the flow control at relay entity buffers. The ACK stopping technique is explained in more detail in the next section. Instead, the protocol executed in the STL is called STPP and will be presented in Section III-F.

Moreover, UTL batches are considered to be composed of n LTL segments, where n is an integer number such that $n \geq 1$. The choice of n is a design parameter that must be optimized jointly considering user and network requirements. In fact, a large n leads to a long end-to-end round tip time (that translates in a long UTL timeout value) and, as a consequence, the connection will be characterized by a long system failures reaction time. On the other hand, when n is small, we have a short network failure reaction time, but the number of UTL ACK's flowing on the backward path increases (leading to a waste of system resources in the reverse path). The choice of n is discussed in more detail in Section IV-A.

E. Relay Module

At the relay module some buffer space is reserved for each connection. Since the terrestrial and satellite segments are characterized by very different bandwidth, this buffer is utilized to harmonize the entering and the outgoing flows.

Substantially, to control the relay buffer occupancy a *stop-and-wait* strategy is used. For instance, focusing on a relay entity placed between the terrestrial segment and the satellite channel (i.e., *Relay1 buffer* in Fig. 2), the relay buffer input flow is due to LTL segments, whereas the output flow is composed by STPP packets.

To avoid buffer overflow problems, as the relay buffer size reaches a given threshold, $B_{\rm th}$, the LTL ACK transmission is inhibited and, as a consequence, the forward LTL flow is also stopped. Moreover, as the buffer occupancy becomes lower than $B_{\rm th}$ the LTL ACK flow is resumed. The LTL forward packet flow will be also resumed after a half wired round trip time $({\rm RTT}_{\rm wired}/2)$, i.e., the time needed for the resumed ACK flow to reach the LTL sending entity.

F. Satellite Transport Protocol Plus (STPP)

The STPP is the protocol responsible of the data integrity over the satellite channel. It is similar to the STP proposed in [3] with the following differences.

- STPP introduces a mechanism that avoids deadlock situations.
- STPP introduces a mechanism which stops the flow to avoid buffer overflows in the REs.

STPP must then counteract for channel errors due to wireless propagation phenomena allowing retransmission for lost packets. The STPP is a sliding-window-based protocol. At the sender side, incoming packets from higher layers (STPP SDUs) are segmented into STPP packets (STPP PDUs) and transmitted over the satellite channel. At the receiver side, these packets are ordered and the original higher level packets are reconstructed. Retransmissions follow a NACK-based selective-repeat ARQ mechanism as explained in the following.

1) At the STPP sender packets are taken from the STPP input queue (LTL SDUs) and segmented into STPP PDUs (the packet units handled at the STPP level). Then, a cyclic redundancy check (CRC) field is attached to each STPP PDU before its insertion into the STPP transmission queue.
2) At the sender, a retransmission buffer is accounted for. Retransmissions are requested by the receiver by means of special status messages, as described below. All the packets indicated in these status messages are inserted in the retransmission queue. At a given time, if no retransmission requests are pending (the sender retransmission queue is empty), then a new PDU is taken from the transmission queue and is transmitted over the channel. Instead, if the sender retransmission queue is not empty then the first PDU is taken from that queue and transmitted over the channel. The retransmission of each packet is inhibited for a full satellite RTT from the instant of its transmission, i.e., all retransmission requests for that packet during this time period are ignored.
3) At the STPP sender, associated with each transmitted PDU, there is a timeout timer, which is initially set to $T_o({\rm STPP})$ seconds. If this timer expires for a given PDU, then such PDU is scheduled for retransmission by inserting it into the STPP retransmission queue.
4) At the STPP receiver, correctness of received packets is determined by using the CRC field. In addition, the STPP receiver identifies the gaps in the received PDUs sequence numbers to infer *lost* packets.
5) If erroneous or lost PDUs are detected, the STPP receiver periodically (every $T_{\rm stat}$ seconds) sends back to the transmitter one unsolicited STATUS message (USTAT) containing the highest PDU sequence number correctly received in order (LAST_IO), the highest sequence number correctly received (LAST_SEQ_NO) and a sequence number mask in which all the sequence numbers of the *erroneous* and *lost* PDUs are listed. USTAT messages are used by the STPP receiver to request the retransmission for lost PDUs. In the case where no errors are detected, USTAT messages containing the LAST_IO identifier are sent back every $T_{\rm stat}$ seconds to advance the sender window.
6) At the STPP sender when an USTAT is received, all the requested PDUs not present in the retransmission queue are scheduled for retransmission by inserting them into the STPP sender retransmission queue. Moreover, all PDUs with sequence number lower than or equal to LAST_IO are deleted from the STPP memory.
7) When the UTL window is utilized for more than the 90% of its maximum size, the STPP sender generates a POLL message triggering an advance of the UTL window in order to avoid deadlock situations. The POLL message is sent to the receiver by setting a special POLL bit in the STPP PDU headers. When the POLL message arrives at the receiver, a solicited STAT message (SSTAT) is immediately sent back to the sender. This message is identical to the USTAT, with the only difference that it is sent *on-demand* (upon the reception of a POLL). The POLL message is resent repeatedly for every following satellite RTT until reception of a SSTAT.

Note that the STPP receiver uses STATUS messages in order to save bandwidth in the feedback channel. In the following, we show that $T_{\rm stat}$ is a crucial parameter because STPP performance is highly dependent on its choice. Moreover, in order to fully utilize the channel resources over the wireless satellite link it is pivotal to correctly set the STPP window size. Specifically, it must be chosen to ensure that the *satellite link is entirely filled by transmitted packets*. Referring to the STPP window as $W_{\rm STPP}$ (expressed in units of STPP PDUs), to the STPP PDU size as ${\rm PDU}_{\rm len}$ (expressed in bit), to the satellite channel bandwidth as $B_{\rm sat}$ (expressed in bits per second) and to the satellite RTT as ${\rm RTT}_{\rm sat}$ (expressed in seconds), the constraint that the STPP window must meet in order to be able to entirely fill the satellite channel with transmitted packets can be written as

$$W_{\rm STPP} \geq \left\lceil \frac{B_{\rm sat}{\rm RTT}_{\rm sat}}{{\rm PDU}_{\rm len}} \right\rceil . \tag{1}$$

The bandwidth available in the forward satellite channel is a known quantity. Therefore, the STPP protocol does not need to incrementally probe the channel for available bandwidth. Accordingly, the STPP level does not rely on algorithms such slow-start and congestion avoidance. Instead, it simply uses a sliding window approach by sending, at any time, the maximum amount of data allowed by the current value of $W_{\rm STPP}$. In addition to the normal operative mode presented above, to avoid buffer overflow at the relay node buffer in the receiving STPP entity, the following *flow-control* policy is implemented.

1) When the number of packets in the relay node buffer becomes larger than a given buffer threshold $(B_{\rm th})$, the

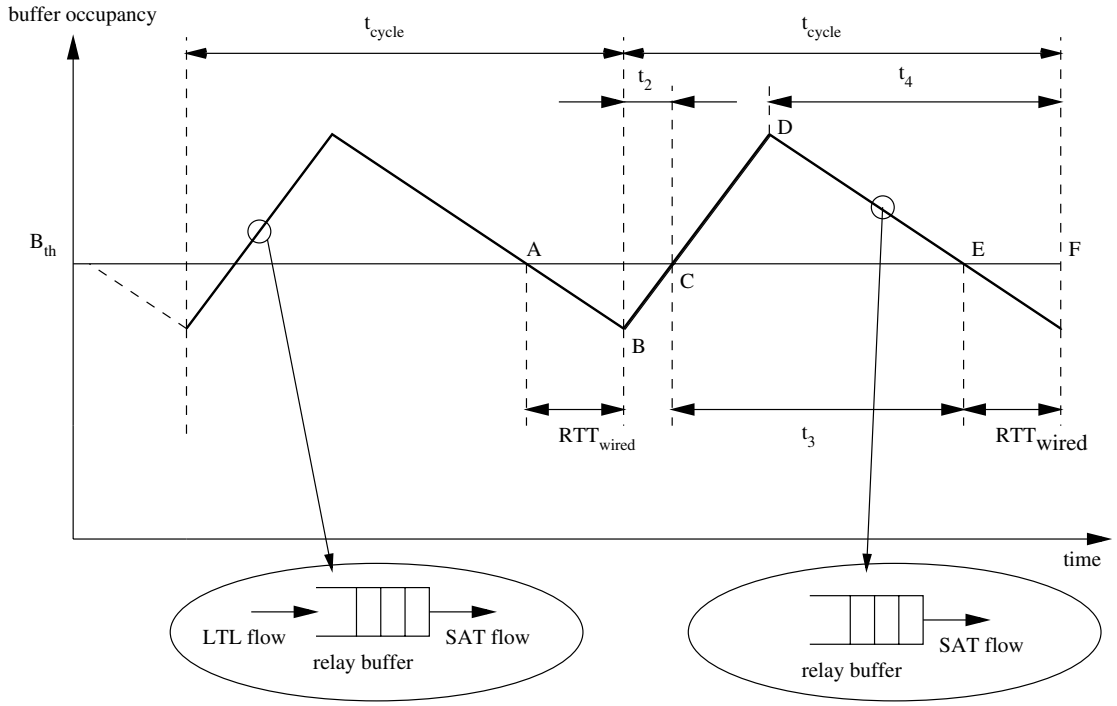

Fig. 3. Relay buffer behavior.

STPP ACK flow is stopped[3] by sending a *stop transmission* (STOP_TX) message.

2) When the number of packets in the receiver relay buffer becomes lower than $B_{\rm th}$, the transmission is resumed by sending back a *resume transmission* (RESUME_TX) message. This message in inserted in a USTAT packet.
3) Upon receiving the RESUME_TX message, the sender restarts transmitting following the rules presented above.

IV. Protocol Parameter Design

The PETRA architecture introduced in the previous sections utilizes several parameters. Here, we show how these parameters should be set.

A. Dimensioning the UTL Timeout

Referring to a generic UTL packet, we define the UTL round trip time $(\mathrm{RTT_{UTL}})$ as the time elapsed between the transmission time of the first bit of that packet and the instant at which its UTL ACK is received. The UTL timeout $(T_o(\mathrm{UTL}))$ must be set such that $T_o(\mathrm{UTL}) \geq \mathrm{RTT_{UTL}}$. In this section, we compute an upper bound for $\mathrm{RTT_{UTL}}$, i.e., where the LTL level has already reached its maximum allowed window size $W_{\rm LTL}^{\max}$ (measured in units of LTL segments).

According to the scenario depicted in Fig. 2, let us consider first the behavior of the relay buffer placed in the first relay entity encountered along the forward path (the leftmost relay entity in Fig. 2). We assume that the available bandwidth over terrestrial paths[4] is larger than the one characterizing the satellite link $(B_{\rm sat})$, i.e., $B_{\rm wired}(i) > B_{\rm sat}, \forall i$. In this case, the relay node buffer is a bottleneck for the incoming LTL flow and the ACK stopping feature at the receiving LTL entity is used to harmonize the incoming and the outgoing flows.

In Fig. 3, we show an example behavior for the relay buffer occupancy as a function of time. Let us start our description from point A of this figure, where the LTL ACK flow is restored because the number of segments in the buffer is below the threshold $B_{\rm th}$. Due to the finite RTT characterizing the first terrestrial link $[\mathrm{RTT_{wired}}(1)]$ no LTL segments are received at the relay node up to point[5] B. Hence, between time A and time B at the relay buffer, we only have the presence of the outgoing satellite link flow $(B_{\rm sat})$. The LTL flow is restored at time B and is stopped again at time C, when the buffer occupancy grows beyond $B_{\rm th}$. Note that after point C, under the steady-state condition, an additional number of $W_{\rm LTL}^{\max}$ packet is received. This is the number of segments that the LTL protocol (that is, a TCP-like protocol) can send without receiving any ACK. Hence, between point C and D, $W_{\rm LTL}^{\max}$ LTL segments are received (also in this interval of time we have the presence of both flows). After time D no more LTL segments are received. The ACK flow is restored again at time[6] E, where the buffer occupancy decreases below $B_{\rm th}$. The behavior of the Relay buffer occupancy continues in this way cyclically until all data has been transferred.

[3]Note that, due to the PDUs in flight over the channel, after this action, at most one further satellite RTT of STPP PDUs is received. This must be considered in the dimensioning of $B_{\rm th}$.

[4]We refer to the bandwidth of the ith terrestrial link encountered along the path, $\mathcal{P}$, as $B_{\rm wired}(i)$ and to the RTT of the ith terrestrial link as $\mathrm{RTT_{wired}}(i)$.

[5]The time difference between point B and point A is given by $\mathrm{RTT_{wired}}(1)$, i.e., the first terrestrial link RTT.

[6]The first LTL packet is received at the Relay node $\mathrm{RTT_{wired}}(1)$ seconds later.

We define t_{cycle} as the time elapsed between B and F (the buffer occupancy function period), where F is the end point of the cycle starting in B (see Fig. 3). To compute RTT_{UTL}, we need to find t_{cycle} and the number of LTL segments transmitted during this period of time.

First of all, we focus our attention on the time interval $[A, C]$. During this time interval the maximum input rate is given by

$$f_{\text{in}}(t) = \begin{cases} 0, & A \le t < B \\ B_{\text{wired}}/\text{LTL}_{\text{len}}, & B \le t \le C \end{cases} \quad (2)$$

where LTL_{len} is the length of a full LTL packet. The output flow is constant and it is due to the satellite link

$$f_{\text{out}}(t) = -B_{\text{sat}}/\text{LTL}'_{\text{len}} \quad (3)$$

where

$$\text{LTL}'_{\text{len}} = \frac{(\text{LTL}_{\text{len}} + \text{RLH}_{\text{len}})\text{PDU}_{\text{len}}}{\text{PDU}_{\text{len}}(\text{payload})}. \quad (4)$$

The parameter RLH_{len} represents the size of the header added by the relay node to each LTL segment to store connection information,[7] whereas PDU_{len} and $\text{PDU}_{\text{len}}(\text{payload})$ are the size of a full STPP packet and of its payload, respectively. In (4), we compute the number of STPP PDUs needed to transmit a full LTL packet ($(\text{LTL}_{\text{len}} + \text{RLH}_{\text{len}})/\text{PDU}_{\text{len}}(\text{payload})$) and then we multiply it by the STPP PDU size. So, the number of transmitted bits per LTL packet at the STPP level can be easily obtained. Since the buffer in points A and C contains the same number of segments we can solve $\int_A^C (f_{\text{in}}(t) + f_{\text{out}}(t))dt = 0$. This integral can be easily solved by noting that $C - B = t_2$ and that $C - A = \text{RTT}_{\text{wired}}(1) + t_2$, hence

$$t_2 = \frac{B_{\text{sat}}\text{RTT}_{\text{wired}}(1)}{\text{LTL}'_{\text{len}}\left(\frac{B_{\text{wired}}}{\text{LTL}_{\text{len}}} - \frac{B_{\text{sat}}}{\text{LTL}'_{\text{len}}}\right)}. \quad (5)$$

Moreover, the number of LTL segments received during time t_2 (interval $[B, C]$) is given by

$$r(t_2) = \frac{B_{\text{wired}}}{\text{LTL}_{\text{len}}} t_2. \quad (6)$$

The time interval t_3 can be easily obtained by observing in Fig. 3 that it is the time necessary for the satellite link to free the buffer from $W_{\text{LTL}}^{\max}$ segments[8]

$$t_3 = \frac{\text{LTL}'_{\text{len}}}{B_{\text{sat}}} W_{\text{LTL}}^{\max}. \quad (7)$$

So, t_{cycle} is computed as $t_{\text{cycle}} = \text{RTT}_{\text{wired}}(1) + t_2 + t_3$.

In the following, we compute the maximum time elapsed between the transmission of the first bit of a UTL packet and the instant at which the last LTL packet composing it is inserted into the relay buffer. We refer to this time as $t(n)$. The worst situation is the one where the first LTL packet is inserted into the buffer at time D, i.e., it is the last of the $W_{\text{LTL}}^{\max}$ segments received after point C. In this case, a time equal to $t_4 = F - D$ has to be waited before the second LTL packet will be inserted into the relay buffer. $D - C$ can be easily computed by noting that it is the time needed to transmit $W_{\text{LTL}}^{\max}$ segments over the wired link. Hence, $D - C = \text{LTL}_{\text{len}} W_{\text{LTL}}^{\max}/B_{\text{wired}}$ and t_4 is obtained as

$$t_4 = t_3 + \text{RTT}_{\text{wired}}(1) - (D - C). \quad (8)$$

Furthermore, by noting that $r(t_{\text{cycle}}) = W_{\text{LTL}}^{\max} + r(t_2)$, LTL segments are inserted into the relay buffer for each following cycle and accounting for the fixed propagation delay ($t_{\text{prop}} = B_{\text{wired}}/\text{LTL}_{\text{len}} + \text{RTT}_{\text{wired}}(1)/2$) for the reception of the first LTL packet (of n) we can upper bound $t(n)$ as follows:

$$t(n) = t_{\text{prop}} + t_4 + \left\lfloor \frac{n-1}{r(t_{\text{cycle}})} \right\rfloor t_{\text{cycle}} + \left(n - 1 - \left\lfloor \frac{n-1}{r(t_{\text{cycle}})} \right\rfloor t_{\text{cycle}}\right) \frac{B_{\text{wired}}}{\text{LTL}_{\text{len}}}. \quad (9)$$

At this point, a worst case estimate of RTT_{UTL} can be computed as

$$\text{RTT}_{\text{UTL}} \le t_{\text{init}} + t(n) + t_{\text{buffer}} + t_{\text{ACK}} + \text{RTT}_{\text{sat}} + \frac{\text{RTT}_{\text{wired}}(1)}{2} + \text{RTT}_{\text{wired}}(2) \quad (10)$$

where

- t_{init} is the time elapsed between the beginning of the LTL transmission and the instant at which the buffer occupancy grows to B_{th} for the first time. t_{init} is computed as the lowest value of t that verifies the following inequality:

$$\left\lceil TX(t - \text{RTT}_{\text{wired}}(1)/2) - \frac{\text{LTL}'_{\text{len}}}{B_{\text{sat}}}(t - \text{RTT}_{\text{wired}}(1)/2) \right\rceil \ge B_{\text{th}} \quad (11)$$

  where the function $\text{TX}(t)$ (derived in the Appendix) represents the number of segments transmitted by the LTL sender at time t, whereas $(\text{LTL}'_{\text{len}})/(B_{\text{sat}})t$ is the number of LTL segments transmitted over the satellite up to and including time t.
- $t(n)$ is an upper bound of the time needed to insert one entire UTL packet (n LTL segments) into the relay buffer.
- t_{buffer}: in the worst case the nth LTL packet arriving at the relay node finds $(W_{\text{LTL}}^{\max} - 1 + B_{\text{th}})$ segments waiting in the relay buffer. In this case, the nth packet is transmitted after a time t_{buffer} given by

$$t_{\text{buffer}} = \frac{\text{LTL}'_{\text{len}}}{B_{\text{sat}}}(W_{\text{LTL}}^{\max} + B_{\text{th}}) \quad (12)$$

- t_{ACK} is the (total) transmission time of a full UTL ACK; it is obtained as the sum of the transmission time experienced by the ACK on all links

$$t_{\text{ACK}} = \sum_{i \in \mathcal{P}} \frac{\text{ACK}_{\text{len}}}{B_{\text{wired}}(i)} + \frac{\text{ACK}'_{\text{len}}}{B_{\text{sat}}} \quad (13)$$

  where we consider the presence of a unique satellite link (B_{sat}) and ACK'_{len} is the number of bits needed to transmit an UTL ACK over the satellite channel.

Equation (10) can be used to obtain the worst case estimate of the UTL RTT in the error free case. Moreover, by considering a lossy link for what concerns the satellite channel, this equation can be still applied replacing B_{sat} with $B_{\text{sat}}^{(\text{eq})}$ in the previous equations, where $B_{\text{sat}}^{\text{eq}}$ is the equivalent satellite bandwidth in presence of satellite channel errors.

[7] To reserve the bandwidth pipe and to keep other flow specific information (flow identifier, priority).

[8] These segments are the ones received after the ACK stopping time, i.e., time C.

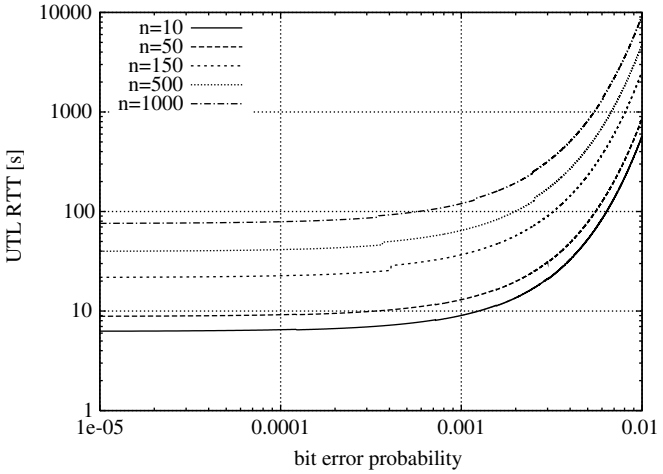

Fig. 4. Estimate of $\mathrm{RTT}_{\mathrm{UTL}}$ as a function of the satellite channel bit-error rate by varying the UTL batch size (n).

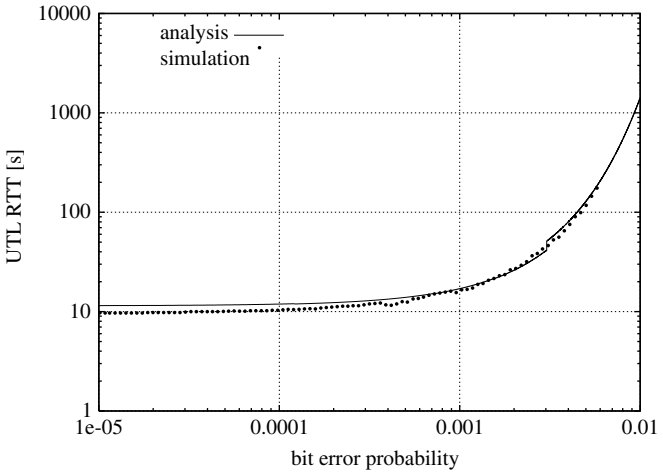

Fig. 5. UTL RTT: comparison between estimate and simulation, $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $B_{\mathrm{sat}} = 128$ Kb/s, $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $n = 100$.

If ε is the STPP PDU error probability,[9] $B_{\mathrm{sat}}^{\mathrm{eq}}$ is computed as $B_{\mathrm{sat}}^{\mathrm{eq}} = B_{\mathrm{sat}}(1-\varepsilon)$. In Fig. 4, we report the $\mathrm{RTT}_{\mathrm{UTL}}$ estimate as a function of the satellite channel bit error probability by varying n. We considered the scenario in Fig. 2 with $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, $\mathrm{LTL}_{\mathrm{len}} = 1$ kB and $B_{\mathrm{sat}} = 128$ Kb/s. In Fig. 5, we report the comparison between simulation points and analysis considering the same parameters for $n = 100$.

B. Dimensioning of the UTL Window Size

The UTL window size must be chosen to ensure that during the time elapsed between the departure of a UTL packet and the instant at which the end-to-end acknowledgment is received, the satellite link is always filled by transmitted segments. By expressing the UTL window size (W_{UTL}) in unit of UTL packets, this condition is verified when the following inequality holds:

$$W_{\mathrm{UTL}} \geq \left\lceil \frac{B_{\mathrm{sat}}\mathrm{RTT}_{\mathrm{UTL}}}{\mathrm{UTL}'_{\mathrm{len}}} \right\rceil \tag{14}$$

[9]Here, we consider the independent error case.

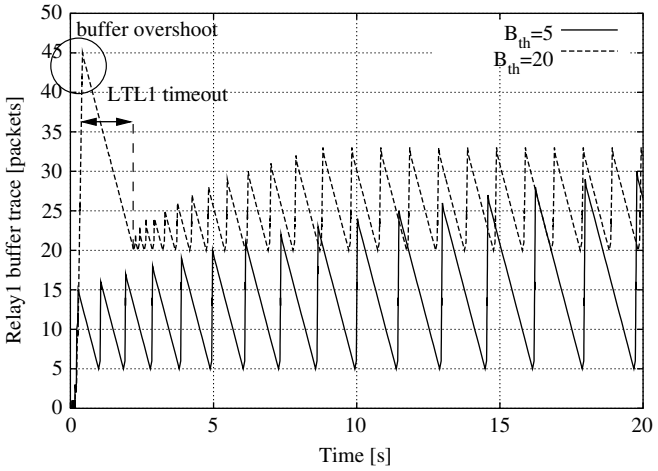

Fig. 6. Relay buffer trace considering $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $B_{\mathrm{sat}} = 128$ kb/s, $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, $W_{\mathrm{max}}^{\mathrm{LTL}} = 32$ LTL segments, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{header}) = 40$ bytes.

where $\mathrm{UTL}'_{\mathrm{len}}$ is the number of bits needed to transmit a full UTL packet over the satellite channel (including STPP overhead). The formula in (14) is valid as long as the satellite link is a bottleneck for the connection, i.e., B_{sat} is lower than the bandwidth of all the links upstream the satellite channel.

C. Dimensioning the Relay Buffer Threshold

The correct dimensioning of the Relay buffer threshold (B_{th}) is a pivotal point in the design of the PETRA architecture. This threshold is responsible of flow control at the relay nodes and (as will be shown in the following) misconfigurations could lead to LTL premature timeouts. Focusing on the scenario depicted in Fig. 2, we consider first the relay node placed in the leftmost side of the figure. When $B_{\mathrm{sat}} \geq B_{\mathrm{wired}}(1)$, the choice of B_{th} at such relay is irrelevant because the buffer in this situation is always empty. In the case where $B_{\mathrm{sat}} < B_{\mathrm{wired}}(1)$ instead a correct dimensioning of the threshold is required. In particular, if the values of B_{th} is too large, the following events would occur.

1) In the initial connection phase a large number of LTL segments (W^*) can be stored into the relay buffer before the LTL ACK flow is stopped (see Fig. 6).
2) The LTL during this time period experiences a very short RTT (equal to $\mathrm{RTT}_{\mathrm{wired}}$) and estimates the timeout $(T_o(\mathrm{LTL}))$ accordingly.
3) After the ACK stopping point, a long period is needed to the satellite link to restore the buffer occupancy below B_{th} and during this period the LTL flow remains stopped.
4) If the length of this period is greater to or equal than $T_o(\mathrm{LTL})$ (the timeout estimate in point 2) then an LTL timeout event has to be waited for.

In order to avoid the LTL timeout problem, B_{th} must be set to a small value. In this way, W^* is limited and the LTL timeout event probability is decreased. In general, the smaller B_{th}, the lower the LTL timeout event probability. However, a necessary condition to ensure that the satellite link is always filled by transmitted segments is that the relay buffer *must be never emptied*.

This condition poses a lower bound on the minimum allowed value of $B_{\rm th}$

$$B_{\rm th} \geq B_{\rm th}^{\rm min}. \tag{15}$$

Now, noting that

- the time elapsed at the relay buffer between the instant when the buffer size decreases below $B_{\rm th}$ (the LTL ACK flow is restored) and the reception of the first LTL packet of the restored flow is equal to $\mathrm{RTT}_{\rm wired}(1)$;
- the satellite channel during this time period consumes $\lceil B_{\rm sat}\mathrm{RTT}_{\rm wired}(1)/\mathrm{LTL}'_{\rm len}\rceil$ LTL segments ($\mathrm{LTL}'_{\rm len}$ is the number of bits needed to transmit a full LTL packet over the satellite link).

Condition in (15) can be written as follows:

$$B_{\rm th} \geq \left\lceil \frac{B_{\rm sat}\mathrm{RTT}_{\rm wired}}{\mathrm{LTL}'_{\rm len}} \right\rceil. \tag{16}$$

Hence, the $B_{\rm th}$ that minimizes the timeout event probability is the minimum value verifying the condition in (16), i.e., $B_{\rm th}^{\rm min} = \lceil B_{\rm sat}\mathrm{RTT}_{\rm wired}/\mathrm{LTL}'_{\rm len}\rceil$. As an example, in Fig. 6, we report the relay buffer occupancy as a function of the time for the two cases where $B_{\rm th} = 20 (\gg B_{\rm th}^{\rm min} = 1)$ and $B_{\rm th} = 5$. In the last case, the timeout event is avoided and the satellite channel is always filled by transmitted PDUs. On the contrary, when $B_{\rm th}$ is too large a spurious timeout event is observed for the reasons discussed above. Note that the problem addressed here could heavily degradate the performance of the LTL running over terrestrial paths, especially in the case where $B_{\rm sat} \ll B_{\rm wired}(1)$. Moreover, in Fig. 6 ($B_{\rm th} = 20$), we have only one timeout event (at the beginning of the connection); after this timeout the LTL connection stabilizes and no more timeouts are observed. However, some particular cases exist in which timeout events occur repeatedly when the buffer threshold is misconfigured. In Fig. 7, we show how for particular values of the system parameters, a misconfigured $B_{\rm th}$ can lead to multiple LTL timeout events, whereas in Fig. 8, we report the discontinuous transmission effect on the STPP flow due to a too small $B_{\rm th}$ value ($B_{\rm th}^{\rm min}$ in this case is equal to 24 LTL segments). Similar considerations apply to the relay node in the rightmost side of Fig. 2.

D. Dimensioning the Relay Buffer Size

We first proceed to the dimensioning of the buffer on the leftmost side of Fig. 2.

Consider the buffer behavior reported in Fig. 3. We observe that in order to avoid buffer losses, the relay buffer size ($B_{\rm size}$) must be greater than the value reached in point D. Specifically, at time C exactly $B_{\rm th}$ LTL segments are stored into the buffer, while during the interval $(C, D]$, at most $W_{\rm max}$ further segments arrive at the relay node. Moreover, during $(C, D]$ exactly $(B_{\rm sat}/\mathrm{LTL}'_{\rm len})(D-C)$ segments are consumed by the satellite level. Hence, the buffer size in point D, $\mathrm{BS}(D)$ is given by

$$\begin{aligned}\mathrm{BS}(D) &= B_{\rm th} + W_{\rm max}^{\rm LTL} - \frac{B_{\rm sat}}{\mathrm{LTL}'_{\rm len}}\left(t_{\rm LTL}W_{\rm max}^{\rm LTL}\right) \\ &= B_{\rm th} + W_{\rm max}^{\rm LTL} - \frac{B_{\rm sat}}{B_{\rm wired}(1)}\frac{\mathrm{LTL}_{\rm len}}{\mathrm{LTL}'_{\rm len}}W_{\rm max}^{\rm LTL}\end{aligned} \tag{17}$$

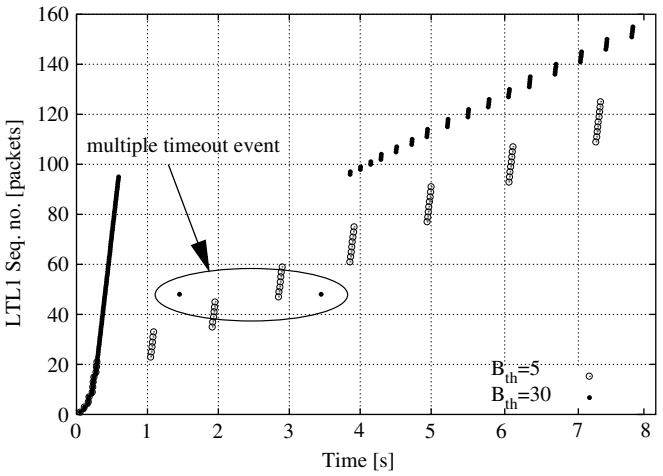

Fig. 7. LTL packet trace with $W_{\rm max}^{\rm LTL} = 50$ LTL segments, $B_{\rm wired}(1) = B_{\rm wired}(2) = 2$ Mb/s, $B_{\rm sat} = 128$ kb/s, $\mathrm{RTT}_{\rm wired}(1) = \mathrm{RTT}_{\rm wired}(2) = 100$ ms, $\mathrm{RTT}_{\rm sat} = 0.5$ s, $\mathrm{LTL}_{\rm len}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\rm len}(\mathrm{header}) = 40$ bytes.

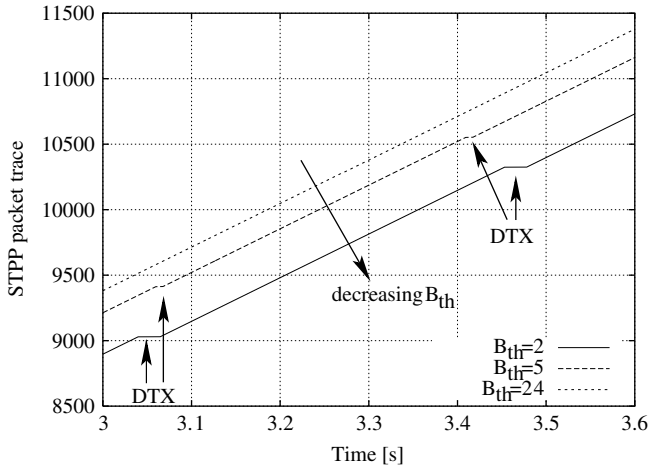

Fig. 8. STPP PDU trace: discontinuous transmission problem, $W_{\rm max}^{\rm LTL} = 32$ LTL segments, $B_{\rm wired}(1) = B_{\rm wired}(2) = 4$ Mb/s, $B_{\rm sat} = 2$ Mb/s, $\mathrm{RTT}_{\rm wired}(1) = \mathrm{RTT}_{\rm wired}(2) = 100$ ms, $\mathrm{RTT}_{\rm sat} = 0.5$ s, $\mathrm{LTL}_{\rm len}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\rm len}(\mathrm{header}) = 40$ bytes.

where $t_{\rm LTL}$ is the transmission time of one LTL packet over the first terrestrial link, $t_{\rm LTL} = \mathrm{LTL}_{\rm len}/B_{\rm wired}(1)$. If the satellite bandwidth is completely utilized the correct value of $B_{\rm size}$ is given by

$$B_{\rm size} = \left\lceil B_{\rm th} + W_{\rm max}^{\rm LTL} - \frac{B_{\rm sat}}{B_{\rm wired}(1)}\frac{\mathrm{LTL}_{\rm len}}{\mathrm{LTL}'_{\rm len}}W_{\rm max}^{\rm LTL} \right\rceil. \tag{18}$$

Due to channel impairments, the available bandwidth $B_{\rm sat}$ may be less than the one used in (17). If this is the case, the correct value of the buffer size is computed as follows:

$$B_{\rm size} = B_{\rm th} + W_{\rm max}^{\rm LTL}. \tag{19}$$

We suggest to use (19) in order to set the relay buffer size at node 1 under all possible satellite channel error conditions.

Note that, for the correct working of the presented network architecture, the dimensioning of the relay buffer is a crucial point. In fact, if a packet is lost due to buffer overflow, no end-to-end

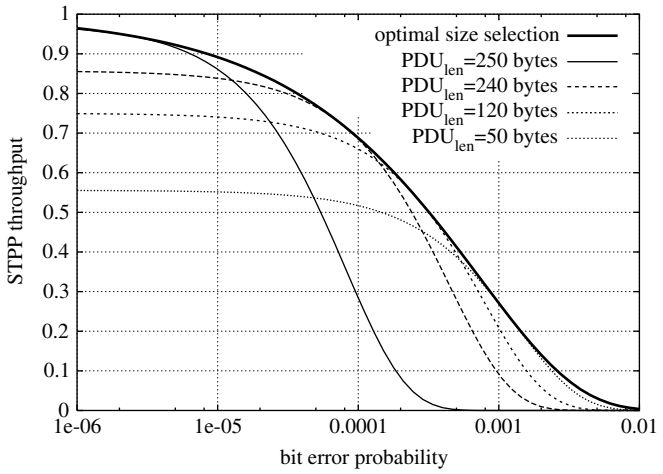

Fig. 9. STPP throughput as a function of the satellite channel bit error probability. Comparison between fixed $\mathrm{PDU_{len}}$ and optimal size selection.

retransmissions of the UTL batch including that packet are performed by UTL and a UTL timeout event has to be waited for. In this case, the connection is closed by UTL upon timeout timer expiration. For this reason, the buffer size *must be* dimensioned in order to completely avoid the buffer overflow problem. Similar reasoning apply to the dimensioning of the buffer placed on the rightmost part of Fig. 2, where input and output buffer flows are due to STPP and LTL, respectively.

E. Dimensioning of the STPP Packet Size

Another issue in the design of the STPP protocol is the choice of the STPP packet size. Regarding this point, there is a tradeoff between the overhead added by STPP PDU headers and CRC fields and the STPP PDU error probability. Defining P_e as the bit error probability at the STPP level,[10] the maximum STPP throughput is given by[11]

$$f(P,O,P_e)=\frac{P}{P+O}(1-(1-P_e)^{P+O}) \tag{20}$$

where P and O are $\mathrm{PDU_{len}(payload)}$ and $\mathrm{PDU_{len}}-\mathrm{PDU_{len}(payload)}$ as defined in Section IV-A. To find the optimal value for P given $O=O^*$ and $P_e=P_e^*$ it is sufficient to solve

$$\left.\frac{\partial f(P,O,P_e)}{\partial P}\right|_{O=O^*,P_e=P_e^*}=0. \tag{21}$$

The optimal value of the PDU payload size is then given by

$$P_{\mathrm{opt}}=\frac{-O^*\sqrt{p}+\sqrt{O^*(4+pO^*)}}{2\sqrt{p}} \tag{22}$$

where $p=|\log(1-P_e^*)|$. In Fig. 9, we report the STPP throughput as a function of the bit error probability considering both the case where O is constant and the case where (solid bold line in Fig. 9) O is chosen in order to maximize the STPP throughput [using (22)].

[10]It is the bit error probability after channel coding and interleaving operations. Here, an independent error process is considered.

[11]Considering that the STPP level is rightly configured so to have a continuous transmission flow, i.e., the available bandwidth over the satellite link (B_{sat}) is fully exploited.

F. Dimensioning the Bandwidth in the Reverse Channel

In order to dimension the satellite reverse channel, the bandwidth requirements must be evaluated. Let first evaluate the average size of a STAT message $\mathrm{SSTAT_{len}}$. This is given by the size of the STPP PDU header and the average size of the mask utilized to identify the STPP PDUs which have been received corrupted or have not been received at all. Accordingly, the average size of the STAT message $\mathrm{SSTAT_{len}}$ can be evaluated as follows:

$$\mathrm{SSTAT_{len}}=O+\alpha\cdot n_e \tag{23}$$

where

- O STPP PDU header size;
- α number of bits required to signal an erroneous STPP PDU in the STAT mask;
- n_e average number of corrupted PDUs in a time interval equal to the satellite round-trip time $\mathrm{RTT_{sat}}$; if we denote $\varepsilon_{\mathrm{PDU}}$ the PDU error probability in the satellite forward channel, n_e can be computed as follows:

$$n_e=\varepsilon_{\mathrm{PDU}}\cdot B_{\mathrm{sat}}\cdot \mathrm{RTT_{sat}}/\mathrm{PDU_{len}}. \tag{24}$$

Now, the average bandwidth required in the satellite reverse channel $B_{\mathrm{STPP}}^{(\mathrm{Rev})}$ can be easily evaluated

$$B_{\mathrm{STPP}}^{(\mathrm{Rev})}=\mathrm{SSTAT_{len}}/T_{\mathrm{stat}}. \tag{25}$$

V. Performance Evaluation

In this section, PETRA performance is evaluated by using analysis where possible and simulation otherwise. PETRA performance will be compared with the performance obtained in the following cases.

- *TCP E2E Case*: Traditional TCP-NewReno is utilized end-to-end. Moreover, in the satellite segment a selective repeat ARQ scheme is implemented at the link layer to counteract wireless channel errors.
- *Split-TCP Case*: A splitted-TCP solution is utilized.[12] In the satellite channel, a selective repeat ARQ technique which assigns higher priority to retransmission is considered. The receiver sends back a NACK for each erroneous PDU. At least one acknowledgment message per RTT is transmitted to advance the sender window.

The last solution allows keeping a constant information about the status since each erroneous packet is tracked. However, when the forward channel error rate is relevant this scheme is bandwidth consuming. The measure of the reverse bandwidth gain obtained by using PETRA with respect to split-TCP is reported in the following Section V-C.

A. Effects of Link Errors in the Satellite Forward Channel

Results shown in this section have been obtained by simulating the transfer of a 10 MB file. In Fig. 10, we show the throughput performance versus the bit error probability achieved using PETRA. We considered different values of T_{stat} and assumed

[12]We consider split-TCP because it achieves the best throughput performance [12].

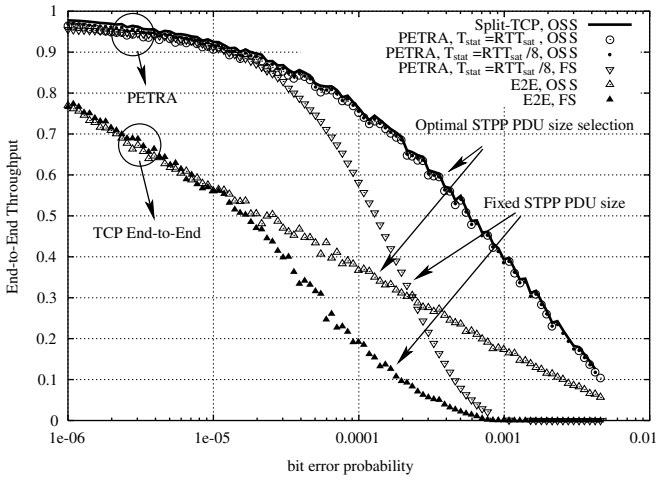

Fig. 10. End-to-end throughput as a function of the satellite channel error probability. $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, $B_{\mathrm{sat}} = 384$ kb/s, $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{header}) = 40$ bytes, $W_{\mathrm{max}}^{\mathrm{LTL}} = 32$ LTL segments, error-free terrestrial segments, 10-MB data transfer, $T_o(\mathrm{UTL}) = 20$ s.

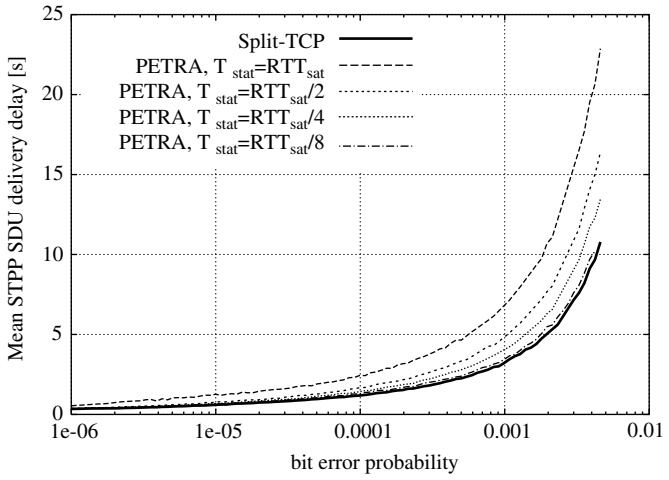

Fig. 11. Mean STPP SDU delivery delay as a function of the satellite channel bit-error rate. $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, $B_{\mathrm{sat}} = 384$ kb/s, $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{header}) = 40$ bytes, $W_{\mathrm{max}}^{\mathrm{LTL}} = 32$ LTL segments, error-free terrestrial segments, 10 MB data transfer, $T_o(\mathrm{UTL}) = 20$ s.

the satellite feedback channel and the wired segments to be error free. For the sake of comparison, in the same figure throughput performance obtained in the split-TCP and in the E2E case are also reported. Throughput has been evaluated as follows:

$$\text{throughput} = \frac{\text{data transferred [bit]}}{\text{transfer time [s]} \times B_{\mathrm{sat}}[\mathrm{bit/s}]}. \tag{26}$$

In Fig. 10, it is evident that the choice of T_{stat} does not have significant impact on the throughput performance (it is only necessary that at least one STAT message per RTT is correctly received to advance the window at the sender). The throughput achieved by PETRA is considerably higher than in the E2E Case and close to that achieved in the split-TCP case. This is due to the following reasons.

- Optimal values of the architecture parameters are continuously computed and utilized.
- A correct management of the relaying buffer is performed so that losses due to buffer overflow and LTL timeout events are avoided.

Obviously, the good PETRA performance is obtained at the expense of higher computational complexity and the overhead introduced by the double transport layering. Moreover, to evaluate the impact of an optimal STPP PDU size selection as a function of the forward channel error rate, in Fig. 10 two sets of results are shown.

- *OSS*: The optimal STPP PDU size is utilized.
- *FS*: A constant STPP PDU size is utilized for any value of the bit error probability. In Fig. 10, the payload size has been set to 600 bytes, whereas the STPP overhead (header and CRC fields) is fixed to 20 bytes in both cases (OSS and FS).

Fig. 11 shows the average delivery delay of the STPP segments versus the bit error probability in the satellite forward channel. In other words, while Fig. 10 contains the average end-to-end throughput, Fig. 11 allows measuring what happens over the satellite portion of the network and focusing on the role of the parameter T_{stat}. If the bit error probability is low, the impact of T_{stat} on the delay is negligible, but if the channel is severely errored, the performance heavily depends on its choice. In more detail, the performance of PETRA is close to the one achieved in the split-TCP case if the STATUS message is transmitted frequently ($T_{\mathrm{stat}} = \mathrm{RTT}_{\mathrm{sat}}/8$); the delay increases when the information about the status is operated after a larger lapse of time. It is interesting to observe that the delay of the STPP segments over the satellite link has no impact on the end-to-end throughput, as clear from Fig. 10, where the performance offered by "PETRA $T_{\mathrm{stat}} = \mathrm{RTT}_{\mathrm{sat}}$" and "PETRA $T_{\mathrm{stat}} = \mathrm{RTT}_{\mathrm{sat}}/8$" is completely equivalent. It means that the overall architecture, by means of properly dimensioned buffering at the REs, can match the delay increase without affecting the end-to-end throughput performance. This is not true if the feedback channel is errored, as will be clear in the next figures.

B. Effects of Link Errors in the Satellite Feedback Channel

Here, we assume a very low (10^{-9}) satellite forward channel bit error probability, while we consider the satellite feedback channel unreliable. In Figs. 12 and 13, we report the throughput performance and average LTL packet delivery delay, respectively. Both throughput and delay are given versus the bit error probability in the satellite feedback channel for different values of the STAT message period T_{stat}. The results presented in both figures were obtained by considering the following simulation parameters: STPP $\mathrm{PDU}_{\mathrm{len}} = 620$ bytes, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 10$ Mb/s, $B_{\mathrm{sat}} = 384$ kb/s, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{header}) = 40$ bytes, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{payload}) = 1$ kB, the forward channel bit error probability is 10^{-9}.

In Figs. 12 and 13, we observe that performance improves as the T_{stat} value decreases. This is because lower values of T_{stat} result in a larger number of STAT messages which allow a more efficient error recovery, as well as a faster advancement of the sliding window. As envisaged in the previous subsection, if, as in this case, the feedback channel is severely errored,

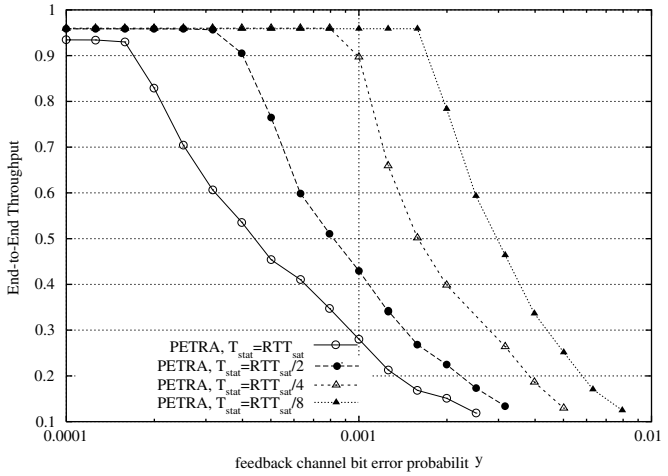

Fig. 12. PETRA end-to-end throughput as a function of the reverse channel bit error probability by varying $T_{\rm stat}$.

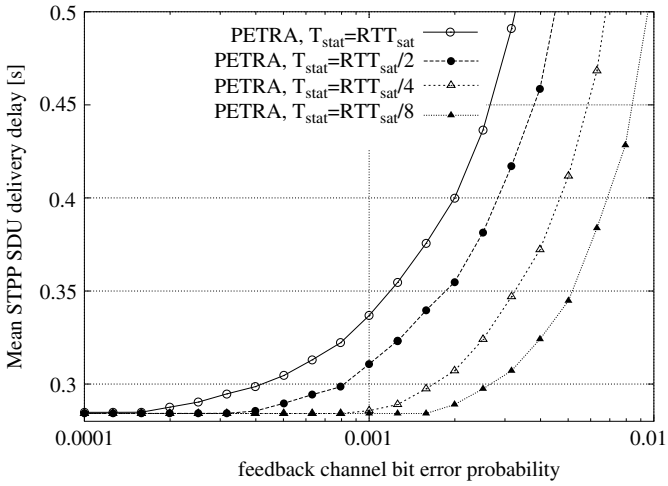

Fig. 13. PETRA, mean STPP SDU delivery delay as a function of the reverse channel bit error probability by varying $T_{\rm stat}$.

also a slight delay of the order of tens of milliseconds (Fig. 13) corresponds to an heavy end-to-end performance degradation (Fig. 12). This highlights the importance of the correct delivery of the STATUS information for the system performance, i.e., *at least one STATUS message must be correctly received in every* $\mathrm{RTT_{sat}}$ *in order to advance the STPP window*.

Anyway, the results offered by PETRA scheme are really satisfying: on one hand, bit error probabilities above 0.001 are a real worst case and very rare over real satellite systems; on the other hand, also in these cases, PETRA can be configured (e.g., $T_{\rm stat} = \mathrm{RTT_{sat}}/8$) so to be very robust against forward and feedback channel errors and to offer performance suited for any type of application.

C. STPP Performance

Let $B^{(\mathrm{Rev})}_{\mathrm{Split-TCP}}$ the average bandwidth utilized in the satellite reverse channel for the *split-TCP* case. Then, we can compute the *relative bandwidth gain* as

$$\mathrm{gain} = \frac{B^{(\mathrm{Rev})}_{\mathrm{Split-TCP}} - B^{(\mathrm{Rev})}_{\mathrm{STPP}}}{B^{(\mathrm{Rev})}_{\mathrm{STPP}}}. \qquad (27)$$

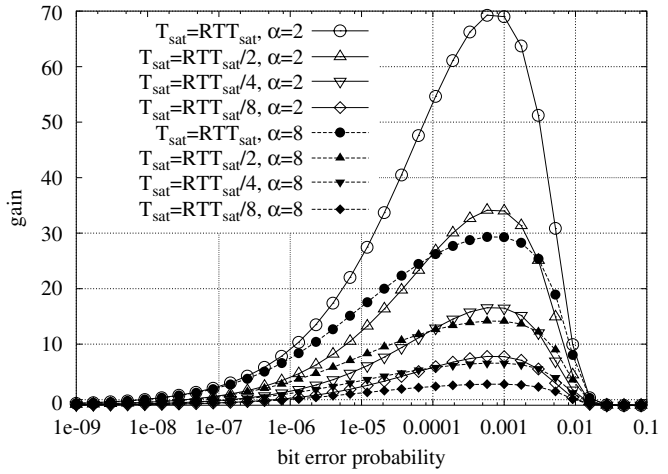

Fig. 14. Relative bandwidth gain between STPP and NACK based algorithm (split-TCP).

Observe that analytical evaluation of the gain value can be easily achieved. In fact, the value of $B^{(\mathrm{Rev})}_{\mathrm{STPP}}$ can be computed as given in (25), whereas $B^{(\mathrm{Rev})}_{\mathrm{Split-TCP}}$ is given by

$$B^{(\mathrm{Rev})}_{\mathrm{Split-TCP}} = (\max(1, n_e) \cdot O)/\mathrm{RTT_{sat}} \qquad (28)$$

where n_e is given in (24).

In Fig. 14, we report the relative feedback bandwidth gain as a function of the satellite channel bit-error rate. We considered the optimal PDU size and different values of α. Observe, that the gain heavily depends on the way in which the error mask is stored into STAT PDU's (α factor, defined in Section IV-F as the average number of bits required to signal an erroneous STPP PDU in a STATUS message). TCP split is a bandwidth consuming scheme; PETRA allows to save bandwidth; the measure of the saving depends on the average number of bits employed to signal the status. A simple algorithm could be implemented by transmitting a vector (the STAT mask) containing: the sequence number of the first packet whose status is described ($\mathrm{SNfirst}, 8$ bits), the sequence number of the last packet ($\mathrm{SNlast}, 8$ bits) and ($\mathrm{SNlast} - \mathrm{SNfirst} + 1$) fields of 1 bit, where the symbol "0" means *correct* and "1" means *errored*. In this case, if the status of 100 segments is transmitted, the average number of bits employed (α) is $116/100 = 1.16$. Alternatively, if all the sequence numbers were transmitted, 100 fields of 8 bits each would be required ($\alpha = 8$). Other schemes, including compression algorithms developed for image processing, can be applied, so reducing the α value (also below 1). On the other hand, some redundancy may be necessary to protect status information and, in this case, the α value would increase. A couple of example values have been chosen for our tests: $\alpha = 2$ and $\alpha = 8$.

At low bit-error rates the NACK based solution can outperforms the STPP one (when $\mathrm{RTT_{sat}}/T_{\rm stat} > 1$). This is due to the fact that STPP always transmits $\mathrm{RTT_{sat}}/T_{\rm stat}$ STAT messages in each $\mathrm{RTT_{sat}}$. Instead, at low error rates, the number of NACKs sent in a RTT in the split-TCP case is equal to one. Moreover, observe that the relative bandwidth gain increases as $T_{\rm stat}$ increases. It is important to observe that the solution "PETRA $T_{\rm stat} = \mathrm{RTT_{stat}}/8$," which offers good and robust performance, can guarantee also a relevant bandwidth gain if employed with a low α value. The value $\alpha = 2$, shown in the results, may be further reduced, as said above.

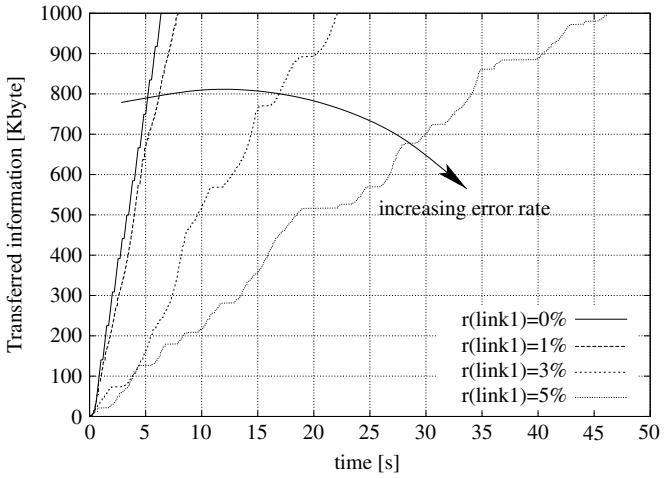

Fig. 15. End-to-end transferred data: $\mathrm{RTT}_{\mathrm{wired}}(1) = \mathrm{RTT}_{\mathrm{wired}}(2) = 100$ ms, $\mathrm{RTT}_{\mathrm{sat}} = 0.5$ s, satellite channel bit-error rate $P_e = 0.0001$, $B_{\mathrm{sat}} = 2$ Mb/s, $B_{\mathrm{wired}}(1) = B_{\mathrm{wired}}(2) = 2$ Mb/s, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{payload}) = 1$ kB, $\mathrm{LTL}_{\mathrm{len}}(\mathrm{header}) = 40$ bytes, $W_{\mathrm{max}}^{\mathrm{LTL}} = 32$ LTL segments, 1 MB data transfer, $T_o(\mathrm{UTL}) = 20$ s.

D. Effects of Additional Errors on Terrestrial Paths

Referring to Fig. 2, we label as link1 and link2 the leftmost and the rightmost terrestrial links, respectively. Moreover, we denote the packet error rate affecting the two above links as $r(\mathrm{link1})$ and $r(\mathrm{link2})$.

Here, as an example we consider only the case in which errors affect the first terrestrial link, i.e., link1. In Fig. 15, we show the simulation traces of the end-to-end transmitted bytes for different values of the link error rate $r(\mathrm{link1})$. As expected, as $r(\mathrm{link1})$ increases, performance decreases. As an example, when compared with the error free case, the LTL *transfer time* triples when the packet error rate is $r(\mathrm{link1}) = 3\%$, whereas it becomes nine times longer by considering an error rate of 5%. This is because due to the above errors, the wired segment becomes the bottleneck and, therefore, the satellite pipe cannot be filled.

VI. Conclusion

In this paper, we have presented a satellite adapted transport architecture (PETRA) which can be deployed to improve performance in communication scenarios involving satellite systems.

PETRA is based on the split-TCP policy. The end-to-end connection is divided into different segments, where a different transport protocols can be deployed in each network segment. Accordingly, customized solutions can be used in the satellite segments, while traditional TCP should be applied in the terrestrial segments. This provides fairness with respect to other TCP flows in the network, and allows standard TCP applications to be easily ported to our architecture. In order to guarantee reliability, the end-to-end semantic is maintained by introducing an appropriate UTL.

The proposed architecture has been presented in detail, and all the design parameters have been appropriately dimensioned. Both analysis and simulation results have been presented to assess our architecture, which has shown very good performance. A proper tuning of the parameters guarantees satisfying results, as well as bandwidth savings, also in presence of relevant bit errors rates both over forward and feedback channels.

Appendix
Derivation of the Function $\mathrm{TX}(\cdot)$

We refer to standard TCP that, after the connection setup phase, is initiating the transmission phase. At the beginning of this phase, the TCP is in slow-start, i.e., its congestion window, W, is incremented by one full segment for each received ACK. This phase finishes when W reaches the TCP window threshold value, W_{thresh} that at the beginning of the connection is always set to the maximum congestion window size W_{max}. Here, we are interested in the computation of the function $\mathrm{TX}(t)$ (assuming no packet errors), i.e., on the number of full transmitted TCP segments between time 0 (where the connection started) and time $t \geq 0$. We call t_{tx} the time needed to transmit a full TCP packet, $t_{\mathrm{tx}} = TCP_{\mathrm{len}}/B_{\mathrm{wired}}$, where $\mathrm{TCP}_{\mathrm{len}}$ is the total length (header and payload)[13] of a TCP packet expressed in bits, whereas B_{wired} is the available bandwidth on the terrestrial part of the connection. We approximate[14] the number of TCP segments transmitted in one entire $\mathrm{RTT}_{\mathrm{wired}}$ with the quantity: $n_{\mathrm{rtt}} = \lfloor \mathrm{RTT}_{\mathrm{wired}}/t_{\mathrm{tx}} \rfloor$. Moreover, the instant t can be written as the sum of two terms

$$t = t_1 + t_2 = \left\lfloor \frac{t}{\mathrm{RTT}_{\mathrm{wired}}} \right\rfloor \mathrm{RTT}_{\mathrm{wired}} + \Delta_t \qquad (29)$$

where the first term (t_1) accounts for the number of complete rounds, while the second $(t_2 = \Delta_t)$ gives the fraction of time elapsed in the current round. In the no delayed ACK case, the window size at the generic round i is equal to: $W(i) = \min(2^{i-1}, W_{\mathrm{max}})$. The number corresponding to the current round (starting from round 1) is given by $c_r = \lfloor (t/\mathrm{RTT}_{\mathrm{wired}}) \rfloor + 1$. Hence, the maximum window W_{cr} reached in the current round (c_r) can be computed as $W_{\mathrm{cr}} = W(c_r)$. $\mathrm{TX}(t)$ can be computed by summing $\mathrm{TX}(t_1)$ to $\mathrm{TX}(t_2)$. $\mathrm{TX}(t_1)$ is computed as follows:

$$\mathrm{TX}(t_1) = \begin{cases} \sum_{i=1}^{c_r-1} \min(2^{i-1}, n_{\mathrm{rtt}}), & c_r > 1 \\ 0, & \text{elsewhere} \end{cases}. \qquad (30)$$

For what concerns $\mathrm{TX}(t_2)$

$$\mathrm{TX}(t_2) = \begin{cases} p, & W_{\mathrm{cr}} \geq p \\ n_{\mathrm{rtt}}, & W_{\mathrm{cr}} < p \end{cases} \qquad (31)$$

where p is the integer number of transmission periods in t_2: $p = \lfloor (t - t_1)/t_{\mathrm{tx}} \rfloor$. Finally, $\mathrm{TX}(t)$ is obtained by $\mathrm{TX}(t) = \mathrm{TX}(t_1) + \mathrm{TX}(t_2)$.

Acknowledgment

The authors wish to thank E. Kristiansen for his many valuable comments and suggestions.

[13] In this paper, we consider the TCP segments to be of fixed size given by the maximum segment size, MSS.

[14] The error introduced in the computation of $\mathrm{TX}(t)$ is negligible when B_{wired} is high. When this error is not negligible (at very low B_{wired}), $\mathrm{TX}(t)$ is always less than the actual number of transmitted segments, hence the analysis performed in this document is still valid (being a worst case dimensioning).

REFERENCES

[1] J. Postel, "Transmission control protocol—Darpa Internet program—Protocol specification," IETF, RFC 793, Sept. 1981.

[2] S. Kota, R. Jain, and R. Goyal, "Broadband satellite network performance," *IEEE Commun. Mag.*, vol. 37, pp. 94–95, July 1999.

[3] T. R. Henderson and R. H. Katz, "Transport protocols for internet-compatible satellite networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 326–344, Feb. 1999.

[4] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," IETF, RFC3135, June 2001.

[5] M. Marchese, "TCP modifications over satellite channels: Study and performance evaluation," *Int. J. Satell. Commun., Special Issue on IP*, vol. 19, no. 1, pp. 93–110, Jan./Feb. 2001.

[6] D. E. Brooks, C. Buffinton, D. R. Beering, A. W. William, D. Ivancic, M. Zernic, and D. J. Hoder, "ACTS 118x Final Report, high speed TCP interoperability testing," NASA, NASA/TM-1999-209 272, July 1999.

[7] W. Ivancic, D. Brooks, B. Frantz, D. Hoder, D. Shell, and D. Beering, "NASA's broadband satellite networking research," *IEEE Commun. Mag.*, vol. 37, pp. 40–47, July 1999.

[8] ARTES 3, The ESA multimedia initiative [Online]. Available: http://www.estec.esa.nl/artes3

[9] D. Adami, M. Marchese, and L. S. Ronga, "TCP/IP based multimedia applications and services over satellite links: Experience of an ASI/CNIT project," *IEEE Pers. Commun. Mag., Special Issue Multimedia Commun. Over Satellites*, vol. 8, pp. 20–27, June 2001.

[10] *Consultative Committee for Space Data Systems (CCSDS), Space Communications Protocol Specification-Transport Protocol*, CCSDS 714.0-B-1, Blue Book, May 1999.

[11] Eur. Telecommun. Standards Inst.—ETSI [Online]. Available: http://www.etsi.org

[12] C. Partridge and T. J. Shepard, "TCP/IP performance over satellite links," *IEEE Network*, vol. 11, pp. 44–49, Sept./Oct. 1997.

[13] T. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.

[14] N. Ghani and S. Dixit, "TCP/IP enhancement for satellite networks," *IEEE Commun. Mag.*, vol. 37, pp. 64–72, July 1999.

[15] M. Allman, S. Dawkinks, D. Glover, J. Griner, T. Henderson, J. Heidemann, S. Ostermann, K. Scott, J. Semke, J. Touch, and D. Tran, "Ongoing TCP research related to satellites," IETF, RFC2760, Feb. 2000.

[16] C. Barakat, E. Altman, and W. Dabbous, "On TCP performance in a heterogeneous network: A survey," *IEEE Commun. Mag.*, vol. 38, pp. 40–46, Jan. 2000.

[17] I. Akyildiz, G. Morabito, and S. Palazzo, "Research issues for transport protocols satellite IP networks," *IEEE Pers. Commun. Mag.*, vol. 8, pp. 44–48, June 2001.

[18] A. Ephremides, "Guest Editor," *Int. J. Satell. Commun., Special Issue IP*, vol. 19, no. 1, Jan./Feb. 2001.

[19] V. Bharadwaj, J. S. Baras, and N. P. Butts, "An architecture for internet service via broadband satellite networks," *Int. J. Satell. Commun., Special Issue IP*, vol. 19, no. 1, pp. 29–50, Jan./Feb. 2001.

[20] Y. Zhang, D. D. Lucia, B. Ryu, and S. K. Dao, "Satellite communications in the global Internet: Issues, pitfalls, and potential," presented at the Internet Global Summit Conf. (INET'97), Kuala Lumpur, Malaysia, June 1997.

[21] H. Kruse, M. Allman, J. Griner, and D. Tran, "Experimentation and modeling of HTTP over satellite channels," *Int. J. Satell. Commun., Special Issue IP*, vol. 19, no. 1, pp. 51–69, Jan./Feb. 2001.

[22] M. Marchese, "Performance analysis of the TCP behavior in a GEO satellite environment," *Comput. Commun. J., Special Issue Performance Evaluation of Telecommunication Systems: Models, Issues and Applications*, vol. 24, no. 9, pp. 877–888, May 2001.

[23] ——, "Proposal of a modified version of the slow start algorithm to improve TCP performance over large delay satellite channels," *Proc. IEEE ICC*, vol. 10, pp. 3145–3149, June 2001.

[24] R. Goyal, R. Jain, M. Goyal, S. Fahmi, B. Vandalore, S. Kota, N. Butts, and T. vonDeak, "Buffer management and rate guarantees for TCP over satellite-ATM networks," *Int. J. Satell. Commun., Special Issue IP*, vol. 19, no. 1, pp. 93–110, Jan./Feb. 2001.

[25] M. Marchese, "TCP/IP-Based Protocols Over Satellite Systems: A Telecommunication Issue," in *Reliability, Survivability and Quality of Large Scale Telecommunication Systems*, P. Stavroulakis, Ed. Chichester: Wiley, 2003, pp. 167–198.

[26] V. Jacobson and R. Braden, "TCP Extensions for Long-Delay Paths," IETF, RFC2018, Oct. 1988.

[27] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP over satellite channels using standard mechanisms," IETF, RFC2488, Jan. 1999.

[28] I. Akyildiz, G. Morabito, and S. Palazzo, "TCP-peach: A new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Networking*, vol. 9, pp. 307–321, June 2001.

[29] ——, "TCP-peach for satellite networks: Analytical model and performance evaluation," *Int. J. Satell. Commun.*, vol. 19, no. 5, pp. 429–442, Sept./Oct. 2001.

[30] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. ACM MOBICOM'01*, Rome, Italy, July 2001, pp. 287–297.

Mario Marchese (S'94–M'97) was born in Genoa, Italy, in 1967. He received the "Laurea" degree (*cum laude*) from the University of Genoa, Genoa, Italy, in 1992, the Qualification degree as Professional Engineer in April 1992, and the Ph.D. (Italian "Dottorato di Ricerca") degree in telecommunications from the University of Genoa, in 1996.

After a short industrial experience with Marconi S.p.A., Genoa, he was a Member of the Research Staff of the Telecommunication Networking Research Group, University of Genoa, with a Postdoctoral Scholarship. Since 1999, he has been with the Italian Consortium of Telecommunications (CNIT), University of Genoa Research Unit, where he is now Head of Research. He is the Official Representative of CNIT within the European Telecommunications Standard Institute (ETSI). He is the author and coauthor of more than 80 scientific works, including international magazines, international conferences, and book chapters. His main research activity concerns: TCP/IP protocols, satellite networks, transport protocols for satellite links, ATM networks and related topics, and best-effort multimedia networks.

Dr. Marchese is Secretary of the IEEE Satellite and Space Communications Technical Committee.

Michele Rossi (S'02) was born in Ferrara, Italy, on October 30, 1974. He received the Laurea degree (*summa cum laude*) in electrical engineering from the University of Ferrara, Ferrara, Italy, in 2000. He is currently working toward the Ph.D. degree at the same university.

During the academic year 2000–2001, he was a Research Fellow in the Department of Engineering. Since April 2003, he is doing research at the Center for Wireless Communications (CWC), University of California at San Diego, La Jolla. His research interests include TCP/IP protocols on wireless networks, TCP/IP header compression, performance analysis of selective repeat link layer retransmission techniques, efficient multicast data delivery, and mobility in 3G cellular networks.

Giacomo Morabito (M'02) received the Laurea degree in electrical engineering and the Ph.D. degree in electrical, computer, and telecommunications engineering from the University of Catania, Catania, Italy, in 1996 and 2000, respectively.

From November 1999 to April 2001, he was with the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, as a Research Engineer. He is with the School of Engineering at Enna, University of Catania, since May 2001, where he is currently an Assistant Professor. His research interests include mobile and satellite networks, quality-of-service, and traffic management.

Dr. Morabito has served and/or is serving on the Editorial Boards of *Computer Networks* and the *IEEE Wireless Communications Magazine*, Guest Editor of *Computer Networks* and *MONET*, and is a Member of the technical program committee on several conferences. He will be the Technical Program Cochair of Med-Hoc-Net 2004.