Satellite Transport Protocol Handling Bit Corruption, Handoff and Limited Connectivity

MAGED E. ELAASAR
IBM Rational Software

MICHEL BARBEAU, Member, IEEE

EVANGELOS KRANAKIS
Carleton University
Canada

ZHEYIN LI
Nortel Networks

Being both wireless and mobile, low Earth obiting (LEO) satellite access networks have a unique set of link errors including bit corruption, handoff, and limited connectivity. Unfortunately, most transport protocols are only designed to handle congestion-related errors common in wired networks. This inability to handle multiple kinds of errors results in severe degradation in effective throughput and energy saving, which are relevant metrics for a wireless and mobile environment. A recent study proposed a new transport protocol for satellites called STP that addresses many of the unique problems of satellite networks. There was, however, no explicit attempt to implement a differentiating error control strategy in that protocol. This paper proposes grafting a new probing mechanism in STP to make it more responsive to the prevailing error conditions in the network. The mechanism works by investing some time and transmission effort to determine the cause of error. This overhead is, however, recouped by handsome gains in both the connection's effective throughput and its energy efficiency.

Manuscript received January 16, 2004; revised September 27, 2004; released for publication November 18, 2004.

IEEE Log No. T-AES/41/2/845780.

Refereeing of this contribution was handled by T. F. Roome.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS).

Authors' current addresses: M. E. Elassar, IBM Rational Software, 770 Palladium, Kanata, Ontario, Canada K2V 1C6; M. Barbcau and E. Kranakis, School of Computer Science, Carleton University, 5302 Herzberg Labs, 1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6; Z. Li, Amazon.com, 1200 12th Ave. S., Suite 1200, Seattle, WA 98144-2734.

I. INTRODUCTION

The work presented here focuses on enhancing the behavior of transport protocols over networks containing low Earth orbiting (LEO) satellite links. Although wireless and satellite links certainly share a lot of common characteristics, like high bit-error rates (BERs) and intermittent connectivity, they also have enough distinct properties to be taken as different environments for data transport. As surveyed by space communication protocol standards (SCPS) [1], these properties include highly variable round trip times, asymmetric up and down link capacities, limited computing resources (power, memory and speed), and relatively higher throughput.

Katz and Henderson [2] proposed the satellite transport protocol (STP) for satellite networks. This paper proposes a new error control strategy for STP that makes it more adaptive to the unique error conditions in satellite access networks. The new strategy is based on an end-to-end probing mechanism that uses the persistence of the error condition as an indication to the kind of prevailing error in the network. Whenever an error is detected, data transmission is suspended. Time and transmission effort are invested to sense the current delays in the network. Error conditions accompanied by noticeable delays are associated with network congestion. Otherwise, errors are associated with different link error events. The strategy proposed here is based on an earlier one proposed by Tsaoussidis and Badr in the context of transmission control protocol (TCP) called TCP-probing [3]. However, the new strategy leverages many unique features of STP to enhance the quality of error control. It reuses STP's acknowledgment polling cycle as a probing mechanism as well as for early error detection. It also uses STP's selective negative acknowledgement as an explicit error indication and a way to detect premature activation. Finally, it makes use of STP's lack of reliance on timeouts to finish faster. The new probing mechanism preserves the end-to-end semantics of STP and is a configurable option of the sender only.

Using simulation, the work presented here shows that the new probing mechanism for STP indeed improves the effective throughput and increases the energy efficiency of the protocol under various network error conditions. The simulation is carried in PIX (Protocol Implementation Framework for Linux) [4]. The main component of the simulation is the extended STP (XSTP). XSTP is the STP protocol extended with our probing mechanism, called XSTP-probing.

The rest of the paper is organized as follows. Section II reviews the satellite link properties and literature about the available transport protocols and some proposed extensions for improving them in satellite access networks. Section III describes the design of the new XSTP protocol. Section IV details the new XSTP-probing mechanism. Section V explains the simulation framework. Section VI presents and reflects on the simulation results. Integration testing and performance over packet radio are discussed in Section VII. Section VIII concludes with a summary.

II. TRANSPORT PROTOCOLS OVER LEO SATELLITES

A. Link Properties

Since a satellite link is a special kind of wireless link, it naturally inherits all of a wireless link's characteristics. Satellite links are described as having a natural broadcast capability and an inherent ability to reach mobile users [5, 6, 1].

Satellite links also cover large areas, resulting in reduced switching and forwarding overhead. Being radio-based, satellite links can have limited bandwidth due to the natural restrictions and international agreements that control the allocation of the radio spectrum.

Satellite networks are characterized by having asymmetric links, usually due to the high cost of the needed technology to support both directions. This asymmetry is typically manifested in the links' speeds, bandwidth, or both. It impacts the performance of transport protocols that use acknowledgements in the reverse direction as a self-clocking mechanism. Having slower arriving acknowledgments hinders transmission speeds in the forward direction. Another restricting feature of LEO satellites is their limited computing resources largely due to power and size limitations. Power restrictions in particular can affect the stability of a connection and the frequency of errors, putting a damper on the overall performance of a transport protocol.

Similar to other wireless links, satellite links are lossy with potentially high BERs, resulting in frequent packet drops. This effect is mainly due to several environmental factors (rain, pollution) that cause noise, multi-path distortion and shadowing of radio channels. However, advanced error control coding is sometimes used in the link layer to mitigate such problems. It is still challenging though to hide the side effect of those solutions from unaware transport protocols. In fact, most reliable transport protocols have an inherent assumption that any loss is due to network congestion. Therefore, when a segment loss is detected, these transport protocols switch to a more conservative stance, effectively underutilizing the link.

Another problem for LEO satellites is their intermittent connectivity due to their constant movement in their orbits. The window of connectivity to a given satellite is around 10% of the time [1]. A handoff occurs when one LEO satellite goes out of range and another one goes in range. It can also happen when a LEO satellite leaves the range of some base-station and enters the range of another. The connectivity can also be lost for more extended periods due to a physical obstruction of the satellite signal or an inefficient distribution of the LEO satellites by the service provider. Both handoff and limited connectivity can lead to periods of blackout, during which all packets get dropped. As mentioned by Allman et al. [5], these periods are usually confusing for unaware transport protocols and may lead them to take wrong decisions (like invoking congestion control) that severely affect their performance.

B. Unique Challenges

All data networks, including satellite access networks, are run by communication protocol stacks. The performance of any data connection is a direct reflection of the aggregate performance of all protocols in the stack. The transport protocol is crucial. The work presented here explores transport protocols that provide end-to-end reliable (complete, correct, in-sequence and without duplication) transmission service. Many standard transport protocols (like TCP) are generally unaware of the specific characteristics of their underlying networks. According to Tsaoussidis and Matta [7], these protocols are originally designed to address the problems and satisfy the transport goals of wired networks. Therefore, such protocols assume continuous connectivity, data loss resulting from network congestion and balanced bidirectional links. These protocols are also calibrated to overcome the problems of stability and heterogeneity in terms of receiver buffers, network bandwidth, and delay. In addition, such protocols strive for fairness in bandwidth consumption and efficiency in link utilization by adopting proper congestion control mechanisms.

These features perform well for wired, high-speed networks. However, they fail miserably for satellite data networks, rendering their performance unacceptable. As explained by researchers at SCPS [1], transport protocols targeted for satellite data networks should strive to provide fair link access, high aggregate throughput and high reliability through differentiating between unique error conditions. Transport protocols should also base bandwidth utilization on a precedence policy, maximize link utilization and provide an optional semi-reliable service when needed. The foregoing is in addition to providing new or updated algorithms to handle the unique properties of satellite networks mentioned earlier. These objectives call for revisiting the standard transport protocols and possibly proposing new ones. Specifically, Tsauoussidis and Matta [7] outlined major features that transport protocol designers have to work on: correct detection of the nature of error (sojourn time, frequency, etc.), implementation of different error-recovery strategies (aggressive, conservative or more fine tuned) which are sensitive to the nature of error detected, optimization of energy expenditure and connection time utilization and bypassing the problems of asymmetric links or having different mechanisms handling problems in each direction. Tsauoussidis and Matta added new performance metrics, other than traditional effective throughput and total expended time, to fairly judge the protocol's performance. They proposed the use of energy efficiency and transmission overhead as more appropriate metrics. These metrics address the nature of the battery-powered devices that typically empower wireless and satellite networks.

C. Error Control Strategies

Many studies showed that the ability of a protocol to correctly classify the nature of the detected error can make all the difference for the performance [3, 8–14]. Not less important is the ability to take the right action in light of the perceived loss and to predict future losses. Protocols that lack the ability to distinguish errors run the risk of taking an aggressive stance in response to deteriorating link conditions or wasting bandwidth resources in response to infrequent transient errors. Both situations lead to inefficient energy utilization by the protocol.

In the absence of explicit network feedback, transport protocols need to rely on other methods to distinguish the different error conditions. Congestion errors, common in wired networks, occur when one or more intermediate routers overflow as a result of being overwhelmed by the incoming traffic. These errors are usually accompanied by noticeable increase in delays. All reliable transport protocols should respond to this event by slowing down their transmission rate, or in other words shrinking their sending window. Failure to do so can affect the fair sharing of bandwidth between competing connections. A more catastrophic result can happen when the network reaches congestive collapse, a situation where large numbers of packets get dumped from the routers.

On the other hand, link errors can vary in nature. According to Tsaoussidis and Matta [7], link errors are usually characterized by both their sojourn time and frequency. Generally, the more frequent the link errors, the worse the throughput gets. As standard transport protocols mistake these errors for congestion, they slow down their transmission, and hence become unnecessarily overconservative. Transport protocols with better error control strategies expend some transmission overhead investigating the error condition. Determining the exact cause of a link error is quite intriguing. However, it is usually enough to conclude on the burstiness and frequency of such an error.

D. State of the Art

The TCP has become the defacto standard for the Internet today. More than twenty years of research have produced a protocol that is perfect for today's high performance networks. Naturally, TCP was one of the first chosen transport protocols for the new wireless, satellite, and heterogeneous networks. Quickly, it became obvious that the protocol needs some improvements to perform as well in these new environments. Many researchers took the approach of proposing extensions to TCP to make it more efficient in these networks. The availability of current test-beds and the strong user base are definite advantages to this approach. Akyildiz, Morabito, and Palazzo proposed TCP-Peach [15]. TCP-Peach addresses satellite networks with long propagation delays and high link error rates. Two new algorithms embedded into TCP are introduced, namely, sudden start and rapid recovery. Both the sender and receiver need to be modified to support the new algorithms. Sudden start addresses the problem of long propagation delays that lengthens significantly the duration of the TCP slow start. Rapid recovery addresses the problem where TCP interprets segment losses as congestion whereas on satellite links it is more probably due to transmission errors. Because of the misinterpretation, TCP helplessly decreases the transmission rate. They integrate an algorithm that can distinguish between congestion situations and transmission error conditions. The availability of network resources is probed with dummy segments (i.e., segment without data traffic) and ACKs for dummy segments. These are distinguished from other segments using control bits that were unused. Prioritization of segments is required by the routers. Dummy segments are given low priority, and therefore are the first to be discarded by routers in case of congestion. Dummy segments are sent and their successful acknowledgements are interpreted as an indication of available bandwidth. The transmission rate can be augmented. Casetti, Gerla, Mascolo, Sanadidi, and Wang have introduced TCP Westwood [16]. TCP Westwood also consists of an improvement to TCP. It doesn't, however, require new messages. Modifications are required on the sender side only. The central idea in TCP Westwood is the continuous measurement of the bandwidth of the channel by monitoring the arrival rate of acknowledgements. An estimate of the bandwidth is obtained by dividing the amount of data, which delivery is confirmed by an acknowledgement; by the acknowledgement interarrival time. This value is smoothed over time using a low-pass filter. Congestion is detected by the reception of three duplicate acknowledgements. Instead of dividing the congestion window by two, as done in normal TCP, the bandwidth estimate is used. Duplicate acknowledgements indicate also an out-of-sequence segment delivery. There is an ambiguity, though, about which segment is exactly delivered. This ambiguity is resolved by taking the average size of the sent segments. More work in that direction can be found in two recent special issues of the *IEEE Journal on Selected Areas in Communications* [17, 18].

Other researchers took the approach of developing new transport protocols that are more tailored to the characteristics of their network environment. Unlike the first approach, this approach does not involve the painful process of retrofitting proposed extensions into TCP, but rather involves incorporating a lot of them in the original design of the new protocol. This approach usually adds more integrity and less complication to the protocol. However, it runs the risk of not complying to the standards. This approach also lacks a rigorous and comprehensive testing schema, usually defined and available for TCP.

One of those protocols, proposed by Katz and Henderson [2], is the STP. This protocol is designed exclusively for satellite networks. The authors try to keep interface and feature parity with TCP, while designing their protocol from the ground up with satellite networks in mind. Specifically, they address the problems of asymmetry, variable round trip times, and degraded performance in the presence of multiple errors per round trip. These features among others make this protocol better suited than TCP for use in satellite environments. However, the STP protocol is lacking one fundamental feature that is essential to any transport protocol targeted for heterogeneous networks. This missing feature is a discriminating error control mechanism. Unfortunately, STP inherits the assumption that any error results from network congestion. When any error occurs, congestion control measures are applied. This inability to classify and properly handle different error types usually compromises both the protocol's throughput and expended energy.

III. XSTP: EXTENDED SATELLITE TRANSPORT PROTOCOL

XSTP is an extension of the STP protocol. XSTP is typically deployed on top of the Internet protocol. XSTP provides a connection-oriented reliable byte streaming service to application level protocols. This section explains data transfer in the XSTP protocol. Descriptions of other aspects, such as connection establishment can be found in [19].

When it is neither sending nor receiving, an XSTP connection maintains a keep-alive timer to periodically check (by sending a POLL message) whether its peer session is still alive. A connection maintains smoothed round trip time and variance estimates.

When a data segment arrives, it is ignored (if invalid), presented to upper protocols (if next expected) or cached. The cache is a sorted list of segments, received out-of-sequence, and creating gaps in the byte stream. A gap is defined as two consecutive cached segments having nonconsecutive sequence numbers. Cached segments are delivered in-sequence to the application protocols as gaps are filled.

An XSTP receiver sends a USTAT message to the sender when a segment gap is discovered. A reported segment gap is described by a start sequence number and an end sequence number. XSTP is configured to delay this notification until a number of missing segments. If meanwhile a new gap is discovered, then the first gap is reported at once and tracking of the new one is started. Unlike STP, XSTP does not require a reported gap to consist solely of absent segments. Instead, XSTP may report gaps that are partially filled. This process ensures that whatever is missing from a gap, it is promptly reported.

Periodically sent POLL messages are time-stamped to eliminate duplicates. Upon the arrival of a POLL message, the receiver returns a STAT message, with the received timestamp, acknowledging received segments and containing a gap list constructed by traversing the cache. The polling rate is a parameter. The higher the rate, the faster the segments get acknowledged, but also the more acknowledgment overhead is incurred. To minimize overhead, POLL messages are piggybacked with data segments if they happen to be scheduled at the same time for transmission. USTAT messages, similar in content to STAT messages, provide unsolicited feedback to a sender. Upon receiving feedback, a sender opens its congestion window either exponentially or linearly depending on whether it is in the slow start or in the congestion avoidance phase.

Messages pushed down from the application are not transmitted right away. The data is put in a send buffer. Lack of buffer space momentarily suspends application message processing. At any point in time, an XSTP sender is transmitting, persisting, or sitting idle. XSTP enters the transmission state when there is enough data to transmit and there is room in the transmission window. The transmission window controls the number of segments that are allowed to be transmitted. This window is calculated as the minimum of the receiver's advertised window size and the sender's congestion window size. The sender's transmission rate is based on a send timer. This timer paces the transmission uniformly across the round trip to minimize the risk of creating huge bursts into the network. An XSTP sender is not able to transmit if a receiver advertises a zero window size. In that case, the sender exits the transmission state and goes into the persistence state. It suspends transmission and periodically sends POLL segments to request window updates. Only after receiving a nonzero window size does the sender return to the transmission state. When the sender is neither transmitting nor persisting, it is in the idle state.

IV. XSTP-PROBING MECHANISM

The new mechanism is called XSTP-probing. Upon detecting a segment loss, the level of congestion in the network is assessed. If congestion is detected, then XSTP-probing responds by invoking congestion control; otherwise it resumes with immediate recovery, which restores the congestion window to the same level as before probing [3]. XSTP-probing also adapts to the level of error in the network by suspending new data transmission and by striving to send only in error-free intervals. XSTP-probing is modeled after TCP-probing, proposed by Tsaoussidis and Badr [3].

The goal is to adapt the sender's transmission rate to the varying network error conditions. It is usually accomplished by taking an aggressive stance when an error is found to be transient and a conservative one when it is found to be persistent. XSTP-probing goes further by probing the connection for possible error-free conditions and only transmitting in those intervals. Upon detecting a loss, data transmission is suspended. A probing phase is initiated to collect RTT statistics. These RTT statistics are compared with the RTT estimate available when the loss was discovered. Interestingly, the duration of the probing cycle is proportional to the level of error in the network This excludes a sender from the error conditions. After the probing phase is finished and if congestion is detected by proliferating RTTs, congestion control is immediately invoked. Otherwise, transmission levels are restored without taking any action.

TCP-probing introduces several new message types and changes to both the sender and receiver. XSTP-probing does not introduce new message types and is a sender-only mechanism. This simplifies the implementation. XSTP-probing reuses the polling cycle of XSTP, which is the protocol's low frequency acknowledgement mechanism, as its probing mechanism. The POLL segment is the probe and the STAT segment is the probe acknowledgment. An XSTP receiver is kept unaware of whether the received POLL is a probe or just a normal POLL.

The behavior of XSTP-probing is pictured by a state diagram in Fig. 1. XSTP-probing is triggered when a loss is discovered either implicitly or explicitly. The implicit method is called early timeout. It consists of a break in the POLL/STAT segment interleaf. XSTP transmits a configurable number of POLL segments every round trip. After the first round trip, STAT segments start to arrive. The rate of arriving STAT segments becomes similar to the rate of leaving POLL segments, producing an interleaved pattern of a sent POLL followed by a received

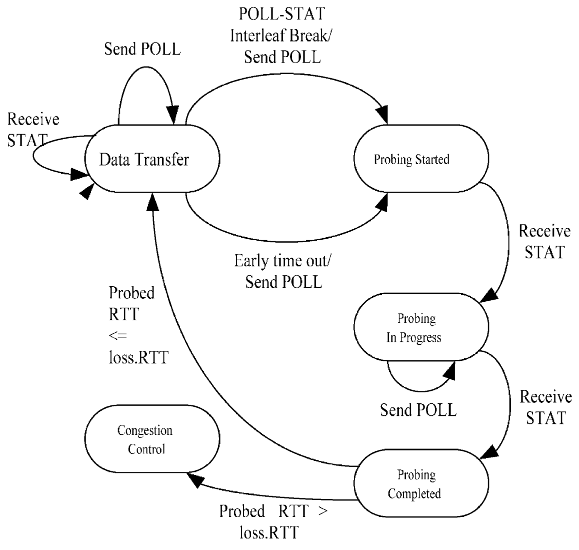


Fig. 1. State diagram of XSTP.

STAT. If either a POLL or a STAT is dropped, then the session detects a break in the interleaf pattern within a maximum of one RTT and a minimum of RTT/POLLS_PER_RTT. The explicit loss detection method relies on receiving feedback from the receiver in the form of either a STAT message or a USTAT message. It is after finding at least one segment worthy of retransmission that XSTP-probing is triggered. This condition avoids premature triggering due to a false alarm. This method can be contrasted to TCP-probing's three DUPACK heuristic, which is only a best effort and may lead to premature probing. When XSTP-probing starts, new data transmission is suspended, the current timestamp and RTT estimate (loss.TS and loss.RTT) are recorded.

Probing is conducted in order to obtain two RTT measurements. A POLL message is sent every RTT. The timestamp of every POLL message is entered in a table (finite number of entries is assumed and old entries are deleted to make room for new entries). Whenever a STAT is received, the RTT measurement is associated to the corresponding timestamp in the table. A STAT message reporting a gap is interpreted as an indication that probing was started prematurely due to packet reordering or a false early timeout. In this case, probing immediately terminates. Data transmission is restored at the previous level. A STAT message not related to a timestamp in the table is received if it took too long to arrive (corresponding timestamp entry got deleted) or it corresponds to a POLL segment that had been sent before probing was started (timestamp is smaller than loss.TS). In the former case, the STAT is ignored. In the latter case, it is associated to the timestamp value loss.TS in the table. In other words, the STAT is considered close in time to the error (sent during the same RTT containing the error). Since XSTP normally sends multiple POLL segments per RTT, several preloss STATs can be received. The loss.TS entry is overridden with every arriving such preloss STAT. Interestingly, preloss STATs can allow the probing cycle to finish quickly (in around one RTT) if the first probe STAT also made it on time (since two measurements are required). The RTT may get a little extended (a common phenomenon in LEO satellite links where the RTT experiences moderate variations). USTAT messages, which explicitly trigger probing, are however ignored while probing is in progress. Gap reports are processed only after probing is done. Probing ends when there are two consecutive entries in the table.

At the end of a probing phase, the two probes' RTT measurements are compared with the loss.RTT. If both RTT measurements are less than or equal to the loss.RTT, congestion is not assumed and the error is considered to be link related. Otherwise, congestion control is applied. In some cases, the RTT can moderately vary (for example due to LEO satellite mobility). The extended RTTs can wrongly be interpreted as an effect of congestion. A configurable RTT tolerance parameter mitigates the effect of that phenomenon.

The congestion control measure depends on how long the probing procedure takes to finish. If it is long enough to reach an XSTP's threshold for idle transmission, then the sender goes back to slow start; otherwise the congestion window is reduced by the congestion avoidance algorithm (the window is halved). This congestion control logic is in contrast to TCP-probing where the sender goes back to slow start if probing is triggered by a timeout event. If the timeout later turns out to be inaccurate, then the connection becomes needlessly over conservative. After probing is done, the missing segments reported by the last STAT message are retransmitted and the normal polling rate is restored.

V. SIMULATION ENVIRONMENT

This section describes the simulation environment, performance metrics and test cases. An implementation of the XSTP protocol has been realized using the PIX framework [4] and C++ over Linux. In addition to XSTP, three other protocols are implemented: an application protocol, a link protocol and a protocol encompassing an error-generating model, see Fig. 2. The application protocol (APP) is a bulk data protocol streaming large files. The continuous flow of bulk data allows demonstrating the capabilities of the error control strategy. The queue link protocol (QLP) is based on POSIX message queues. The extended delay and drop protocol (XDELDROP) models delays and drops of packets. It is similar to VDELDROP of x-Kernel [20]. The model is a continuous time Markov chain with two states. Each state has three parameters: a mean sojourn time t_i to model the persistence of the state, a dropping rate r_i to model the severity of the error condition and a delay range $d_i^{\min}$ to $d_i^{\max}$ ($i = 1,2$) to model the prevailing end-to-end delay. When a state is entered, the duration of the stay is exponentially distributed with mean t_i. During that time, incoming packets are received and either dropped, with a probability r_i, or forwarded, after applying the minimum delay $d_i^{\min}$ or maximum delay $d_i^{\max}$. The choice of either delay is random with a uniform distribution, but is invariant during each stay in a state. The minimum delay aims to correspond to the one way delay across a noncongested network. The maximum delay aims to correspond to some level of network congestion delay.

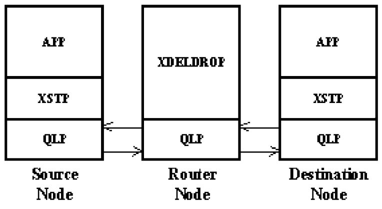


Fig. 2. Simulation configuration.

A LEO satellite access network is simulated (see Fig. 2). There are three nodes: a source, a router, and a destination. In the source and destination, APP, XSTP, and QLP are installed. In the router, XDELDROP over QLP are installed.

The application in the source node is configured to stream a large (10,000,000 byte) file in chunks of 1000 bytes each. This configuration is chosen to avoid any buffering delays and to neutralize Nagle's algorithm, which delays sending small segments. A single connection from the source to the destination is run in each test. Error conditions are simulated: bit corruption, handoff, and limited connectivity.

XDELDROP is in either one of four states: no error, moderate congestion, link error, and severe congestion. The packet forwarding minimum delay is 100 ms and maximum delay is 150 ms. The minimum represents a delay under no congestion. The maximum represents a delay under congestion. The no error state models forwarding all packets after applying the minimum forwarding delay. The moderate congestion state models forwarding all packets after applying the maximum forwarding delay. The link error state models dropping packets while applying the minimum delay to the forwarded ones. The severe congestion state models dropping packets and applying the maximum delays to forwarded ones.

The sender and receiver buffer sizes are set to 64000 bytes, while the maximum segment size (MSS) is set to 1000 bytes, which leads to maximum window size of 64 segments. The polling frequency is also set to three per RTT and sending a POLL with the first burst is enabled. In addition, the USTAT sending threshold is configured to be three out-of-order segments. The initial congestion window is set to one segment size and maximum burst size is set to eight segments. The probing option is configured the same way for all tests in the simulation. The maximum number of traceable probes is set to four (i.e., the size of the table).

The performance metrics are the effective throughput, transmission overhead and throughput/overhead ratio. The effective throughput is defined as the average data rate (in bit/s) from the point of view of the receiver. It is calculated using the following formula:

$$\text{effect. through.} = \text{orig. size}/\text{conn. time.}$$

The transmission overhead is defined as the percentage of extra bytes expended in the transmission of the data bytes. The transmission overhead is calculated as a percentage using the following formula:

$$\text{trans. over.} = ((\text{tot. size} - \text{orig. size})/\text{orig. size}) \times 100.$$

The throughput/overhead ratio is defined as the effective throughput achieved per one percent of expended transmission overhead and is calculated using the following formula:

$$\text{through./over.} = \text{effect. through./tot. trans. over.}$$

It measures the protocol's ability to manage the tradeoff between throughput and overhead.

Tests are conducted to represent three different error conditions: bit corruption, handoff, and limited connectivity. In the bit corruption tests, the nonpacket dropping states (no error or moderate congestion) and packet dropping states (link error or severe congestion) have the same mean sojourn time. In the dropping states, the drop rate is varied from zero to 50%. In the handoff tests, the nondropping states have a larger mean sojourn time than the mean sojourn time of the dropping states. In the dropping states, the drop rate is 100%. Different levels of handoff rates and duration (rendezvous) are tested by varying the mean sojourn time of both the nondropping states and dropping states. To simulate a required handoff rate and rendezvous combination, the mean sojourn time of the dropping state is set to the handoff rendezvous time and the mean sojourn time of the nondropping states is calculated as

$$((\text{rendezvous}/\text{rate}) * 100) - \text{rendezvous}.$$

In the limited connectivity tests, the dropping states have a 100% drop rate. The mean sojourn time of the dropping states is larger than the sojourn time of the nondropping states. Different levels of connectivity rate and duration (rendezvous) are tested by varying the mean sojourn time of the nondropping states and dropping states. To simulate the required connectivity rate and rendezvous combination, the mean sojourn time of the nondropping states is set to the connectivity rendezvous and the mean sojourn time of the dropping states is calculated as

$$((\text{rendezvous}/\text{rate}) * 100) - \text{rendezvous}.$$

VI. SIMULATION RESULTS

The highlights of results of a simulation are discussed in this section. The complete measurement data along with their standard deviation and 95% confidence interval are reported in [19]. XSTP with the probing option turned off/on is referred to as XSTP-OFF/ON. For each test, the total connection time and three kinds of transmission overhead are tracked: the retransmission overhead, forward control overhead, and reverse control overhead.

A. Bit Corruption Tests

In this category of tests, an XSTP-OFF sender usually detects a loss upon the reception of a STAT or a USTAT message reporting gap(s). The sender retransmits the lost segments according to the following conditions. In the case of a USTAT report, the segments should not have been retransmitted before. For a STAT report, enough time (usually one RTT subject to backoff) must have passed since each segment was last transmitted. Should at least one segment need retransmission, the sender performs the required retransmissions before closing its congestion window using the congestion avoidance algorithm. The sender then continues its normal transmission of new data segments only if there is still room in its send window. Although the USTAT acknowledgment mechanism is somehow resilient to premature reporting of gaps due to packet reordering, both the USTAT and STAT mechanisms are not totally immune from reporting bogus gaps. This bogus reporting can occur in the case of unexpected levels of packet reordering in the network.

When an XSTP-ON sender receives a STAT or USTAT gap report, if segments pass the aforementioned retransmission criteria, then it does not perform the retransmission but rather activates the probing. An increase in the RTT compared with the RTT before the probing is taken as a sign of congestion. Measures similar to those applied by XSTP-OFF are applied. If no increase in the RTT is detected, data transmission with immediate recovery is applied.

Fig. 3 shows results of experiments with a mean sojourn time of 10 s. The tests are repeated with different error intensities ranging from 0 to 50%. XSTP-ON achieves consistently higher effective throughput than XSTP-OFF (from 3 to 150%). The total overhead is plotted against the various bit corruption rates in Fig. 4. The probing reduces the overhead by a significant ratio (44% to 50%) at error rates of 20% and above. As XSTP activates probing,

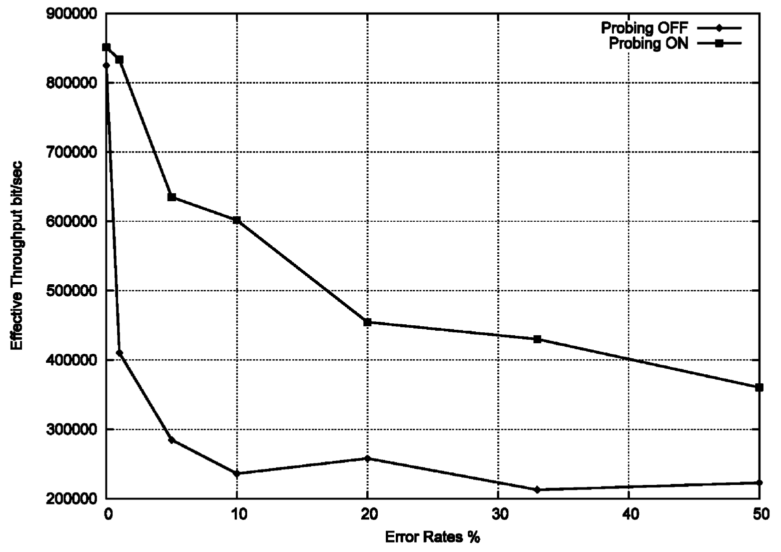


Fig. 3. Effective throughput under bit corruption with 10 s mean sojourn time.

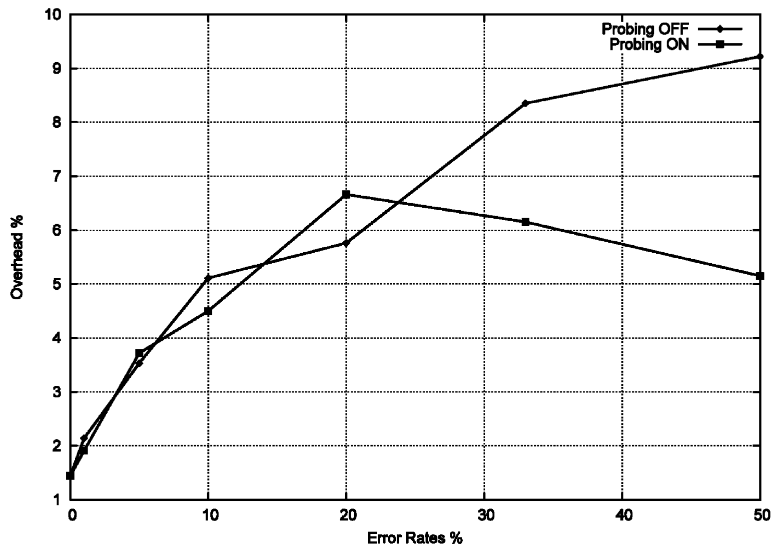


Fig. 4. Overhead under bit corruption with 10 s mean sojourn time.

it suspends new data transmission to protect new packets from getting dropped. Through that probing cycle, the mechanism effectively waits until the error condition clears away before committing to new data transmission. The throughput/overhead ratio is a metric to measure the amount of throughput achieved per one percent of overhead. In Fig. 5 it is shown that this metric is inversely proportional to the error rate. As error deteriorates the amount of throughput achievable as a percentage of overhead decreases. The figures also show that XSTP-probing helps the session achieve more effective throughput with a lower level of overhead expenditure.

B. Handoff Tests

A handoff occurs due to the mobility of both the user terminal and satellite. The event takes place when either a satellite or a base station is switched over. In these tests, packets are dropped in both directions in the link error state with a 100% probability, marking periods of blackouts. Fig. 6 plots results of tests performed with handoff rendezvous of 1 s (suspension of connection) and handoff rates ranging from one to 15%. Reference [19] also presents tests with a handoff rendezvous of other durations.

In the no error state, packets are forwarded after being subjected to one of the two configured delays. If the error condition occurs with noticeable increases in RTT, it is indicative of congestion. The sojourn time and frequency of that condition depends on the competing traffic in the network. However, if that period occurs without any RTT extensions, it is indicative of a handoff event.

XSTP-ON achieves consistently higher effective throughput than XSTP-OFF (from 4 to 86% gain).

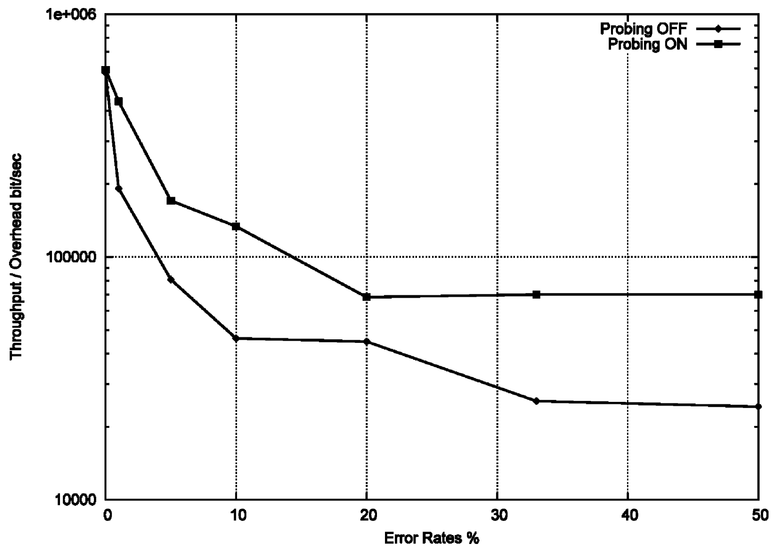


Fig. 5. Throughput/overhead ratio under bit corruption with 10 s mean sojourn time.

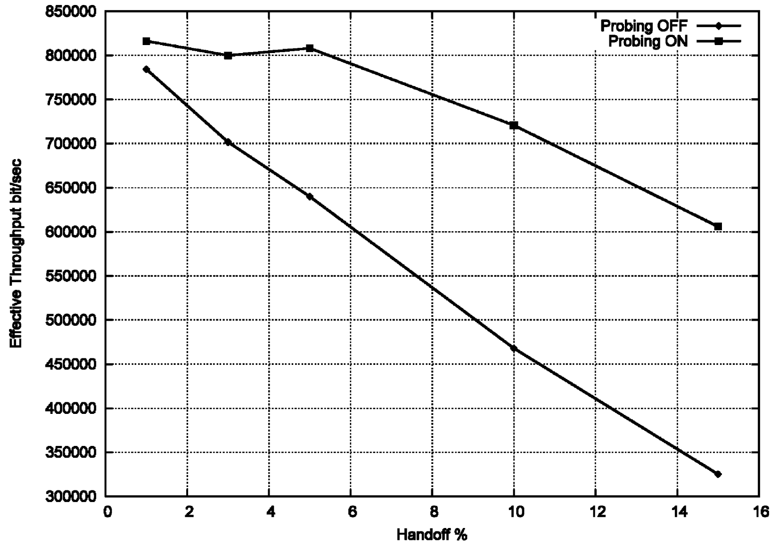


Fig. 6. Effective throughput under handoff with 1 s rendezvous.

An XSTP-OFF sender does not detect the blackout period, hence the loss, until the period is over and either a STAT or a USTAT segment brings back a nonempty gap report. This inability to sense the loss in connectivity makes the session susceptible to draining some or all of the segments in its send window. Also, as a result of applying congestion control measures, the session slows down its transmission speed before retransmitting the reportedly lost segments. On the other hand, an XSTP-ON session can detect the blackout period much quicker by detecting a break in the POLL/STAT interleaf. This ability allows the session to only dump a fraction (1/POLLS_PER_RTT) of its current send window rather than the whole window, which translates into gains in both effective throughput and energy efficiency. As soon as the session detects the loss, it activates the probing mechanism. In a LEO satellite environment, the RTT can moderately fluctuate with the mobility of the satellite, relative to the terminal user. The RTT can also increase due to a congestion condition starting to build up. In these cases, XSTP-ON can prematurely go into probing mode, as a result of the early timeout feature. However, as soon as the presumably lost STAT segment arrives containing an empty gap report, the probing mode is immediately terminated with virtually little if any loss in throughput.

C. Limited Connectivity Tests

In the limited connectivity tests, connectivity (rendezvous) is available during limited periods of time. Tests are performed with a rendezvous of 5 s

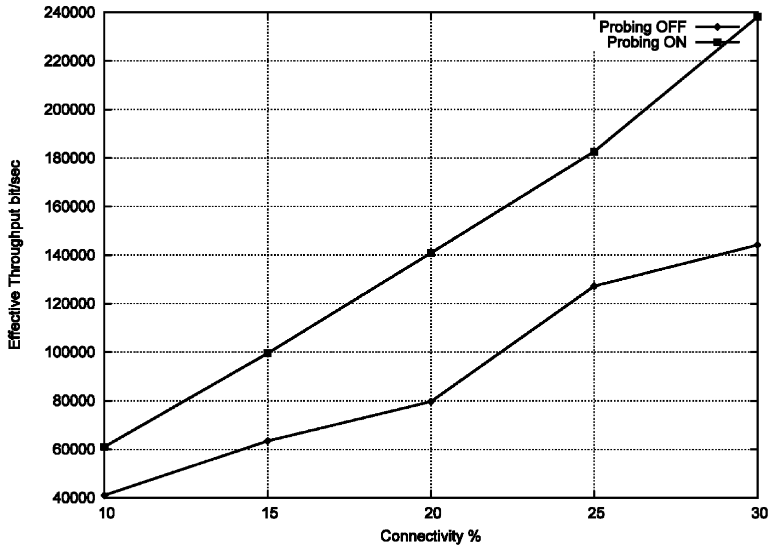


Fig. 7. Effective throughput under limited connectivity with 5 s rendezvous.

and with various connectivity rates ranging from 1 to 30%, as shown in Fig. 7. Reference [19] also presents tests with a connectivity rendezvous of other durations.

In the link error state, packets are dropped in both directions with a 100% probability. In the no error state, packets are forwarded after being subjected to one of the two configured delays. This setup exemplifies extended physical obstruction of satellite signals or unavailability of an in range satellite due to coverage limitations. This condition can occur due to the relative mobility of satellites and mobility of user terminals. When the connection is restored, the RTT can be similar to the one before the interruption.

An XSTP-OFF sender does not detect the loss of connectivity until it is over and either a STAT or a USTAT segment brings back a gap report. This inability to sense the loss in connectivity makes the sender susceptible to drain some or all of the data segments from its send window. While connectivity is lost, the session maintains polling to sense return of connectivity. However, due to the extended connectivity loss periods, the polling gets extensively backed off to a level that compromises the ability of the sender to detect the mainly short connectivity windows. In this case, the session either misses or detects late one or more connectivity windows. As soon as a sender detects the connectivity loss, it slows down its transmission before retransmitting the reportedly missing segments. On the other hand, the XSTP-ON sender can detect the connectivity loss period much quicker as a break in the POLL/STAT interleaf. The sender dumps a fraction (1/POLLS_PER_RTT) of its current send window rather than the whole window, which translates into gains in both throughput and energy efficiency. As soon as a sender detects the loss, it

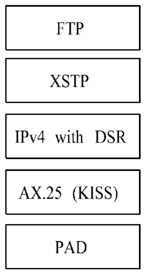


Fig. 8. Protocol stack for packet radio.

activates the probing. Connectivity windows are quickly detected within one RTT. XSTP-ON achieves consistently higher throughput than XSTP-OFF (from 13 to 77% gain).

VII. PERFORMANCE OVER PACKET RADIO

Packet radio refers to breaking up large blocks of data into small units called packets and sending them using radio signals. Characteristics of packet radio are multi hop networks, wide range, slow speed, short frames, and high latency.

In this section, we explore the performance of a protocol stack integrating the following protocols: a file transfer protocol, the XSTP, IPv4 with dynamic source routing (DSR), AX.25, and a packet assembler dissembler (PAD) protocol depicted in Fig. 8. AX.25 [21] and PAD provide the packet radio services. Experiments were conducted at data rates of 1.2 Kbit/ss and 9.6 Kbit/s, frame size 255 bytes, using the carrier sense multiple access (CSMA) medium access control protocol. We first briefly review communications architectures related to the one investigated in this work. Then we present and discuss our performance results. More details about this aspect of our work can also be found in [22].

A. Packet Communications in Space

In space communications, packet radio is used for CubeSat at data rates ranging from 1.2 Kbit/s to 9.6 Kbit/s [23]. PACSATs are also users of packet radio [24]. A PACSAT is a LEO satellite that carries on-board memory for the purpose of data storage and retrieval by ground stations. Its protocol suite is composed of two APPs, namely, a file transfer level 0 (FTL0) protocol and a PACSAT broadcast protocol (PBP) that both operate above the AX.25 protocol. CHIPSat [25], a satellite launched on January 2003, uses TCP/IP and FTP for end-to-end satellite operation. The idea of end-to-end operation on spacecraft over TCP/IP has been demonstrated by the UoSat-12, however, CHIPSat is the first attempt to implement the concept as the primary means of satellite communications. The SCPS [1] provide a suite of protocols to communicate with space vehicles. The stack includes a file protocol (FP), a transport protocol (TP), addressing issues in space, a security protocol (SP), ensuring integrity and security, and a network protocol (NP), both connectionless and connection-oriented. Point-to-point UHF 9600 bit/s data radio was used for the Lander to Rover communications link of the Mars Pathfinder mission. Although it was not packet radio per se, e.g. there were neither encapsulation of data with headers nor addressing or routing, it had some of the transmission characteristics that we find in packet radio such as slow speed.

B. Experimental Environment and Performance Results

For this experiment, two PCs are used to simulate a satellite and a ground station. Each of them is connected to a Kenwood TM-D700A/E FM transceiver, which contains a built-in terminal node controller (implementing framing and the CSMA). Tests were performed in a naturally noisy environment.

Each test involves retrieving a file of certain size from the satellite to the ground station. The average is reported, but results that are dramatically affected by noise are discarded.

We observed that when transferring files of small size (under 1 KB), bandwidth is the primary determinant of the latency, as file transfers at 9.6 Kbit/s are much faster than in 1.2 Kbit/s. The maximum PAD frame size is 255 bytes, an AX.25 header is 17 bytes, an IP header is 60 bytes (including source routing options), an XSTP header is 16 bytes, and an FTP header is 5 bytes, therefore each frame could only carry 157 bytes of FTP data at the most. A 10 KB file need to be fragmented into and transmitted in 66 frames. In this case, the processing time is dominant over transmission time because of the high number of small packets. The throughput results, in bit/s, of transferring files under 10 KB are presented in Table I. It is interesting to find that better throughput is achieved when transferring files of small size, which is the opposite of transferring files in wired communications. This is due to the high processing time of large files as illustrated above. We also express throughput as a fraction of the available bit rate, referred to as normalized throughput. The result is shown as Table II. The average normalized throughput achieved in 1.2 Kbit/s is around 0.24, while as to 9.6 Kbit/s, the normalized throughput is pretty low. AX.25 uses p-persistent CSMA for medium access control (MAC), where *p* is the probability of the transmission when medium is idle. According to analytic models [26], the global throughput can theoretically be pushed very high by increasing the global offered channel traffic. We believe that a normalized CSMA throughput of 0.24 observed locally by a client under real conditions and light load (two nodes only: ground station and satellite) is consistent with theoretical estimations.

TABLE I
Throughput in bit/s

	File Size			
Data Rate	1 B	100 B	1 KB	10 KB
1200 bit/s	334.07	375.21	212.38	220.26
9600 bit/s	758.86	768.16	249.92	248.42

TABLE II
Normalized Throughput

	File Size			
Data Rate	1 B	100 B	1 KB	10 KB
1200 bit/s	0.28	0.31	0.18	0.18
9600 bit/s	0.08	0.08	0.03	0.03

VIII. CONCLUSION

In this paper, a new error control strategy for STP that adapts to the error conditions in the network is presented. The strategy is based on an end-to-end probing mechanism installed at the STP sender and activated when a segment loss is detected. A loss is detected by either an early timeout event or an explicit feedback from an STP receiver. Probing suspends data transmission while waiting until the error situation improves before adjusting the transmission rate. It measures the connection's RTT both before and after probing, to determine if the error condition is congestion or link related. After this comparison, the mechanism retransmits the lost segments and resumes transmission of new data. The new protocol with the probing option is called the XSTP.

In the simulation tests, XSTP-probing consistently achieves higher levels of throughput while maintaining lower levels of overhead. Throughout our experiments conducted over packet radio, we found that the link peak throughput is comparable to the theoretical estimations. We know, however, that the overhead introduced by each protocol layer occupies a big portion of the frame, which impedes the performance. So an option to be considered, to improve the performance, is compression of headers.

REFERENCES

[1] C. C. for Space Data Systems
Space communication protocol specification (SCPS): Rationale, requirements and application notes.
National Aeronautics and Space Administration, CCSDS Secretariat, Program Integration Division (Code OI), Washington, D.C., 20546, Green Book Draft CCSDS 710.0-G-0.4, Aug. 1998.

[2] Henderson, T., and Katz, R.
Satellite transport protocol (STP): An SSCOP-based transport protocol for datagram satellite networks.
In *Proceedings of the 2nd International Workshop on Satellite-Based Information Services* (WOSBIS), Budapest, Hungary, Oct. 1997, 23–34.

[3] Tsaoussidis, V., and Badr, H.
TCP-probing: Towards an error control schema with energy and throughput performance gains.
In *Proceedings of the IEEE 8th International Conference on Network Protocols*, Osaka, Japan, Nov. 2000, 12–21.

[4] Barbeau, M., and Bordeleau, F.
A protocol stack development tool using generative programming.
In *Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering*, Pittsburgh, PA, Oct. 2002, 93–109.

[5] Allman, M., Glover, D., and Sanchez, L.
Enhancing TCP over satellite channels using standard mechanisms.
The Internet Engineering Task Force (IETF), RFC 2488, Jan. 1999.

[6] Allman, M., Dawkins, S., Glover, D., Griner, J., Tran, D., Henderson, T., Heidemann, J., Touch, J., Kruse, H., Ostermann, S., Scott, K., and Semke, J.
Ongoing TCP research related to satellites.
The Internet Engineering Task Force (IETF), RFC 2760, Feb. 2000.

[7] Tsaoussidis, V., and Matta, I.
Open issues on TCP for mobile computing.
The Journal of Wireless Communications and Mobile Computing, **2**, 1 (Feb. 2002), 3–20.

[8] Tsaoussidis, V., Badr, H., Ge, X., and Pentikousis, K.
Energy/throughput tradeoffs of TCP error control strategies.
In *Proceedings of the Fifth IEEE Symposium on Computers and Communications* (ISCC 2000), Antibes, France: July 2000, 106–112.

[9] Biaz, S., and Vaidya, N.
Sender-based heuristics for distinguishing congestion losses from wireless transmission losses.
Technical Report TR98-013, Texas A & M University, June 1998.

[10] Samaraweera, N.
Non-congestion packet loss detection for TCP error recovery using wireless links.
IEE Proceedings of Communications, **146**, 4 (Aug. 1999), 222–230.

[11] Kim, T., Lu, S., and Bharghavan, V.
Improving congestion control performance through loss differentiation.
In *Proceedings of the 8th International Conference on Computer Communications and Networks*, Boston, MA, Oct. 1999, 412–418.

[12] Tsaoussidis, V., Badr, H., and Verma, R.
Wave and wait protocol (WWP): An energy-saving transport protocol for mobile IP-devices.
In *Proceedings of the Seventh Annual International Conference on Network Protocols*, Toronto, Canada, Nov. 1999, 301–310.

[13] Balakrishnan, H., Padmanabhan, V., Seshan, S., and Katz, R.
A comparison of mechanisms for improving TCP performance over wireless links.
ACM SIGCOMM Computer Communication Review, **26**, 4 (Oct. 1996), 256–269.

[14] Holland, G., and Vaidya, N.
Analysis of TCP performance over mobile ad hoc networks.
Wireless Networks, **8**, 2/3 (Mar.–May 2002), 275–288.

[15] Akyildiz, I. F., Morabito, G., and Palazzo, S.
TCP-peach: A new congestion control scheme for satellite IP networks.
IEEE/ACM Transactions on Networking (TON), **9**, 3 (June 2001), 307–321.

[16] Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M., and Wang, R.
TCP Westwood: End-to-end congestion control for wired/wireless networks.
Wireless Networks, **8**, 5 (Sept. 2002), 467–479.

[17] Jamalipour, A., Marchese, M., Cruickshank, H., Neale, J., Verma, S., and Bush, A.
Guest editorial broadband IP networks via satellites—Part I.
IEEE Journal on Selected Areas in Communications, **22**, 2 (Feb. 2004), 213–217.

[18] Jamalipour, A., Marchese, M., Cruickshank, H., Neale, J., Verma, S., and Bush, A.
Guest editorial broadband IP networks via satellites—Part II.
IEEE Journal on Selected Areas in Communications, **22**, 3 (Apr. 2004), 433–437.

[19] Elaasar, M.
XSTP: Extended satellite transport protocol.
Master's thesis, School of Computer Science, Carleton University, Ontario, Canada, Jan. 2003.

[20] Hutchinson, N., and Peterson, L.
The x-kernel: An architecture for implementing network protocols.
IEEE Transactions on Software Engineering, **17**, 1 (Jan. 1991), 64–76.

[21] Karn, P., Price, H., and Diersing, R.
Packet radio in the amateur service.
IEEE Journal on Selected Areas in Communications, **3**, 3 (May 1985), 431–439.

[22] Li, Z.
Performance of generative programming protocol implementation.
Master's thesis, School of Computer Science, Carleton University, Ontario, Canada, May 2003.

[23] Heidt, H., Puig-Sauri, J., Moore, A., Nakasuka, S., and Twiggs, R.
CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation.
In *Proceedings of the 14th Annual AIAA/USU Conference on Small Satellites*, SSC00-V-5, Logan, UT, Aug. 2000.

[24] Price, H., and Ward, J.
PACSat protocol suite—An overview.
In *Proceedings of the 9th Computer Networking Conference*, London, Canada, Aug. 1990, 232–238.

[25] Rubio, K., Janicik, J., and Szielenski, J.
CHIPSat's TCP/IP mission operations architecture—Elegantly simple.
In *Proceedings of the 16th Annual AIAA/USU Conference on Small Satellites*, SSC02-IV-4, Logan, UT, Aug. 2002.

[26] Kleinrock, L., and Tobagi, F. A.
Packet switching in radio channels: Part I—Carrier sense multiple-access modes and their throughput-delay characteristics.
IEEE Transactions on Communications, **23**, 12 (Dec. 1975), 1400–1416.

Maged E. Elaasar received his B.Sc. in computer science from The American University in Cairo (1996) and his M.C.S. from Carleton University, Ontario, Canada (2003).

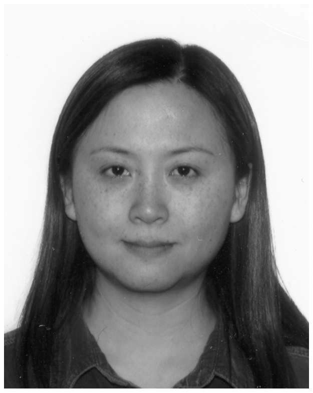
He is a senior software developer at IBM Rational Software. His research interests include mobile and wireless networks, satellite communications and software engineering.

Michel Barbeau has a B.Sc. from Université de Sherbrooke (1985), an M.Sc. from Université de Montréal (1987) and a Ph.D. from Universite de Montréal (1991), all in computer science.

He is a professor at the School of Computer Science at Carleton University, Ontario, Canada.

Evangelos Kranakis has a B.Sc. from the University of Athens (1973), and a Ph.D. from the University of Minnesota, Minneapolis (1980), all in mathematics.

He is a professor at the School of Computer Science at Carleton University, Ontario, Canada. His current research interests include data and communication networks, distributed computing and network security.



Zheyin Li received her B.Economics in international trade and in computer applications from Shandong Institute of Economics in 1998. She received her M.Sc. in information and system science from the School of Computer Science, Carleton University, Ontario, Canada, in 2003.

Currently she is working at Amazon.com as a software development engineer, focusing on multiservice switch crash analysis. Her areas of interest include mobile and wireless communications and performance evaluation of network protocols.