The eXtended Satellite Transport Protocol: Its Design and Evaluation *

Maged E. Elaasar, Zheyin Li, Michel Barbeau and Evangelos Kranakis
School of Computer Science, Carleton University,
1125 Colonel By Drive,
Ottawa (Ontario), Canada K1S 5B6

Abstract

Being both wireless and mobile, 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.

1 Introduction

This paper focuses on enhancing the transport behavior 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 and intermittent connectivity, they also have enough distinct properties to be taken as different environments for data transport. As surveyed by SCPS [SCP98], these properties include highly variable round trip times, disproportional up and down link capacities, limited computing resources (power, memory and speed), and relatively higher throughput.

*The authors graciously acknowledge the financial support received from the following organizations: Natural Sciences and Engineering Research Council of Canada (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS).

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. The mechanism works by suspending data transmission when an error is detected and investing some time and transmission effort to sense the current delays in the network. The mechanism associates error conditions accompanied by noticeable delays with network congestion. Otherwise, it associates these errors with different link error events. The strategy proposed here is based on an earlier one proposed by Tsaoussidis and Badr in the context of TCP called TCP–probing [TB00]. 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, this paper 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) [BB02]. The main component of the simulation is the *eXtended Satellite Transport Protocol* (XSTP). XSTP is the STP protocol extended with our probing mechanism, called XSTP–probing.

The rest of the paper is organized as follows. Sec. 2 reviews the satellite link properties and literature about the available transport protocols and some proposed extensions for improving them in satellite access networks. Sec. 3 describes the design of the new XSTP protocol. Sec. 4 details the new XSTP–probing mechanism. Sec. 5 explains the simulation framework. Sec. 6 presents and

reflects on the simulation results. Integration testing and performance over packet radio are discussed in Sec. 7. Sec. 8 concludes with a summary.

2 Transport over LEO satellites

2.1 Link properties

Since a satellite link is a special kind of wireless link, it naturally inherits all of a wireless link's characteristics. Allman et al. [AGS99, ADG00] and the researchers at SCPS [SCP98] describe satellite links as having a natural broadcast capability and an inherent ability to reach mobile users, which make satellite links ideal substitutes for terrestrial links. Satellite links also cover large areas resulting in reduced switching and forwarding overhead. However, being radio–based, satellite links can have limited bandwidth due to the natural limitation of the radio spectrum and due to certain international agreements that control the allocation of this resource. Also affected are transport protocols as satellite links have limited ability to trade bandwidth to solve other design problems.

Satellite networks are also 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 bit–error rates *(BER)*, 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 under–utilizing 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 [SCP98]. A handoff occurs when one LEO satellite goes out of range and another one goes in range of some satellite access device. 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. [AGS99], 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.

2.2 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 one of the crucial protocols in any communication stack. While some transport protocols provide unreliable service, this paper explores those 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 [TM01], 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 bi–directional 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 [SCP98], 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 [TM01] outlined major features that transport protocol designers have to work on: correct detection of the nature of error (duration, 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 that transport protocols for mobile ad hoc networks need 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.

2.3 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 of such a protocol [TB00, TBGP00, BV98, Sam99, KLB99, TBV00, BPSK96, HV99]. 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 [TM01], link errors are usually characterized by both their duration 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. Even transport protocols with better error control strategies can 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.

2.4 State of the art

The *Transmission Control Protocol* (TCP) has become the defacto standard for the Internet today. More than twenty years of research have produced a protocol that is perfectly optimized 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. Other research efforts went on to verify the achieved results. The availability of current test beds and the strong user base are definite advantages to this approach. However, the burden of staying compatible with legacy systems is a major obstacle confining the improvement of TCP.

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 [HK97, Kat99], is the *Satellite Transport Protocol* (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 for any transport protocol targeted for heterogeneous networks. This missing feature is a discriminating error control mechanism. Unfortunately, STP inherits the as-

sumption that any error results from network congestion and is therefore mitigated by applying congestion control measures. This inability to classify and properly handle different error types usually compromises both the protocol's throughput and expended energy.

3 XSTP: Extended Satellite Transport Protocol

XSTP is an extension of the STP protocol. STP is used to host a new error control strategy, called XSTP–probing, which is introduced in Section 4. This section explains the general behavior of the XSTP protocol, which is similar to STP's general behavior. Differences are indicated.

The XSTP protocol can typically be deployed on top of a network protocol such as the *Internet Protocol* (IP). XSTP provides a connection-oriented reliable byte streaming service to application level protocols such as the *File Transfer Protocol* (FTP).

An XSTP connection consists of two sessions created on the client and server sides. While a client session is created in the CLOSED state, a server session is created in the LISTEN state. After being created, a client session starts a two-way handshaking cycle that involves sending a BGN segment (connection request) to the server. It is only after a BGNAK segment is received by the client that the connection handshaking is completed successfully. The two sessions then move to the EST state. This disconnection process is accomplished by starting a handshaking cycle that involves sending an END segment (a disconnection request). It is only after an ENDAK segment is received by the initiating session that the connection is labelled *half closed*. For the connection to be *fully closed*, the other session has to also complete a similar disconnection handshaking cycle successfully. After that is accomplished, both sessions go back to the CLOSED state. The disconnection handshaking is only attempted when the session is neither sending nor receiving. In that state, a session typically maintains a *keep–alive* timer to periodically check (by sending a POLL segment) whether its peer session is still alive. A session maintains state variables, such as the session's smoothed round trip time and variance estimates.

When an incoming segment is received, XSTP starts by checking the segment's checksum. If the checksum test fails, the segment is discarded; otherwise the protocol continues with a syntax validation check. Once validated, the segment's basic information and extended options are extracted from the header before inspecting the segment's type. If it is an RST segment, the receiver drops the session; otherwise it processes the segment based on its simple or composite type. XSTP allows segments to combine certain types together to save network bandwidth by reducing the number of transmitted segments. However, these composite types are processed in a certain logical order that preserves the semantics of the protocol.

For each segment type, the XSTP has a handler. Each handler is passed the information from the segment that it needs. A handler performs at least one or more of the following actions: trigger a state transition, order a certain response segment and ask for a certain post–action. While state transitions are executed right away by each handler, responses and post actions are collected first from all handlers then checked for consistency before being processed. The currently supported actions include one to keep-alive and another to notify the upper protocol of the peer's closure.

Other major responsibilities of XSTP are received data assembly and presentation to upper protocols. When a data segment arrives, it is either ignored (if invalid), presented or cached in the receive cache. The cache is a sorted list of out–of–sequence segments. If the cached segment is also the next expected segment, then XSTP checks whether it fills the first gap in the cache. If it does, then a new data message is created to hold all the presentable data from the segment. The message is passed to the upper protocol. If the segment is not the next expected one, then the receiver just copies it to the cache in the appropriate order.

XSTP has the responsibility to send USTAT notifications to the peer session when new segment gaps are discovered. A reported segment gap is described by its lower and upper sequence numbers. However, in a network where packet reordering is a possibility, XSTP can be configured to delay this notification up to a number of missing segments. If meanwhile the receiver discovers a new gap, then it reports the first gap at once and starts tracking the new one. Unlike STP, XSTP does not require a gap to consist solely of absent segments when it gets reported. Instead, XSTP allows reporting gaps that are partially filled. This process ensures that whatever is missing from the gap is promptly reported.

XSTP sends POLL segments periodically. These segments are sent to trigger peers to transmit back STAT segments containing acknowledgments for the received segments. The polling rate is a configuration parameter. The higher the rate, the faster the segments get acknowledged, but also the more acknowledgment overhead is incurred. To minimize that overhead, POLL segments are allowed to be piggybacked with data segment if they both happen to be scheduled around the same time for transmission. To implement this feature, both have to be scheduled by the same send timer. With each timer tick, XSTP examines if a transmission of a data segment, a POLL segment or both is due. After that process, XSTP reschedules the next transmission (if any) before restart-

ing the send timer.

A segment timestamp is used to make sure that only new POLL segments are handled. When a POLL segment is received, XSTP replies with a STAT segment. This segment contains a gap list constructed by traversing the cache. A gap is defined as two consecutive cached segments having non–consecutive sequence numbers. For arriving STAT and USTAT segments, XSTP extracts and uses peer's status information to adjust its flow control strategy.

XSTP implements the following flow control strategy. Messages pushed down from the application are not transmitted right away. The message's data is copied to a send buffer. If there is not enough space for the data in the buffer, then processing of the message is suspended. It is resumed when buffer space is available.

At any point in time, XSTP is either 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 is subject to a strategy that controls the number of segments that are allowed to be transmitted at any point in time. 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. Whenever the transmission state is entered, the sender updates the total number of allowed data segments, the timer sending interval and the burst size (the number of segments to be sent back-to-back each time). However, if the number of allowed segments is below a certain configurable threshold, then the sender transmits all of them in one burst. If the sender determines that no new transmissions are currently possible, then it immediately stops the send timer.

An XSTP sender transmits a data burst by creating data segments and pushing them back to back. All segments in a burst except the last one have the maximum allowed segment size. If a POLL is scheduled at the same time as a data burst, then the POLL is piggybacked with the last segment of the burst. The sender can also force a POLL to be piggybacked with a data segment if the sender has transmitted a certain configurable ratio of the send buffer since the last POLL. This process is done to expedite receiving acknowledgments and therefore rapidly freeing the send buffer. Since the send buffer is not segmented, the sender maintains a separate queue of unacknowledged segment entries called the *send cache*. Before transmitting any segment, the sender creates an entry for this segment in the send cache. The entry contains the segment's sequence number, data size, number of retransmissions and last retransmission timestamp. This information is used in the retransmission process.

XSTP might not be able to transmit if a slow peer advertises a zero window to prevent a fast sender from overwhelming with data segments. In this case, the sender exits the transmission state and goes into the persistence state. In that state, the sender suspends data transmission and periodically schedules POLL segments to request window updates from the peer. Only after receiving a non–zero window size does the sender go back to the transmission state. When the sender is neither transmitting nor persisting, it is in the idle state. Only when the sender is in this state can a session be safely closed. However, a session can be forced to go into the idle state. This process is usually done as a session is dropped.

Another responsibility of an XSTP sender is to manage the session's flow control strategy. This strategy is affected by feedback received from the session's peer. This feedback can either be in the form of a STAT or a USTAT segment, which contain both cumulative and selective negative acknowledgments. The cumulative acknowledgment informs the sender about the receiver's next expected in–sequence segment. This allows the sender to advance the left edge of its transmission window and remove entries from the sending cache up to, but not including that segment's entry. The sender also frees data from the send buffer that belongs to those acknowledged segments and releases any pending threads that are waiting for the send buffer to free. After that process, the sender opens its congestion window either exponentially or linearly depending on whether the sender is in the slow start or in the congestion avoidance phase.

Here is how STP originally handles selective negative acknowledgments; the way it is done by XSTP is explained in Sec. 4. A selective negative acknowledgment is a gap report. The sender examines every reportedly missing segment and determines if these segments qualify for retransmission. The sender distinguishes gaps reported by STAT segments from those reported by USTAT segments. While missing segments reported by a USTAT can only be retransmitted once, those reported by a STAT can be retransmitted only if an RTT (subject to backoff) has elapsed since they were last transmitted. To retransmit a segment, STP uses the send cache. After examining all gaps and retransmitting the required segments, STP reduces its congestion window by half. Further retransmissions in the same window does not result in further reducing of the congestion window, since these retransmissions are probably caused by the same loss event. After closing the window for the first time, the sender exits the slow start phase and goes into the congestion avoidance phase. Once there, the sender never goes back to slow start again unless it has been idle for some time (a configuration parameter).

4 XSTP–Probing mechanism

4.1 Motivation

In a typical heterogenous network, these exist many kinds of error conditions that widely vary in their nature. One of the classic problems of transport protocols over heterogenous networks is their inability to detect and effectively react to such different error conditions. An assumption made by protocols, such as TCP, is that network congestion is the cause of all perceived errors. However, in a typical LEO satellite access network there are other types of error conditions including bit corruption, handoff and limited connectivity. These non–congestion link–related errors are systemically perceived as congestion–related by unaware transport protocols. Unfortunately, engaging congestion control in these cases hampers throughput and leaves the session too conservative even for moderate levels of link errors. Also, according to the used congestion control tactics, there can be further increase in the level of segment retransmission, which usually translates into more increases in energy expenditure.

Unfortunately, the STP protocol [Kat99], [HK97] inherits this congestion control bias from its ancestor protocols. Although the protocol can efficiently recover from multiple losses in the same round trip, its error recovery tactics can negatively affect its overall performance. Specifically, slowing down in response to link–related errors can hamper the effective throughput. In this section, a new error control strategy is proposed to stretch a protocol's ability to adapt to the different error kinds found in LEO satellite access networks. The new strategy is integrated in XSTP.

The new strategy consists of a probing mechanism 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 [TB00]. XSTP–probing also adapts to the level of error in the network by suspending new data transmission and by striving to send only in windows of error–free connection. The probing mechanism is modelled after an earlier one, proposed by Tsaoussidis and Badr in the context of TCP called TCP–probing [TB00]. The remainder of this section explains XSTP–probing mainly by comparing it to TCP–probing.

4.2 Description

The goal of any error control strategy is to adapt the sender's transmission rate to the varying error conditions in the network. This goal is usually accomplished by taking an aggressive stance when the error is found to be transient and a conservative one when it is more persistent. The XSTP–probing mechanism is no exception. In fact, this mechanism goes even further by probing the connection for possible error free conditions and only transmitting in those windows. The mechanism accomplishes that task by suspending new data transmission upon detecting a loss and initiating a probing cycle to collect RTT statistics on the connection. The mechanism then compares these RTT statistics to the RTT estimate available when the loss was discovered. Interestingly, the duration of that probing cycle is proportional to the level of error in the network which helps the connection sit out the error conditions. After the cycle is finished and if congestion is detected by proliferating RTTs, congestion control is immediately invoked. Otherwise, transmission levels are restored without taking any action.

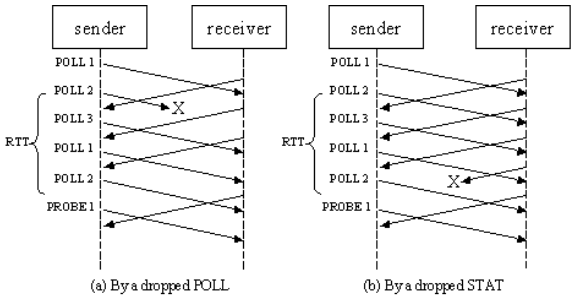

Figure 1. Triggering of XSTP–probing mechanism by an *early timeout* event.

Unlike the TCP–probing mechanism, which introduces changes to both the sending and receiving ends of a connection, XSTP–probing is a sender–only mechanism. This simplifies the implementation. Also, while TCP–probing introduces several new segment types and their associated states, XSTP–probing leverages the unique semantics of XSTP and does not introduce any new segment types. XSTP–probing reuses the polling cycle of XSTP, which is the protocol's low frequency acknowledgement mechanism, as its probing cycle. Specifically, the POLL segment is the probe and the STAT segment is the probe acknowledgment. Fortunately, a XSTP receiver is kept unaware whether the received POLL is a probe or just a normal POLL. Also unlike TCP–probing, XSTP–probing does not introduce several states to track the progress of a probing cycle. That fact makes XSTP–probing more scalable in terms of its ability to configure the probing cycle with different numbers of probing exchanges.

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

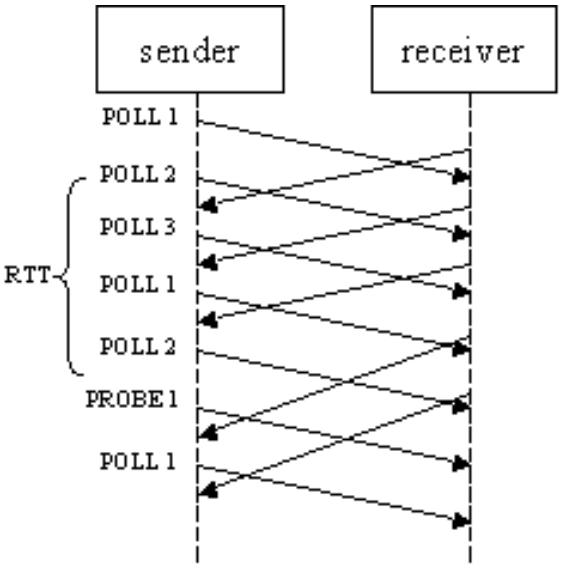

Figure 2. Triggering of XSTP–probing mechanism by a false *early timeout* event.

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 STAT. If either a POLL or a STAT is dropped in the connection, then the session detects a break in the interleaf pattern within a maximum of one RTT and a minimum of $RTT/POLLS\_PER\_RTT$, as shown in Fig. 1, where $POLLS\_PER\_RTT$ is three. This depiction is in contrast to TCP–probing, where a timeout usually spans multiple RTTs.

The explicit loss detection method relies on receiving feedback from the receiving session in the form of either a STAT or a USTAT segment. However, it is only after validating the gap reports in these segments and finding at least one segment worthy of retransmission that XSTP–probing is triggered. These conditions avoid premature triggering due to an invalid 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.

As XSTP–probing starts up, the session goes into probing mode. This mode supersedes the current state in the sender. Again, there is a contrast to TCP–probing going into the newly introduced "PR_X_SENT" states. After that event, the mechanism suspends new data transmission and records the current timestamp and RTT estimate ($loss.TS$ and $loss.RTT$).

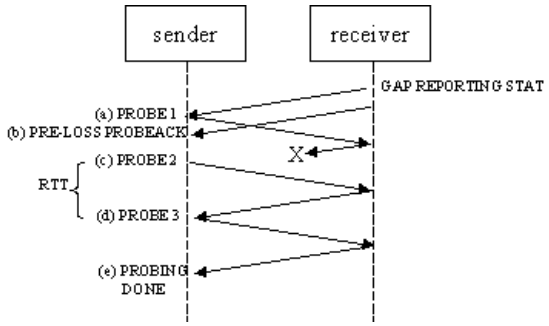

Figure 3. Phases of a probing cycle as it happens in the network.

In the case of TCP–probing, a complete cycle is labelled with a version number that is stored and reflected by all segments (two probes and two probe acknowledgments) participating in the cycle. Only after the first probe exchange is successfully completed, does the other exchange start. If any segment in that cycle gets dropped, then the whole cycle is abandoned and a new one is initiated with a new version number. The loss of a segment is detected by means of a timeout (set to the currently estimated RTT). On the other hand, XSTP–probing does not use a timeout. Rather, XSTP sends a probe segment every RTT irrespective of whether a probe acknowledgment is received or not. The advantage becomes clear when the RTT gets a little extended (a common phenomenon in LEO satellite links where the RTT experiences moderate variations). In this case, the previous exchange is not ignored but is rather given more time (up to one more RTT) to complete. If after that time it is still not completed, then a new exchange begins and the old exchange becomes obsolete (since it is followed by two new probe exchanges). This situation is depicted in Fig. 2, where PROBE1 gets transmitted when no STAT arrived since the last POLL because an extension of the RTT.

XSTP–probing defines a map between the sent probe (POLL) timestamps and their corresponding acknowledgment's (STAT) RTT measurement. Whenever a probe is sent, its timestamp is entered in the map. Also, whenever a valid probe acknowledgment is received, its RTT measurement is associated the corresponding timestamp in the map. The probing cycle does not complete until two consecutive entries in the map get filled with RTT measurements. The probing map has a constant size that is set as a configuration parameter. The map also has a policy of deleting the oldest timestamps to make room for new entries. The map size should be set in proportion to the expected error levels in the network. Fig. 3 illustrates different phases of a typical probing cycle as they

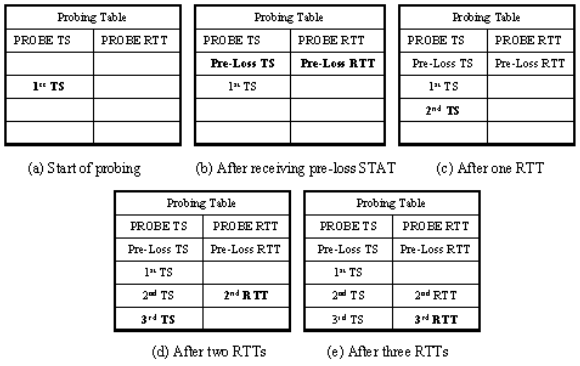

Figure 4. Phases of a probing cycle as reflected in the probing map.

occur in the network. Fig. 4 shows the corresponding stages of the cycle as reflected in the probing map.

There are three more salient points. The first point is related to an arriving probe acknowledgement that brings back an empty gap report. This event is taken to indicate that probing is started prematurely due to packet reordering or a false *early timeout* event. In this case, XSTP–probing immediately terminates and restores new data transmission at the previous level. The second point relates to an arriving probe acknowledgment that does not correspond to existing probes in the map. This scenario appears in one of two situations: either the acknowledgment took so long to arrive that its probe entry got deleted from the map or it corresponds to a POLL segment that had been sent before probing was started (timestamps are smaller than $loss.TS$). XSTP–probing ignores the acknowledgment in the former case, it records it in the map in the latter. During the first RTT of probing, an entry in the map is reserved for this kind of acknowledgments. The rationale to consider this acknowledgment is that being close enough in time to the error (sent during the same RTT containing the error), it provides invaluable information about the network condition at that time. Since XSTP normally sends multiple POLL segments per RTT, several pre-loss acknowledgments can be received. XSTP–probing overrides the reserved entry in the map with every arriving acknowledgment. Interestingly, this ability to handle pre-loss probes can allow the probing cycle to finish much quicker (in around one RTT) if the first probe acknowledgment also made it on time (since two exchanges are required). The third and final point is related to USTAT segments. Although they are used as explicit triggers to the probing mechanism, the segments are totally ignored while the probing mode is on. Their gap reports, usually repeated by consecutive STAT segments, are processed only after probing is done.

One very important side of the probing mechanism is the decision making criteria at the end of the probing cycle to determine the probable cause of the error. Although many possible heuristics can be used, the one introduced by TCP–probing is adopted by XSTP–probing. This heuristic compares the two probes' RTT measurements to the $loss.RTT$. If both probes are less than or equal to the $loss.RTT$, congestion is not detected and the error is considered link–related. Otherwise, congestion control is applied. However, in some cases, the RTT can moderately vary over the round trips (for example due to LEO satellite mobility). In this case, the varying RTTs can confuse the probing mechanism into thinking that the RTT has extended due to congestion. For this scenario, we define a configurable RTT tolerance parameter (ratio) in the probing algorithm to smooth out the effect of that phenomena on the measurements. This parameter should be set based on the expected RTT variance in the network.

Congestion control measure depends on how long the probing cycle takes to finish. If the cycle is long enough to reach XSTP's threshold for idle transmission, the sender goes back to slow start; otherwise the congestion window is reduced by the congestion avoidance algorithm (the window gets 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 overconservative. After probing is done, the missing segments reported by the last probe acknowledgement (STAT) are retransmitted and the normal polling rate is restored.

4.3 Design

The XSTP–probing mechanism has three configuration parameters: the maximum number of trackable probe exchanges MAX_PROBES, number of requested probe exchanges REQ_PROBES, and RTT tolerance ratio $RTT\_TOLERANCE$. The probing object's local state contains a loss descriptor and a probing map. The loss descriptor consists of a loss timestamp ($loss.TS$) and a loss round trip time ($loss.RTT$). The probing map, whose size is set to MAX_PROBES, contains timestamps of probes and their corresponding RTT measurement. The entries are sequentially ordered by timestamp value. The probing object also keeps an activation flag, which is initially set to false.

A sender starts the probing mechanism with the current timestamp and RTT estimate, which are used to initialize its loss descriptor before turning its activation flag on. When XSTP probing is activated, the sender enters the probing mode, which means suspending new data

transmission by closing the data sending interval, changing the polling rate to one POLL per RTT and disabling further RTT backoff. The sender then initiates a probing cycle, in which it sends a probe each round trip. For each probe, the sender records the probe's timestamp. This method puts the timestamps in the next entry in the probing map. When all entries get filled, the probing object starts overriding older entries. It is important to mention that the first probe's timestamp goes into the second not the first entry in the map. The first entry is initially reserved for potential pre–loss probes.

Whenever a probe acknowledgment (STAT) arrives, the sender updates its congestion window by examining the segment's cumulative acknowledgment. After that, the sender examines the segment's gap report. If the report is empty, then the sender immediately aborts the probing mechanism and resets the measurement map and loss descriptor. However, if some segments are reported missing, then the normal gap validation is performed. If at least one segment is determined worthy of retransmission, then the STAT is considered a valid probe acknowledgment. In this case, the sender measures its RTT and records the acknowledgment's timestamp and RTT values.

The acknowledgment's timestamp is compared to the one of the loss descriptor. If the timestamp is smaller, the acknowledgment is considered a pre–loss probe acknowledgment and both its timestamp and RTT values are stored in the first entry in the map. If the received acknowledgment is not the first pre–loss acknowledgement, the method overrides the older data in the first entry (usually pre–loss acknowledgements stop arriving after the first post–loss probe acknowledgment is received). However, if the acknowledgment's timestamp turns out to be greater than the one in the loss descriptor, then the sender looks up the acknowledgment's corresponding timestamp entry in the map and records its RTT in the appropriate corresponding slot.

Finally, the sender checks if there are REQ_PROBES consecutive filled entries in the map. If there are, then the probing cycle ends. The sender calls the congestion detection heuristic. The sender first applies the tolerance ratio $RTT\_TOLERANCE$ to the RTT measurements (only those consecutive REQ_PROBES measurements that ended the probing mode). The sender then compares the RTT measurements to the $loss.RTT$. If any one of the measurements is greater than the $loss.RTT$, then congestion is signaled. The sender then terminates the probing mechanism. If congestion is detected, the sender closes its congestion window according to the congestion avoidance algorithm. The sender then concludes by retransmitting the missing segments, restoring the polling rate and resuming new data transmission.

5 Simulation environment

Using simulation, XSTP–probing is tested in various error conditions and performance is quantified. This section describes the simulation environment, performance metrics and test cases.

For simulation purposes, an implementation of the XSTP protocol has been realized in the PIX framework [BB02] over a recent distribution of Linux using C++. 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. 5. The *application protocol* (APP) is a bulk data protocol streaming large files across the network at a configurable speed. Bulk data transfer is the best traffic pattern to study the effect of errors on the efficiency of transport protocols. The long and continuous flow of data allows to demonstrate the full capabilities of its error control strategy. Configuration parameters for this protocol include the file size, the chunk size and the streaming rate.

The link protocol is called the *Queue Link Protocol* (QLP) and it is based on the POSIX message queues. It represents nodes of a simulated network on a single physical machine. Each session of that protocol is configured with a message queue identifier, representing the node's link interface address, and a static forwarding table mapping of destinations to corresponding message queue identifiers. Using this protocol, it is possible to simulate networks of various topologies.

The protocol encompassing an error generating model is called the *eXtended Delay and Drop protocol* (XDELDROP). It is based on a similar protocol of x–Kernel [HP91] called VDELDROP. It models the delays and drops of packets in a heterogeneous network (like a LEO satellite access network). Each session of that protocol is modelled as a continuous–time Markov chain with two states. Each state has three configurable 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 in the network. When a state is entered, the duration of the stay is exponentially distributed with mean t_i During that time, the session receives incoming packets and either drops them with a probability r_i, or forwards them after applying the minimum delay ${d_i}^{min}$ or the 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 an uncongested network. The maximum delay aims to correspond to some level of network congestion delay. XDELDROP is configured above the link layer in intermediate nodes (router nodes).

A LEO satellite access network is simulated (see

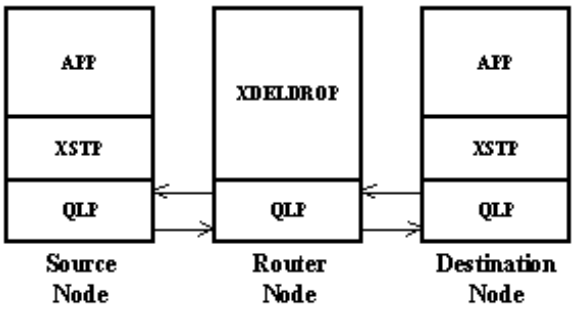

Figure 5. The simulation configuration.

Fig. 5). There are three nodes: a source, a router, and a destination. In the source and destination, a communication suite consisting (from top to bottom) of APP, XSTP and QLP is installed. In the router, a suite consisting XDELDROP directly over QLP is installed. Each node is run in a separate process on the same physical machine.

The application in the source node is configured to be a data source, streaming a large (10,000,000-byte) file in chunks of 1000 bytes each, which is equivalent to the XSTP session's maximum segment size (MSS). This configuration is chosen to avoid any buffering delays and to neutralize some XSTP algorithms such as the Nagle's algorithm, which delays sending small segments for some time until previous acknowledgements are received. A single connection from the source to the destination is run in each test. XDELDROP in the router models the effect of a complex network. Different error conditions are simulated including a bit corruption condition, a handoff condition and a limited connectivity condition. Every one of these conditions is coupled with a congestion condition.

XDELDROP is configured to have two distinct states: an error–free state (called the good state); and an error–prone state (called the bad state). A parameter fixed for both states throughout the tests is the packet forwarding delay range, which is set to a minimum of 100 msec and a maximum of a 150 msec. The minimum represents a hypothetical normal one–way delay when there is no congestion, while the maximum represents the delay under a congestion. The values are chosen to distinguish congestion from non–congestion.

XDELDROP is in either one of four states: no error, moderate congestion, link error, and severe congestion. The no error state model forwarding all packets (good state) after applying the minimum forwarding delay. The link error state models dropping packets (bad state) while applying the minimum delay to the forwarded ones if any. Similarly, the moderate congestion state models forwarding all packets (good state) after applying the maximum forwarding delay. The severe congestion state models dropping packets (bad state) and applying the maximum delays to forwarded ones if any.

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 trackable probes is set to four. The number of consecutive RTT measurements sufficient to finish the probing cycle is set to two measurements.

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

$$effect.through. = orig.size/conn.time$$

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

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

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

$$through./over. = effect.through./tot.trans.over.$$

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

Tests are conducted to represent three different satellite link error conditions: bit corruption, handoff and limited connectivity. For each category, XDELDROP is configured to produce phases of a category of link error in addition to phases of congestion. This configuration permits the analysis of the mechanism's effect on performance for each type of error independently with congestion. For each test, two versions are configured: one with the XSTP mechanism on, and another one with it off. Each test is replicated twenty times (to smoothen out the statistical variations) with only one statistical data sample taken at the end of each test (after all data is sent from the source to the destination).

In the bit corruption tests, the good and bad states have the same mean duration time. While the good state has a drop rate equal to zero, the bad state's drop rate is varied from 0-50%. In the handoff tests, the good state is configured to have a zero drop rate, which makes it

forward all packets in both directions. The bad state is configured to have a 100% drop rate. The duration of the good state is larger than the duration of the bad state in this test. Different levels of handoff rates and duration (rendezvous) are tested by varying the mean duration time of both the good and bad state. To simulate a required handoff rate and rendezvous combination, the bad state's mean duration is set to the handoff rendezvous and the good state duration is calculated as:

$$((rendezvous/rate) * 100) - rendezvous$$

In the limited connectivity tests, the good state is configured to have a zero drop rate The bad state is configured to have a 100% drop rate. The duration of the bad state is larger than the duration of the good state in this test. Different levels of connectivity rate and duration (rendezvous) are tested by varying the mean duration time of both the good and bad state. To simulate the required connectivity rate and rendezvous combination, the good state's mean duration is set to the connectivity rendezvous and the bad state duration is calculated as:

$$((rendezvous/rate) * 100) - rendezvous$$

6 Simulation results

In this Section, the results of the simulation, defined in Sec. 5 are reported and discussed. XSTP with the probing option turned off/on is referred to as XSTP–OFF/ON. It is important to keep in mind that the XSTP protocol is already more tuned to carry out error control than standard TCP. Katz and Henderson [Kat99], [HK97] showed that the protocol can effectively recover from multiple errors (including link errors) in the same round trip. It is also interesting to mention that TCP–probing was only compared against the standard versions of TCP. Therefore, in contrast to TCP–probing, the performance gains achieved by the XSTP–probing mechanism are relative to a more extended protocol, which gives the results further significance.

For each test in the simulation, the total connection time and three kinds of transmission overhead are tracked: the retransmission overhead, forward control overhead and reverse control overhead. While it is generally sufficient to observe the total transmission overhead, in some cases it helps to look at an individual overhead component to gain more insight. The complete measurement data along with their standard deviation and 95% confidence interval are reported in [Mage03]. This data is used to calculate three different metric values: the effective throughput, total transmission overhead, and throughput/overhead ratio. The analysis is outlined first by the error category and second by the affected metric.

6.1 Bit corruption tests

In these tests, packets are dropped in both the good state and bad state with a certain probability. On the other hand, packets surviving the drop are forwarded after being subjected to one of the two configured delays. The same error condition is applied to both directions of the connection at the same time. Usually, when packet drops are accompanied with extensions to the RTT, a congestion condition in the network, resulting from network over–buffering is assumed. Otherwise, the event is only indicative of link bit errors.

In this category of tests, an XSTP–OFF sender usually detects a loss by a STAT or a USTAT segment reporting back a non–empty gap report. The session responds by updating the next unacknowledged segment, which may expand the congestion window. Then, the session examines the gap report and validates each reportedly missing segment. In the case of a USTAT report, the segments should not have been retransmitted before. For a STAT report, on the other hand, enough time (usually one RTT subject to backoff) must have passed since each segment was last transmitted. After examining all the gaps, and 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.

Similarly, when an XSTP–ON session receives a STAT or USTAT gap report, it performs the usual validation on the report. However, should the session determine that one or more segments pass the criteria and hence need retransmission, it does not perform the retransmission but rather activate the probing mechanism. This activation involves suspending the new data transmission and initiating a probing cycle. The cycle does not finish until a specific number (two in this case) of consecutive probe exchanges are completed. If the probing mechanism detects an increase in the RTT of the network compared to the RTT prevailing before the cycle got triggered, the mechanism takes this increase as a sign of congestion. The probing mechanism reacts by applying measures similar to those applied by XSTP-OFF. However, if no increase in the RTT is detected, the mechanism resumes the new data transmission with *Immediate Recovery*. Moreover, if any probe acknowledgment brings back an empty gap report, the probing mechanism

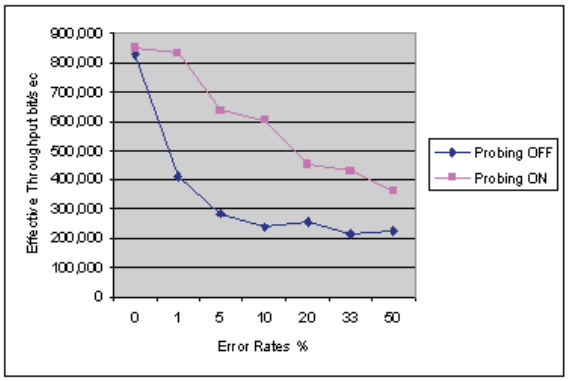

Figure 6. Effective throughput under bit corruption with 10 seconds mean duration

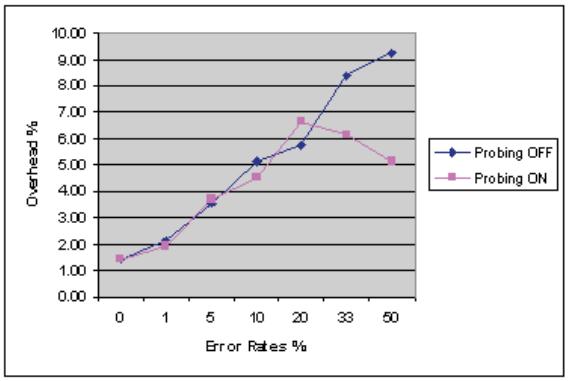

Figure 7. Overhead under bit corruption with 10 seconds mean duration

considers that a sign of premature probing. The mechanism terminates and resumes transmission with *Immediate Recovery*.

In this category of tests, the experiments are performed with a mean duration of 10 seconds, as shown in Fig. 6. Additional tests with a mean duration of one second are presented in Ref. [Mage03].

The tests are repeated with different error intensities ranging from 0 to 50%. The effective throughput is measured in each case. One important observation is that XSTP–ON achieves consistently higher effective throughput than XSTP–OFF (from 3 to 150%). This is due in large to the probing mechanism's ability to proceed with *Immediate Recovery* when link errors are detected, and therefore avoid to unnecessarily reduce the transmission rate. However, the advantage is more apparent in the 10–second tests than in the 1–second tests. The reason relates to the insufficient time in the 1–second tests for the sender to demonstrate its full error control capabilities. Sessions in both cases expand their congestion window size in the error–free phase. However, that size reaches a more dramatic level under a 10–second phase than under a 1–second one. In fact, the size after the 1–second cycle is usually not that different from the initial window size.

Another observation is that the more intense the error condition, the longer it takes the probing cycle to finish, due to the high probability of probe or acknowledgement loss. During that cycle, the session is effectively sitting out the error condition. Also, in case of severe error conditions with deteriorating RTTs that prevent an acknowledgment from coming back for sufficiently long time, the session can eventually go back to slow start after the probing is finished, which further stretches the connection time. This tends to make the session more conservative and therefore lowers throughput. On the other hand, if the error condition is light, the probing cycle finishes much more quickly. If no RTT extensions are detected, the session resumes transmission at the same speed as before probing.

It is also important to realize that the XSTP-OFF session does not reduce the congestion window further in response to more reported losses until the previous window of segments is confirmed received. This sequence has the beneficial effect of avoiding to reduce the congestion window multiple times in response to the same loss event. In addition, XSTP–OFF suspends its data transmission and only maintains a low frequency polling cycle until the window is reopened. This polling cycle becomes much like the XSTP–probing cycle, except with a different frequency that could also be subjected to backoffs in the absence of arriving STATs. Under severe error conditions, the polling cycle can be extended enough to miss windows of error–free transmission in the connection, which increases the overall connection time and hence lowers the achieved effective throughput. On the other hand, XSTP–probing maintains the probing frequency at one probe per RTT. The RTT value can still be updated with each arriving probe acknowledgment to approximate the RTT prevailing in the network. Also, since POLLs (probes) are being sent even before probing started, the chance to quickly collect the requested number of acknowledgments (STATs) improves considerably under moderate error conditions. The aforementioned generally favors effective throughput as transmission is resumed much quicker after appropriately adapting to the error condition.

The total overhead is plotted against the various bit corruption rates in Fig. 7.

The probing mechanism generally reduces the total transmission overhead. As XSTP–probing activates the probing cycle, it suspends new data transmission to pro-

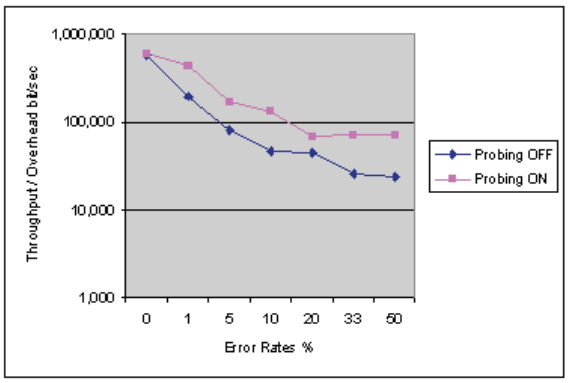

Figure 8. Throughput/overhead ratio under bit corruption with 10 seconds mean duration

tect 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. In addition, the probing mechanism tends to keep the session's *send* window at higher levels due to its *Immediate Recovery* option when link errors are detected. However, this also has the side effect of increasing the number of initial losses incurred by the session at the start of a loss event. At some error levels (like at 5% and 20% when the duration is 10–second), the reduction in overhead achieved by suspending transmission is more than recouped by the overhead incurred as a result of having a large window at the start of an error phase. Another observation is that at some level of increasing error rates (20% in this case), the probing mechanism manages to actually reduce the overhead. This is in large due to ability of the probing mechanism to start its cycle earlier using the *early timeout* feature. The probing cycle also gets extended because of the deteriorating error rates. For these reasons, the session becomes more conservative, which leads to an overall reduction in the retransmission overhead. In most cases, the probing mechanism succeeds in lowering the overhead by a significant ratio (44% at the 50% point in 10–second duration tests).

The Throughput/overhead ratio is a metric to measure the amount of throughput achieved per one percent of overhead. In Fig. 8, it is shown that this metric is inversely proportional to error rate. As error deteriorates, the amount of throughput achievable per 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.

6.2 Handoff tests

In these tests, packets in both directions are dropped in the bad state with a 100% probability, marking a period of "blackout", where connectivity is temporarily lost. In the good 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 an intermediate router over–buffering, a symptom of a serious congestion condition in the network. The duration and frequency of that condition depends on the transport behavior of competing traffic in the network. However, if that period occurs without any RTT extensions, it is indicative of a handoff event. 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. This period can vary in duration (rendezvous) or frequency depending on the speeds of both the satellite and user terminal. In general for tests in this category, the connection is mainly in the good state and only frequently in the bad state.

An XSTP–OFF session 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 non–empty gap report in the following good state. This inability to sense the loss in connectivity makes the session susceptible to drain 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 presumedly lost STAT segment arrives containing an empty gap report, the probing mode is immediately terminated with virtually little if any loss in throughput.

In this category, tests are performed with handoff rendezvous of 0.5 and 1 second, as shown in Figures 9 and 10.

In each case, the tests are repeated with various handoff rates ranging from 1 to 15%. The effective throughput is measured in each case. The first observation is that

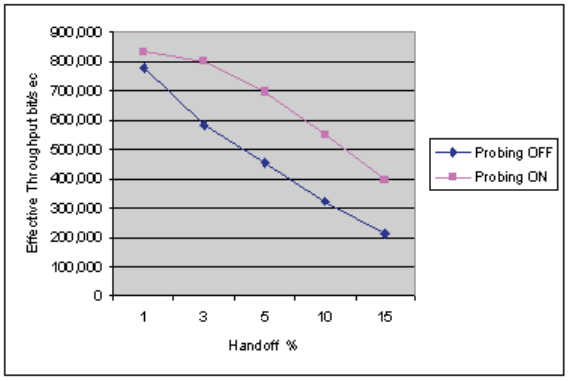

Figure 9. Effective throughput under handoff with 0.5 second rendezvous

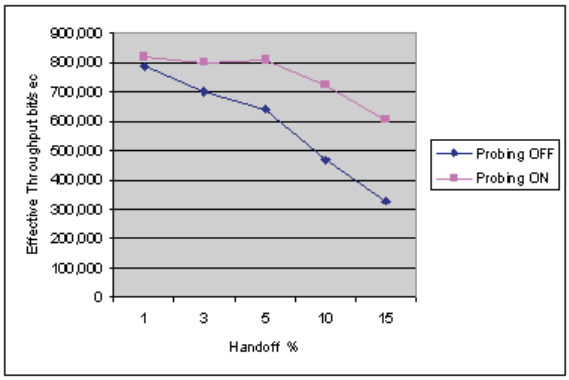

Figure 10. Effective throughput under handoff with 1 second rendezvous

throughput is inversely proportional to the rate (hence the frequency) of handoff. The reason relates to each handoff occurrence costing the connection a portion of the send window worth of retransmissions, which increases the total connection time. Also, the throughput is expected to be inversely proportional to the rendezvous duration, due to the longer suspension of transmission. However, a comparison of the 0.5 and 1 second figures does not confirm this expectation. The reason for that relates to the rendezvous being relatively short when compared to the RTT, while the good phase duration is an order of magnitude larger than the RTT. For example, in the 1–second case, the rendezvous time is twice as much as the 0.5–case, but the good phase durations are also twice as much. The loss in throughput realized by doubling the rendezvous is more than made up for by also doubling the good phase durations. This effect usually makes the connection obtain larger send windows and therefore shorter total connection time.

Another important observation is that XSTP–ON achieves consistently higher effective throughput than XSTP–OFF (from 4 to 86% gain). The main reason for that observation is the ability of XSTP-ON to differentiate congestion from handoff situations. This ability makes the session more conservative in case of deteriorating RTTs, and more aggressive otherwise by keeping a higher congestion window. Another reason for the observation is the ability of the protocol to quickly suspend transmission as soon as it detects a “blackout” period, and therefore avoid many retransmissions that usually extend the connection time. The advantage of the probing mechanism is more apparent in the 1–second tests due to the longer good state duration, which results in a bigger send window when the session enters the handoff state. Therefore, more packets get dropped for the 1–second tests than for the other tests. Also, the number of packet drops saved by the probing mechanism translates directly into handsome gains for the connection effective throughput.

The total overhead is plotted against the various handoff rates in Figures 11 and 12.

The general observation is that the expended overhead is proportional to the handoff rates. This observation is mainly due to the increase in segment loss and hence the retransmission resulting from more frequent handoff events. The overhead is also expected to be proportional to the handoff rendezvous. However, a comparison of the overhead levels in the 0.5 and 1 second cases does not confirm this expectation. In the 1–second case, the rendezvous is doubled but the good state duration is also doubled, with respect to the 0.5–case. The increase of overhead achieved by doubling the rendezvous is more than wiped out by the overhead savings resulting from faster transmission in the doubled good phase duration.

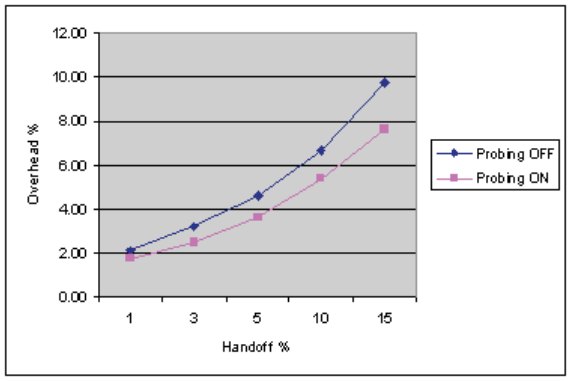

Figure 11. Overhead under handoff with 0.5 second rendezvous

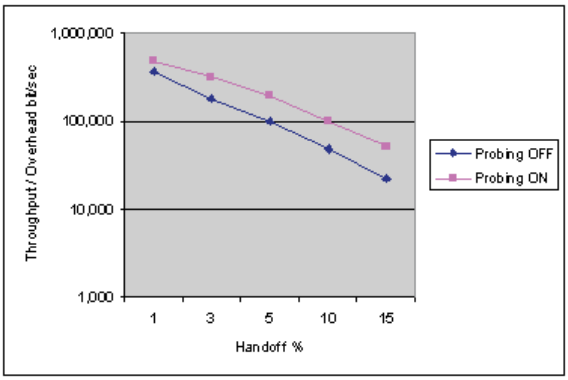

Figure 13. Throughput/overhead ratio under handoff with 0.5 second rendezvous

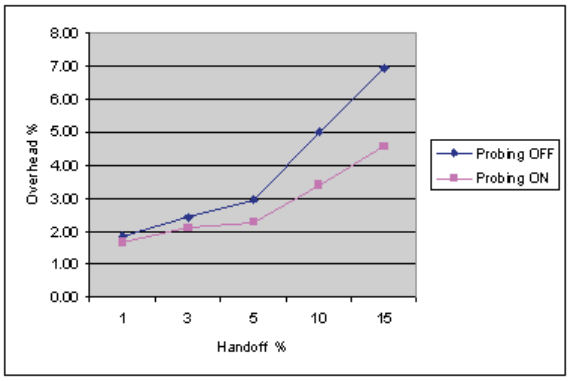

Figure 12. Overhead under handoff with 1 second rendezvous

The XSTP–ON session suspends new data transmission as soon as it activates probing by the early–timeout event. That suspension effectively reduces the number of segments lost from the send window and therefore reduces the retransmission overhead. Also for the control overhead, XSTP–ON produces slightly less POLL and STAT traffic than does XSTP–OFF. This reduction is mainly due to the probing mechanism adjusting the polling (probing) rate to once per RTT while the mechanism is on. On the other hand, XSTP-OFF keeps its regular polling rate. It is interesting to realize that this much smaller polling rate does not noticeably affect overhead since the handoff rendezvous is usually quite short. Generally, the probing mechanism succeeds in lowering the overhead by a significant amount (34% at the 50% point in 1–second rendezvous tests).

In Figures 13 and 14, it is shown that the throughput/overhead ratio is inversely proportional to the handoff rate. Also for the same handoff rate, the efficiency is directly proportional to the handoff rendezvous, mainly because it is more proportional to the good state duration than it is inversely proportional to the rendezvous. The figures also show that using XSTP–probing helps the session achieve more throughput with the same level of overhead expenditure. The efficiency advantage increases from the 0.5 to the 1–second tests, mainly due to the impressive gains in the throughput and the reduction in the overhead achieved in the 1–second tests.

6.3 Limited connectivity tests

In these tests, packets in both directions are dropped in the bad state with a 100% probability, marking a period of lost connectivity. In the good state, which is usually much smaller, packets are forwarded after being subjected to one of the two configured delays. This setup

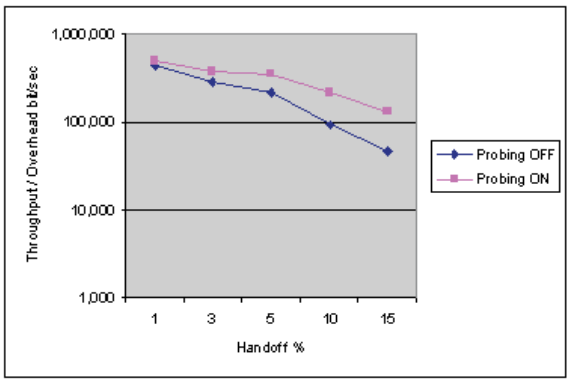

Figure 14. Throughput/Overhead ratio under handoff with 1 second rendezvous

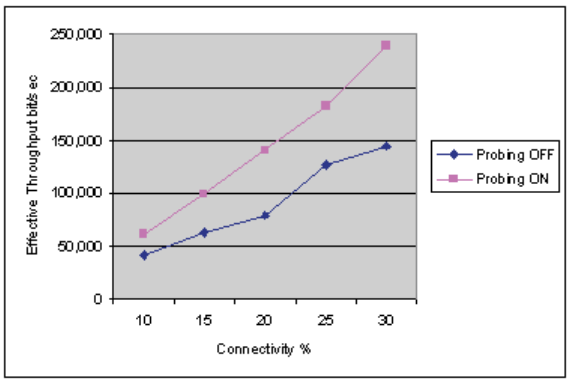

Figure 15. Effective throughput under limited connectivity with 5 seconds rendezvous

exemplifies limited connectivity conditions that result either from extended physical obstruction to the satellite signal, or from the unavailability of a satellite in range due to limitations in the coverage. This condition can also occur due to the constant mobility of the satellite and the frequent mobility of the terminal user. When the connection is later restored, it can either have a similar RTT to that before the interruption or a much deteriorated RTT that is usually indicative of congestion. Limited connectivity can vary in duration (rendezvous) or frequency depending on the satellite service coverage area and on the local environment (physical obstacles). For tests in this category, the connection is mainly in the bad state and only frequently in the good state.

An XSTP–OFF session does not detect the loss of connectivity, hence the segment loss, until it is over and either a STAT or a USTAT segment brings back a non–empty gap report in the good state. This inability to sense the loss in connectivity makes the session susceptible to drain some or all of the data segments from its send window. While connectivity is lost, the session maintains its usual polling cycles to sense when connectivity gets back. However, due to the extended connectivity loss periods, the polling cycles get extensively backed off to a level that compromises the ability of the session to detect the mainly short connectivity windows. In this case, the session either misses one or more connectivity windows or at best detects a window long after one started. As soon as the session detects the connectivity loss, it slows down its transmission before retransmitting the reportedly missing segments.

On the other hand, the XSTP–ON session can detect the connectivity loss period much quicker by detecting a break in the POLL/STAT interleaf. This detection ability makes the session only dump 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 the session detects the loss, it activates the probing mechanism. The mechanism maintains a probe–per–RTT cycle without applying any backoff. This feature has the advantage of quickly detecting the connectivity windows within one RTT. On the other hand, it can also have an adverse effect on the control overhead due to the amount of POLL traffic transmitted during the extended probing cycles. In the case of a premature timeout– event (a false alarm), the probing mode is terminated as soon as the next STAT arrives holding an empty gap report.

In this category, tests are performed with a connectivity rendezvous of 5 seconds, as shown in Fig. 15. Ref. [Mage03] also presents tests with a connectivity rendezvous of 10 seconds.

Tests are repeated with various connectivity rates ranging from 1 to 30%. The effective throughput is measured in each case. The first observation is that the more frequent the connectivity, the larger the chance of transmitting new data packets and the shorter the overall connection time. Effective throughput is also proportional to the rendezvous of the connectivity, especially when the rendezvous is orders of magnitude larger than the RTT. The reason relates to the fact that during longer connectivity periods, the congestion window grows much bigger than during shorter periods, resulting in better throughput, hence shorter connection time.

Another important observation is that XSTP–ON achieves consistently higher throughput than XSTP–OFF (from 13 to 77% gain). The main reason relates to the ability of XSTP-ON to differentiate congestion from connectivity loss situations. This ability makes the session more conservative in case of deteriorating RTTs, and more aggressive otherwise by maintaining a higher

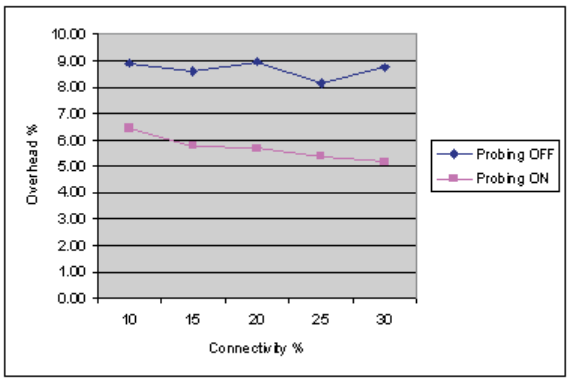

Figure 16. Overhead under limited connectivity with 5 seconds rendezvous

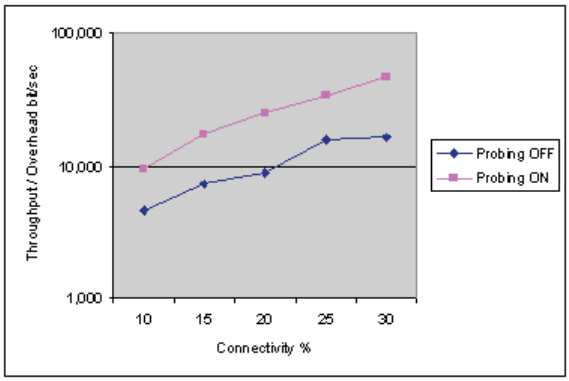

Figure 17. Throughput/overhead ratio under limited connectivity with 5 seconds rendezvous

congestion window. Another reason for this observation is the ability of the protocol to quickly suspend transmission as soon as it detects a loss in connectivity, and therefore avoid many retransmissions that usually extend the connection time.

The total overhead is plotted against the various connectivity rates in Fig. 16.

The general observation is that the expended overhead is not changing much in response to the changing connectivity rates. The reason for that observation becomes apparent when a closer look at the overhead components is taken. The major component is the retransmission overhead, which is usually incurred with every connectivity loss event. The value of that overhead is roughly the product of the number of connectivity loss events and average number of drops per event. In the tests, the bigger the connectivity rate, the higher the average send window the session maintains. That higher window translates into more average drops in the bad state. At the same time, the higher window also means that more packets on average will make it through in the good state, leading to a shorter connection time and therefore a lower number of connectivity loss events. In short, one part of the product increases while the other decreases. Therefore, the total retransmission overhead is kept somewhat in a tight range.

In addition, it is also shown that XSTP–ON sessions incur less overhead than XSTP–OFF sessions. The reason relates to the ability of the probing mechanism to detect the connectivity loss event much quicker using its early–timeout heuristic. That ability reduces the amount of segments dropped per loss event. Also by adapting to the error conditions in the network, the probing mechanism maintains a higher level of throughput and therefore a shorter connection time. This feature allows the connections to experience fewer loss events in total. It is also interesting to notice that the probing mechanism's advantage is lower in the 10–second tests than in the other tests. This lower advantage is mainly due to XSTP-OFF going back to slow start when connectivity is resumed. In this case, XSTP–OFF maintains a higher average throughput, which shortens the total connection time. Again, a shorter connection experiences fewer connectivity loss events and hence has less retransmission overhead. Another reason is the extension of the probing cycle in the longer connectivity loss events. This extension leads to an increase in the control overhead incurred by the probing mechanism (a probe per RTT), which lowers the overall advantage of the probing mechanism for these tests.

In Fig. 17, it is shown that throughput/overhead ratio is mainly proportional to the connectivity rate. Also for the same connectivity rate, the efficiency is proportional to the connectivity rendezvous. The figures also show that using XSTP–probing helps the session achieve more throughput with the same level of overhead expenditure.

7 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 Extended Satellite Transfer Protocol (XSTP), IPv4 with Dynamic Source Routing (DSR), AX.25 and a Packet Assembler Dissembler (PAD) protocol depicted in Fig. 18. AX.25 [Karn85] and PAD provide the packet radio services. Experiments were con-

ducted at data rates of 1.2 Kbps and 9.6 Kbps, 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 Ref. [Li03].

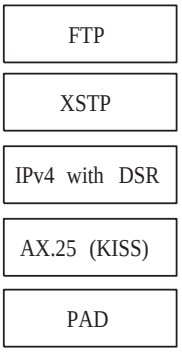

Figure 18. Protocol stack for packet radio.

7.1 Packet communications in space

In space communications, packet radio is used for CubeSat at data rates ranging from 1.2 Kbps to 9.6 Kbps [Heid00]. PACSATs are also users of packet radio [Pric90]. 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 application protocols, namely, a File Transfer Level 0 (FTL0) protocol and a PACSAT Broadcast Protocol (PBP) that both operate above the AX.25 protocol. CHIPSat [Daws02], 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 Space Communications Protocol Standards (SCPS) [SCP98] 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 bps 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.

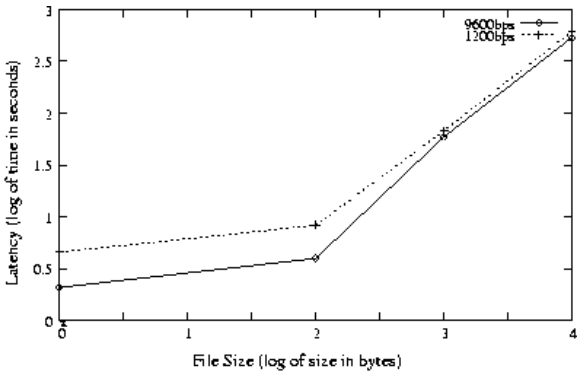

Figure 19. Latency ($\log_{10} - \log_{10}$ scale).

7.2 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. Each test is performed for 20-30 times. The average is reported, but results that are dramatically affected by noise are discarded.

Fig. 19 plots the results of transferring files of sizes from 1 B to 10 KB with data rates 1.2 Kbps and 9.6 Kbps. Horizontal axis is $\log_{10}$ of file size in bytes while vertical axis is $\log_{10}$ of time in seconds. We observe that when transferring files of small size (under 1 KB), bandwidth is the primary determinant of the latency, as file transfers at 9.6 Kbps are much faster than in 1.2 Kbps. 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 bps, of transferring files under 10 KB are presented in Table 1. 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 2. The average normalized throughput achieved in 1.2 Kbps is around 0.24, while

Data rate	File size			
	1 B	100 B	1 KB	10 KB
1200 bps	334.07	375.21	212.38	220.26
9600 bps	758.86	768.16	249.92	248.42

Table 1. Throughput in bps.

Data	File size			
	1 B	100 B	1 KB	10 KB
1200 bps	0.28	0.31	0.18	0.18
9600 bps	0.08	0.08	0.03	0.03

Table 2. Normalized throughput.

as to 9.6 Kbps, 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 [Klei75], 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.

8 Conclusion

In this paper, a new error control strategy for STP that adapts to the available 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 a STP receiver. The idea of the probing mechanism is to suspend new data transmission upon discovering a loss and wait until the error situation improves before adjusting the transmission rate. The mechanism compares 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 *eXtended Satellite Transport Protocol* (XSTP). In the tests, XSTP–probing consistently helps the protocol achieve higher levels of throughput/overhead ratio mainly by increasing the connection effective throughput while maintaining lower levels of overhead.

An important point of note is that the probing mechanism is most helpful in mixed link and congestion error conditions, especially when link errors are more prevalent. However, the more the errors turn out to be congestion–related, the more conservative the connection becomes on average. This effect can counter the throughput gains achieved by the probing mechanism for non–congestion errors. At some point, the aggregate negative impact on throughput can balance and sometimes exceed the positive impact. The same situation can occur to overhead as well. It is prudent to leave the investigation of that pivotal point to a future work.

From the analysis of 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

[ADG00] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. Ongoing TCP research related to satellites. RFC 2760, February 2000.

[AGS99] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP over satellite channels using standard mechanisms. RFC 2488, January 1999.

[BB02] M. Barbeau and F. Bordeleau. A protocol stack development tool using generative programming. In *Proceedings of Generative Programming and Component Engineering (GPCE)*, 2002. Lecture Notes in Computer Science 2487.

[BPSK96] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of ACM SIGCOMM '96*, Stanford, CA, August 1996.

[BV98] S. Biaz and N. Vaidya. Sender–based heuristic for distinguishing congestion losses from wireless losses. Technical report, Texas A&M University, June 1998. Tech. Rep. TR98-013.

[Daws02] S. Dawson, J. Wolff and J. Szielenski. CHIPSat's TCP/IP Mission Operations Architecture - Elegantly Simple. Proceedings of the 16th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, 2002.

[Mage03] M.E. Elaasar. XSTP: eXtended Satellite Transport Protocol. Master of Computer Science, School of Computer Science, Carleton University, January 2003.

[Heid00] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka and R. Twiggs. CubeSat: A New Generation

of Picosatellite for Education and Industry Low-Cost Space Experimentation. Proceedings of the Thirteenth Annual AIAA/USU Small Satellite Conference, Logan, UT, 2000.

[HK97] T. Henderson and R. Katz. Satellite transport protocol (STP): An SSCOP-based transport protocol for datagram satellite networks. In *Proceedings of 2nd Workshop on Satellite-Based Information Systems*, 1997.

[HP91] N. C. Hutchinson and L. L. Peterson. The x–Kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, January 1991.

[HV99] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of IEEE/ACM MOBICOM '99*, pages 219–230, August 1999.

[Karn85] P.R. Karn, H.E. Price and R.J. Diersing. Packet Radio in the Amateur Service. IEEE Journal on Selected Areas in Communications, Vol. SAC-3, No. 3, May 1985, pp. 431-439.

[Kat99] R. Katz. *Satellite Transport Protocol*. PhD thesis, December 1999.

[KLB99] T. Kim, S. Lu, and V. Bharghavan. Improving congestion control performance through loss differentiation. Eighth International Conference on Computer Communications and Networks, October 1999.

[Klei75] L. Kleinrock and F.A. Tobagi. Packet Switching in Radio Channles: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. IEEE Transactions on Communications, Vol. COM-23, No. 12, Dece,ber 1975, pp. 1400-1416.

[Li03] Z. Li. Performance of Generative Programming Protocol Implementation. Master of Science, Information and System Science, School of Computer Science, Carleton University, May 2003.

[Pric90] H.E. Price and J. Ward. PACSAT Broadcast Protocol. ARRL 9th Computer Networking Conference, pp. 232-238, August 1990.

[Sam99] N. Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. In *IEEE Proceedings of Communications*, volume 146, pages 222–230, August 1999.

[San99] R. Sangal. Performance analysis of the transmission control protocol over low earth orbit satellite communication systems. Master's thesis, College of Engineering and Technology, Ohio University, August 1999.

[SCP98] Space communication protocol specification (SCPS): Rationale, requirements and application notes. Technical report, Consultative Committee for Space Data Systems (CCSDS), August 1998. Green Book, Draft 0.4.

[TBGP00] V. Tsaoussidis, H. Badr, X. Ge, and K. Pentikousis. Energy / throughput tradeoffs of TCP error control strategies. In *Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC)*, 2000.

[TB00] V. Tsaoussidis and H. Badr. TCP-probing: Towards an error control schema with energy and throughput performance gains. In *Proceedings of the 8th IEEE Conference on Network Protocols*, Japan, November 2000.

[TBV00] V. Tsaoussidis, H. Badr, and R. Verma. Wave & wait protocol (WWP): An energy–saving transport protocol for mobile IP–devices. In *Proceedings of the 8th IEEE conference on networks*, pages 469–473, September 2000.

[TM01] V. Tsaoussidis and I. Matta. Open issues on TCP for mobile computing. Technical Report 2001-013, 3 2001.