Network Working Group
Request for Comments: 5325
Category: Informational

S. Burleigh
NASA/Jet Propulsion Laboratory
M. Ramadas
ISTRAC, ISRO
S. Farrell
Trinity College Dublin

September 2008

Licklider Transmission Protocol - Motivation

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

IESG Note

This RFC is not a candidate for any level of Internet Standard. It represents the consensus of the Delay Tolerant Networking (DTN) Research Group of the Internet Research Task Force (IRTF). See RFC 3932 for more information.

Abstract

This document describes the motivation for the development of the Licklider Transmission Protocol (LTP) designed to provide retransmission-based reliability over links characterized by extremely long message round-trip times (RTTs) and/or frequent interruptions in connectivity. Since communication across interplanetary space is the most prominent example of this sort of environment, LTP is principally aimed at supporting "long-haul" reliable transmission in interplanetary space, but it has applications in other environments as well.

In an Interplanetary Internet setting deploying the Bundle protocol, LTP is intended to serve as a reliable convergence layer over single-hop deep-space radio frequency (RF) links. LTP does Automatic Repeat reQuest (ARQ) of data transmissions by soliciting selective-acknowledgment reception reports. It is stateful and has no negotiation or handshakes.

This document is a product of the Delay Tolerant Networking Research Group and has been reviewed by that group. No objections to its publication as an RFC were raised.

Table of Contents

1. Introduction

The Licklider Transmission Protocol (LTP) is designed to provide retransmission-based reliability over links characterized by extremely long message round-trip times and/or frequent interruptions in connectivity. Communication in interplanetary space is the most prominent example of this sort of environment, and LTP is principally aimed at supporting "long-haul" reliable transmission over deep-space RF links. Specifically, LTP is intended to serve as a reliable "convergence layer" protocol, underlying the Delay-Tolerant Networking (DTN) [DTN] Bundle protocol [BP], in DTN deployments where data links are characterized by very long round-trip times.

This document describes the motivation for LTP, its features, functions, and overall design. It is part of a series of documents describing LTP. Other documents in the series include the main protocol specification document [LTPSPEC] and the protocol extensions document [LTPEXT].

The protocol is named in honor of ARPA/Internet pioneer JCR Licklider.

2. Problem

2.1. IPN Operating Environment

There are a number of fundamental differences between the environment for terrestrial communications (such as seen in the Internet) and the operating environments envisioned for the Interplanetary Internet (IPN) [IPN].

The most challenging difference between communication among points on Earth and communication among planets is round-trip delay, of which there are two main sources, both relatively intractable: physics and economics.

The more obvious type of delay imposed by nature is signal propagation time. Round-trip times between Earth and Jupiter's moon Europa, for example, run between 66 and 100 minutes.

Less obvious and more dynamic is the delay imposed by occultation. Communication between planets must be by radiant transmission, which is usually possible only when the communicating entities are in line of sight of each other. During the time that communication is impossible, delivery is impaired and messages must wait in a queue for later transmission.

Round-trip times and occultations can at least be readily computed given the ephemerides of the communicating entities. Additional delay that is less easily predictable is introduced by discontinuous transmission support, which is rooted in economics.

Communicating over interplanetary distances requires expensive special equipment: large antennas, high-performance receivers, etc.

For most deep-space missions, even non-NASA ones, these are currently provided by NASA's Deep Space Network (DSN) [DSN]. The communication resources of the DSN are currently oversubscribed and will probably remain so for the foreseeable future. Radio contact via the DSN must therefore be carefully scheduled and is often severely limited.

This over-subscription means that the round-trip times experienced by packets will be affected not only by the signal propagation delay and occultation, but also by the scheduling and queuing delays imposed by the management of Earth-based resources: packets to be sent to a given destination may have to be queued until the next scheduled contact period, which may be hours, days, or even weeks away.

These operating conditions imply a number of additional constraints on any protocol designed to ensure reliable communication over deep-space links.

- Long round-trip times mean substantial delay between the transmission of a block of data and the reception of an acknowledgment from the block's destination, signaling arrival of the block. If LTP postponed transmission of additional blocks of data until it received acknowledgment of the arrival of all prior blocks, valuable opportunities to utilize what little deep-space transmission bandwidth is available would be forever lost. Multiple parallel data block transmission "sessions" must be in progress concurrently in order to avoid under-utilization of the links.

- Like any reliable transport service employing ARQ, LTP is "stateful". In order to ensure the reception of a block of data it has sent, LTP must retain for possible retransmission all portions of that block that might not have been received yet. In order to do so, it must keep track of which portions of the block are known to have been received so far and which are not, together with any additional information needed for purposes of retransmitting part or all of that block.

- In the IPN, round-trip times may be so long and communication opportunities so brief that a negotiation exchange, such as an adjustment of transmission rate, might not be completed before connectivity is lost. Even if connectivity is uninterrupted, waiting for negotiation to complete before revising data transmission parameters might well result in costly under-utilization of link resources.

- Another respect in which LTP differs from TCP is that, while TCP connections are bidirectional (blocks of application data may be flowing in both directions on any single connection), LTP sessions are unidirectional. This design decision derives from the fact that the flow of data in deep-space flight missions is usually unidirectional. (Long round-trip times make interactive spacecraft operation infeasible, so spacecraft are largely autonomous and command traffic is very light.) Bidirectional data flow, where possible, is performed using two unidirectional links in opposite directions and at different data rates.

- Finally, the problem of timeout interval computation in the environment for which LTP is mainly intended is different from the analogous problem in the Internet. Since multiple sessions can be conducted in parallel, retardation of transmission on any single session while awaiting a timeout need not degrade communication performance on the association as a whole. Timeout intervals that would be intolerably optimistic in TCP don't necessarily degrade LTP's bandwidth utilization.

  But the reciprocal half-duplex nature of LTP communication makes it infeasible to use statistical analysis of round-trip history as a means of predicting round-trip time. The round-trip time for transmitted segment N could easily be orders of magnitude greater than that for segment N-1 if there happened to be a transient loss of connectivity between the segment transmissions. A different mechanism for timeout interval computation is needed.

2.2. Why Not TCP or SCTP?

These environmental characteristics -- long and highly variable delays, intermittent connectivity, and relatively high error rates -- make using unmodified TCP for end-to-end communications in the IPN infeasible. Using the TCP throughput equation from [TFRC] we can calculate the loss event rate (p) required to achieve a given steady-state throughput. Assuming the minimum RTT to Mars from planet Earth is 8 minutes (one-way speed of light delay to Mars at its closest approach to Earth is 4 minutes), assuming a packet size of 1500 bytes, assuming that the receiver acknowledges every other packet, and ignoring negligible higher-order terms in p (i.e., ignoring the second additive term in the denominator of the TCP throughput equation), we obtain the following table of loss event rates required to achieve various throughput values.

Throughput	Loss event rate (p)
10 Mbps	4.68 * 10^(-12)
1 Mbps	4.68 * 10^(-10)
100 Kbps	4.68 * 10^(-8)
10 Kbps	4.68 * 10^(-6)

Note that although multiple losses encountered in a single RTT are treated as a single loss event in the TCP throughput equation [TFRC], such loss event rates are still unrealistic on deep-space links.

For the purposes of this discussion, we are not considering the more aggressive TCP throughput equation that characterizes HighSpeed TCP [HSTCP].

The TCP characteristic of an initial three-way handshake for each new connection, followed by slow-start, is a further obstacle, because the delay of the three-way handshake and the additional delay of slow-start could be exorbitant in a long-delay environment.

The Stream Control Transmission Protocol (SCTP) [SCTP] can multiplex "chunks" (units of application data) for multiple sessions over a single-layer connection (called an 'association' in SCTP terminology) as LTP does, but it still requires multiple round trips prior to transmitting application data for session setup and so clearly does not suit the needs of the IPN operating environment.

3. Protocol Overview

3.1. Nominal Operation

The nominal sequence of events in an LTP transmission session is as follows.

Operation begins when a client service instance asks an LTP engine to transmit a block of data to a remote client service instance.

LTP regards each block of data as comprising two parts: a "red-part", whose delivery must be assured by acknowledgment and retransmission as necessary, followed by a "green-part" whose delivery is attempted, but not assured. The length of either part may be zero; that is, any given block may be designated entirely red (retransmission continues until reception of the entire block has been asserted by the receiver) or entirely green (no part of the block is acknowledged or retransmitted). Thus, LTP can provide both TCP-like and UDP-like functionality concurrently on a single session.

Note that in a red-green block transmission, the red-part data does NOT have any urgency or higher-priority semantics relative to the block's green-part data. The red-part data is merely data for which the user has requested reliable transmission, possibly (though not necessarily) data without which the green-part data may be useless, such as an application-layer header or other metadata.

The client service instance uses the LTP implementation's application programming interface to specify to LTP the identity of the remote client service instance to which the data must be transmitted, the location of the data to be transmitted, the total length of data to be transmitted, and the number of leading data bytes that are to be transmitted reliably as "red" data. The sending engine starts a transmission session for this block and notifies the client service instance that the session has been started. Note that

LTP communication session parameters are not negotiated but are instead asserted unilaterally, subject to application-level network management activity; the sending engine does not negotiate the start of the session with the remote client service instance's engine.

The sending engine then initiates the original transmission: it queues for transmission as many data segments as are necessary to transmit the entire block, within the constraints on maximum segment size imposed by the underlying communication service. The last segment of the red-part of the block is marked as the end of red-part (EORP) indicating the end of red-part data for the block, and as a checkpoint (identified by a unique checkpoint serial number) indicating that the receiving engine must issue a reception report upon receiving the segment. The last segment of the block overall is marked end of block (EOB) indicating that the receiving engine can calculate the size of the block by summing the offset and length of the data in the segment.

LTP is designed to run directly over a data-link layer protocol, but it may instead be deployed directly over UDP in some cases (i.e., for software development or in private local area networks). In either case, the protocol layer immediately underlying LTP is here referred to as the "local data-link layer".

At the next opportunity, subject to allocation of bandwidth to the queue into which the block data segments were written, the enqueued segments and their lengths are passed to the local data-link layer protocol (which might be UDP/IP) via the API supported by that protocol, for transmission to the LTP engine serving the remote client service instance.

A timer is started for the EORP, so that it can be retransmitted automatically if no response is received.

The content of each local data-link layer protocol data unit (link-layer frame or UDP datagram) is required to be an integral number of LTP segments, and the local data-link layer protocol is required never to deliver incomplete LTP segments to the receiving LTP engine. When the local data-link layer protocol is UDP, the LTP authentication [LTPEXT] extension should be used to ensure data integrity unless the end-to-end path is one in which either the likelihood of datagram content corruption is negligible (as in some private local area networks) or the consequences of receiving and processing corrupt LTP segments are insignificant (as perhaps during software development). When the LTP authentication extension is not

used, LTP requires the local data-link layer protocol to perform integrity checking of all segments received; specifically, the local data-link layer protocol is required to detect any corrupted segments that are received and to discard them silently.

Received segments that are not discarded are passed up to the receiving LTP engine via the API supported by the local data-link layer protocol.

On reception of the first data segment for the block, the receiving engine starts a reception session for this block and notifies the local instance of the relevant client service that the session has been started. In the nominal case, it receives all segments of the original transmission without error. Therefore, on reception of the EORP data segment, it responds by (a) queuing for transmission to the sending engine a report segment indicating complete reception and (b) delivering the received red-part of the block to the local instance of the client service: on reception of each data segment of the green-part, it responds by immediately delivering the received data to the local instance of the client service.

All delivery of data and protocol event notices to the local client service instance is performed using the LTP implementation's application programming interface.

Note that since LTP data flows are unidirectional, LTP's data acknowledgments -- "reception reports" -- can't be piggybacked on data segments as in TCP. They are instead carried in a separate segment type.

At the next opportunity, the enqueued report segment is immediately transmitted to the sending engine and a timer is started so that the report segment can be retransmitted automatically if no response is received.

The sending engine receives the report segment, turns off the timer for the EORP, enqueues for transmission to the receiving engine a report-acknowledgment segment, and notifies the local client service instance that the red-part of the block has been successfully transmitted. The session's red-part transmission has now ended.

At the next opportunity, the enqueued report-acknowledgment segment is immediately transmitted to the receiving engine.

The receiving engine receives the report-acknowledgment segment and turns off the timer for the report segment. The session's red-part reception has now ended and transmission of the block is complete.

3.1.1. Link State Cues

Establishing a communication link across interplanetary distances may entail hardware configuration changes based on the presumed operational state of the remote communicating entity, for example:

- orienting a directional antenna correctly;
- tuning a transponder to pre-selected transmission and/or reception frequencies; and
- diverting precious electrical power to the transponder at the last possible moment, and for the minimum necessary length of time.

We therefore assume that the operating environment in which LTP functions is able to pass information on the link status (termed "link state cues" in this document) to LTP, telling it which remote LTP engine(s) should currently be transmitting to the local LTP engine and vice versa. The operating environment itself must have this information in order to configure communication link hardware correctly.

3.1.2. Deferred Transmission

Link state cues also notify LTP when it is and isn't possible to transmit segments. In deep-space communications, at no moment can there ever be any expectation of two-way connectivity. It is always possible for LTP to be generating outbound segments -- in response to received segments, timeouts, or requests from client services -- that cannot immediately be transmitted. These segments must be queued for transmission at a later time when a link has been established, as signaled by a link state cue.

In concept, every outbound LTP segment is appended to one of two queues -- forming a queue-set -- of traffic bound for the LTP engine that is that segment's destination. One such traffic queue is the "internal operations queue" of that queue set; the other is the application data queue for the queue set. The de-queuing of a segment always implies delivering it to the underlying communication system for immediate transmission. Whenever the internal operations queue is non-empty, the oldest segment in that queue is the next segment de-queued for transmission to the destination; at all other times, the oldest segment in the application data queue is the next segment de-queued for transmission to the destination.

The production and enqueuing of a segment and the subsequent actual transmission of that segment are in principle wholly asynchronous.

In the event that (a) a transmission link to the destination is currently active and (b) the queue to which a given outbound segment is appended is otherwise empty and (c) either this queue is the internal operations queue or else the internal operations queue is empty, the segment will be transmitted immediately upon production. Transmission of a newly queued segment is necessarily deferred in all other circumstances.

Conceptually, the de-queuing of segments from traffic queues bound for a given destination is initiated upon reception of a link state cue indicating that the underlying communication system is now transmitting to that destination; i.e., the link to that destination is now active. It ceases upon reception of a link state cue indicating that the underlying communication system is no longer transmitting to that destination; i.e., the link to that destination is no longer active.

3.1.3. Timers

LTP relies on accurate calculation of expected arrival times for report and acknowledgment segments in order to know when proactive retransmission is required. If a calculated time were even slightly early, the result would be costly unnecessary retransmission. On the other hand, calculated arrival times may safely be several seconds late: the only penalties for late timeout and retransmission are slightly delayed data delivery and slightly delayed release of session resources.

Since statistics derived from round-trip history cannot safely be used as a predictor of LTP round-trip times, we have to assume that round-trip timing is at least roughly deterministic -- i.e., that sufficiently accurate RTT estimates can be computed individually in real time from available information.

This computation is performed in two stages:

- We calculate a first approximation of RTT by simply doubling the known one-way light time to the destination and adding an arbitrary margin for any additional anticipated latency, such as queuing and processing delay at both ends of the transmission. For deep-space operations, the margin value is typically a small number of whole seconds. Although such a margin is enormous by Internet standards, it is insignificant compared to the two-way

light time component of round-trip time in deep space. We choose to risk tardy retransmission, which will postpone delivery of one block by a relatively tiny increment, in preference to premature retransmission, which will unnecessarily consume precious bandwidth and thereby degrade overall performance.

- Then, to account for the additional delay imposed by interrupted connectivity, we dynamically suspend timers during periods when the relevant remote LTP engines are known to be unable to transmit responses. This knowledge of the operational state of remote entities is assumed to be provided by link state cues from the operating environment.

The following discussion is the basis for LTP's expected arrival time calculations.

The total time consumed in a single "round trip" (transmission and reception of the original segment, followed by transmission and reception of the acknowledging segment) has the following components:

- Protocol processing time: The time consumed in issuing the original segment, receiving the original segment, generating and issuing the acknowledging segment, and receiving the acknowledging segment.

- Outbound queuing delay: The delay at the sender of the original segment while that segment is in a queue waiting for transmission, and delay at the sender of the acknowledging segment while that segment is in a queue waiting for transmission.

- Radiation time: The time that passes while all bits of the original segment are being radiated, and the time that passes while all bits of the acknowledging segment are being radiated. (This is significant only at extremely low data transmission rates.)

- Round-trip light time: The signal propagation delay at the speed of light, in both directions.

- Inbound queuing delay: Delay at the receiver of the original segment while that segment is in a reception queue, and delay at the receiver of the acknowledging segment while that segment is in a reception queue.

- Delay in transmission of the original segment or the acknowledging segment due to loss of connectivity -- that is, interruption in outbound link activity at the sender of either segment due to occultation, scheduled end of tracking pass, etc.

In this context, where errors on the order of seconds or even minutes may be tolerated, protocol processing time at each end of the session is assumed to be negligible.

Inbound queuing delay is also assumed to be negligible because, even on small spacecraft, LTP processing speeds are high compared to data transmission rates.

Two mechanisms are used to make outbound queuing delay negligible:

- The expected arrival time of an acknowledging segment is not calculated until the moment the underlying communication system notifies LTP that radiation of the original segment has begun. All outbound queuing delay for the original segment has already been incurred at that point.

- LTP's deferred transmission model minimizes latency in the delivery of acknowledging segments (reports and acknowledgments) to the underlying communication system. That is, acknowledging segments are (in concept) appended to the internal operations queue rather than the application data queue, so they have higher transmission priority than any other outbound segments, i.e., they should always be de-queued for transmission first. This limits outbound queuing delay for a given acknowledging segment to the time needed to de-queue and radiate all previously generated acknowledging segments that have not yet been de-queued for transmission. Since acknowledging segments are sent infrequently and are normally very small, outbound queuing delay for a given acknowledging segment is likely to be minimal.

Deferring calculation of the expected arrival time of the acknowledging segment until the moment at which the original segment is radiated has the additional effect of removing from consideration any original segment transmission delay due to loss of connectivity at the original segment sender.

Radiation delay at each end of the session is simply segment size divided by transmission data rate. It is insignificant except when the data rate is extremely low (for example, 10 bps), in which case the use of LTP may well be inadvisable for other reasons (LTP header overhead, for example, could be too much under such data rates). Therefore, radiation delay is normally assumed to be negligible.

We assume that one-way light time to the nearest second can always be known (for example, provided by the operating environment).

So the initial expected arrival time for each acknowledging segment is typically computed as simply the current time at the moment that radiation of the original segment begins, plus twice the one-way light time, plus 2*N seconds of margin to account for processing and queuing delays and for radiation time at both ends. N is a parameter set by network management for which 2 seconds seem to be a reasonable default value.

This leaves only one unknown, the additional round-trip time introduced by loss of connectivity at the sender of the acknowledging segment. To account for this, we again rely on external link state cues. Whenever interruption of transmission at a remote LTP engine is signaled by a link state cue, we suspend the countdown timers for all acknowledging segments expected from that engine. Upon a signal that transmission has resumed at that engine, we resume those timers after (in effect) adding to each expected arrival time the length of the timer suspension interval.

3.2. Retransmission

Loss or corruption of transmitted segments may cause the operation of LTP to deviate from the nominal sequence of events described above.

Loss of one or more red-part data segments other than the EORP segment triggers data retransmission: the receiving engine returns a reception report detailing all the contiguous ranges of red-part data received (assuming no discretionary checkpoints were received, which are described below). The reception report is normally sent in a single report segment that carries a unique report serial number and the scope of red-part data covered. For example, if the red-part data covered block offsets [0:1000] and all but the segment in range [500:600] were received, the report segment with a unique serial number (say 100) and scope [0:1000] would carry two report entries: (0:500) and (600:1000). The maximum size of a report segment, like all LTP segments, is constrained by the data-link MTU; if many non-contiguous segments were lost in a large block transmission and/or the data-link MTU was relatively small, multiple report segments need to be generated. In this case, LTP generates as many report segments as are necessary and splits the scope of red-part data covered across multiple report segments so that each of them may stand on their own. For example, if three report segments are to be generated as part of a reception report covering red-part data in range [0:1,000,000], they could look like this: RS 19, scope [0:300,000], RS 20, scope

[300,000:950,000], and RS 21, scope [950,000:1,000,000]. In all cases, a timer is started upon transmission of each report segment of the reception report.

On reception of each report segment, the sending engine responds as follows:

- It turns off the timer for the checkpoint referenced by the report segment, if any.

- It enqueues a reception-acknowledgment segment acknowledging the report segment (to turn off the report retransmission timer at the receiving engine). This segment is sent immediately at the next transmission opportunity.

- If the reception claims in the report segment indicate that not all data within the scope have been received, it normally initiates a retransmission by enqueuing all data segments not yet received. The last such segment is marked as a checkpoint and contains the report serial number of the report segment to which the retransmission is a response. These segments are likewise sent at the next transmission opportunity, but only after all data segments previously queued for transmission to the receiving engine have been sent. A timer is started for the checkpoint, so that it can be retransmitted automatically if no responsive report segment is received.

- On the other hand, if the reception claims in the report segment indicate that all data within the scope of the report segment have been received, and the union of all reception claims received so far in this session indicates that all data in the red-part of the block have been received, then the sending engine notifies the local client service instance that the red-part of the block has been successfully transmitted; the session’s red-part transmission has ended.

On reception of a report-acknowledgment segment, the receiver turns off the timer for the referenced report segment.

On reception of a checkpoint segment with a non-zero report serial number, the receiving engine responds as follows:

- It returns a reception report comprising as many report segments as are needed in order to report in detail on all data reception within the scope of the referenced report segment, and a timer is started for each report segment.

- If at this point all data in the red-part of the block have been received, the receiving engine delivers the received block's red-part to the local instance of the client service and, upon reception of reception-acknowledgment segments acknowledging all report segments, the session's red-part reception ends and transmission of the block is complete. Otherwise, the data retransmission cycle continues.

Loss of a checkpoint segment or the report segment generated in response causes timer expiry; when this occurs, the sending engine normally retransmits the checkpoint segment. Similarly, the loss of a report segment or the corresponding report-acknowledgment segment causes the report segment's timer to expire; when this occurs, the receiving engine normally retransmits the report segment.

Note that the redundant reception of a report segment (i.e., one that was already received and processed by the sender), retransmitted due to loss of the corresponding report-acknowledgment segment for example, causes another report-acknowledgment segment to be transmitted in response but is otherwise ignored. If any of the data segments retransmitted in response to the original reception of the report segment were lost, further retransmission of those data segments will be requested by the reception report generated in response to the last retransmitted data segment marked as a checkpoint. Thus, unnecessary retransmission is suppressed.

Note also that the responsibility for responding to segment loss in LTP is shared between the sender and receiver of a block: the sender retransmits checkpoint segments in response to checkpoint timeouts, and retransmits missing data in response to reception reports indicating incomplete reception, while the receiver retransmits report segments in response to timeouts. An alternative design would have been to make the sender responsible for all retransmission, in which case the receiver would not expect report-acknowledgment segments and would not retransmit report segments. There are two disadvantages to this approach:

First, because of constraints on segment size that might be imposed by the underlying communication service, it is at least remotely possible that the response to any single checkpoint might be multiple report segments. An additional sender-side mechanism for detecting and appropriately responding to the loss of some proper subset of those reception reports would be needed. We believe that the current design is simpler.

Second, an engine that receives a block needs a way to determine when the session can be closed. In the absence of explicit final report acknowledgment (which entails retransmission of the report in case of the loss of the report acknowledgment), the alternatives are (a) to close the session immediately on transmission of the report segment that signifies complete reception and (b) to close the session on receipt of an explicit authorization from the sender. In case (a), loss of the final report segment would cause retransmission of a checkpoint by the sender, but the session would no longer exist at the time the retransmitted checkpoint arrived. The checkpoint could reasonably be interpreted as the first data segment of a new block, most of which was lost in transit, and the result would be redundant retransmission of the entire block. In case (b), the explicit session termination segment and the responsive acknowledgment by the receiver (needed in order to turn off the timer for the termination segment, which in turn would be needed in case of in-transit loss or corruption of the termination segment) would somewhat complicate the protocol, increase bandwidth consumption, and retard the release of session state resources at the sender. Here again we believe that the current design is simpler and more efficient.

3.3. Accelerated Retransmission

Data segment retransmission occurs only on receipt of a report segment indicating incomplete reception; report segments are normally transmitted only at the end of original transmission of the red-part of a block or at the end of a retransmission. For some applications, it may be desirable to trigger data segment retransmission incrementally during the course of red-part original transmission so that the missing segments are recovered sooner. This can be accomplished in two ways:

- Any red-part data segment prior to the EORP can additionally be flagged as a checkpoint. Reception of each such "discretionary" checkpoint causes the receiving engine to issue a reception report.

- At any time during the original transmission of a block’s red-part (that is, prior to reception of any data segment of the block’s green-part), the receiving engine can unilaterally issue additional asynchronous reception reports. Note that the CFDP protocol’s "Immediate" mode is an example of this sort of asynchronous reception reporting [CFDP]. The reception reports generated for accelerated retransmission are processed in exactly the same way as the standard reception reports.

3.4. Session Cancellation

A transmission session may be canceled by either the sending or the receiving engine in response either to a request from the local client service instance or to an LTP operational failure as noted earlier. Session cancellation is accomplished as follows.

The canceling engine deletes all currently queued segments for the session and notifies the local instance of the affected client service that the session is canceled. If no segments for this session have yet been sent to or received from the corresponding LTP engine, then at this point the canceling engine simply closes its state record for the session and cancellation is complete.

Otherwise, a session cancellation segment is queued for transmission. At the next opportunity, the enqueued cancellation segment is immediately transmitted to the LTP engine serving the remote client service instance. A timer is started for the segment, so that it can be retransmitted automatically if no response is received.

The corresponding engine receives the cancellation segment, enqueues for transmission to the canceling engine a cancellation-acknowledgment segment, deletes all other currently queued segments for the indicated session, notifies the local client service instance that the block has been canceled, and closes its state record for the session.

At the next opportunity, the enqueued cancellation-acknowledgment segment is immediately transmitted to the canceling engine.

The canceling engine receives the cancellation-acknowledgment, turns off the timer for the cancellation segment, and closes its state record for the session.

Loss of a cancellation segment or of the responsive cancellation-acknowledgment causes the cancellation segment timer to expire. When this occurs, the canceling engine retransmits the cancellation segment.

4. Security Considerations

There is a clear risk that unintended receivers can listen in on LTP transmissions over satellite and other radio broadcast data links. Such unintended recipients of LTP transmissions may also be able to manipulate LTP segments at will.

Hence, there is a potential requirement for confidentiality, integrity, and anti-DoS (Denial of Service) security services and mechanisms.

In particular, DoS problems are more severe for LTP compared to typical Internet protocols because LTP inherently retains state for long periods and has very long time-out values. Further, it could be difficult to reset LTP nodes to recover from an attack. Thus, any adversary who can actively attack an LTP transmission has the potential to create severe DoS conditions for the LTP receiver.

To give a terrestrial example: were LTP to be used in a sparse sensor network, DoS attacks could be mounted resulting in nodes missing critical information, such as communications schedule updates. In such cases, a single successful DoS attack could take a node entirely off the network until the node was physically visited and reset.

Even for deep-space applications of LTP, we need to consider certain terrestrial attacks, in particular those involving insertion of messages into an ongoing session (usually without having seen the exact bytes of the previous messages in the session). Such attacks are likely in the presence of firewall failures at various nodes in the network, or due to Trojan software running on an authorized host. Many message insertion attacks will depend on the attacker correctly "guessing" something about the state of the LTP peers, but experience shows that successful guesses are easier than might be thought [DDJ].

We now consider the appropriate layer(s) at which security mechanisms can be deployed to increase the security properties of LTP, and the trade-offs entailed in doing so.

The Application layer (above-LTP)

Higher-layer security mechanisms clearly protect LTP payload, but leave LTP headers open. Such mechanisms provide little or no protection against DoS type attacks against LTP, but may well provide sufficient data integrity and ought to be able to provide data confidentiality.

The LTP layer

An authentication header (similar to IPsec [AH]) can help protect against replay attacks and other bogus packets. However, an adversary may still see the LTP header of segments passing by in the ether. This approach also requires some key management infrastructure to be in place in order to provide strong authentication, which may not always be an acceptable overhead. Such an authentication header could mitigate many DoS attacks.

Similarly, a confidentiality service could be defined for LTP payload and (some) header fields. However, this seems less attractive since (a) confidentiality is arguably better provided either above or below the LTP layer, (b) key management for such a service is harder (in a high-delay context) than for an integrity service, and (c) forcing LTP engines to attempt decryption of incoming segments can in itself provide a DoS opportunity.

Further, within the LTP layer we can make various design decisions to reduce the probability of successful DoS attacks. In particular, we can mandate that values for certain fields in the header (session numbers, for example) be chosen randomly.

The Data-link layer (below-LTP)

The lower layers can clearly provide confidentiality and integrity services, although such security may result in unnecessary overhead if the cryptographic service provided is not required for all data. For example, it can be harder to manage lower layers so that only the data that requires encryption is in fact encrypted. Encrypting all data could represent a significant overhead for some LTP use cases. However, the lower layers are often the place where compression and error-correction is done, and so may well also be the optimal place to do encryption since the same issues with applying or not applying the service apply to both encryption and compression.

In light of these considerations, LTP includes the following security mechanisms:

The optional LTP Authentication mechanism is an LTP segment extension comprising a ciphersuite identifier and optional key identifier that precede the segment’s content, plus an authentication value (either a message authentication code or a digital signature) that follows the segment’s content; the ciphersuite ID is used to indicate the length and format of the authentication value. The authentication mechanism serves to assure the segment’s integrity and, depending on the ciphersuite selected and the key management regime, its authenticity.

The optional LTP cookie mechanism is an LTP segment extension comprising a "cookie" -- a randomly chosen numeric value -- that precedes the segment’s content. By increasing the number of bytes in a segment that cannot be easily predicted by an inauthentic data source, and by requiring that segments lacking the correct values of these bytes be silently discarded, the cookie mechanism increases the difficulty of mounting a successful denial-of-service attack on an LTP engine.

The above mechanisms are defined in detail in the LTP extensions document [LTPEXT].

In addition, the serial numbers of LTP checkpoints and reports are required to be randomly chosen integers, and implementers are encouraged to choose session numbers randomly as well. This randomness adds a further increment of protection against DoS attacks. See [PRNG] for recommendations related to randomness.

5. IANA Considerations

Please see the IANA Considerations sections of [LTPSPEC] and [LTPEXT].

6. Acknowledgments

Many thanks to Tim Ray, Vint Cerf, Bob Durst, Kevin Fall, Adrian Hooke, Keith Scott, Leigh Torgerson, Eric Travis, and Howie Weiss for their thoughts on this protocol and its role in Delay-Tolerant Networking architecture.

Part of the research described in this document was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was performed under DOD Contract DAA-B07-00-CC201, DARPA AO H912; JPL Task Plan No. 80-5045, DARPA AO H870; and NASA Contract NAS7-1407.

Thanks are also due to Shawn Ostermann, Hans Kruse, and Dovel Myers at Ohio University for their suggestions and advice in making various design decisions. This work was done when Manikantan Ramadas was a graduate student at the EECS Dept., Ohio University, in the Internetworking Research Group Laboratory.

Part of this work was carried out at Trinity College Dublin as part of the SeNDT contract funded by Enterprise Ireland’s research innovation fund.

7. References

7.1. Informative References

[LTPSPEC] Ramadas, M., Burleigh, S., and S. Farrell, "Licklider Transmission Protocol - Specification", RFC 5326, September 2008.

[LTPEXT] Farrell, S., Ramadas, M., and S. Burleigh, "Licklider Transmission Protocol - Security Extensions", RFC 5327, September 2008.

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[BP] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[CFDP] CCSDS File Delivery Protocol (CFDP). Recommendation for Space Data System Standards, CCSDS 727.0-B-2 BLUE BOOK Issue 1, October 2002.

[DDJ] I. Goldberg and E. Wagner, "Randomness and the Netscape Browser", Dr. Dobb's Journal, 1996, (pages 66-70).

[DSN] Deep Space Mission Systems Telecommunications Link Design Handbook (810-005) web-page, "http://eis.jpl.nasa.gov/deepspace/dsndocs/810-005/"

[DTN] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets", In Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, Aug 2003.

[IPN] InterPlanetary Internet Special Interest Group web page, "http://www.ipnsig.org".

[TFRC] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

[HSTCP] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[SCTP] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[PRNG] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Authors' Addresses

Scott C. Burleigh
Jet Propulsion Laboratory
4800 Oak Grove Drive
M/S: 301-485B
Pasadena, CA 91109-8099
Telephone: +1 (818) 393-3353
Fax: +1 (818) 354-1075
EMail: Scott.Burleigh@jpl.nasa.gov

Manikantan Ramadas
ISRO Telemetry Tracking and Command Network (ISTRAC)
Indian Space Research Organization (ISRO)
Plot # 12 & 13, 3rd Main, 2nd Phase
Peenya Industrial Area
Bangalore 560097
India
Telephone: +91 80 2364 2602
EMail: mramadas@gmail.com

Stephen Farrell
Computer Science Department
Trinity College Dublin
Ireland
Telephone: +353-1-896-1761
EMail: stephen.farrell@cs.tcd.ie

Full Copyright Statement

Intellectual Property