Delay Tolerant Networking Research Group
Internet Draft
<draft-irtf-dtnrg-ltp-motivation-01.txt>
July 2005
Expires January 2006

S. Burleigh
NASA/Jet Propulsion Laboratory
M. Ramadas
Ohio University
S. Farrell
Trinity College Dublin

Licklider Transmission Protocol - Motivation

Status of this Memo

Abstract

This document describes 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 has applications in other environments as well.

In an Interplanetary Internet setting deploying the Bundling protocol being developed by the Delay Tolerant Networking Research Group, LTP is intended to serve as a reliable convergence layer over single hop deep-space RF links. LTP does ARQ of data transmissions by soliciting selective-acknowledgment reception reports. It is stateful and has no negotiation or handshakes.

Table of Contents

1. Introduction

The Licklider Transmission Protocol (LTP) described in this memo 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.

Since 1982 the principal source of standards for space communications has been the Consultative Committee for Space Data Systems (CCSDS) [CCSDS]. Engineers of CCSDS member agencies have developed communication protocols that function within the constraints imposed by operations in deep space. These constraints differ sharply from those within which the Internet typically functions:

- Extremely long signal propagation delays, on the order of seconds, minutes, or hours rather than milliseconds.
- Frequent and lengthy interruptions in connectivity.
- Low levels of traffic coupled with high rates of transmission error.
- Meager bandwidth and highly asymmetrical data rates.

The CCSDS File Delivery Protocol (CFDP) [CFDP] in particular, automates reliable file transfer across interplanetary distances by detecting data loss and initiating the requisite retransmission without mission operator intervention.

CFDP by itself is sufficient for operating individual missions, but its built-in networking capabilities are rudimentary. In order to operate within the IPN environment it must rely on the routing and incremental retransmission capabilities of the Bundling protocol [BP] defined for Delay-Tolerant Networks [DTN]. LTP is intended to serve as a reliable "convergence layer" protocol underlying Bundling in DTN regions whose links are characterized by very long round-trip times. Its design notions are directly descended from the retransmission procedures defined for CFDP.

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

document [LTPEXT] respectively.

2. Motivation

2.1 IPN Operating Environment

There are a number of fundamental differences between the environment for terrestrial communications and the operating environments envisioned for the 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: natural law and economics.

The more obvious type of delay imposed by nature is signal propagation time. Our inability to transmit data at speeds higher than the speed of light means that while round-trip times in the terrestrial Internet range from milliseconds to a few seconds, minimum round-trip times to Mars range from 8 to 40 minutes, depending on the planet's position. Round-trip times between Earth and Jupiter's moon Europa 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. An entity residing on a planetary surface will be periodically taken out of sight by the planet's rotation (it will be "on the other side of" the planet); an entity that orbits a planet will be periodically taken out of sight by orbital motion (it will be "behind" the planet); and planets themselves lose mutual visibility when their trajectories take them to opposite sides of the Sun. 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. While studies have been done as to the feasibility of upgrading or replacing the current DSN, the number of deep space missions will probably continue to grow faster

than the terrestrial infrastructure that supports them, making over-subscription a persistent problem. 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 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. While queuing and scheduling delays are generally known well in advance except when missions need emergency service (such as during landings and maneuvers), the long and highly variable delays make the design of timers, and retransmission timers in particular, quite difficult.

Another significant difference between deep space and terrestrial communication is bandwidth availability. The combined effects of large distances (resulting in signal attenuation), the expense and difficulty of deploying large antennas to distant planets, and the difficulty of generating electric power in space all mean that the available bandwidth for communication in the IPN will likely remain modest compared to terrestrial systems. Maximum data rates on the order of a few tens of megabits per second will probably be the norm for the next few decades.

Moreover, the available bandwidths are highly asymmetrical: data are typically transmitted at different rates in different directions on the same link. Current missions are usually designed with a much higher data "return" rate (from spacecraft to Earth) than "command" rate (from Earth to spacecraft). The reason for the asymmetry is simple: nobody ever wanted a high-rate command channel, and, all else being equal, it was deemed better to have a more reliable command channel than a faster one. This design choice has led to data rate asymmetries in excess of 100:1, sometimes approaching 1000:1. A strong desire for a very robust command channel will probably remain, so any transport protocol designed for use in the IPN will need to function with a relatively low-bandwidth outbound channel to spacecraft and landers.

The difficulty of generating power on and around other planets will also result in relatively low signal-to-noise ratios and thus high bit error rates. Current deep-space missions operate with raw bit error rates on the order of 10^(-1) to 10^(-3); while heavy coding is used to reduce error rates, the coding overhead further reduces the residual bandwidth available for data transmission.

Signal propagation delay is the only truly immutable characteristic that distinguishes the IPN from terrestrial communications. Queuing

and scheduling delays, low data rates, intermittent connectivity, and high bit error rates can all be mitigated or eliminated by adding more infrastructure. But this additional infrastructure is likely to be provided (if at all) only in the more highly developed core areas of the IPN. We see the IPN growing outwards from Earth as we explore more and more planets, moons, asteroids, and possibly other stars. This suggests that there will always be a "fringe" to the fabric of the IPN, an area without a rich communications infrastructure. The delay, data rate, connectivity, and error characteristics mentioned above will probably always be an issue somewhere in the IPN.

2.2 Why not Standard Internet Protocols?

These environmental characteristics - long delays, low and asymmetric bandwidth, 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 from [TFRC], such loss event rates are still unrealistic on deep space links.

The above values are upper bounds on steady-state throughput. Since the number of packets in an episode of connectivity will generally be under 10,000 due to the low available bandwidth, TCP performance would be dominated by its behavior during slow-start. This means that even when Mars is at its closest approach to Earth it would take a TCP session nearly 100 minutes to ramp up to an Earth-Mars transmission rate of 20kbps.

Note: Lab experiments using a channel emulator and standard

applications show that even if TCP could be pushed to work efficiently at such distances, many applications either rely on several rounds of handshaking or have built-in timers that render them non-functional when the round-trip-time is over a couple of minutes. For example, it typically takes eight round trips for FTP to get to a state where data can begin flowing. Since an FTP server may time out and reset the connection after 5 minutes of inactivity, a conformant standard FTP server could be unusable for communicating even with the closest planets.

The SCTP [SCTP] protocol can multiplex bundles (Note : defined differently from the DTN "bundle") for multiple sessions over a single layer connection just as LTP, but 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. Features

The design of LTP differs from that of TCP in several significant ways. The common themes running through these differences are two central design assumptions, both of which amount to making virtues of necessity.

First: given the severe innate constraints on interplanetary communication discussed above, we assume that the computational resources available to LTP engines will always be ample compared to the communication resources available on the link between them.

Certainly in many cases the computational resources available to a given LTP engine - such as one on board a small robotic spacecraft - will not be ample by the standards of the Internet. But in those cases we expect that the associated communication resources (transmitter power, antenna size) will be even less ample, preserving the expected disproportion between the two.

Second, we note that establishing a communication link across interplanetary distance entails enacting several hardware configuration measures based on the presumed operational state of the remote communicating entity like:

- orienting a directional antenna correctly;
- tuning a transponder to pre-selected transmission and/or reception frequencies;
- 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 Massive State Retention

Like any reliable transport service employing ARQ, LTP is "stateful". In order to assure the reception of a block of data it has sent, LTP must retain for possible retransmission all portions of that block which 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.

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.

For this reason, LTP is based in part on a notion of massive state retention. Any number of requested transmissions may be concurrently "in flight" at various displacements along the link between two LTP engines, and the LTP engines must necessarily retain transmission status and retransmission resources for all of them. Moreover, if any of the data of a given block are lost en route, it will be necessary to retain the state of that transmission during an additional round trip while the lost data are retransmitted; even multiple retransmission cycles may be necessary.

In sum, LTP transmission state information persists for a long time because a long time must pass before LTP can be assured of transmission success - so LTP must retain a great deal of state information. Since the alternatives are non-reliability on the one hand and severe under-utilization of transmission opportunities on the other, we believe such massive statefulness is cost-justified (though probably not for all LTP applications).

3.1.1 Multiplicity of Protocol State Machines

This design decision is reflected in a significant structural difference between LTP and TCP.

Both TCP and LTP provide mechanisms for multiplexing access by a variety of higher-layer services or applications: LTP's "client service IDs" correspond to TCP's port numbers. Also, both TCP and LTP implement devices for encapsulating threads of retransmission protocol (protocol state machines): LTP's "sessions" functionally correspond to TCP connections. At any moment each such thread of retransmission protocol is engaged in conveying a single block of data from one protocol end-point to another.

However, a single TCP association (local host address, local port number, foreign host address, foreign port number) can accommodate at most one connection at any one time. In contrast, a single LTP association (local engine ID, local client service ID, foreign engine ID, foreign client service ID) can accommodate multiple concurrent sessions, one for each block of data in transit on the association.

3.1.2 Session IDs

In TCP, the fact that any single association is occupied by at most one connection at any time enables the protocol to use host addresses and port numbers to demultiplex arriving data to the appropriate protocol state machines. LTP's possible multiplicity of sessions per association makes it necessary for each segment of application data to include an additional demultiplexing token, a "session ID" that uniquely identifies the session in which the segment was issued and, implicitly, the block of data being conveyed by this session.

3.1.3 Use of Non-volatile Storage

Another important implication of massive statefulness is that implementations of LTP should consider retaining transmission state information in non-volatile storage of some kind, such as magnetic disk or flash memory.

Transport protocols such as TCP typically retain transmission state in dynamic RAM. If the device on which the software resides is rebooted or powered down then, although all transmissions currently in progress are aborted, the resulting degree of communication disruption is limited because the number of concurrent connections is limited. Rebooting or power-cycling a computer on which an LTP engine resides could abort a much larger number of transmission sessions in various stages of completion, at a much larger total cost.

3.2 Absence of Negotiation

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.

For this reason, LTP communication session parameters are asserted unilaterally, subject to application-level network management activity that may not take effect for hours, days, or weeks.

3.3 Partial Reliability

For environments where application data is not critical, overall link bandwidth utilization may be improved if the data is transmitted on a "best efforts" basis, i.e., without being subject to acknowledgment and retransmission. However, we believe that for many applications, unreliable transmission of data is likely to be useful only if any application headers/meta-data describing the actual data are received reliably. For example, suppose a block transmission involves a high-definition photograph (and can afford to be sent on "best efforts"): the first 40 bytes of the block might be a prologue containing information such as camera settings and time of exposure, without which the photograph data is useless, while the actual photograph data is an array of fixed-length scan lines. In this case the assured delivery of the first 40 bytes of the block is critical for interpreting the data, but the loss of a few individual scan lines may not be important enough to justify the cost of retransmission. A more typical example would be when the bundling protocol [BP] was the upper-layer protocol operating over LTP : if a bundle needs to be transmitted on "best efforts", it would at least be expected to have the bundle protocol header received reliably, or the bundle itself would be meaningless to the receiving bundling node.

The motivation for "partially reliable" transmission, as opposed to an alternative unreliable mode, is therefore to provide a mechanism for upper layer protocols to get any of their header and meta-data transmitted reliably (as necessary) but have the actual data transmitted unreliably. LTP regards each block of data as comprising two parts: a "red-part", whose delivery must be assured by acknowledgment and retransmission as necessary, and 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 association.

Note that in a red-green block transmission, the red-part data does NOT convey any urgency or higher-priority semantics relative to the block's green-part data; the red-part data is merely intended to carry imperative meta-data without which green-part data reception is likely to be futile.

3.4 Laconic Acknowledgment

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.)

One could imagine an LTP instance, upon being asked to transmit a block of data, searching through all existing sessions in hopes of finding one that was established upon reception of data from the new block's destination; transmission of the new block could be piggybacked on the acknowledgment traffic for that session. But the prevailing unidirectionality of space data communications means that such a search would frequently fail and a new unidirectional session would have to be established anyway. Session bidirectionality therefore seemed to entail somewhat greater complexity unmitigated by any clear performance advantage, so we abandoned it. Bidirectional data transfer is instead accomplished by opening two individual LTP sessions.

Since they are not piggybacked on data segments, LTP data acknowledgments - "reception reports" - are carried in a separate segment type. To minimize consumption of low and asymmetric transmission bandwidth in the IPN, these report segments are sent infrequently; each one contains a comprehensive report of all data received within some specified range of offsets from the start of the transmitted block. The expectation is that most data segments will arrive safely, so individual acknowledgment of each one would be expensive in information-theoretical terms: the real information provided per byte of acknowledgment data transmitted would be very small. Instead, report segments are normally sent only upon encountering explicit solicitations for reception reports - "checkpoints" - in the sequence of incoming data segments.

The aggregate nature of reception reports gives LTP transmission an

episodic character:

- "Original transmissions" are sequences of data segments issued in response to transmission requests from client services.

- "Retransmissions" are sequences of data segments issued in response to the arrival of report segments that indicate incomplete reception.

Checkpoints are mandatory only at the end of the red-part of each original transmission and at the end of each retransmission. For applications that require accelerated retransmission (and can afford the additional bandwidth consumption entailed), reception reporting can be more aggressive. Additional checkpoints may optionally be inserted at other points in the red-part of an original transmission, and additional reception reports may optionally be sent on an asynchronous basis during reception of the red-part of an original transmission.

3.5 Adjacency

TCP reliability is "end to end": traffic between two TCP endpoints may traverse any number of intermediate network nodes, and two successively transmitted segments may travel by entirely different routes to reach the same destination. The underlying IP network-layer protocol accomplishes this routing. Although TCP always delivers data segments to any single port in order and without gaps, the IP datagrams delivered to TCP itself may not arrive in the order in which they were transmitted.

In contrast, LTP is a protocol for "point to point" reliability on a single link: traffic between two LTP engines is expected not to traverse any intermediate relays. Point-to-point topology is innate in the nature of deep space communication, which is simply the exchange of radiation between two mutually visible antennae. No underlying network infrastructure is presumed, so no underlying network-layer protocol activity is expected; the underlying communication service is assumed to be a point-to-point link-layer protocol such as CCSDS Telemetry/Telecommand [TM][TC] (or, for terrestrial applications, PPP). The contents of link-layer frames delivered to LTP are always expected to arrive in the order in which they were transmitted, though possibly with any number of gaps due to data loss or corruption.

Note that building an interplanetary network infrastructure - the DTN-based architecture of the IPN - *on top of* LTP does not conflict with LTP design principles. Bundling functions as an overlay network protocol, and LTP bears essentially the same relationship to Bundling

that a reliable link protocol (for example, the ARQ capabilities of LLC) would bear to IP.

The design of LTP relies heavily on this topological premise, in at least two ways:

If two successively transmitted segments could travel by materially different routes to reach the same destination, then the assumption of rough determinism in timeout interval computation discussed below would not hold. Our inability to estimate timeout intervals with any accuracy would severely compromise performance; while spurious timeouts cause redundant retransmissions wasting precious bandwidth, overly conservative timeout intervals delay loss recovery.

If data arrived at an LTP engine out of transmission order, then the assumptions based on which the rules for reception reporting are designed would no longer hold. A more complex and/or less efficient retransmission mechanism would be needed.

3.6 Optimistic and Dynamic Timeout Interval Computation

TCP determines timeout intervals by measuring and recording actual round trip times, then applying statistical techniques to recent RTT history to compute a predicted round trip time for each transmitted segment.

The problem is at once both simpler and more difficult for LTP:

Since multiple sessions can be conducted on any single association, 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.

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, as discussed earlier.

3.7 Deferred Transmission

Link state cues also notify LTP when it is and isn't possible to transmit segments by passing them to the underlying communication service.

Continuous duplex communication is the norm for TCP operations in the Internet; when communication links are not available, TCP simply does not operate. In deep space communications, however, 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.

4. Overall Operation

4.1 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 to a remote client service instance. The sending engine opens a Sending State Record (SSR) for a new session, thereby

starting the session, and notifies the client service instance that the session has been started. The sending engine then initiates an 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 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.

At the next opportunity, subject to allocation of bandwidth to the queue into which the block data segments were written, the enqueued segments are transmitted 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.

On reception of the first data segment for the block, the receiving engine opens a Receiving State Record (RSR) for the new session 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.

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, notifies the local client service instance that the red-part of the block has been successfully transmitted, and closes the SSR for the session.

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

The receiving engine receives the report-acknowledgment segment,

turns off the timer for the report segment, and closes the RSR for the session.

Closing both the SSR and RSR for a session terminates the session.

4.2 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:

Rather than returning a single reception report indicating complete reception, the receiving engine returns a reception report comprising as many report segments as are needed in order to report in detail on all red-part data reception for this session (other than data reception that was previously reported in response to any discretionary checkpoints, described later), within the constraints on maximum segment size imposed by the underlying communication service. [Still, only one report segment is normally returned; multiple report segments are needed only when a large number of segments comprising non-contiguous intervals of red-part block data are lost or when the receiver-to-sender path MTU is small.] A timer is started for each report segment.

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 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 indicate 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 and closes the SSR for the session.

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

It first turns off the timer for the referenced report segment.

It then 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, within the constraints on maximum segment size imposed by the underlying communication service; 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, closes the RSR for the session. Otherwise the data retransmission cycle continues.

Loss of any checkpoint segment or of the responsive report segment causes the checkpoint timer to expire. When this occurs, the sending engine normally retransmits the checkpoint segment. Similarly, loss of a report segment or of the responsive report-acknowledgment segment causes the report segment's timer to expire. When this occurs, the receiving engine normally retransmits the report segment.

Note that reception of a previously received report segment that was retransmitted due to loss of the report-acknowledgment segment causes another responsive report-acknowledgment segment to be transmitted, but is otherwise ignored; if any of the data retransmitted in response to the previously received report segment were lost, further retransmission of those data will be requested by one or more new report segments issued in response to that earlier retransmission's 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 it retransmits missing data in response to reception reports indicating incomplete reception, while the receiver additionally

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 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 that 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.

4.3 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 [Sec 6]. The reception reports generated for accelerated retransmission are processed in exactly the same way as the standard reception reports.

4.4 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 normally retransmits the cancellation segment.

5. Functional Model

The functional model underlying the specification of LTP is one of deferred, opportunistic transmission, with access to the active transmission link apportioned between two (conceptual) outbound traffic queues. The accuracy of LTP retransmission timers depend in large part on a faithful adherence to this model.

5.1 Deferred Transmission

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.

5.2 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.

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 [Sec 5.1] 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 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.

6. Tracing LTP back to CFDP

LTP in effect implements most of the "core procedures" of the CCSDS File Delivery Protocol (CFDP) specification, minus flow labels and all features that are specific to operations on files and filestores (file systems). In the IPN architecture we expect that file and filestore operations will be conducted by file transfer application protocols -- notably CFDP itself -- operating on top of the DTN Bundling protocol. Bundling's QoS features serve the same purposes as CFDP's flow labels, so flow labeling is omitted from LTP. Bundling in effect implements the CFDP "extended procedures" in a more robust and scalable manner than is prescribed by the CFDP standard.

The fundamental difference between LTP and CFDP is that, while CFDP delivers named files end-to-end, LTP is used to transmit arbitrary, unnamed blocks of data point-to-point.

Some differences between LTP and CFDP are simply matters of terminology; the following table summarizes the correspondences in language between the two.

LTP	CFDP
LTP engine	CFDP entity
Segment	Protocol Data Unit (PDU)
Reception Report	NAK
Client service request	Service request
Client service notice	Service indication

CFDP specifies four mechanisms for initiating data retransmission, called "lost segment detection modes". LTP effectively supports all

four:

"Deferred" mode is implemented in LTP by the Request flag in the EORP data segment, which triggers reception reporting upon receipt of the EORP.

"Prompted" mode is implemented in LTP by turning on Request flags in data segments that precede the EORP; these additional checkpoints trigger interim reception reporting under the control of the source LTP engine.

"Asynchronous" mode is implemented in LTP by the autonomous production, under locally specified conditions, of additional reception reports prior to arrival of the EORP.

"Immediate" mode is simply a special case of asynchronous mode, where the condition that triggers autonomous reception reporting is detection of a gap in the incoming data.

CFDP uses a cyclic timer to iterate reception reporting until reception is complete. Because choosing a suitable interval for such a timer is potentially quite difficult, LTP instead flags the last data segment of each retransmission as a checkpoint, sent reliably; the cascading reliable transmission of checkpoint and RS segments assures the continuous progress of the transmission session.

As the following table indicates, most of the functions of CFDP PDUs are accomplished in some way by LTP segments.

----------LTP----------	----------CFDP----------
Data segments	File data and metadata PDUs
Flags on data segments	EOF (Complete), Prompt (NAK), Prompt (Keep Alive)
Report segment	ACK (EOF Complete), NAK, Keep Alive, Finished (Complete)
Report-acknowledgment	ACK (Finished Complete)
Cancel segment	EOF (Cancel, Protocol Error) Finished (Cancel, Protocol Error)
Cancellation Acknowledgment	ACK (EOF (Cancel, Protocol Error), Finished (Cancel, Protocol Error))

But some CFDP PDUs have no LTP equivalent because in an IPN

architecture they will likely be implemented elsewhere. CFDP's EOF (Filestore error) and Finished (Filestore error) PDUs would be implemented in an IPN application-layer file transfer protocol, e.g., CFDP itself. CFDP's Finished [End System] PDU is a feature of the Extended Procedures, which would in effect be implemented by the Bundling protocol.

7. Security Considerations

Not relevant for this document.

8. IANA Considerations

Not relevant for this document. Please follow the IANA Considerations sections of the internet-drafts on the series (main protocol specification and protocol extensions).

9. 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.

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.

10. References

10.1 Normative References

[B97] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[LTP] Ramadas, M., Burleigh, S., and Farrell, S., "Licklider Transmission Protocol - Specification", draft-irtf-dtnrg-ltp-03.txt

(Work in Progress), July 2005.

[LTPEXT] Farrell, S., Ramadas, M., and Burleigh, S., "Licklider Transmission Protocol - Extensions", draft-irtf-dtnrg-ltp-extensions-01.txt (Work in Progress), July 2005.

10.2 Informative References

[BP] K. Scott, and S. Burleigh, "Bundle Protocol Specification", Work in Progress, October 2003.

[CCSDS] Consultative Committee for Space Data Systems web page, "http://www.ccsds.org".

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

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

[TM] Packet Telemetry Specification. Recommendation for Space Data System Standards, CCSDS 103.0-B-2 BLUE BOOK Issue 2, June 2001.

[TC] Telecommand Part 2 - Data Routing Service. Recommendation for Space Data System Standards, CCSDS 202.0-B-3 BLUE BOOK Issue 3, June 2001.

[ECS94] D. Eastlake, S. Crocker, and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[SCTP] R. Stewart et al, "Stream Control Transmission Protocol", RFC 2960, October 2000.

11. Author's Addresses

Scott C. Burleigh

Jet Propulsion Laboratory
4800 Oak Grove Drive
M/S: 179-206
Pasadena, CA 91109-8099
Telephone +1 (818) 393-3353
FAX +1 (818) 354-1075
Email Scott.Burleigh@jpl.nasa.gov

Manikantan Ramadas
Internetworking Research Group
301 Stocker Center
Ohio University
Athens, OH 45701
Telephone +1 (740) 593-1562
Email mramadas@irg.cs.ohiou.edu

Stephen Farrell
Distributed Systems Group
Computer Science Department
Trinity College Dublin
Ireland
Telephone +353-1-608-3070
Email stephen.farrell@cs.tcd.ie

12. Copyright Statement