Possible items for an LTP corrigendum:

1. A note on the suitability of LTP for ultra-high-speed implementations

LTP was designed to be highly bit-efficient; part of the design includes the use of self-delimiting numeric values (SDNVs) in some of the protocol fields.  While this allows LTP to maintain bit efficiency over a wide range of operational environments (e.g. when sending different sized LTP blocks), it results in headers where the size MAY vary from segment to segment and that require parsing multiple SDNVs.  This makes it difficult to implement LTP in high-speed environment or extremely resource-constrained environments where the LTP processing is moved to field programmable gate arrays (FPGAs) or other hardware accelerators.

To address the performance issues at high rates, CCSDS is developing another optionally-reliable link layer shim that provides many of the same features as LTP but using a header format that allows for the determination of all of the field sizes by reading a single byte in the header.  This will facilitate FPGA/ASIC implementation of an LTP-like high-speed reliable mechanism suitable for near-Earth or lunar optical communications (or other high-rate links), while LTP may continue to be suitable for lower-rate high-delay links such as deep space links. While LTP and this new protocol will not be directly interoperable, they are expected to be used as converge layers under the Bundle Protocol in a delay tolerant network. Thus, coexistence of both protocols in different links of the same delay tolerant network is possible.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

2. Note on Session Closure and Removal of Session Information

The state associated with an LTP session should be removed by the receiver when the report confirming reception of the red-part has been acknowledged.  In particular, if the block also contains green-part data, no persistent state information should be kept once the report reporting reception of the red part has been acknowledged.  This is necessary because the End-of-Block (EOB) marker will fall on a Green segment and hence will be unreliable.  If the EOB marker is lost and the receiver still has reception state for the session, there will be no protocol mechanisms to clear that state.

3. Clarification on Reception of Green-Part Data

Green-part data reception should be stateless at the receiving LTP engine.  Green-part segments should be delivered to the LTP client at the earliest opportunity and implementations should maintain minimal state for green LTP sessions. If implementers wish to implement green-part LTP in conjunction with link suspension, they should take care to reuse the session identifiers which were utilized prior to suspension.

4. Kiyo had a description of a session closure issue with the state machine.

We should say something about that.  I think it was that once the sender sent the report acknowledgement for the report covering the entire block that the sender would then close the session.  If the report acknowledgement is lost, the receiver will retransmit the report but the sender, on receiving the report segment, will not have an extant session to match it to and will ignore it.  The receiver will thus continue to emit report segments until it runs out of retransmission attempts.  [Kiyo: was it this or something else?]  I don't think we can propose a normative fix for this without reopening the book and doing interoperability testing (which I don't think we want), but we might suggest that implementations do something like keep enough information from terminated sessions (for a while) to respond to such events.

-------------------------
(1) It'd be nice to have some data about the total amount of overhead to send blocks of sizes (100, 1000, ...) to throw in here.

5. Clarification on Asynchronous Reception Reports

Receiver implementations may send an asynchronous reception report if the last received red segment completed the red part of a session’s block even if the received segment is not a checkpoint. This reduces the number of unnecessary retransmissions when segments are received out-of-order, which may cause the last checkpoint to be received before other data segments that departed before the checkpoint.

6. Automatic Cancellation of Idle Sessions

LTP sender implementations may automatically cancel sessions if a long enough time interval has elapsed without reception of a report segment corresponding to checkpoint segment. This time interval may be calculated as one plus the number of allowable transmission problems set by management at the transmit LTP engine, times the value of the start Checkpoint Timer, plus an optional time offset.

LTP receiver implementation may automatically cancel sessions if a long enough time interval has elapsed without reception of a report acknowledgement segment corresponding to a report segment. This time interval may be calculated as one plus the number of allowable transmission problems set by management at the receive LTP engine, times the value of the RS Timer, plus an optional time offset.

LTP sender and receiver implementations may notify the client service of this automatic cancellation event by using the reason-code RLEXC, if the time interval for automatic retransmission was set based on that parameter, or using the reason-code USR\_CNCLD otherwise.

7. LTP Service Data Aggregation

LTP Service Data Aggregation allows to concatenate several LTP blocks into a single LTP block by concatenating SDA Client Data Capsule. These consist of a single SDNV contain the LTP client service ID and the complete client data unit as passed to SDA. However, no generic way to determine the length of the client data unit at the receiving side is specified, this mechanism is not interoperable unless additional agreements on the structure or the exact types of client data and their respective client service IDs are agreed. It is expected that Service Data Aggregation will be removed from LTPv2 and that eventual aggregation shall be performed by upper layers if deemed necessary.

8. Deprecation of LTP Security

Can we remove the entire section 3.9 of the current spec. RFC 5326 (LTP spec) does not require RFC 5327 (LTP security) anywhere, so we could simply say that implementations compliant with the CCSDS profile do not have to implement RFC 5327.

9. Deprecation of the Service Data Aggregation

10. Discourage use of mixed sessions

11. Out-of-order delivery of Segments/Blocks