Working Group Draft for TCPCLv4

Brian Sipos

RKF Engineering Solutions

IETF104

Motivations for Updates to TCPCL

1. During implementation of TCPCLv3, Scott Burleigh found an ambiguity in bundle acknowledgment and refusal.
2. For use in a terrestrial WAN, author has a need for TLS-based authentication and integrity. TCPCLv3 mentions TLS but does not specify its use. IETF strongly in favor of TLS for new general-use protocols.
3. Reduced sequencing variability from TCPCLv3
4. Adding extension capability for TCPCL sessions and transfers.

Goals for TCPCLv4

- Do not change scope or workflow of TCPCL.
  - As much as possible, keep existing requirements and behaviors. The baseline spec was a copy-paste of TCPCLv3.
  - Still using single-phase contact negotiation, re-using existing headers and message type codes.
  - Allow existing implementations to be adapted for TCPCLv4.

Last Draft Edits

- Changes are in draft-ietf-dtn-tcpclv4-11.
- Removed separate XFER_INIT message and moved transfer extension items into first XFER_SEGMENT message (when START bit is set).
  - This avoids overhead of extra message and simplifies message sequencing logic.
  - The transfer Total Length has been moved into an extension item (further discussion in later slides).
- Reduced total extension list length from 64-bit to 32-bit.
  - Strong guidance provided in spec to limit the size of extension items.
  - This still allows “large” extensions (for some relative amount of largeness).
- Clarified default and minimum session timeout behaviors.
  - Restored recommended default from TCPCLv3.
- Added a “reply” marking to SESS_TERM message to avoid trivial feedback loop.
  - Now a termination initiation is distinguishable from its acknowledgement.
- Removed encoding variability in SESS_TERM reason code.
  - An “unknown” code is used where previously there was no encoded value.

Minimal TCPCLv4 Implementation

- In the case where a user wants to achieve least-overhead on a reliable private network:
  - No TLS use, no EID exchange, no extensions
  - Always single-segment transfers
- Sequence:
  - Contact header (each direction): 6 octets
  - SESS_INIT (each direction): 25 octets
  - XFER_SEGMENT out: 22 octets + bundle size
  - XFER_ACK in: 18 octets
  - SESS_TERM (each direction): 2 octets
- Overhead for session: 33 octets
- Overhead for each transfer: 40 octets

Transfer Length Extension

- The total length of a segmented transfer is now included in an extension item.
- Moving this data from (removed) XFER_INIT message to extension item saved 2 octets.
  - XFER_INIT+XFER_SEGMENT overhead was 37 octets, now 35 octets when the Transfer Length extension is used.

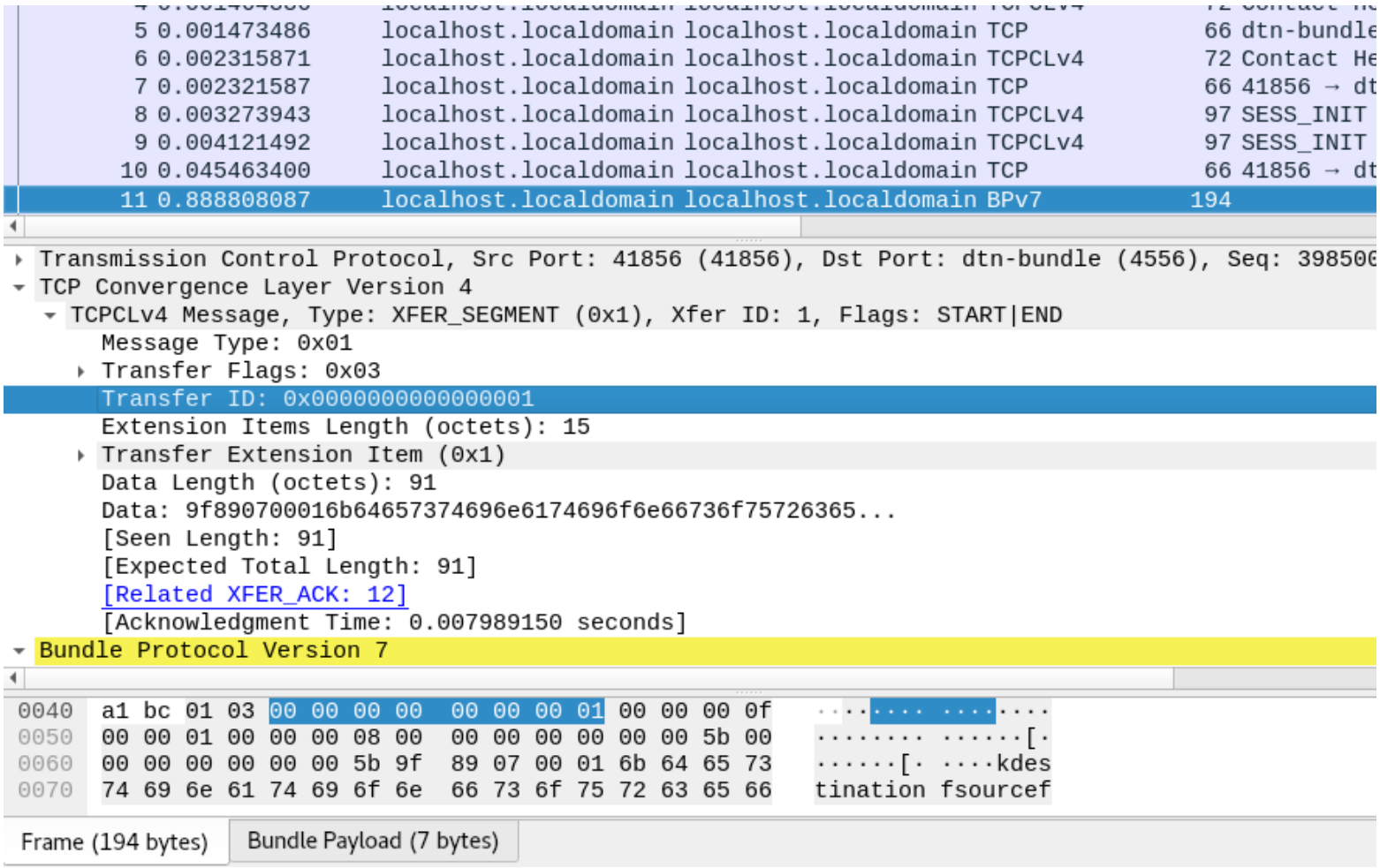
Demo CL Agent Changes

- The python example agent has been updated to follow new -11 message sequencing.
- New behaviors:
  - Agents are not fully bidirectional and D-Bus controlled to allow multiple sessions both incoming and outgoing.
  - Performs graceful SESS_TERM sequencing on `KeyboardInterrupt` (Ctrl+C) or D-Bus command.
  - Implemented segment-scaling algorithm to target a desired time-to-acknowledge as a proof of concept.
- Also implemented random message generator to exercise demo agent and wireshark plugin.

New Wireshark Dissectors

- For TCPCLv4:
  - Decodes Contact Header and all defined Message types.
  - Handles TLS in sessions.
  - Decodes session and transfer extension items.
  - Performs several sequence checks with warnings.
  - Performs SEGMENT--ACK cross-linking and timing.
  - Reassembles segments of a transfer into a single data block.
  - Validates CBOR decoding of the bundle content.
- For BPv7:
  - Verifies proper bundle header/footer.
  - Decodes primary and canonical blocks.
  - Decodes type-specific data defined in the core spec.
  - Ran into issues with CRC use, may need to clarify in BP spec.

Wireshark Screenshot



Open Issues from Feedback

- Concern about allowed extension item encodings.
  - Currently the Extension Item data Length field is 32-bit.
  - This is oversized from minimum expected use.
  - This also avoids any possible issue with large extension items.
  - Is it worth shaving octets to possibly run into size-overflow issues?
    - An Extension Item Length of 16-bits could be used with more complex multiple-item sequencing to implement larger data payloads.
  - Are we concerned with two octets in an optional mechanism?

Way Forward for TCPCLv4

- Further set of editorial changes to fix some typos and to include type/reason codes in the spec body tables (not just in the IANA tables).
- Working implementation is available for interoperability testing
  - Implemented in `scapy/python` for ease of understanding.
  - Handles concurrent sessions and asynchronous socket events.
  - Does not implement BP agent behavior, only CL behavior.
- Working Wireshark protocols for troubleshooting implementations and analyzing traffic.
  - These supersede the “Bundle” protocols in stock Wireshark 2.6