

# Working Group Draft for TCPCLv4

---

Brian Sipos

RKF Engineering Solutions

IETF104

A solid orange horizontal bar at the bottom of the slide.

# 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

The screenshot displays a packet capture in Wireshark. The top pane shows a list of packets, with packet 11 selected. The middle pane shows the details of this packet, and the bottom pane shows the raw packet bytes.

No.	Time	Source	Destination	Protocol	Length
5	0.001473486	localhost.localdomain	localhost.localdomain	TCP	66
6	0.002315871	localhost.localdomain	localhost.localdomain	TCPCLv4	72
7	0.002321587	localhost.localdomain	localhost.localdomain	TCP	66
8	0.003273943	localhost.localdomain	localhost.localdomain	TCPCLv4	97
9	0.004121492	localhost.localdomain	localhost.localdomain	TCPCLv4	97
10	0.045463400	localhost.localdomain	localhost.localdomain	TCP	66
11	0.888808087	localhost.localdomain	localhost.localdomain	BPv7	194

Transmission Control Protocol, Src Port: 41856 (41856), Dst Port: dtn-bundle (4556), Seq: 398506

TCP Convergence Layer Version 4

- TCPCLv4 Message, Type: XFER\_SEGMENT (0x1), Xfer ID: 1, Flags: START|END
  - Message Type: 0x01
    - Transfer Flags: 0x03
      - Transfer ID: 0x0000000000000001
      - Extension Items Length (octets): 15
        - Transfer Extension Item (0x1)
          - Data Length (octets): 91
          - Data: 9f890700016b64657374696e617469666e66736f75726365...
          - [Seen Length: 91]
          - [Expected Total Length: 91]
          - [\[Related XFER\\_ACK: 12\]](#)
          - [Acknowledgment Time: 0.007989150 seconds]

## Bundle Protocol Version 7

0040	a1 bc 01 03 00 00 00 00 00 00 00 01 00 00 00 0f	.....
0050	00 00 01 00 00 00 08 00 00 00 00 00 00 00 5b 00	.....[.
0060	00 00 00 00 00 00 5b 9f 89 07 00 01 6b 64 65 73	.....[. ....kdes
0070	74 69 6e 61 74 69 6f 6e 66 73 6f 75 72 63 65 66	termination fsourcef

Frame (194 bytes) Bundle Payload (7 bytes)

# 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