Overview

• Motivations and goals for change
• Individual protocol changes
• Status of proposed changes
• Path forward for TCPCL
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, I have a need for TLS-based authentication and integrity. TCPCLv3 mentions TLS but does not specify its use.

3. Contact negotiation in TCPCLv3 is limited and relatively hard-coded.

4. Allow an endpoint to positively reject a message (rather than simply ignoring it).
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.

• Re-use existing encoding, type and reason codes.
  ◦ Avoid duplication of IANA assignments.
  ◦ Since workflow is preserved, majority of message types are retained.
  ◦ This inherits limitations from TCPCLv3 for the sake simpler implementation changes.
Bundle Identification Problem

• In TCPCLv3 there is strict ordering of bundles (no interleaving of segments from different bundles) but there is no identification of individual bundles.

• It is possible for a TCPCL transmitter to send 2+ bundles before receiving any ACK/REFUSE, and it is also possible for a receiver to send multiple REFUSE messages for the same bundle.

• The transmitter has no way to correlate refusals to particular bundles.
Bundle Identification Solution

• In TCPCLv4 each bundle-related message (LENGTH, DATA_SEGMENT, ACK_SEGMENT, REFUSE_BUNDLE) now includes a 'unique' Bundle Identifier SDNV.

• This allows exact correlation of all messages related to the 'same' bundle transfer, and avoids the problem ambiguity.

• Bit-length and reuse of bundle IDs is left to implementation. Spec provides some guidance.

• Goal of TCPCLv4 is to avoid overhead while still disambiguating bundles.
Transport Security Problem

• In TCPCLv3 spec, there is only one statement regarding transport security:

   Nothing in TCPCL prevents the use of the Transport Layer Security (TLS) protocol [RFC5246] to secure a connection.

• Possible interpretations of this statement:
  ◦ Use same TCP port number – have interoperability problems
  ◦ Use different port number – either non-standard port or have “duplicate” port assignments for one protocol
Transport Security Solution

• Same issue has come up for HTTP, FTP, LDAP, SMTP, POP3, IMAP, etc. in the past.

• RFC 7605 *Recommendations on Using Assigned Transport Port Numbers*, Section 7.4 states “The overall preference is for use of a single port…”

• TCPCLv4 takes same method as LDAP, SMTP, IMAP via in-band “STARTTLS” upgrade message.

• This behavior allows transport security to be negotiated within a TCPCL contact.

• Each TCPCL endpoint can apply its own security policies to the contact (e.g. allow or disallow insecure use).

• Goal of TCPCLv4 is to avoid reliance on TLS, allow endpoints to ignore or negotiate its use.
Secured Negotiation

• Use of STARTTLS in TCLCLv4 follows existing best practices; contact header negotiation is repeated after connection is secured.

• This adds some connection establishment overhead, but no differently than other widely deployed and well-used protocols.

• This avoids the statement in TCPCLv3 that the contact header Endpoint ID is not to be trusted.
Parameter Negotiation Problem

- In TCPCLv3, there is a fixed-width bit field for negotiation of contact parameters.
- This mechanism must be extended for TCPCLv4 needs, and is not extensible for future or network-specific needs.
- Negotiation in TCPCLv3 sets parameters for both directions of the connection, but some really apply to receiver-side only. TCPCLv4 will clarify the scope of each parameter.
- Goal of TCPCLv4 is to provide no loss of fidelity when negotiating connection parameters.
Parameter Negotiation Solution

• Follow established behavior of PPP negotiation, using type-length-value (TLV) parameters.
  ◦ TCPCLv4 still uses single-phase negotiation, does not use multiple-phase negotiation of PPP. Only one contact header message is sent.

• This has several benefits:
  ◦ More refined negotiation options than Boolean “enable/disable”. Current spec has IGNORE/ALLOW/REQUIRE for message handling negotiation.
  ◦ Each parameter is optional. If all parameters of an endpoint are non-negotiable then TCPCLv4 contact header is actually shorter than TCPCLv3 header.
  ◦ Allows network-specific parameters to be added with no change to TCPCL proper. Extensibility!
Parameters to Negotiate

• Same as TCPCLv3:
  ◦ Provide EID
  ◦ Keepalive time interval

• Changed in TCPCLv4:
  ◦ Determine use of LENGTH, ACK_SEGMENT, and REFUSE_BUNDLE messages (now uni-directional)

• New in TCPCLv4:
  ◦ Supported Bundle Protocol versions
  ◦ Maximum RX segment size
  ◦ TLS support
Message Rejection Problem

• In TCPCLv3 if an endpoint receives an unknown or unexpected message, the only recourse is to ignore it.

• This has some implications for interoperability and troubleshooting:
  ◦ A transmitting endpoint has no way to determine whether or not a sent message
Message Rejection Solution

• TCPCLv4 adds a new “REJECT” message to allow an endpoint to signal an invalid message reception.
  ◦ Important point: this is not required behavior.
  ◦ A minimal implementation on closed network can avoid this messaging.

• Includes reason code for rejection
  ◦ Can be either: not understood or not expected (in workflow)

• Current motivation is to allow rejection of STARTTLS messages.
  ◦ Rather than having a distinct “TLS reject”, this is simply a generic “message reject”.
TCPCL Protocol Versioning

- There is a worrisome TCPCLv3 requirement, from RFC 7242: *If a node receives a contact header containing a version that is greater than the current version of the protocol that the node implements, then the node SHOULD interpret all fields and messages as it would normally.*

- This “forward compatibility” effectively disallows changes to message formats.
  - There is some benefit to this behavior, but the message type is only four-bits (16 types) so new message types are expensive.

- This requirement was removed from TCPCLv4, but remains in TCPCLv3.
  - This proposed draft supersedes RFC 7242 anyway.
  - TCPCLv4 could change header “magic” string if deemed necessary.

- Thoughts from WG members?
Bundle Protocol Versioning

• The current draft TCPCLv4 allows endpoints to negotiate use of specific BP versions.

• BPbis adds additional wrinkle of different BPv6 encoding formats.

• How should TCPCL (or any other CL) handle this?

• BBbis can add an IANA registry of enumerated encoding formats.
  ◦ CBOR would be first entry in registry.

• Is encoding negotiated per-connection or per-bundle?
Protocol Status

• Current draft spec should be complete enough to review for content and editorial changes
  ◦ May need to remove more vestigial statements, especially in Section 7 “Security Considerations”.
  ◦ URL: https://tools.ietf.org/html/draft-sipos-dtn-tcpclv4-01

• A rough but usable implementation is being worked on GitHub
  ◦ Currently used for prototyping new behaviors, not one-for-one with draft specification.
  ◦ URL: https://github.com/BSipos-RKF/dtn-bpbis-tcpcl
Working Group Adoption

• Current spec truly intended as a rough draft to allow implementing and bashing on actual requirements.

• Any objections to proposed changes?

• Any usefulness to BPbis approval by IESG?