draft-piraux-tcpls

TCPLS: Modern Transport Services with TCP and TLS

Maxime Piraux, Olivier Bonaventure,
UCLouvain
Florentin Rochet
University of Edinburgh

NGI POINTER

Content

- Introduction
- Using TLS for transport protocol extensibility
- Opportunities for the transport stack
- The TCPLS protocol
- Conclusion

The design of MPTCP

- In 2009, the mptcp WG formed with an initial design involving TCP Options
- In 2013, v0 shipped and enabled
  - Bandwidth aggregation of several TCP subflows
  - Failover in case of network failure
  - Backwards compatibility with TCP
- MPTCP has several issues
  - Address exchange is not secure, improved in MPTCP v1
  - TCP is prone to middlebox interference
  - Can be difficult to implement
    - 7-year journey from specification to mainline Linux

The design of QUIC

- In 2016, the quic WG formed to design an UDP-based transport protocol
- In 2021, QUIC v1 shipped and enabled:
    - Stream multiplexing
    - Connection migration, failover
- TLS secures most of the QUIC header and all QUIC payloads
- QUIC can be implemented in user-space and shipped with applications

Using TLS for transport protocol extensibility

- TLS is the most used protocol atop TCP
- TLS version 1.3 used encryption to extend the protocol
  - Encrypted TLS records and Encrypted Extensions allows securely exchanging control and application data
- TCP support in the network and in operating systems remains wider
- **Given the ubiquity of TLS, can we provide new transport services with TCP and TLS ?**

Opportunities for the transport stack

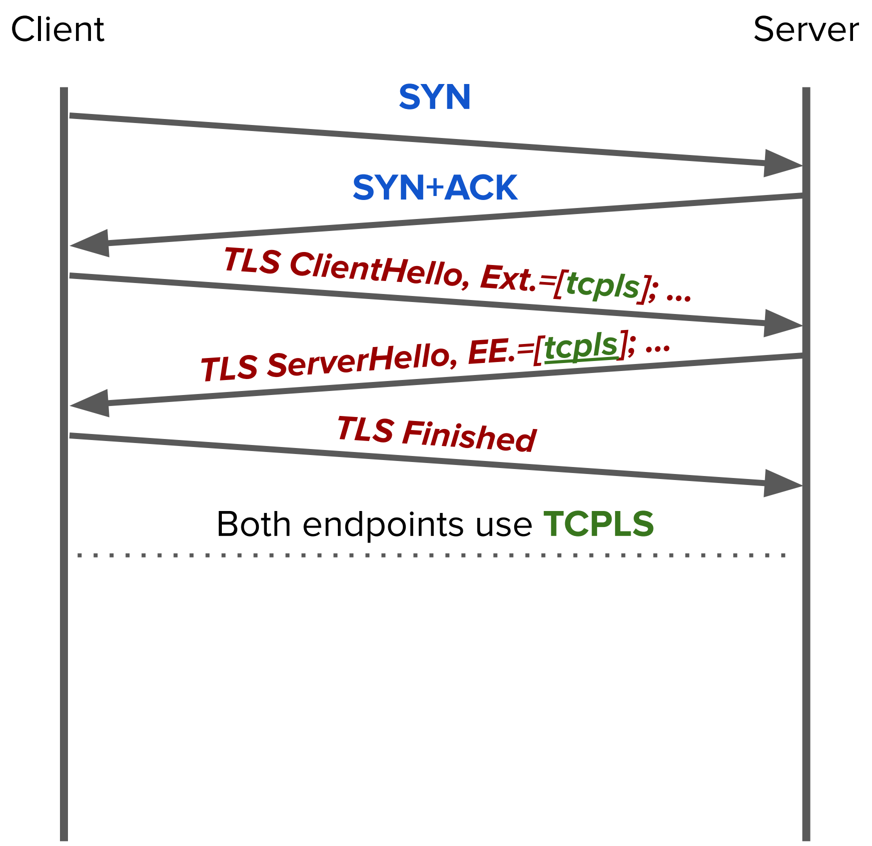
- Build an encrypted transport protocol
  - Stream multiplexing
    - App-chosen HoL blocking resilience
  - Connection Migration
    - Based on app triggers and network conditions
  - Multipath
    - Scheduling at the TLS record level
- More efficient than the HTTP/2+TLS+MPTCP stack
  - Built on a strict layering assumption
- Clean slate for other transport extensions

The TCPLS protocol

- Session establishment
- Exchanging application and control data
- Adding TCP connections
- Record acknowledgements
- Modern Transport Services
  - Stream multiplexing
  - Failover
  - Bandwidth aggregation

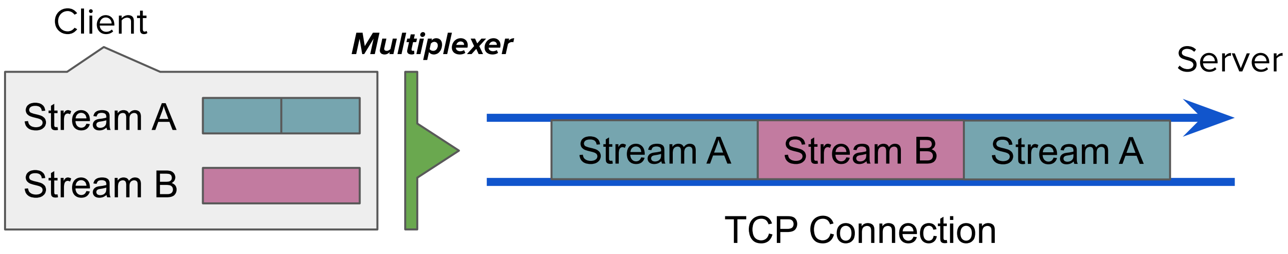
Session establishment

- TCPLS does not modify the TCP and TLS handshake
- ***tcpls*** is a TLS Extension indicating the support of TCPLS
- Compatible with TCP TFO and TLS 0-RTT Handshake



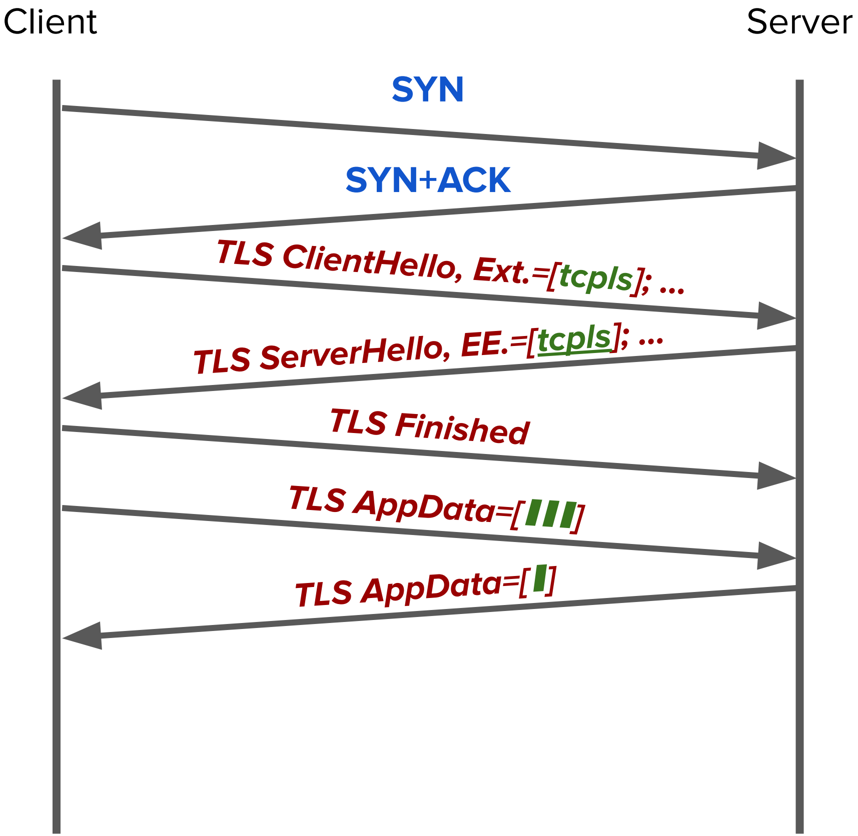
Stream multiplexing

- Streams provide concurrent bytestreams to applications
- TCPLS manages the streams and multiplexes them

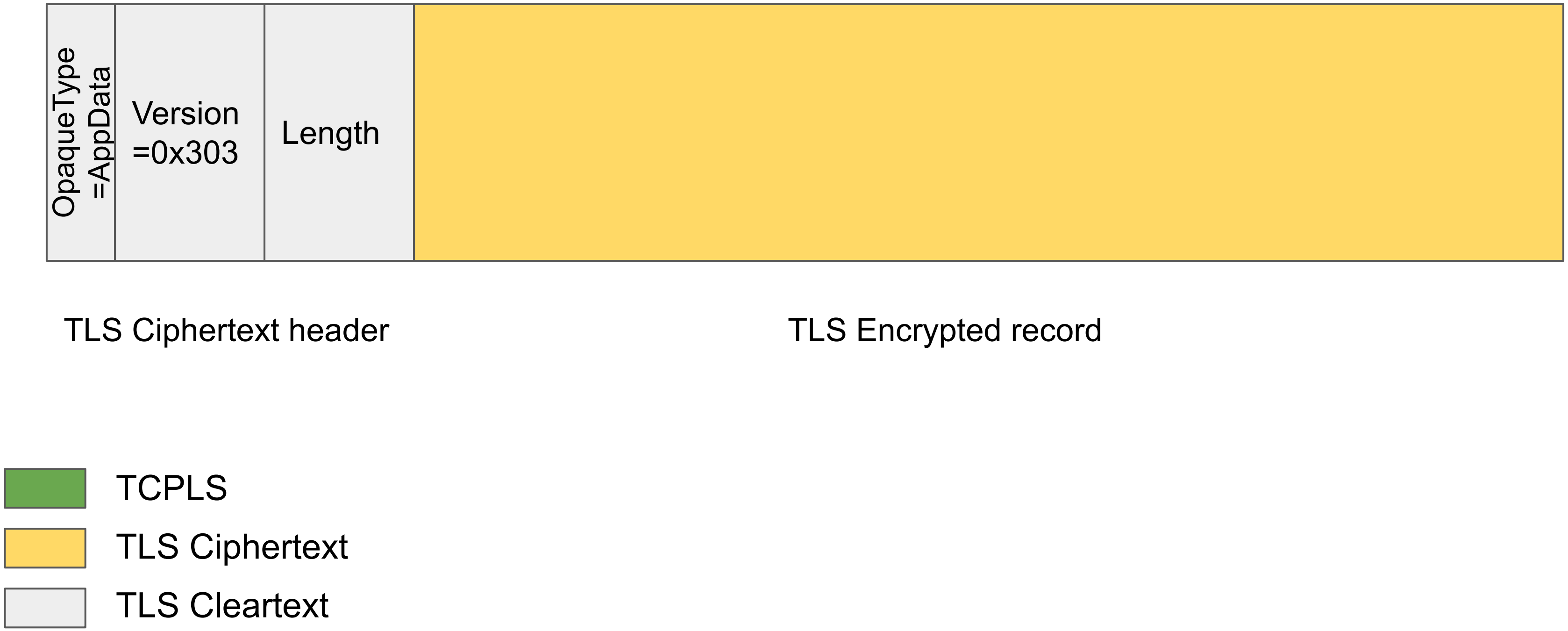


Exchanging data

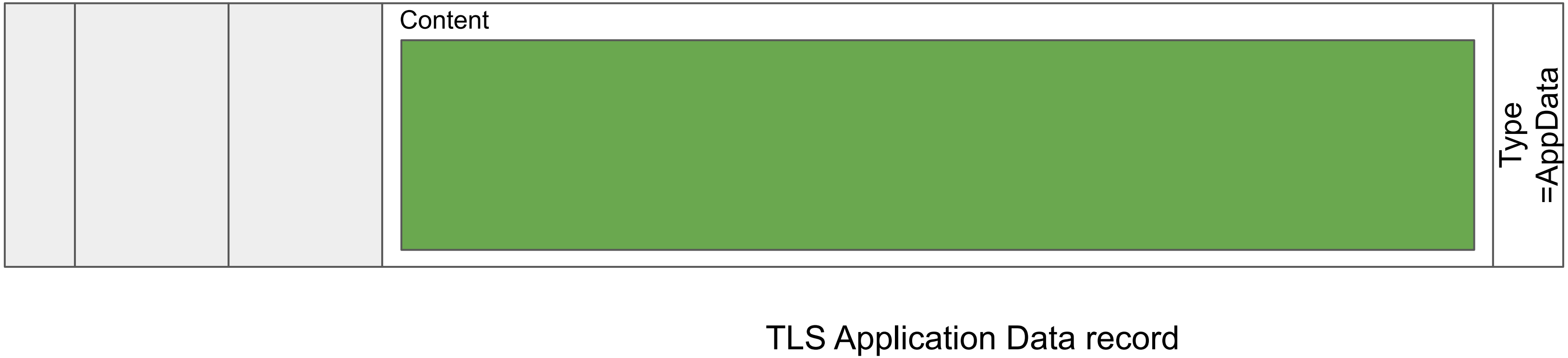
- Application and control data can then be sent in TLS encrypted records using TCPLS frames
- Frames compose TLS records



Example: A TLS record containing a TCPLS Stream frame



Example: A TLS record containing a Stream frame

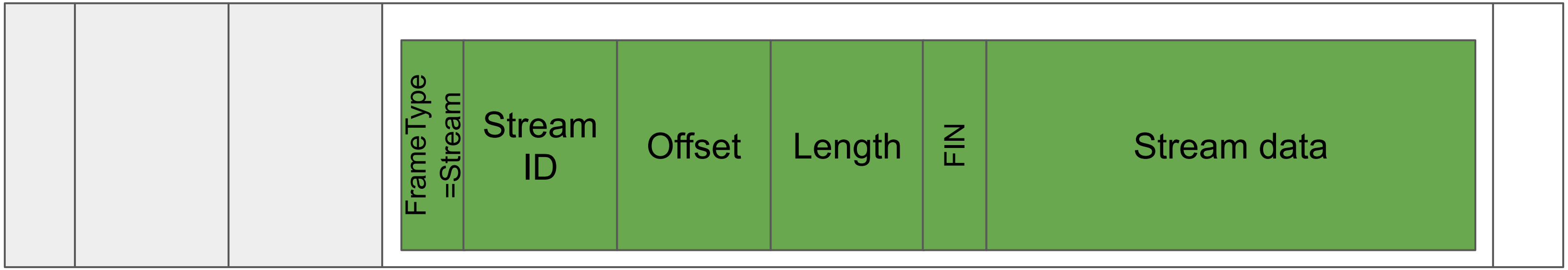


TCPLS

TLS Ciphertext

TLS Cleartext

Example: A TLS record containing a Stream frame



Stream frame

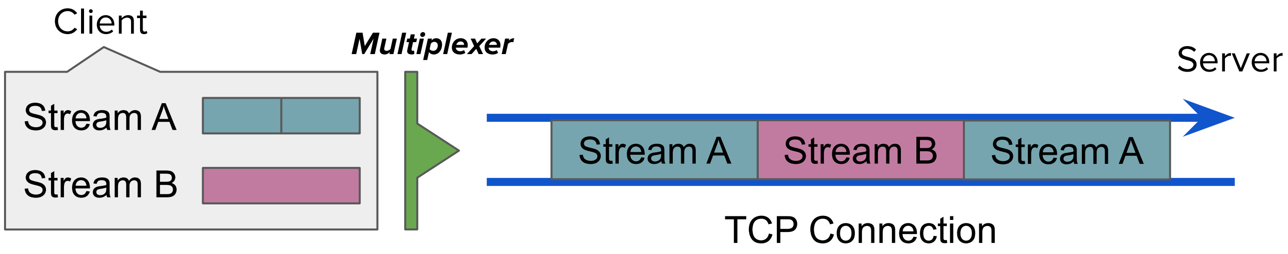
TCPLS

TLS Ciphertext

TLS Cleartext

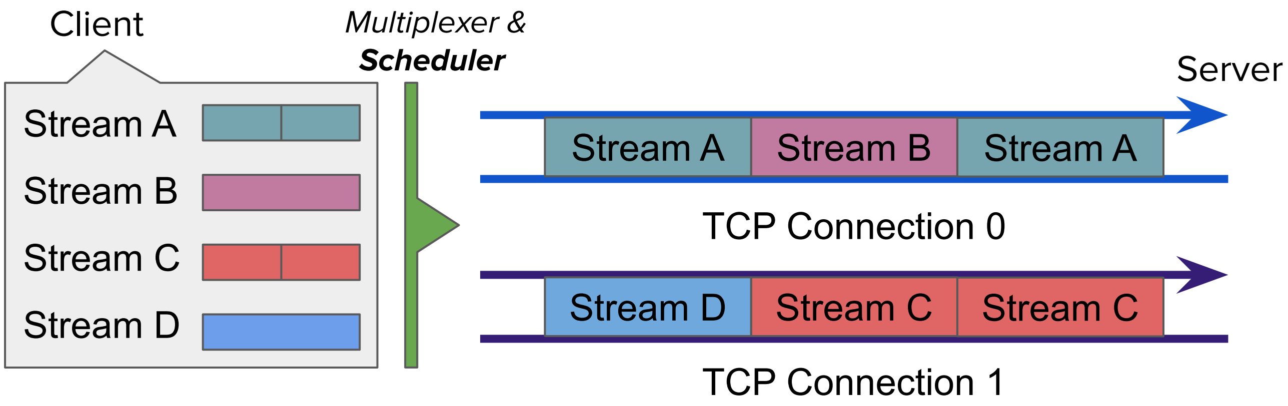
Stream multiplexing

- Streams provide concurrent bytestreams to applications
- TCPLS manages the streams and multiplexes them
- Streams multiplexed on a single connection are subject to HoL blocking



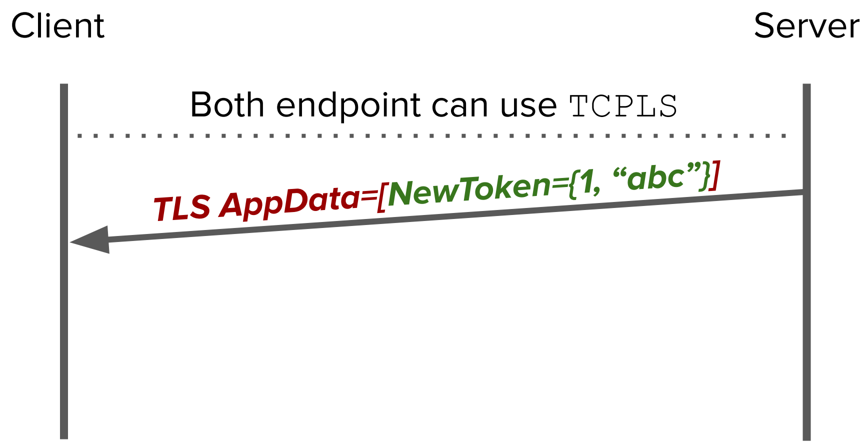
Stream multiplexing

- TCPLS manages TCP connections and schedules the TLS records
- By mapping streams to connections, the app choose the streams it wants to protect, and the ones that are bound together



Adding TCP connections

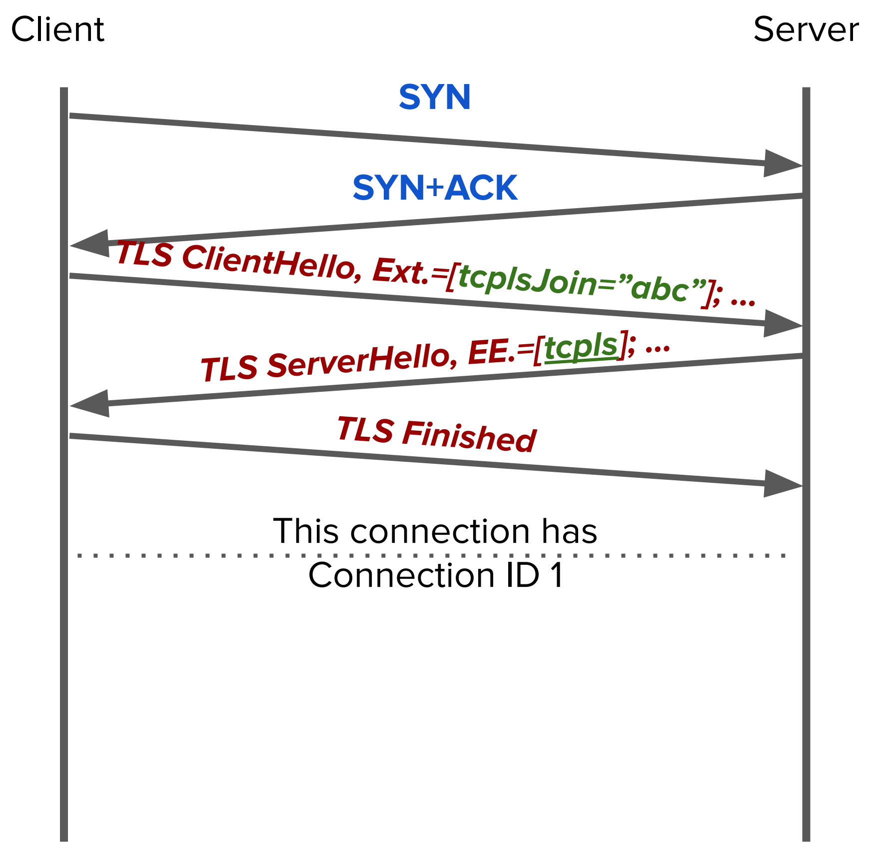
- Server gives tokens to the client
- Each token can be used by the client to open and join an additional TCP connection
- Server can limit the connections by limiting the tokens
- The Sequence number of the Token becomes the Connection ID



New Token frame

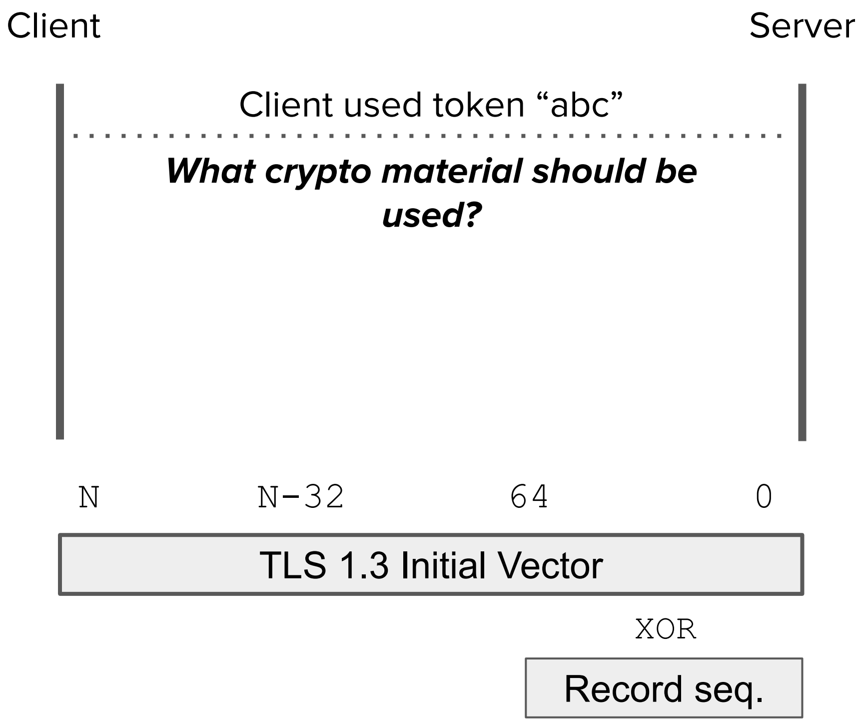
Adding TCP connections

- The client put the token in the ***tcplsJoin*** TLS Extension
- The server validates the token and joins the TCP connection to the session



Adding TCP connections

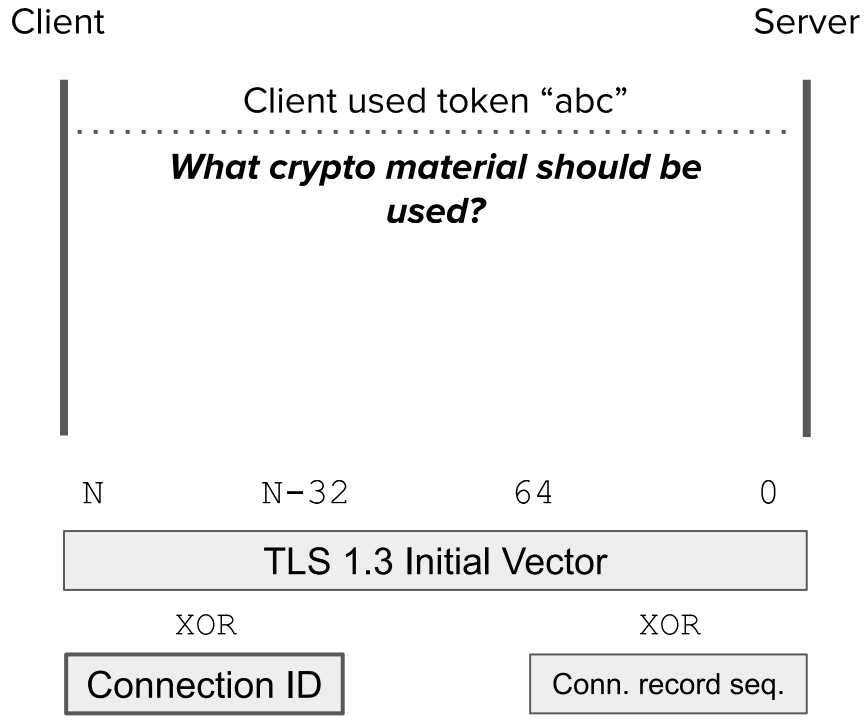
- Each TLS record is encrypted with a unique nonce
- Record sequence is kept implicit
- The record sequence cannot be shared among TCP connections
- **We do not want to do a full TLS handshake, which is costly**



TLS Per-record Nonce

Adding TCP connections

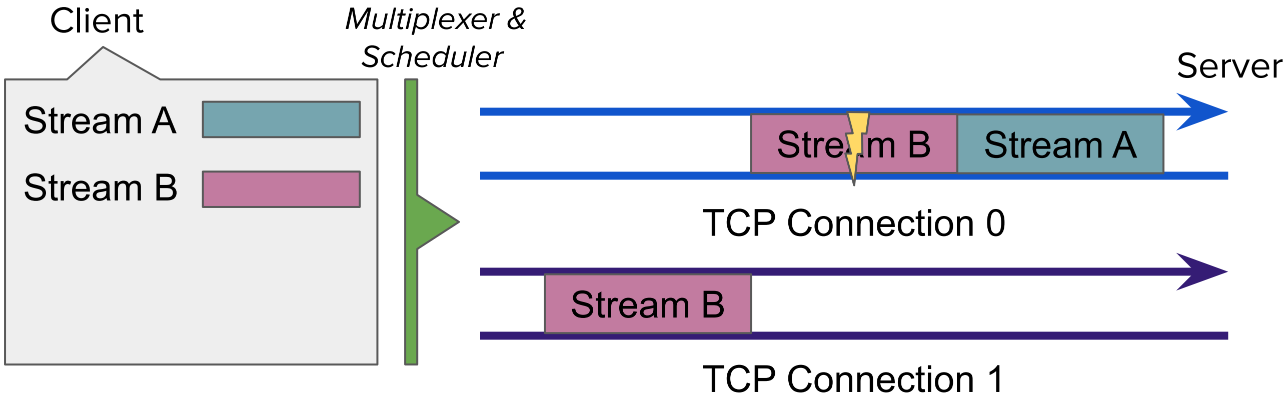
- Each TLS record is encrypted with a unique nonce
- Record sequence is kept implicit
- The record sequence cannot be shared among TCP connections
- **We XOR the Connection ID to the nonce and add a per-connection record sequence**



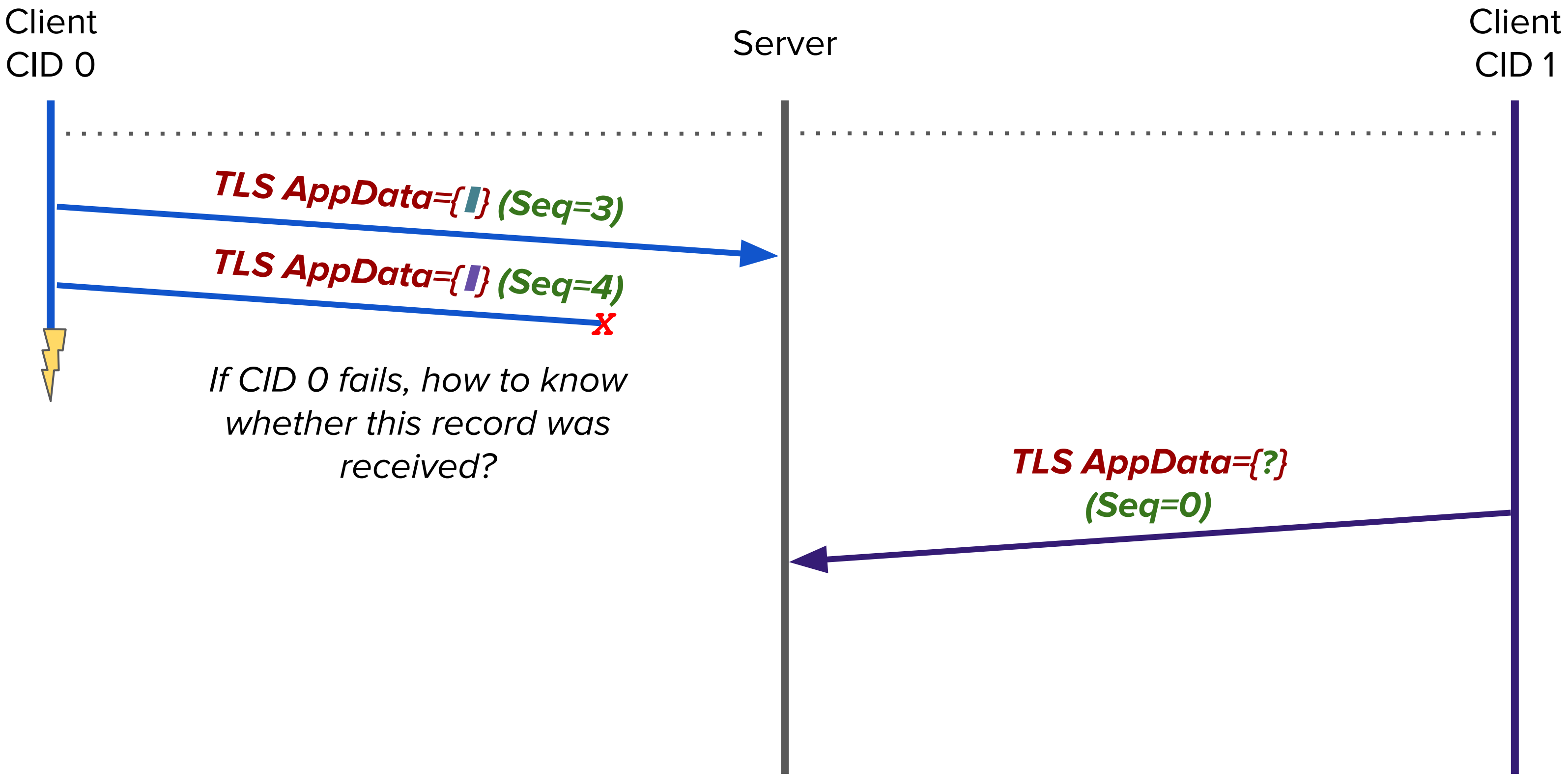
TCPLS Per-record Nonce

Failover

- Endpoints can reinject frames from lost records onto other TCP connections
- They know which records have been received

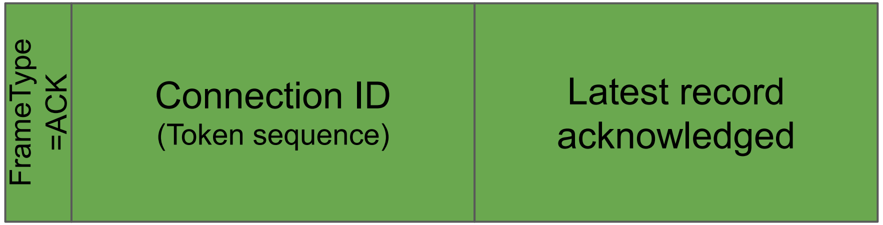


Record acknowledgements



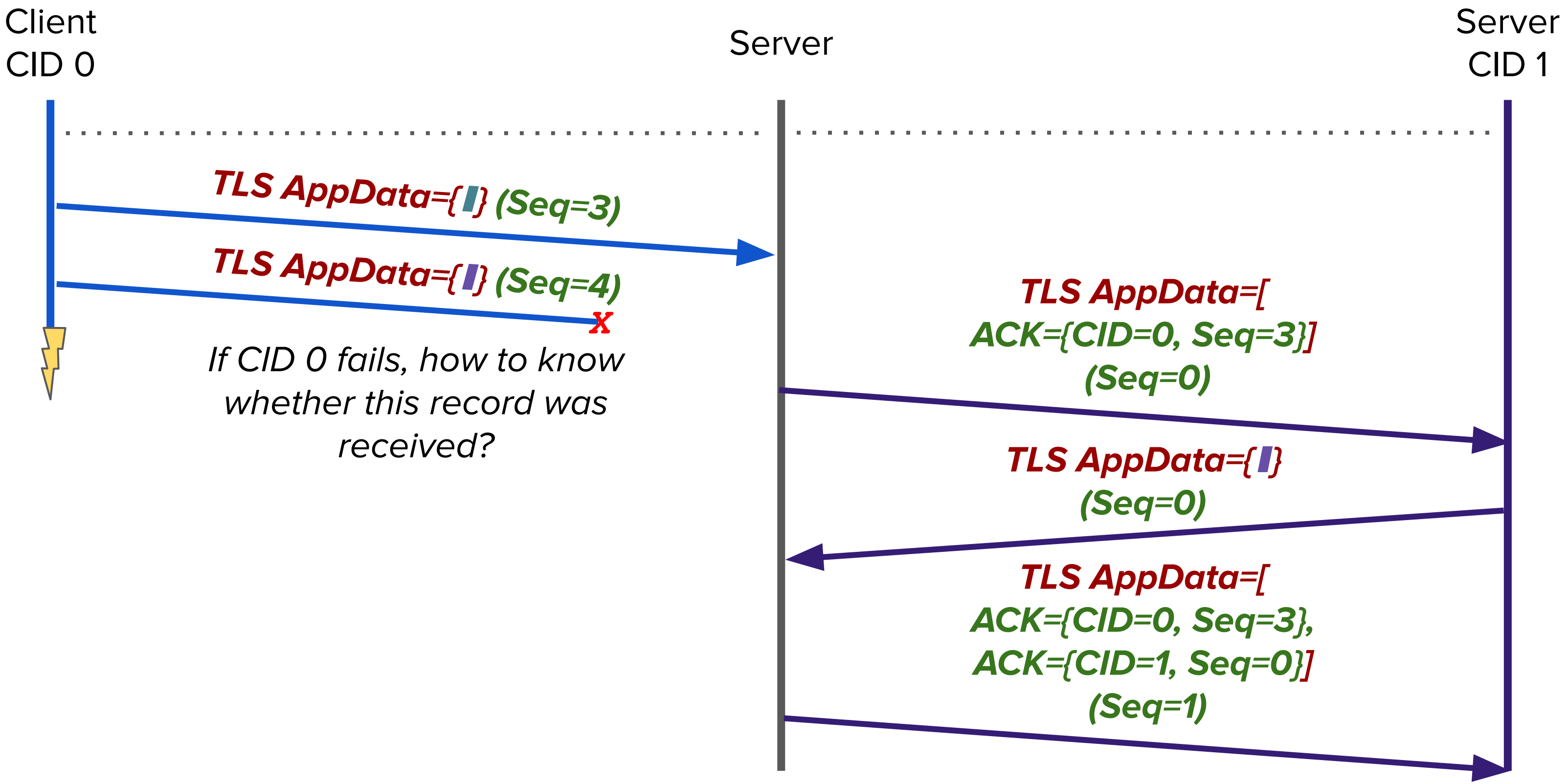
Record acknowledgements

- Each record is identified by:
  - its TLS record sequence number
  - the Connection ID (=Token sequence number) it was sent on
- ACK frame indicates the sequence number of the latest record received over a connection
- As TCP delivers data in sequence, only cumulative ACKs are needed



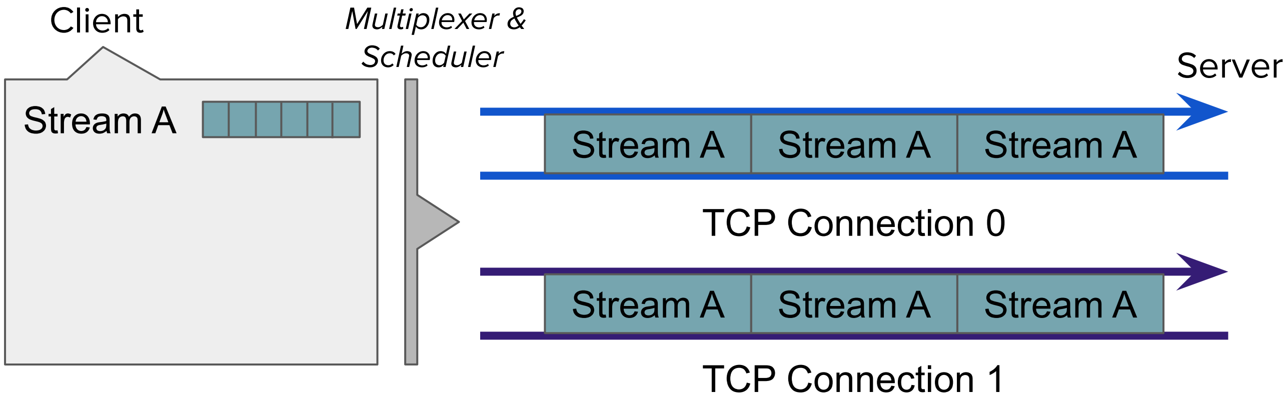
ACK frame

Record acknowledgements



Bandwidth aggregation

- Endpoints can send Stream frames of a given stream on several TCP connections, benefiting from bandwidth aggregation



draft-piraux-tcpls

- It describes the protocol presented here
- We welcome feedback and comments on the draft
    - For both the protocol and the use-cases
- We will continue working on improving the protocol
- Some parts will be discussed in future versions
    - Congestion control
    - Flow control
- Followed a preliminary version of the TCPLS protocol presented at CoNEXT’21 [1]

[1] Rochet, F., Assogba, E., Piraux, M., Edeline, K., Donnet, B., and O. Bonaventure, "TCPLS - Modern Transport Services with TCP and TLS", (CoNEXT'21).

Prototype

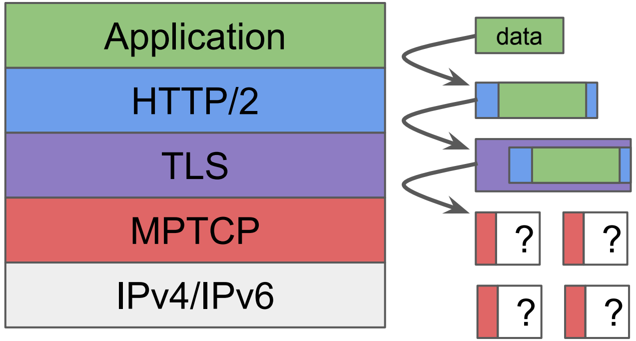
- We implemented *draft-piraux-tcpls-01* on top of `picotls`, a TLS 1.3 implementation in C
- We modified 50 lines of `picotls` for the required TCPLS interface
- The prototype implements stream multiplexing, failover and multipath
- It consists of 2.5k lines of C
- We will release the prototype under an open-source license

Conclusion

- TCPLS is a secure, user-space, transport protocol bringing
  - Stream multiplexing
  - Connection migration, Failover
  - Multipath
- TCPLS leverages in-kernel high performance TCP implementations
- We implemented a prototype in 2.5k lines of C
  - We will publish the code
- We are interested pursuing this work within the IETF
  - Should we start with a dedicated mailing list ?

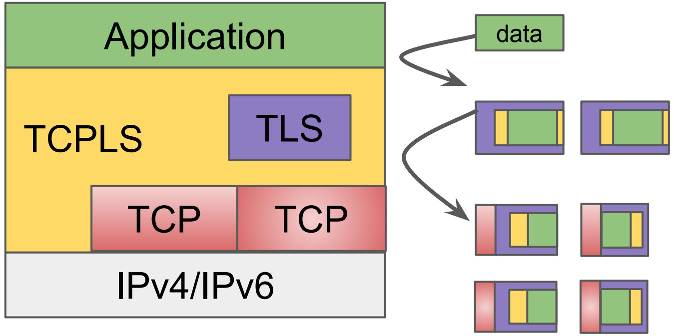
Backup – HTTP/2

- HTTP/2+TLS+MPTCP is built on strict layering assumption
- TCPLS offers more control to the application over the TCP connections of the session



How to divide in:

- Frame?
- Record?
- Packets?



TCPLS offers a better control and leverages several TCP connections