TCPLS: Modern Transport Services with **TCP** and **TLS**

Published in: CoNEXT `21

Summarized by

Sangwon Lim (sangwonlim@snu.ac.kr)

2022-08-31

Contents

Introduction

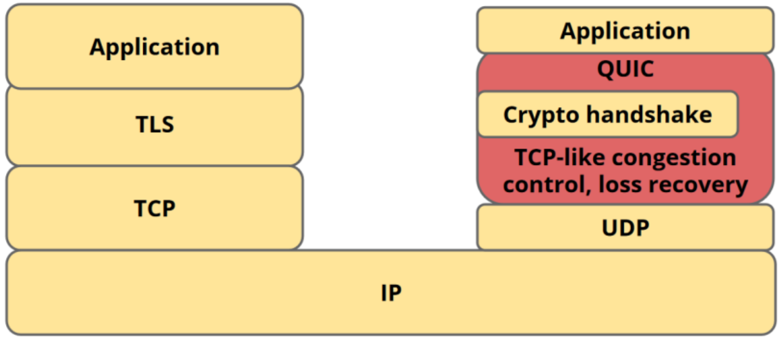
- The Transmission Control Protocol (TCP) is one of the most critical protocols in today’s Internet
  - ✓ TCP provides connection abstraction, reliability, and congestion control

- During the late nineties, and early 2000s, transport protocol researchers explored alternatives to TCP
  - ✓ DCCP: provides a way to gain access to congestion-control mechanisms at the application layer
  - ✓ SCTP: provides multihoming support where one or both endpoints of a connection can consist of more than one IP address

Introduction

- Extending TCP today is not feasible anymore as middleboxes severely interfere with changes to the TCP header and options

- To overcome this problem, Google started QUIC combining functions usually found in TCP, TLS, and HTTP/2
  - ✓ QUIC leverages encryption to prevent middlebox interference and
  - ✓ proposes to revisit the layered model of the Internet to improve the transport services
  - ✓ QUIC runs atop UDP, it can be implemented and deployed as a user-space library

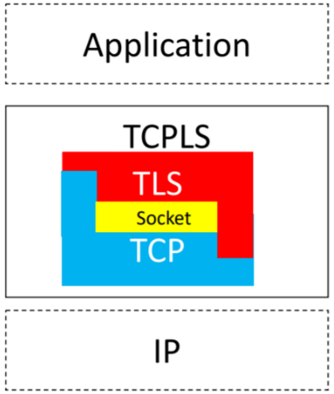


Introduction

- Does the standardization of QUIC mark the end of the TCP era?

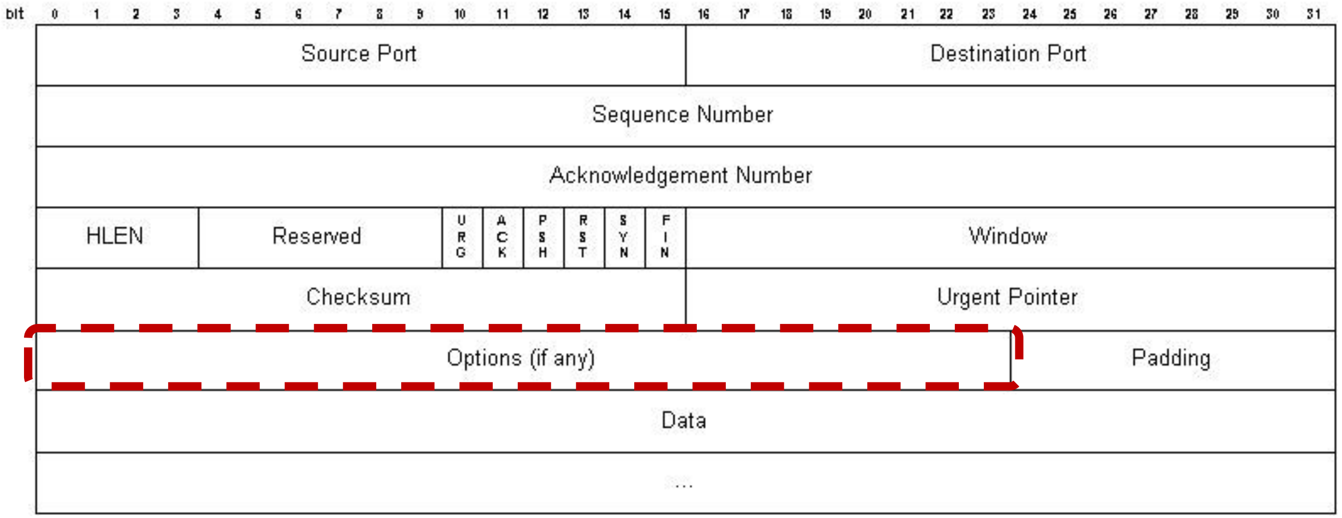
- TCP remains a fallback because of its greater support in networks, and TCP also still serves many applications

- The authors revisit how transport services can be provided with TCP and TLS today
  - ✓ (1) How can TCP and TLS be combined to improve extensibility and middlebox resilience?
  - ✓ (2) What are the new transport services that this combination can offer?



Background

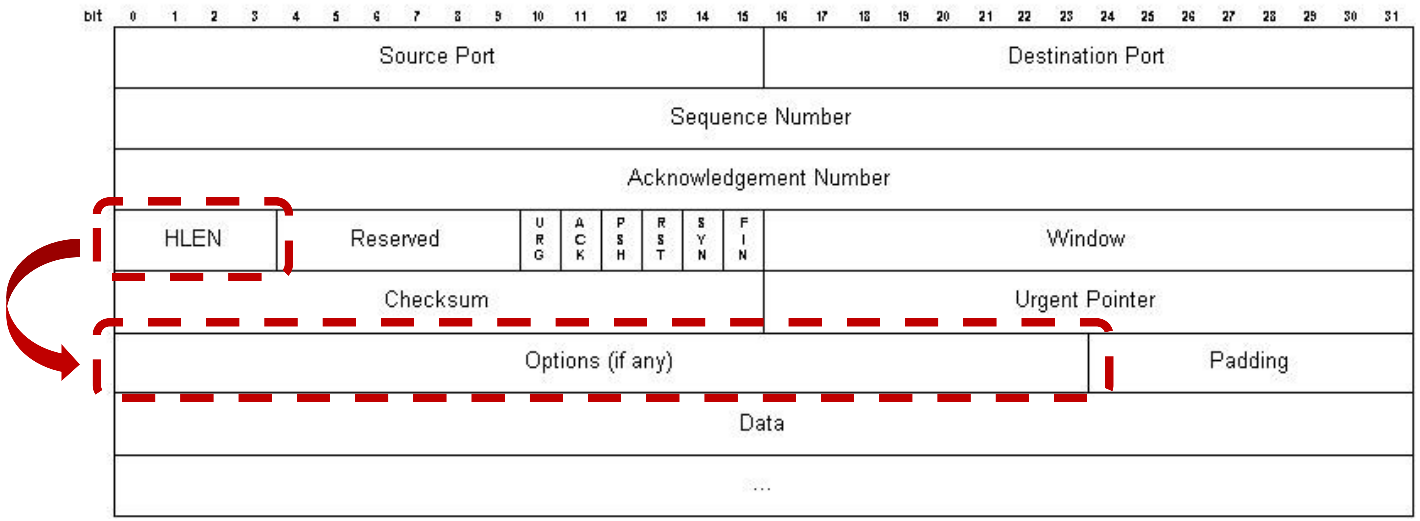
- Middleboxes interfere with TCP or its extensions
  - ✓ Firewalls can discard packets containing TCP Options that were not known when the firewall was designed
  - ✓ Firewalls can replace unknown TCP Options with the NOP TCP Option



< TCP Segment Format >

Background

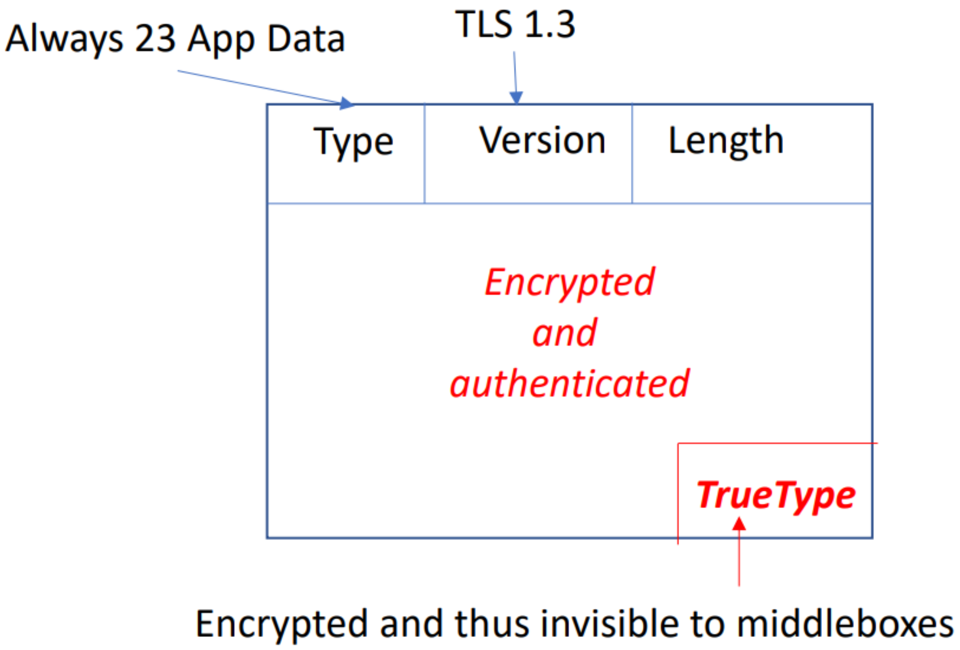
- TCP extensions are hard to extend
  - ✓ The amount of bytes for extensions in the TCP header is limited to 40 bytes
  - ✓ TCP is often implemented as a part of the OS kernel, which leads to complexity to implement and deploy any modification



< TCP Segment Format >

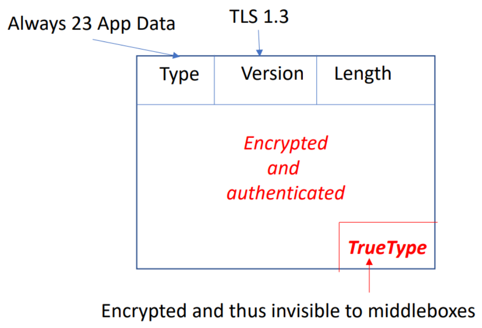
Background

- Modern applications rarely use TCP alone and they often combine TCP with Transport Layer Security (TLS)
  - ✓ TCPLS extends the encrypted TLS records to convey control and application data

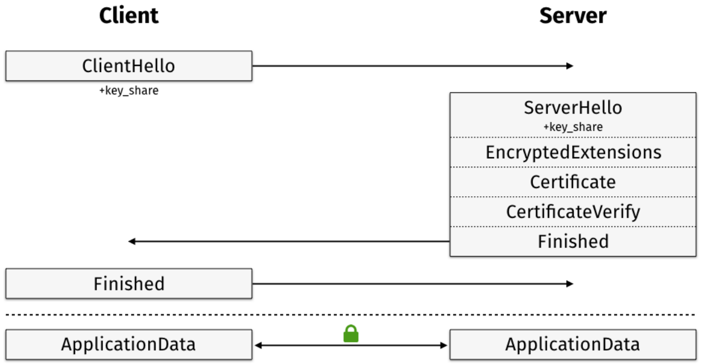


TCPLS Design

- (1) How can TCP and TLS be combined to improve extensibility and middlebox resilience ?
  - **Reliable exchange of TCP extension**
    - ✓ Transport level control data in TLS records
    - ✓ TCPLS can provide a large range of new transport services

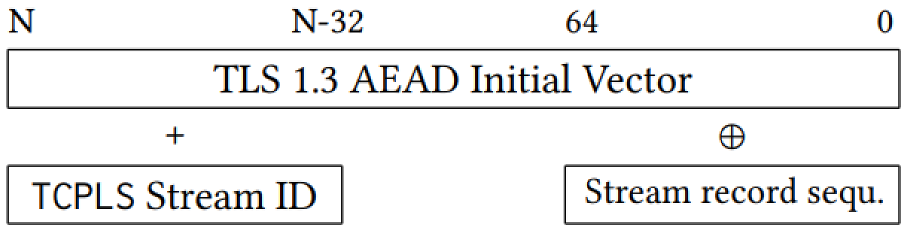


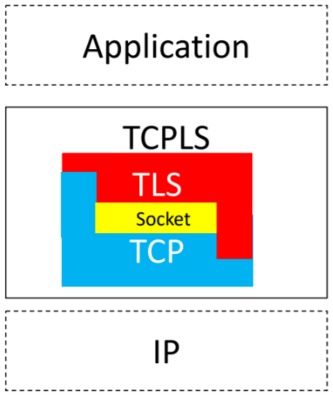
  - **More options during the handshake**
    - ✓ TCPLS can leverage TLS Encrypted Extensions to negotiate during the handshake some of the new transport services



TCPLS Design

- (2) What are the new transport services that this combination can offer?
  - **Quick Resumption**
    - ✓ TCP's Fast Open + TLS's 0'RTT
  - **Stream Multiplexing**
    - ✓ The AEAD[1)] Nonce of TCPLS Streams is derived from TLS 1.3

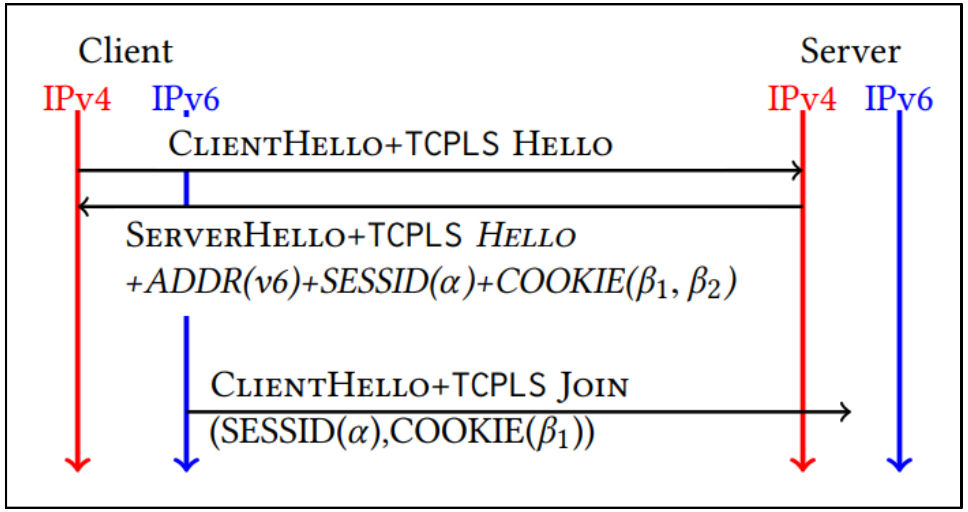


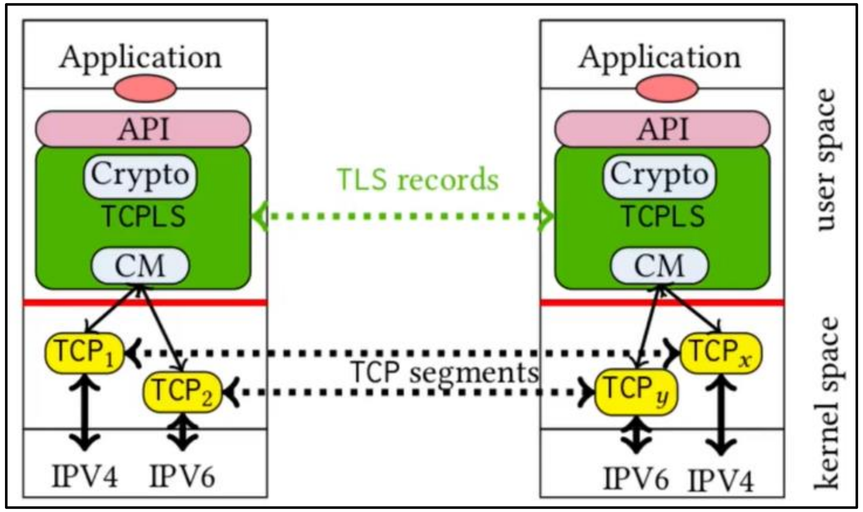


1) Authenticated Encryption with Associated Data

TCPLS Design

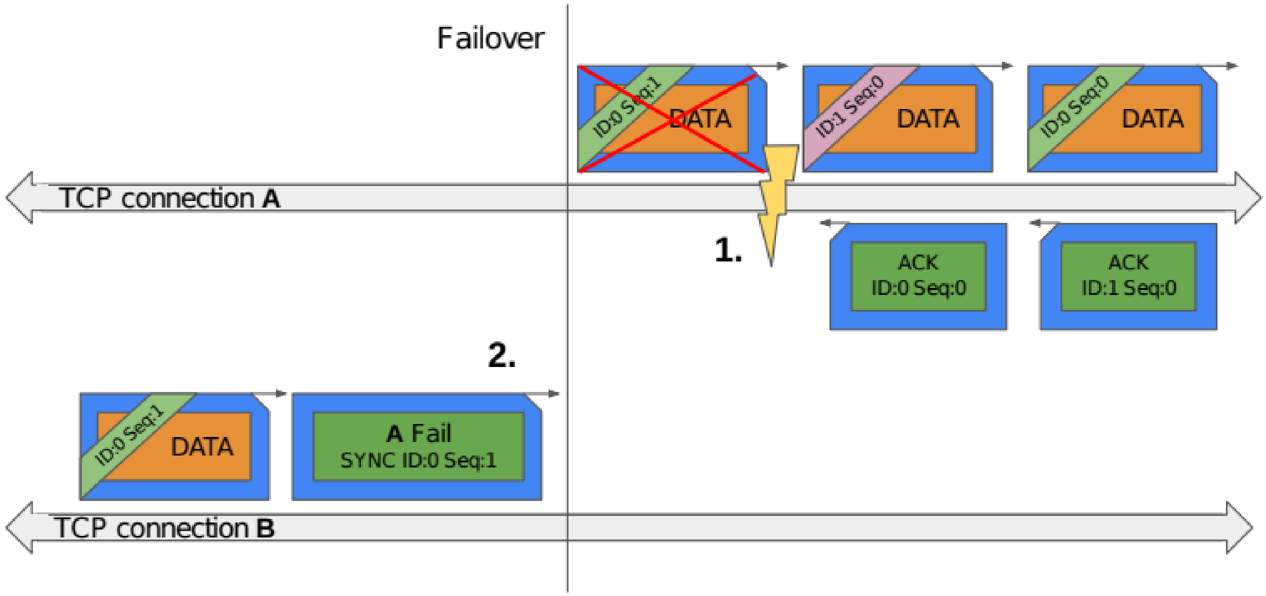
- (2) What are the new transport services that this combination can offer?
  - **Joining TCP connections**





TCPLS Design

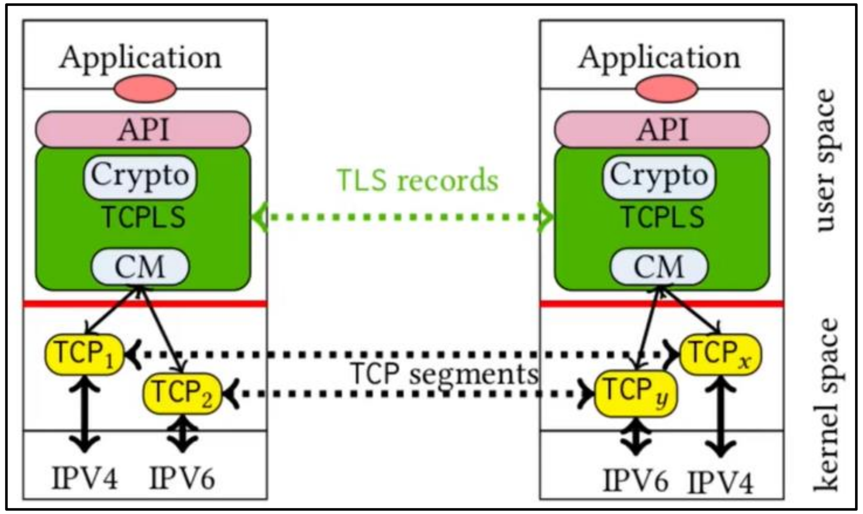
- (2) What are the new transport services that this combination can offer?
  - **Fail over**



< Failover resynchronizes and retransmits lost TCPLS records from a failed TCP connection to another >

TCPLS Design

- (2) What are the new transport services that this combination can offer?
  - **Application-triggered Connection Migration**
    - ✓ e.g., Migration from LTE to Wi-Fi



TCPLS Design

- (2) What are the new transport services that this combination can offer?
  - **Multipath Capabilities**
    - ✓ Stream Steering
      - ✓ The application has full control in exchange of a bit of work
      - ✓ No head-of-line blocking
    - ✓ Coupling streams for aggregated bandwidth
      - ✓ TCPLS exposes the sender side TCPLS record scheduler to the application
      - ✓ This enables the application to actively decide the TCP connection
    - ✓ Securing Multipath TCP
      - ✓ Security concern on MPTCP: Token is exchanged inside SYN/SYN+ACK
      - ✓ With TCPLS: Derive token from TLS secrets

TCPLS Prototype Implementation

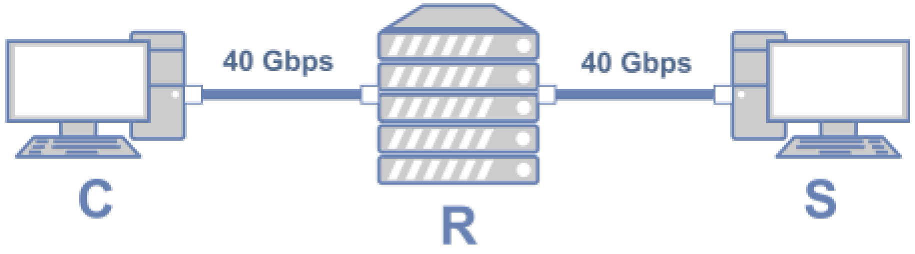
- The prototype is a fork of the picotls TLS 1.3 implementation
  - The authors added only 9k lines of C code to implement TCPLS

- eBPF[1] Code Remote Attachment
  - ✓ eBPF can run sandboxed programs in an operating system kernel
  - ✓ Since Linux kernel version 5.6, an application can attach congestion control schemes entirely implemented in eBPF
  - ✓ TCPLS prototype enables the server to attach a new eBPF congestion controller to the client over the TCPLS session

1) extended Berkeley Packet Filter

TCPLS Evaluation

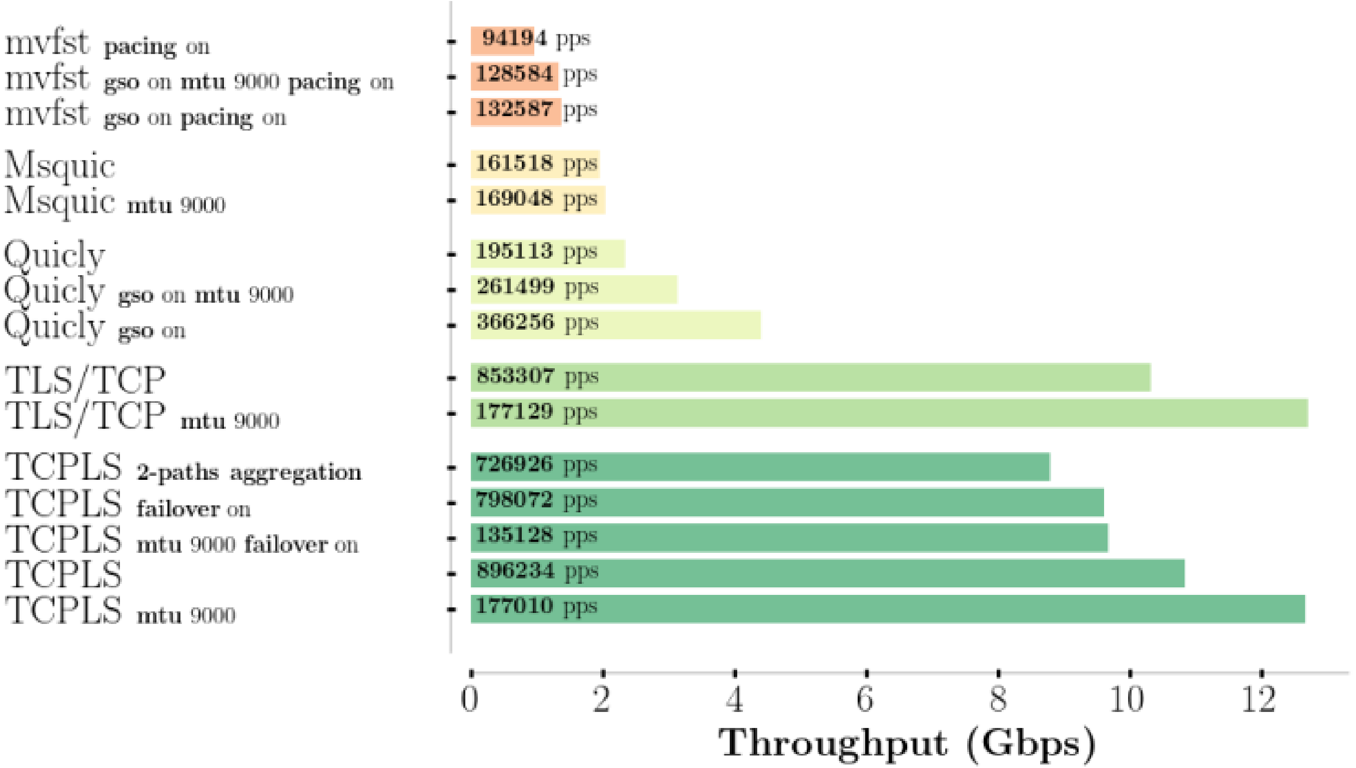
- Performance Measurements Setup
  - Intel Xeon CPU E5-2630 2.40GHz, 16 GB RAM
  - Debian with Linux 5.9 and 5.7 kernels
  - Intel XL710 2x40 Gbps NIC (MTU: 9000 bytes, 1500 bytes)



(C = Client. R = Router/Middlebox. S = Server.)

TCPLS Evaluation

- TCPLS offers better raw performance than several QUIC implementations



< Throughput comparison between TCPLS, TCP/TLS, and three QUIC implementations >

Why QUIC is slower?

1) GSO[1)] is often not supported by the NIC
2) Userspace pacing
3) Ack in userspace
4) packets are smaller units

1) Generic Segmentation Offload

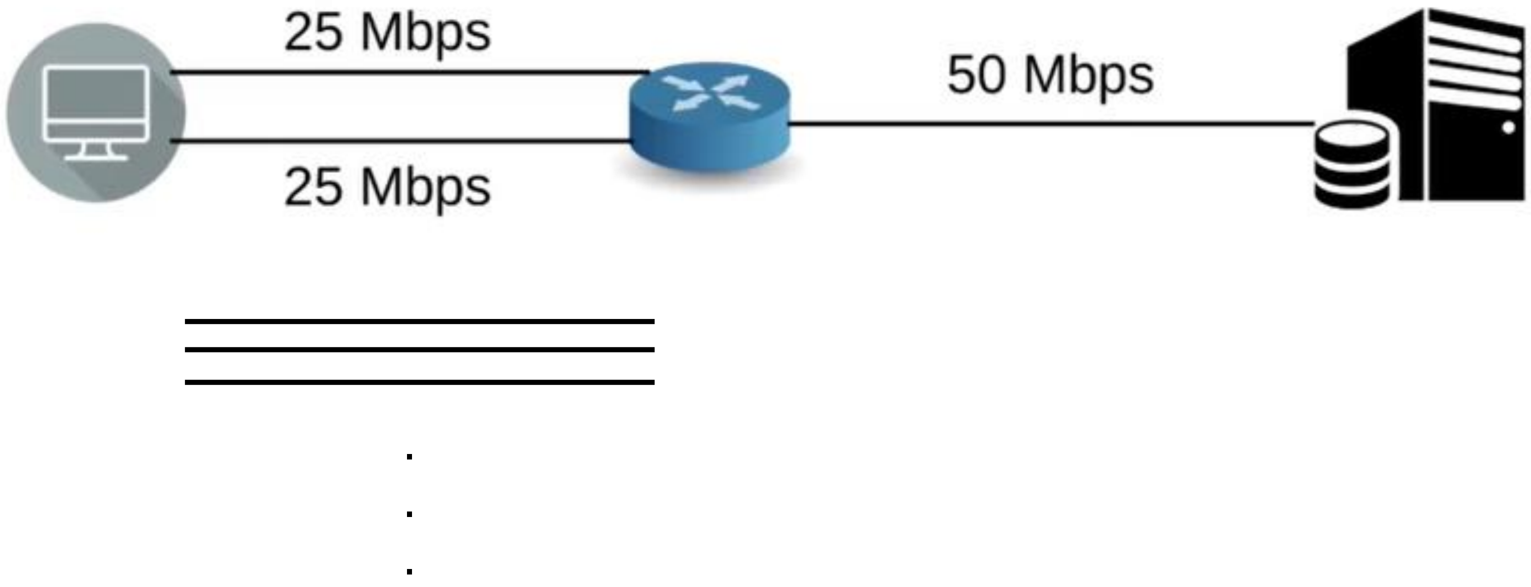
TCPLS Evaluation

- The authors tested TCPLS against different opensource and commercial stateful firewalls and proxy implementations (i.e., pfSense, IPFire, Cisco ASAv, mitmproxy)
  - ✓ They found no unexpected interference

- When faced with middleboxes that modify TLS 1.3, some TCPLS messages can be impacted
  - ✓ TCPLS Hello, TCPLS Join, SESSID, and COOKIE
  - ✓ Then, the client can implicitly fall back to TLS and continue with the handshake

TCPLS Evaluation

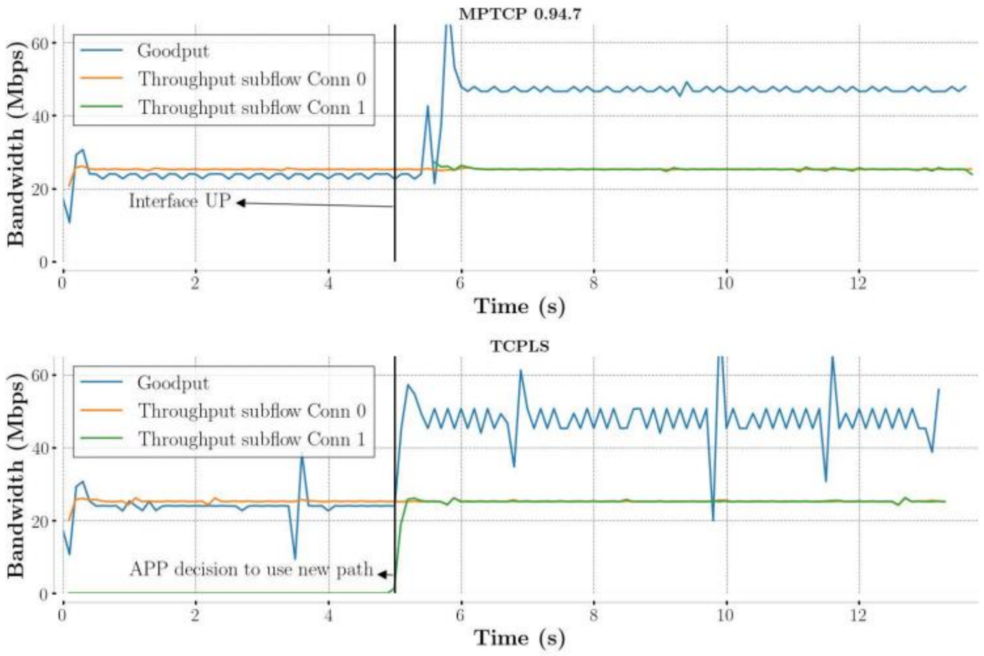
- Mininet comparison of MPTCP[1)] and TCPLS



1) MultiPath TCP

TCPLS Evaluation

- TCPLS offers a bandwidth aggregation service similar to the one offered by state-of-the-art MPTCP



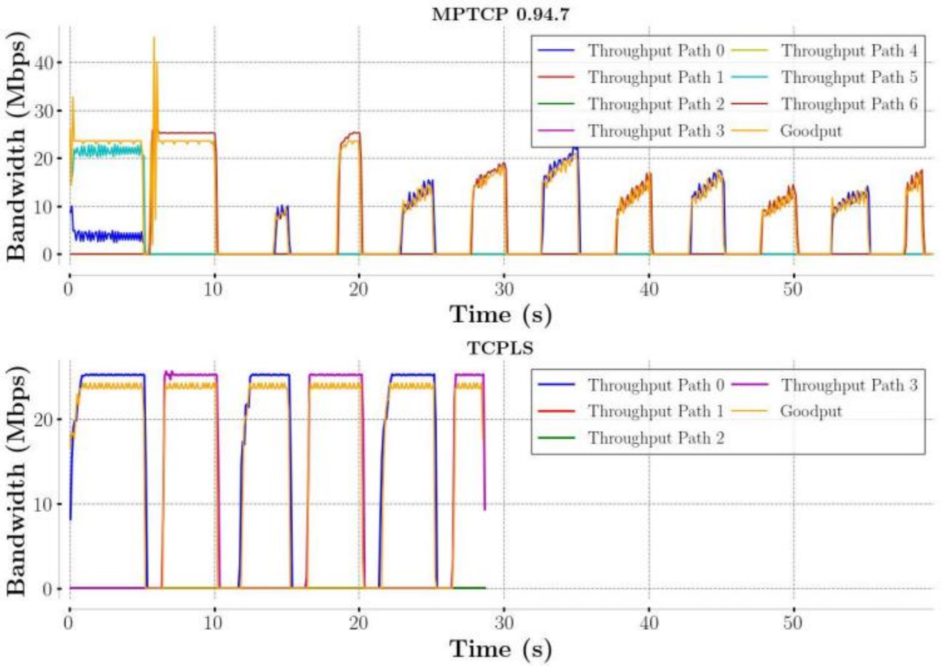
1) for MPTCP, there is a delay before it becomes fully utilized
   ∵ Linux Kernel requires the time to configure the new network interface

2) TCPLS's aggregated goodput seems less stable than MPTCP
   ∵ Bigger payload size

< Bandwidth aggregation comparison between MPTCP (top plot) and TCPLS (bottom plot) with a record payload size of 16,384 bytes >

TCPLS Evaluation

- Failover recovery speed analysis
  - MPTCP has difficulties reacting to successive network outages during a 60MB file download TCPLS reacts quickly to such outages and completes the file transfer faster

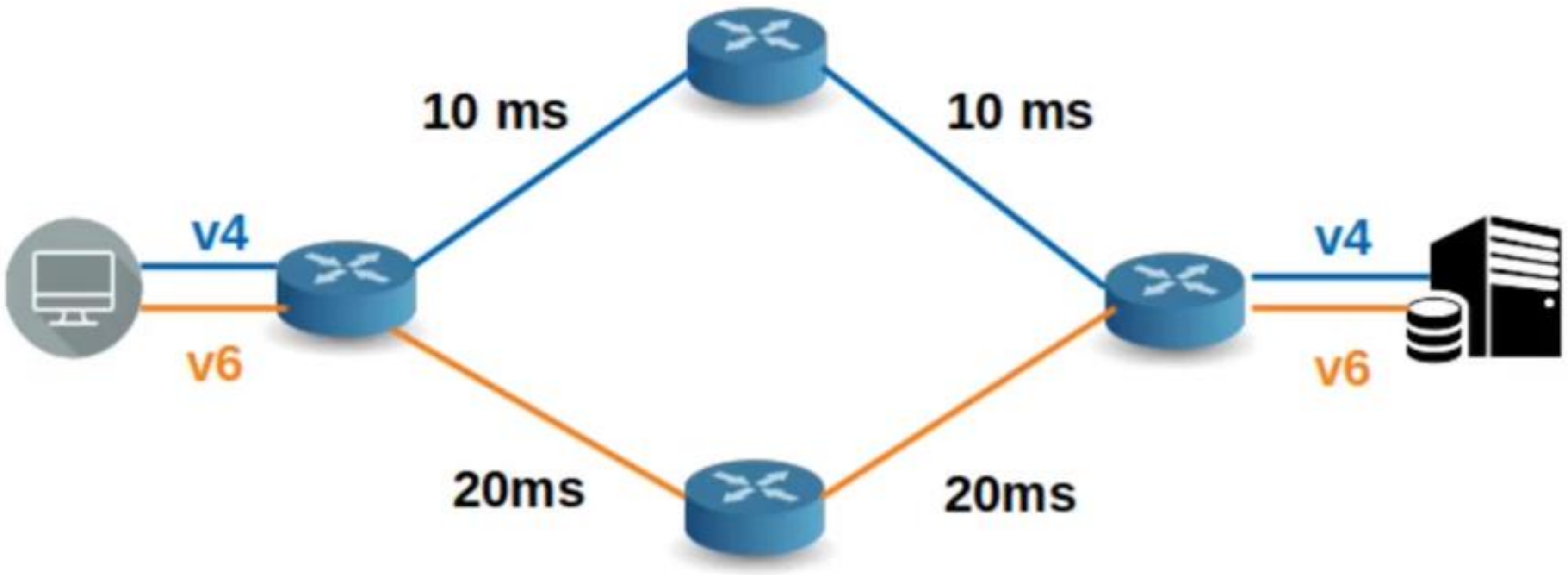


Why TCPLS faster?

1) Exchange the TCP User Timeout option through TCPLS records

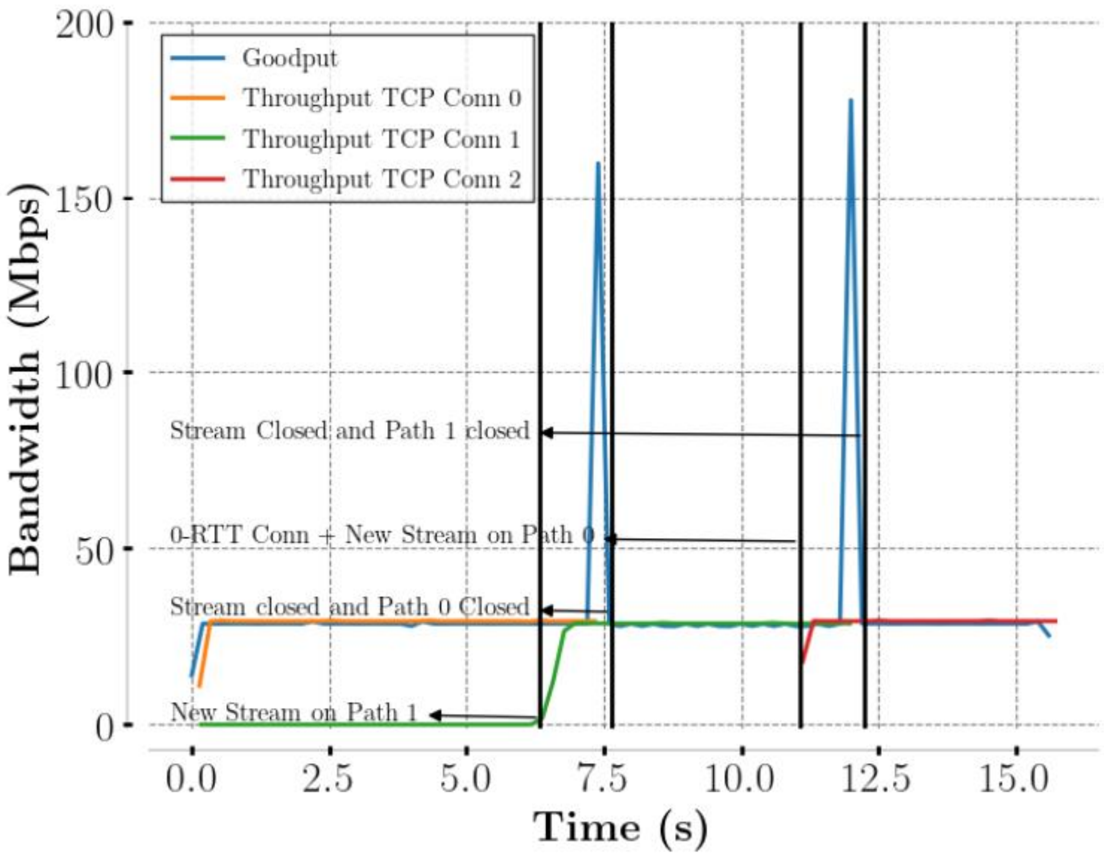
TCPLS Evaluation

- Mininet



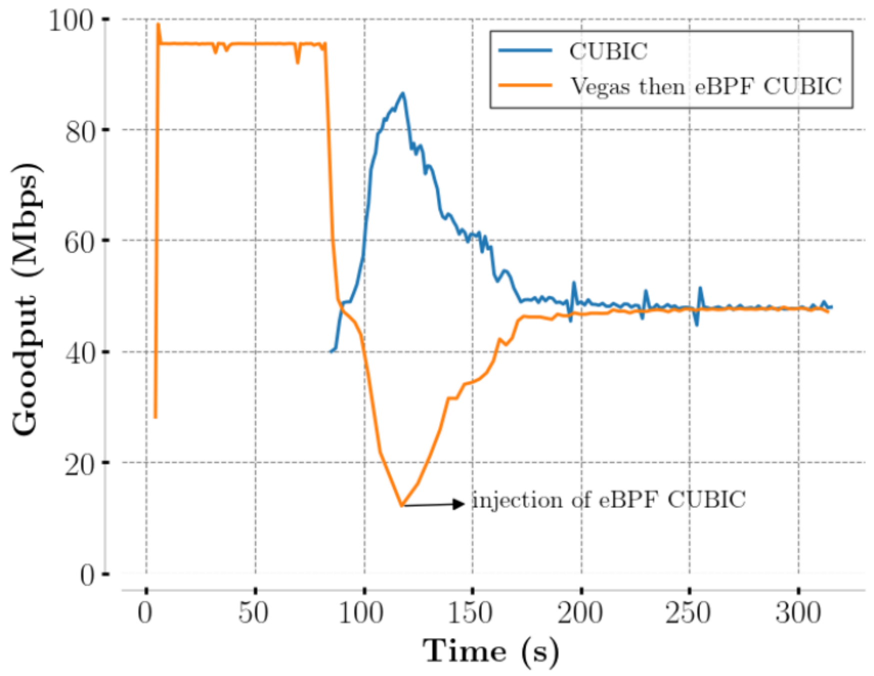
TCPLS Evaluation

- Application-level Migration
  - ✓ The application can trigger a connection migration and sustain its bandwidth during the process
  - ✓ TCPLS temporarily aggregates the two network paths during such a migration



TCPLS Evaluation

- TCPLS hosts can exchange eBPF congestion controllers and enable them during a TCPLS session
  - The bandwidth distribution becomes fair after the server sends an eBPF bytecode implementing the CUBIC congestion controller
  - *Mininet network with a 100 Mbps link*

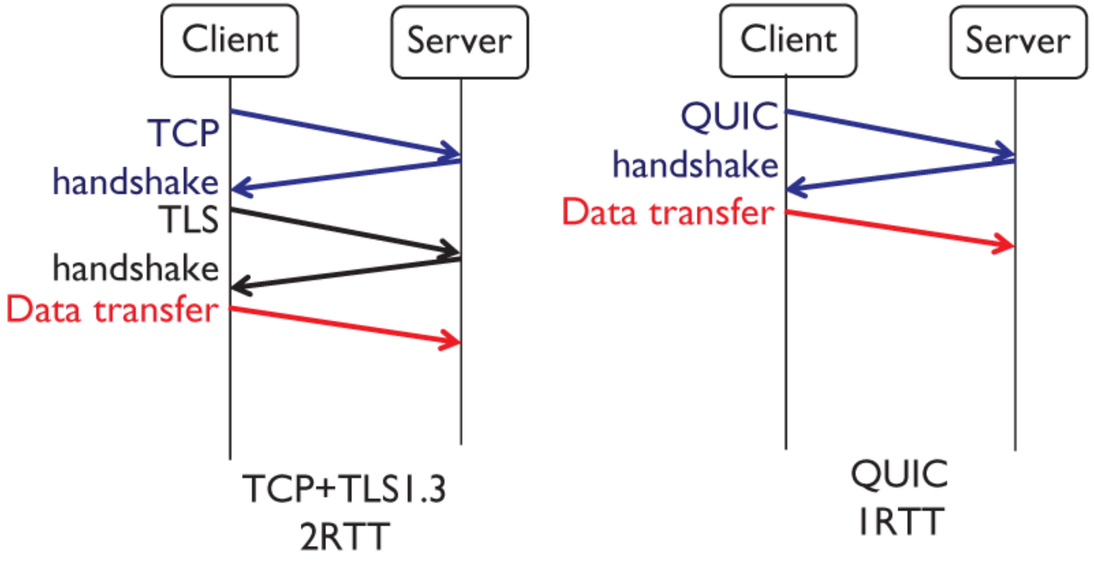


Conclusion

- There are benefits to a cross-layer approach for TLS/TCP
  - For capabilities, performance, extensibility, and security & privacy
- TCPLS can be implemented simply with existing TLS libraries
  - Without any kernel change in contrast to MPTCP
- TCPLS can be a powerful contender to QUIC for modern services
  - Over TCP vs. UDP
  - Bigger unit size

Critique

- QUIC offers a quick first response



- QUIC is optimized for web content delivery