Designing Transport-Level Encryption for Datacenter Networks

Tianyi Gao, Xinshu Ma, Suhas Narreddy, Eugenio Luo,
Steven W. D. Chien, and Michio Honda
University of Edinburgh
18 March 2026, IETF meeting 125

For the paper to appear at IEEE Symposium on Security & Privacy 2026

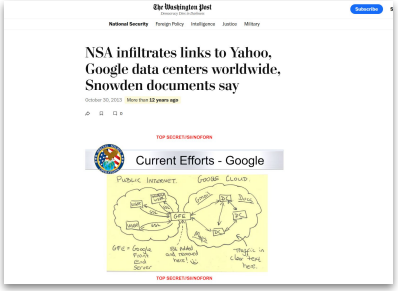
THE UNIVERSITY of EDINBURGH
informatics

Encryption in the Internet

- -2010
  - Encryption was exceptional [Bittau et al., USENIX Sec'10]
- 2010-
  - Google rolled out default SSL encryption
- 2015-
  - More encryption: QUIC, ECH, DoH, ML-KEM etc.

Encryption in datacenters

- 2013
  - Plaintext data breach
- Today
  - TLS is ubiquitous
  - Bad actors exist
    - Malicious insiders [Mogul and Wilkes, HotNets'23]
    - Compromised tenants [Arzani et al., NSDI'20]



The Washington Post, 2013

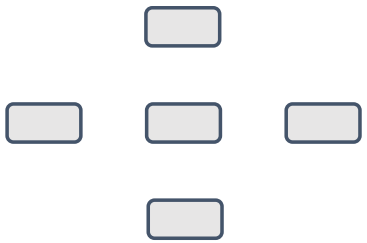
Engineering at Meta, 2019

TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput,
  low-latency RPCs
  - Software overhead and
    head-of-line blocking matter

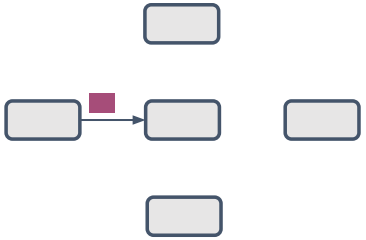
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



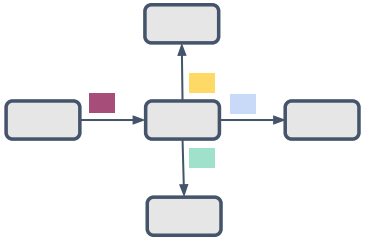
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



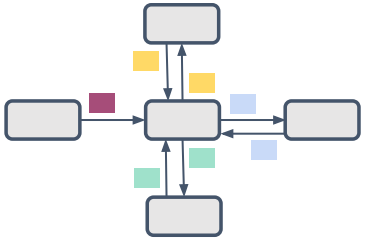
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



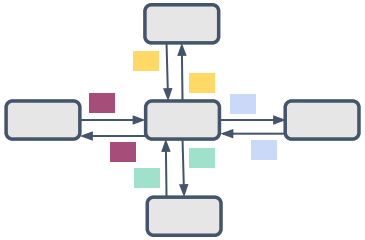
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



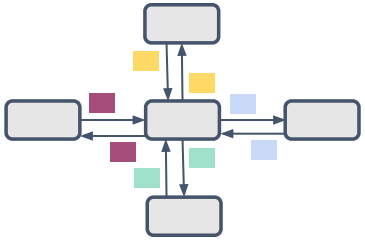
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



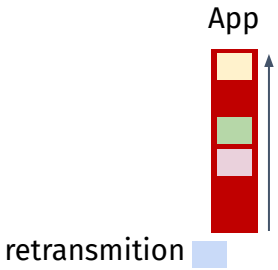
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter



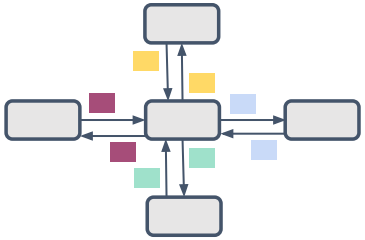
Intra-connection HoLB

on packet loss



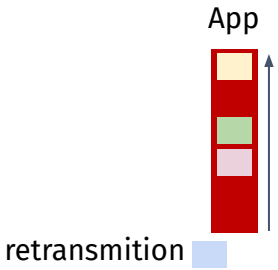
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter

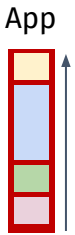


Intra-connection HoLB

on packet loss

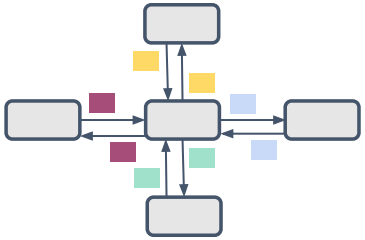


on large message



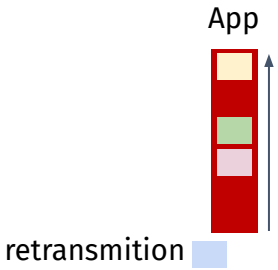
TCP (and thus TLS/TCP) is unfit to datacenters

- High-throughput, low-latency RPCs
  - Software overhead and head-of-line blocking matter

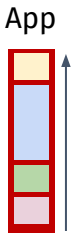


Intra-connection HoLB

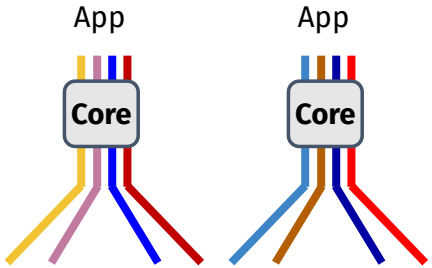
on packet loss



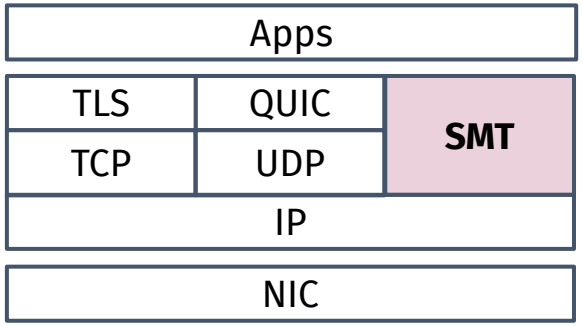
on large message



Inter-connection HoLB on CPU cores

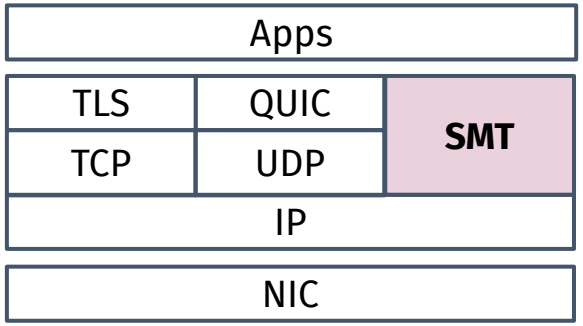


SMT design goals



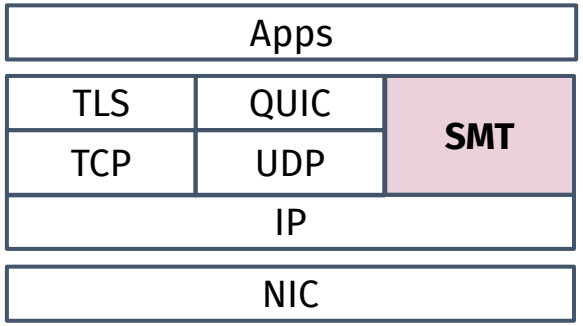
SMT design goals

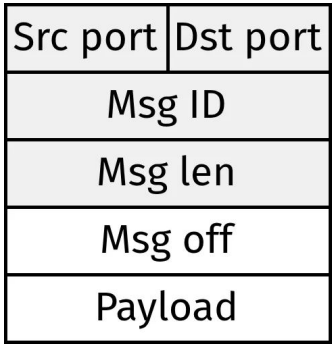
- TLS 1.3 security guarantees
  - TLS/TCP trust model



SMT design goals

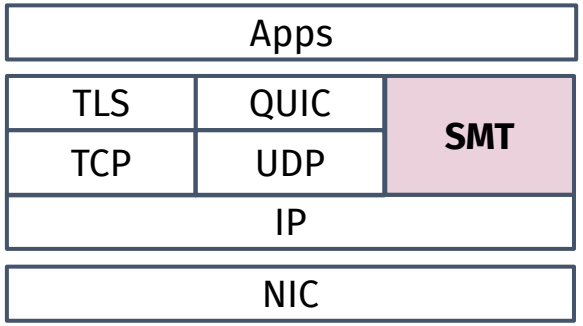
- TLS 1.3 security guarantees
  - TLS/TCP trust model
- Message-based abstractions
  - Avoid HoLB

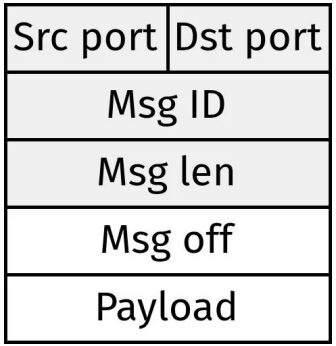




SMT design goals

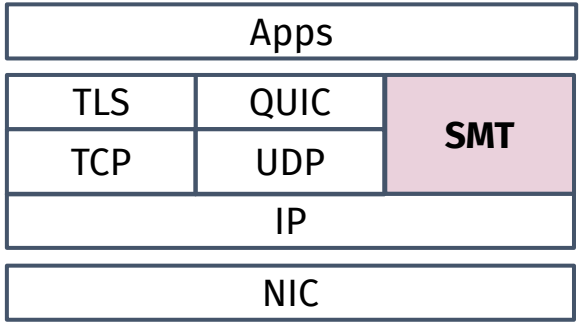
- TLS 1.3 security guarantees
  - TLS/TCP trust model
- Message-based abstractions
  - Avoid HoLB
- Opportunistic hardware offload
  - TSO and TLS encryption

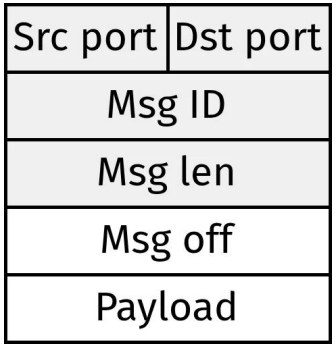




SMT design goals

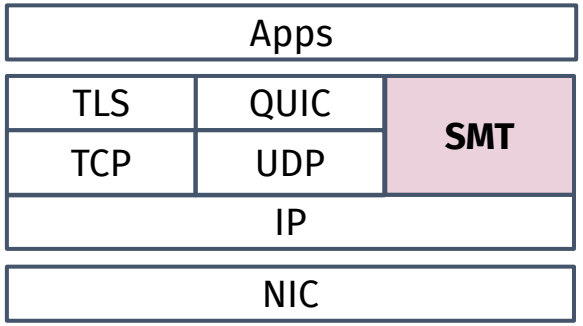
- TLS 1.3 security guarantees
  - TLS/TCP trust model
- Message-based abstractions
  - Avoid HoLB
- Opportunistic hardware offload
  - TSO and TLS encryption
- Plaintext transport-level headers
  - Useful for in-network computing

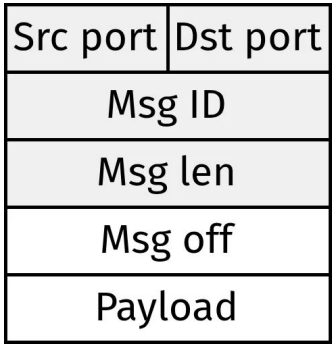




SMT design goals

- TLS 1.3 security guarantees
  - TLS/TCP trust model
- Message-based abstractions
  - Avoid HoLB
- Opportunistic hardware offload
  - TSO and TLS encryption
- Plaintext transport-level headers
  - Useful for in-network computing
- Native transport
  - No UDP encapsulation





Why not QUIC?

	QUIC's design point	**SMT's design point**
	Internet	**Datacenter**
Latency	1–100 ms	**1–100 µs**
HoLB	Path	**Path & CPU cores**
Middlebox Interference	High	**Low**
HW Offload	Optional	**Must**

Other options?

- Encrypted transports
  - TcpCrypt [USENIX Sec'10], TCPLS [CoNEXT'21]
    - Inherit the TCP problems

Other options?

- Encrypted transports
  - TcpCrypt [USENIX Sec'10], TCPLS [CoNEXT'21]
    - Inherit the TCP problems
- (Unencrypted) Message-based transports
  - SRD, Falcon [Sigcomm'25]
    - Hardware specific
  - NDP [Sigcomm'17], Homa [Sigcomm'18]

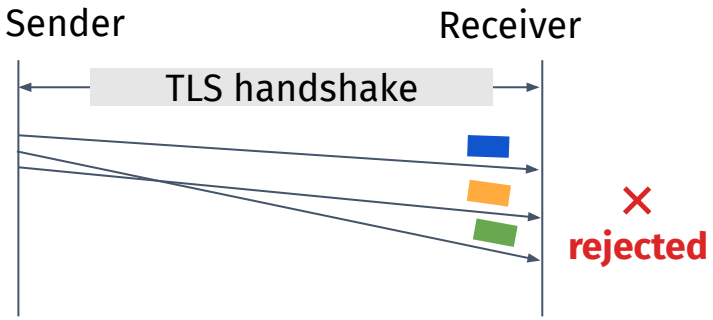
Other options?

- Encrypted transports
  - TcpCrypt [USENIX Sec'10], TCPLS [CoNEXT'21]
    - Inherit the TCP problems
- (Unencrypted) Message-based transports
  - SRD, Falcon [Sigcomm'25]
    - Hardware specific
  - NDP [Sigcomm'17], Homa [Sigcomm'18]

No existing approach combines TLS encryption + message abstraction, but Homa could be a middleground

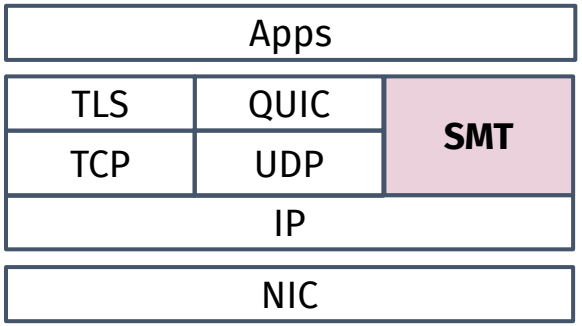
Why not TLS *over* a message-based transport?

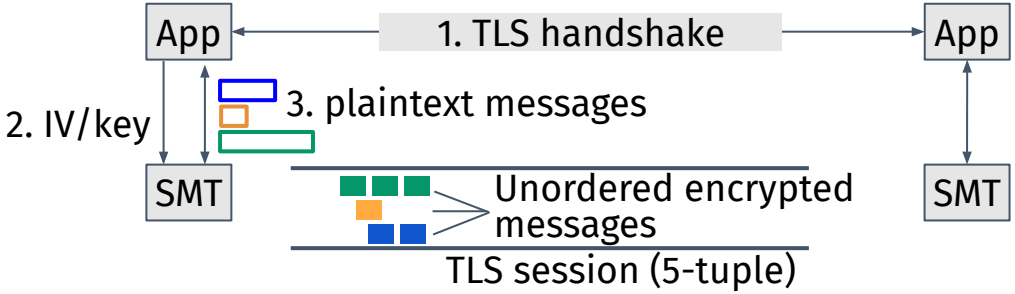
- TLS assumes in-order bytestream
  - Record rejection for out-of-order messages
  - Per-message handshake is too slow



SMT architecture

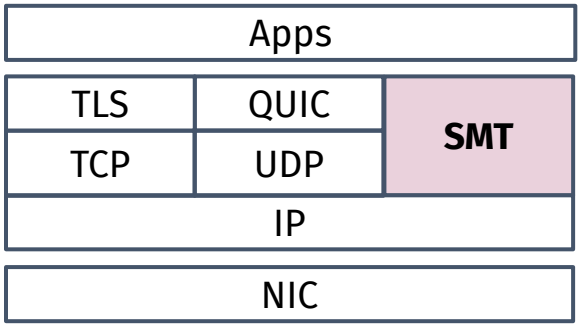
- Handshake in the app
- Boundary-preserved, reliable but out-of-order messages

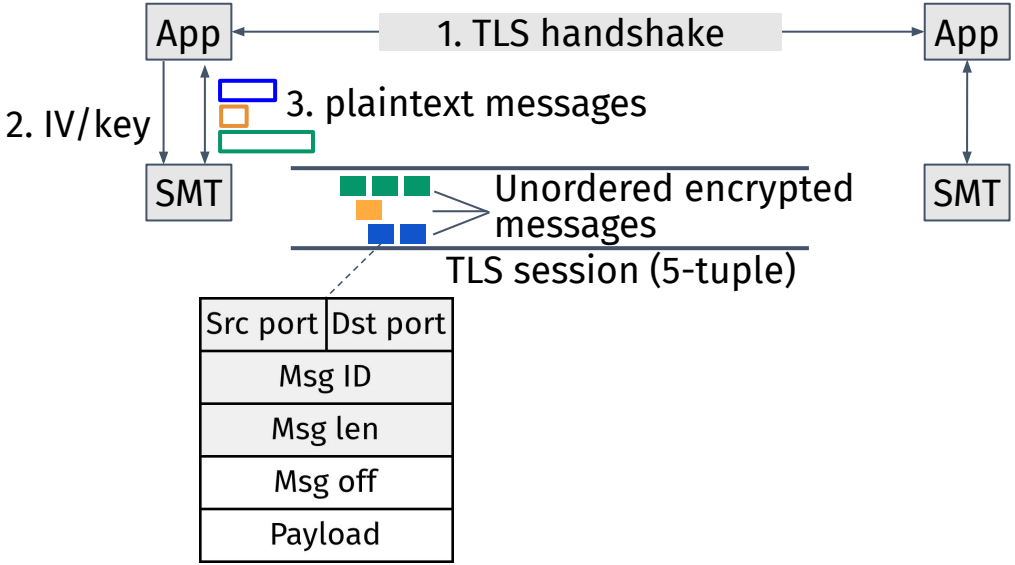




SMT architecture

- Handshake in the app
- Boundary-preserved, reliable but out-of-order messages





SMT key design components

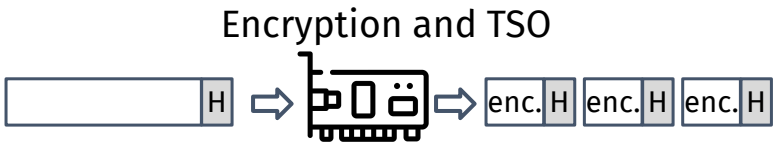
- Message format based on the TLS record protocol
  - TLS offload with TSO
- Per-message record sequence number space
  - Transport integration enables replay protection

SMT message and packet format

- **Goal: TSO/GSO with TLS offload**

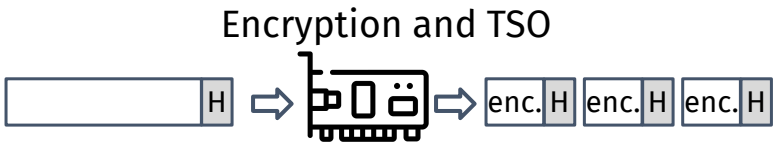
SMT message and packet format

- **Goal: TSO/GSO with TLS offload**



SMT message and packet format

- **Goal: TSO/GSO with TLS offload**

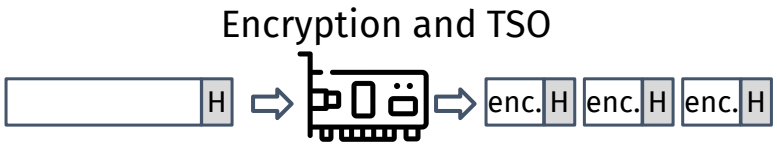


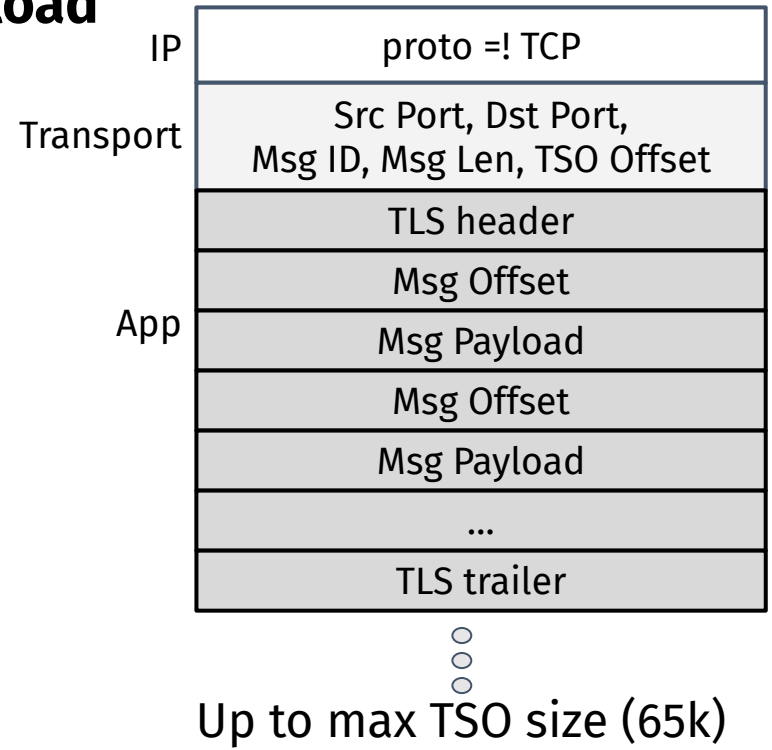
Key finding

TLS offload with TSO works for non-TCP packets as long as the header structure is preserved

SMT message and packet format

- **Goal: TSO/GSO with TLS offload**



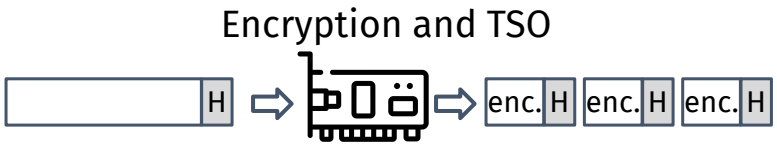


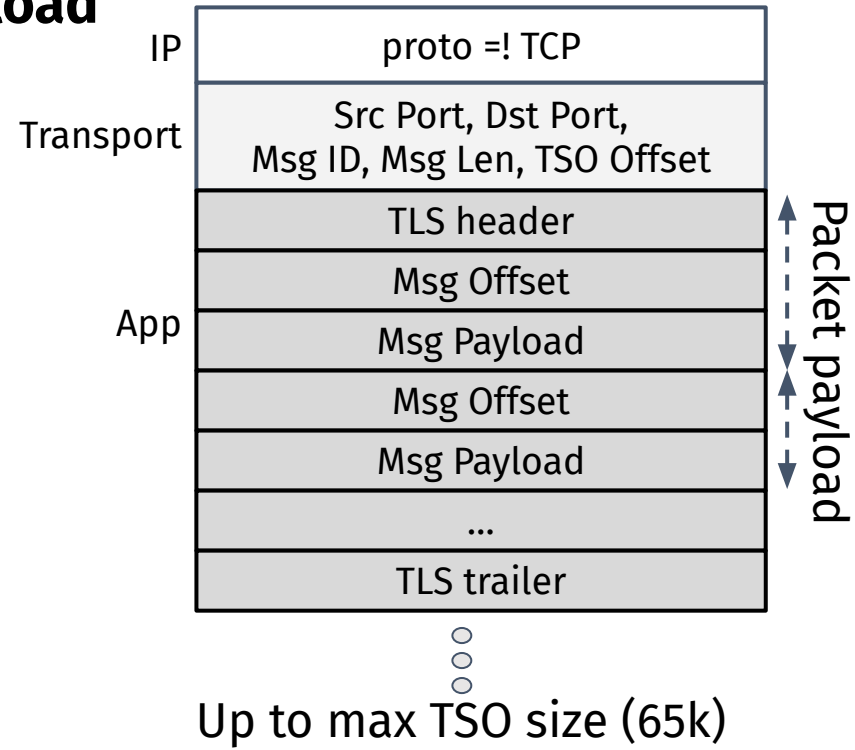
Key finding

TLS offload with TSO works for non-TCP packets as long as the header structure is preserved

SMT message and packet format

- **Goal: TSO/GSO with TLS offload**



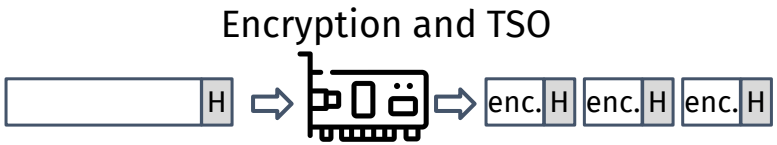


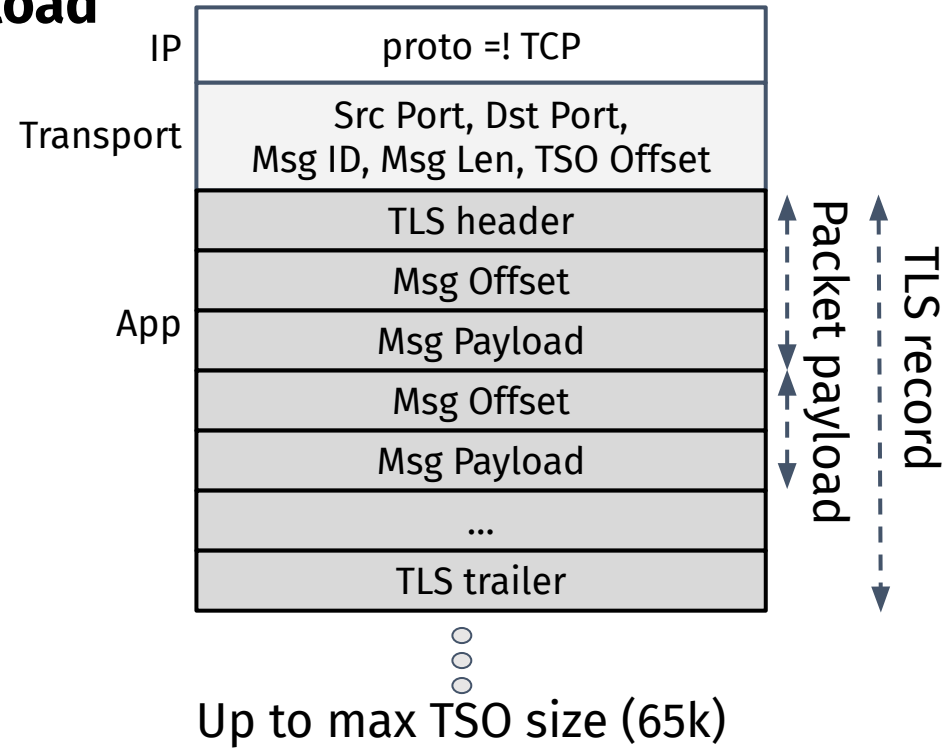
Key finding

TLS offload with TSO works for non-TCP packets as long as the header structure is preserved

SMT message and packet format

- **Goal: TSO/GSO with TLS offload**



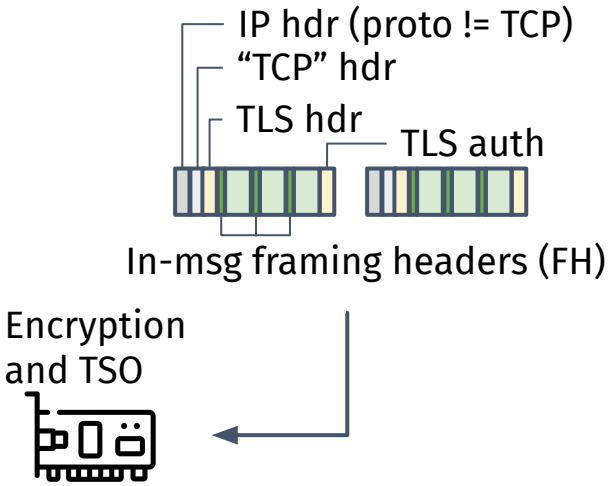


Key finding

TLS offload with TSO works for non-TCP packets as long as the header structure is preserved

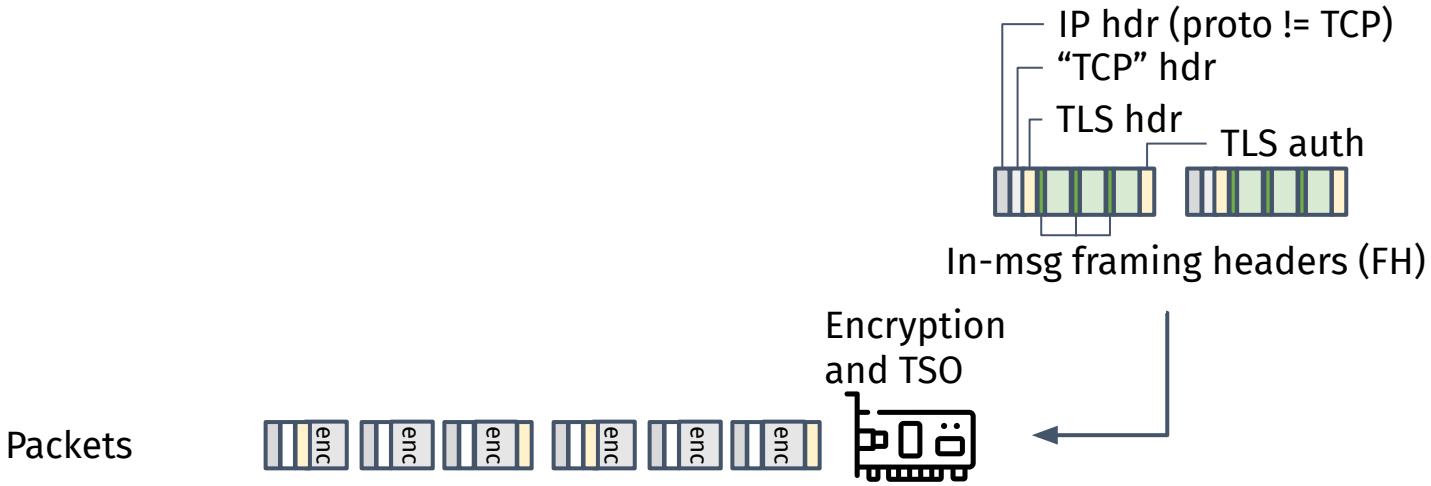
Segmentation and reassembly in action

one message over two TSO segments, each with one TLS record



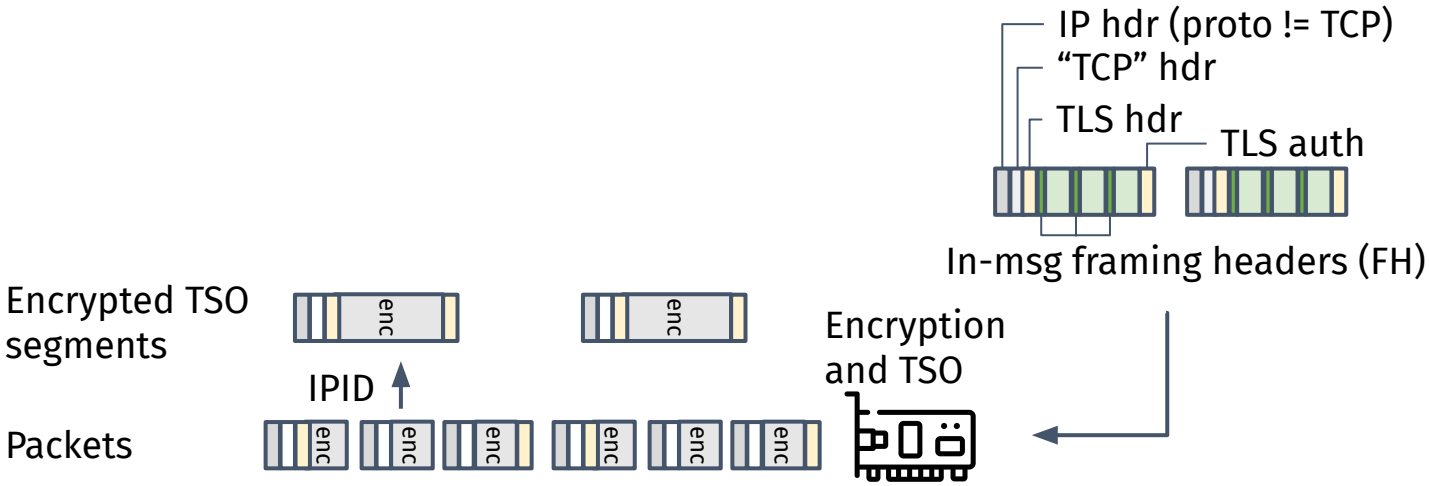
Segmentation and reassembly in action

one message over two TSO segments, each with one TLS record



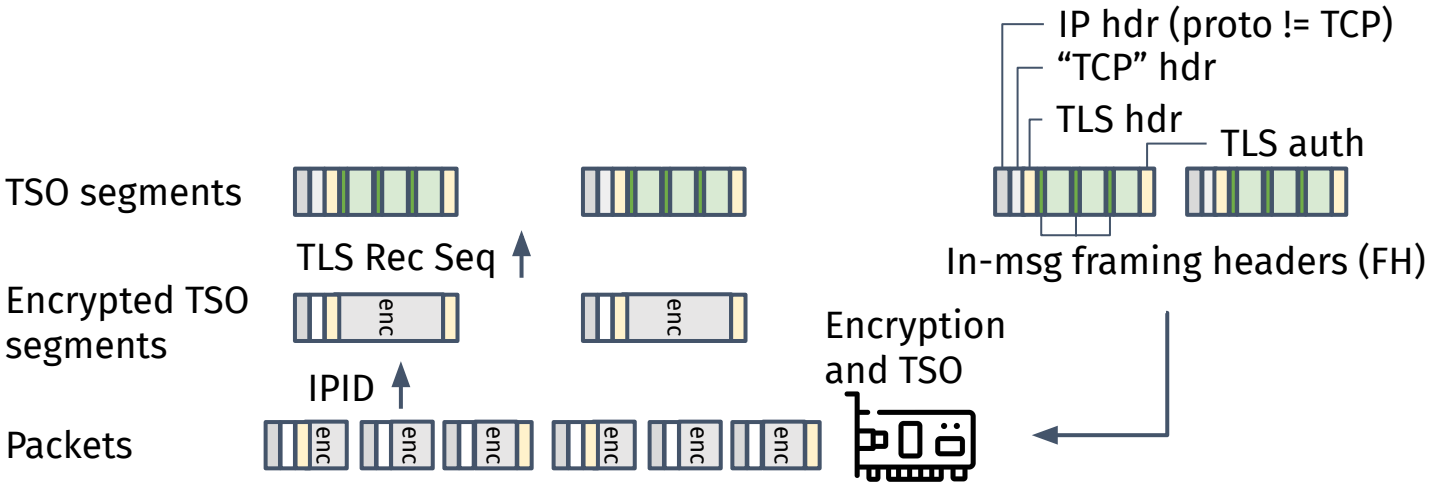
Segmentation and reassembly in action

one message over two TSO segments, each with one TLS record



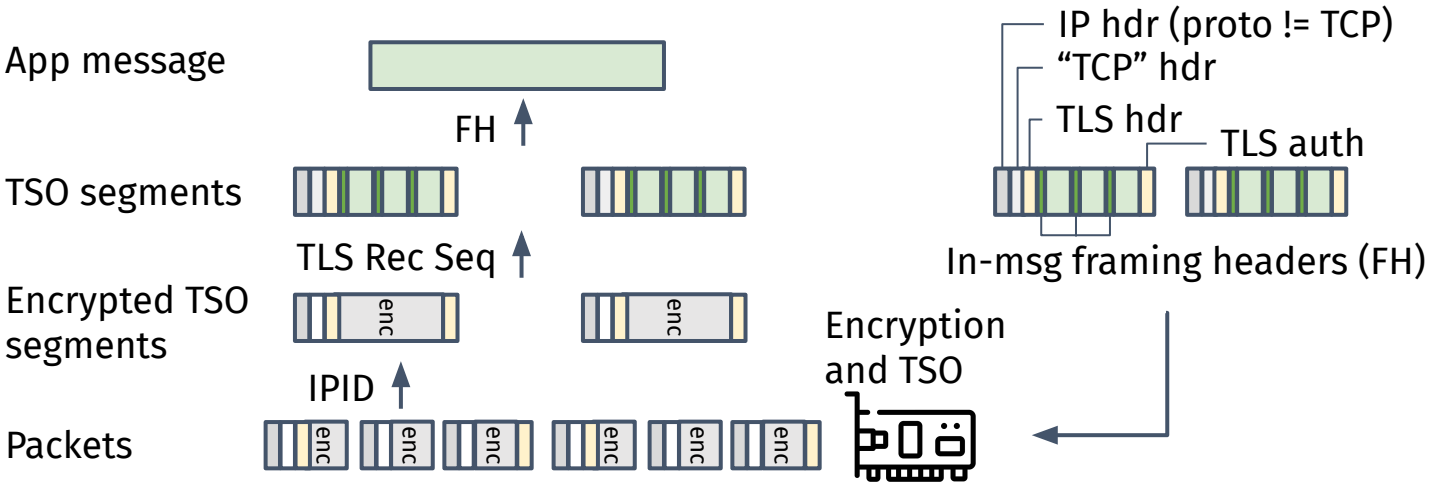
Segmentation and reassembly in action

one message over two TSO segments, each with one TLS record



Segmentation and reassembly in action

one message over two TSO segments, each with one TLS record



Per-message record sequence number spaces

- **Goal: Out-of-order message delivery**

  Problem

  Multiple seqno spaces can create duplicate seqno, considered as replay

Per-message record sequence number spaces

- **Goal: Out-of-order message delivery**

  Problem

  Multiple seqno spaces can create duplicate seqno, considered as replay

  Solution

  We can avoid duplicate by combining with a unique message IDs in the session

Per-message record sequence number spaces

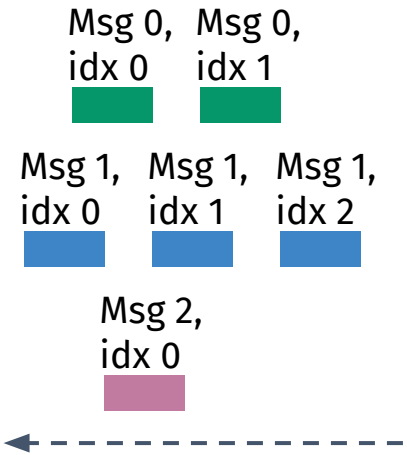
- **Goal: Out-of-order message delivery**

Problem

Multiple seqno spaces can create duplicate seqno, considered as replay

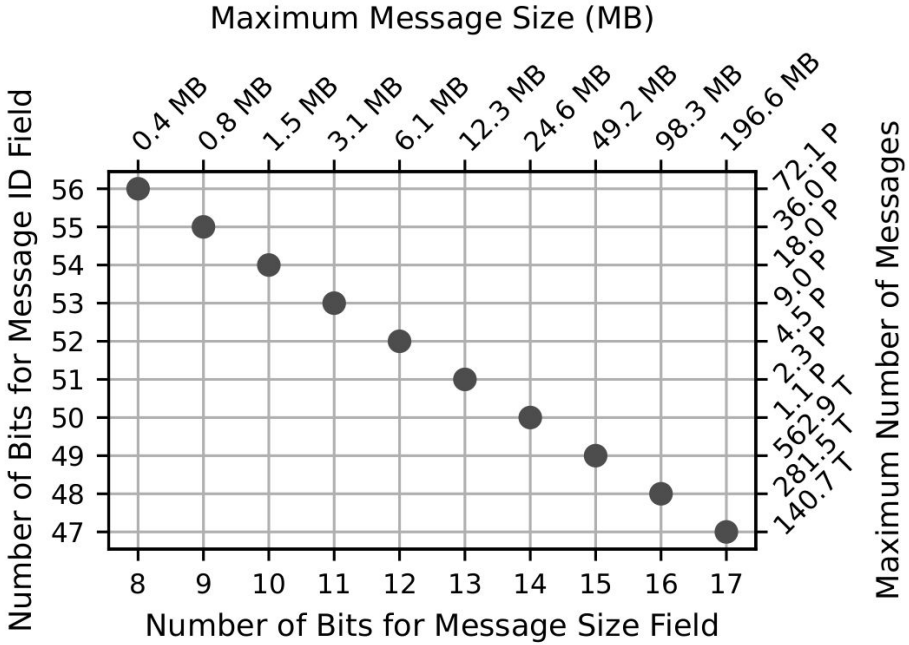
Solution

We can avoid duplicate by combining with a unique message IDs in the session



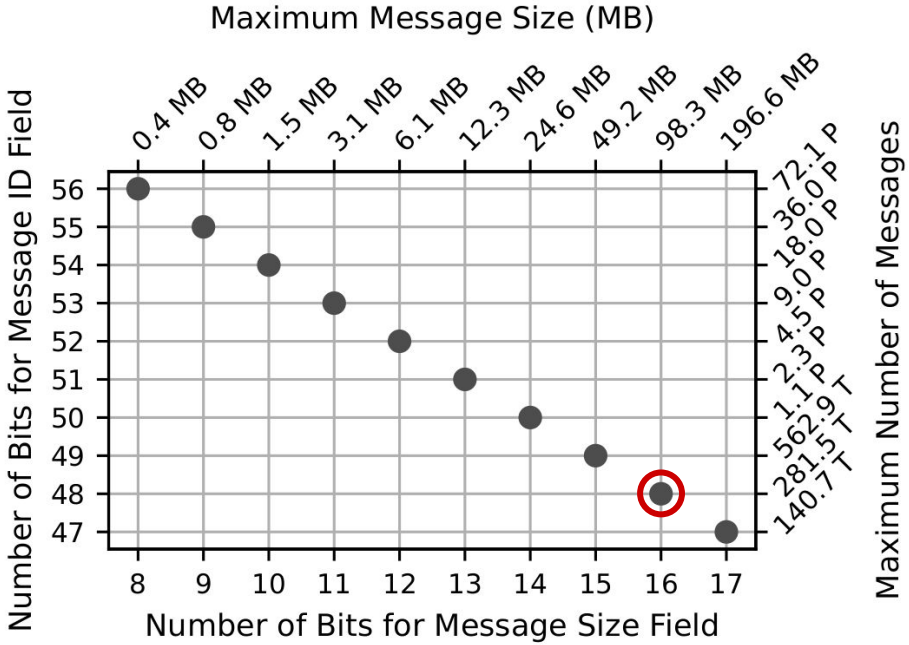
Message count/size trade-off

- Bit allocation can be flexible to support larger messages sizes



Message count/size trade-off

- Bit allocation can be flexible to support larger messages sizes



SMT and TLS 1.3 security guarantees

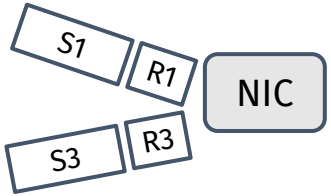
Authentication	TLS 1.3 handshake	**Same as TLS/TCP**
Confidentiality & Integrity	AEAD (AES-GCM); SMT also provides message integrity (unlike Homa)	**Same as TLS/TCP**
Order protection	Per-message via per-message record seqno; monotonic within message	**Per-message**
Non-replayability	Composite 48-bit msg ID + 16-bit record index ensures session uniqueness	**Guaranteed**
Length concealment	Compatible with TLS padding; msg length field can include padding	**Supported**

Out-of-order TLS offload

- Per-message record seqno is essential for TLS offload
  - NIC expects in-order record arrival
    - We can update NIC's expectation,
      but not atomically
  - Message-based transports send messages
    in any order

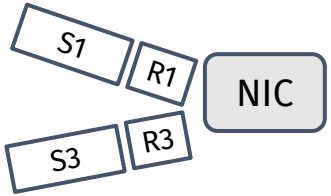
Out-of-order TLS offload

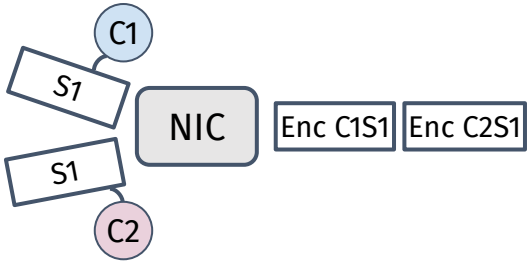
- Per-message record seqno is essential for TLS offload
  - NIC expects in-order record arrival
    - We can update NIC's expectation, but not atomically
  - Message-based transports send messages in any order



Out-of-order TLS offload

- Per-message record seqno is essential for TLS offload
  - NIC expects in-order record arrival
    - We can update NIC's expectation, but not atomically
  - Message-based transports send messages in any order
    - Per-message record seqnos allow the sender to use separate *flow contexts*



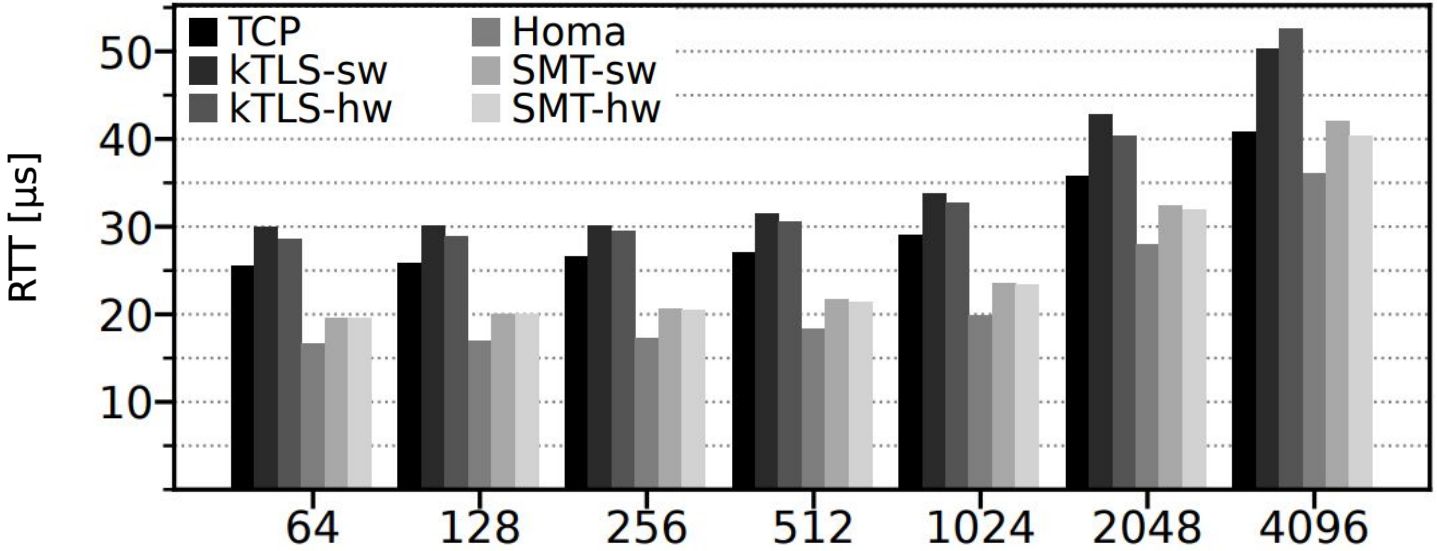


Implementation

- 2800 LoC patch on Homa/Linux kernel module
- 300 LoC patch on NVIDIA mlx5 driver
  - Driver change: adjust encryption offset + generalize flow context
- Open source: github.com/uoenoplab/smt
  - New version coming soon

Unloaded latency

- SMT outperforms kTLS by 21–32% with hw offload and 16–35% without it

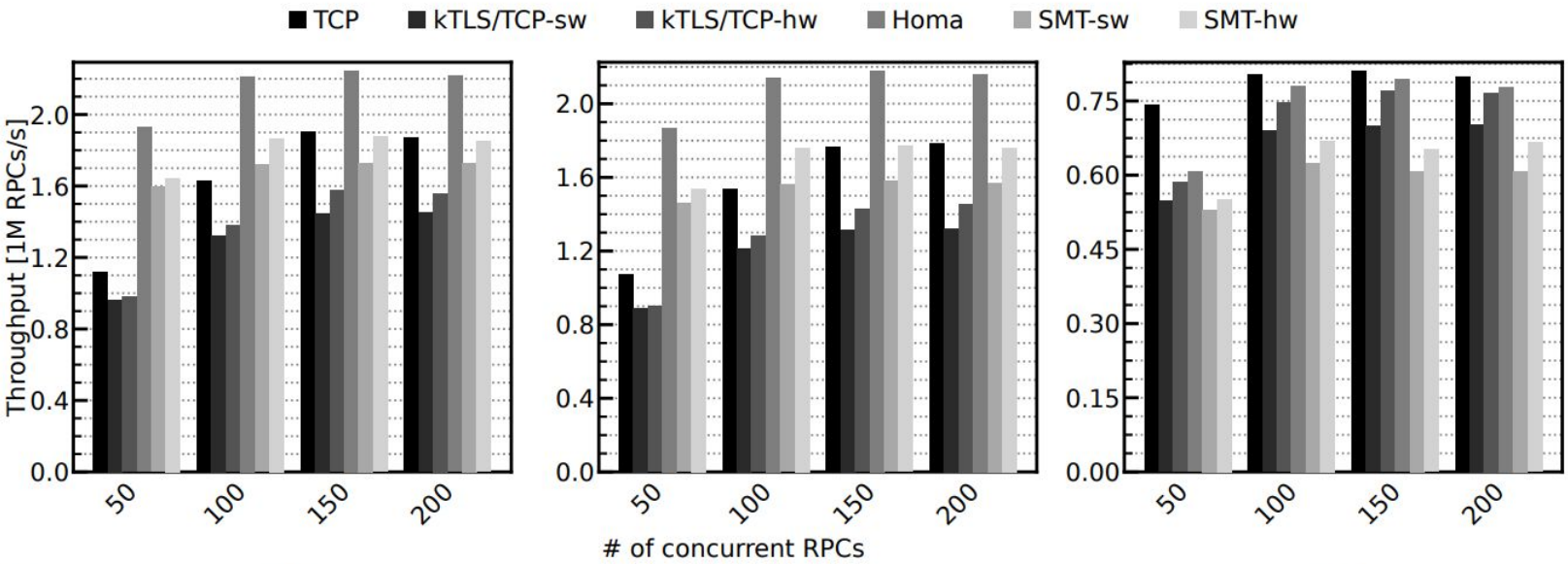


Loaded throughput

16-40% improvements

16-41% improvements

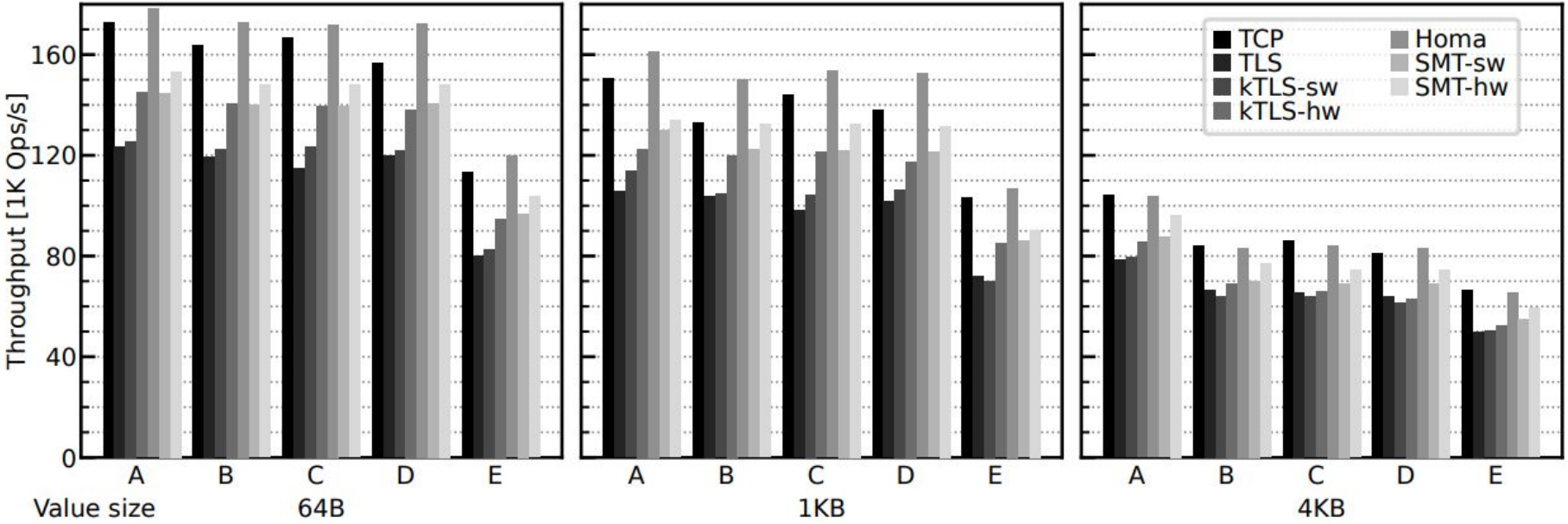
Homa is unoptimized large message throughput yet



RPC size 64B 1KB 8KB

Redis throughput

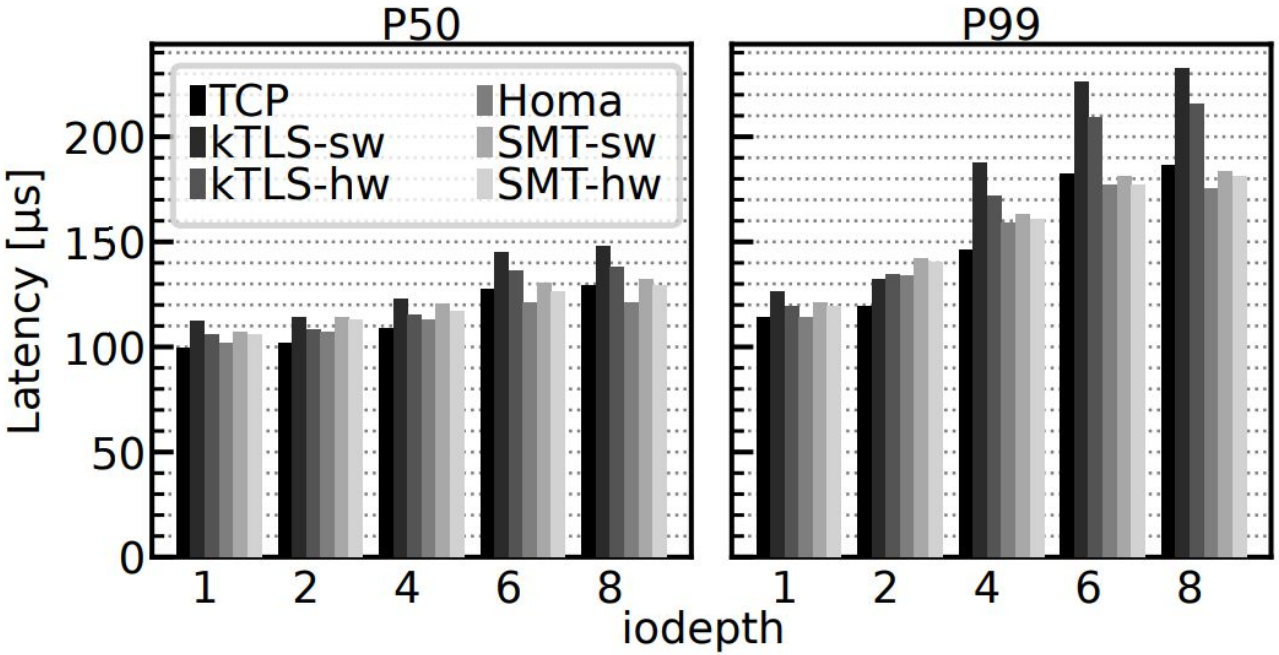
- SMT outperforms kTLS by 5–13 % with TLS offload and 8–17 % without it



Workload A: Update heavy; Workload C: Read only

NVMe-oF (In-kernel application) throughput

- P50 - up to 7 % (with TLS offload) or 15 % (without it) improvement
- P99 - up to 16 % (with TLS offload) or 21 % (without it) improvement



Discussion & future directions

Message integrity	Integrity via AEAD — Homa lacks checksum offload
IPv6 / Segmentation	No IPID in IPv6 — use TSO for up to 2 segments; hopefully NIC vendors embed “TCP” seqno to non-TCP packets
Receive-side offload	NIC doesn't route non-TCP packets to crypto engine; need more flexible/programmable parser
PSP support	Could support PSP in addition to TLS; challenges remain without NIC support
In-network compute	Compatible with MTP [NSDI’25] — plaintext msg ID/length enables boundary detection
Post-quantum	Inherits TLS 1.3 PQ resistance; NIC supports 256-bit keys; larger HW benefit

Conclusion

- SMT: Message-based, encrypted transport for datacenters
  - Linux kernel implementation as Homa/Linux extension
- Ongoing work
  - Improving implementation and performance
  - Detailed specification
    - targeting I-D submission for IETF 126

See more details in our IEEE S&P paper