Designing Transport-level Encryption For Datacenter Networks

Tianyi Gao, Xinshu Ma, Suhas Narreddy,
Eugenio Luo, Steven Chien, Michio Honda

University of Edinburgh

8th Aug 2025
APNet'25

THE UNIVERSITY of EDINBURGH
informatics

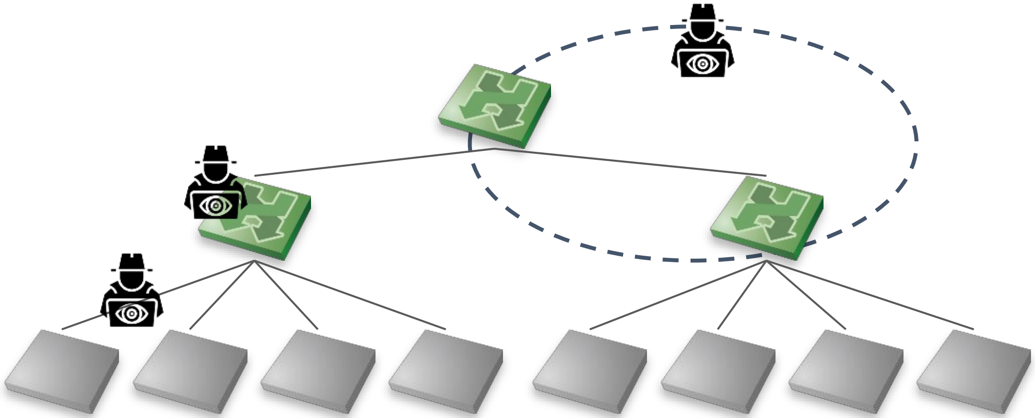
Background: Datacenter transport status quo

	2010-	2015-	2020-
Simulator/ DPDK		pHost [2]/dcPIM [3] /NDP [4]/Homa [5]	
Linux	DCTCP [1]		Homa/Linux [6]
HW			EQDS [7] Falcon

[1] Alizadeh et al, SIGCOMM'10 [2] Gao et al, CoNEXT'15 [3] Cai et al, SIGCOMM'22 [4] Hand
SIGCOMM'17 [5] Montazeri et al, SIGCOMM'18 [6] Ousterhout et al, ATC'21 [7] Olteanu et al,

Datacenters need end-to-end encryption

- Multi tenancy
- Third-party network/hardware/software on the path



Datacenter transports need message abstract

Efficient RPC (request-response) support is essenti

- Head-of-line blocking avoidance [1]
  - Unordered message delivery
- In-network compute (INC) support [2]
  - e.g., Per-message load balancing
  - Network needs message-level buffering with bounded time
- In-host load balancing [1]
  - Flow-based CPU core affinity creates CPU hotspots

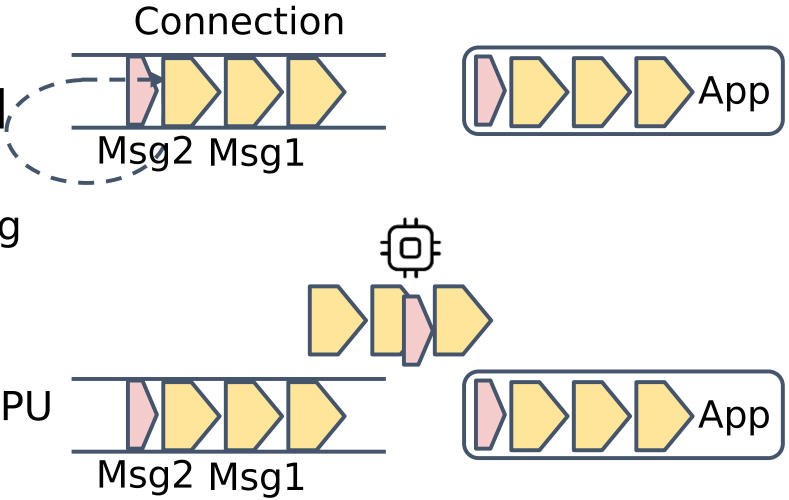
[1] Homa/Linux (ATC’21) [2] MTP (NSDI’25)

Datacenter transports need message abstract

Efficient RPC (request-response) support is essenti

- Head-of-line blocking avoidance [1]
  - Unordered message delivery
- In-network compute (INC) support [2]
  - e.g., Per-message load balancing
  - Network needs message-level buffering with bounded time
- In-host load balancing [1]
  - Flow-based CPU core affinity creates CPU hotspots

In-order bytestream is unfit



[1] Homa/Linux (ATC'21) [2] MTP (NSDI'25)

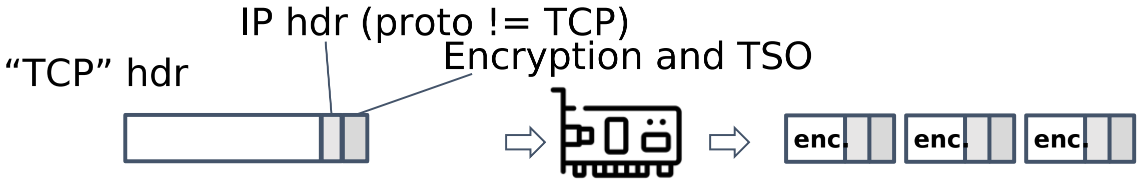
Design space

	Encrypt.	Abstract.	Offload	Protocol	Parallelism	
TcpCrypt[4]	TcpCrypt	Stream	N	TCP	Conn.	
QUIC[25]	TLS	Stream	N	UDP	Conn.	
TCPLS[53]	TLS	Stream	TSO	TCP	Conn.	
TLS/TCP[45]	TLS	Stream	Crypto+TSO	TCP	Conn.	
SDP	TLS	Msg.	Crypto+TSO	New	Msg.	
Homa[40]/NDP[21]	-	Msg.	TSO	New	Msg.	
MTP[62]	-	Msg.	TSO	UDP	N/A	
Falcon [12]	PSP	Msg.	Full	UDP	Msg.	Custom NIC
SRD[58]	-	Dgram.	Full	N/A	Packet	Custom NIC
KCM[28]/μTCP[38]	-	Msg.	TSO	TCP	Conn.	

- Crypto offload with commodity NICs
  - No compromise from TLS/TCP
- Native transport
  - Flexible protocol design and easy network management

Key question - can we use existing TLS offloac

- Autonomous offload [1] (NVIDIA ConnectX-6/7)
  - Mainstream today
  - Likely similar architecture in Fungible (Microsoft) and Netronome NICs
- **It works for non-TCP!**



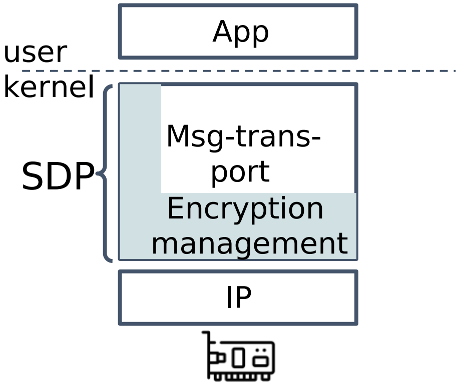
[1] Pismenny et al, ASPLOS'21 [2] https://docs.kernel.org/networking/tls-offloa

SDP overview

- One-to-many style socket

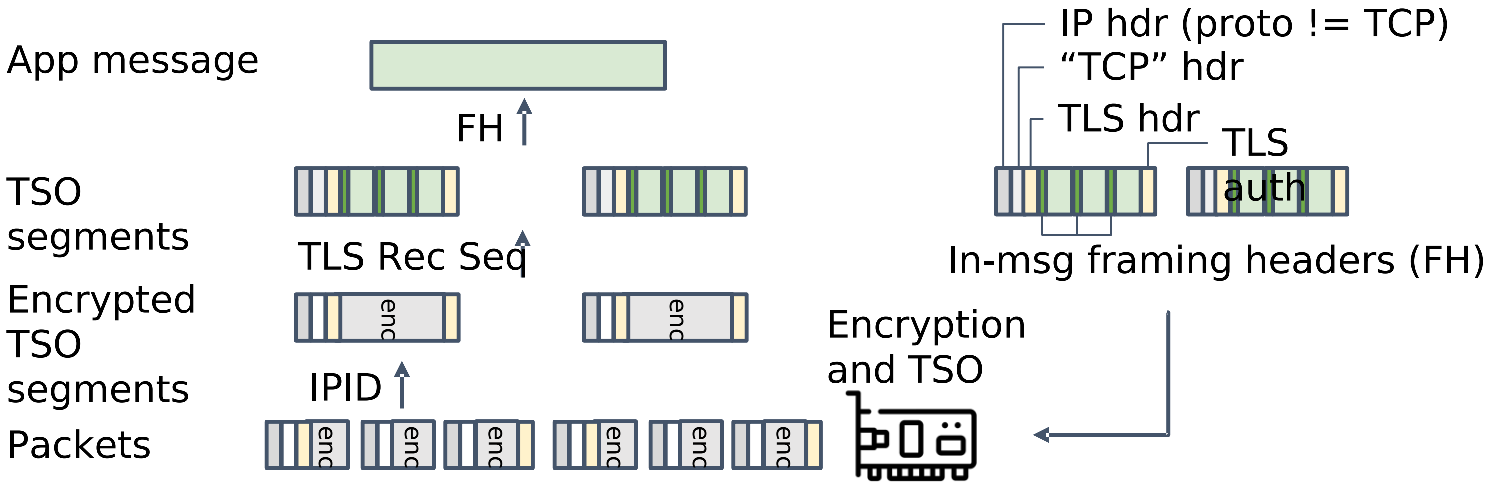
```
fd = socket(SOCK_DGRAM, IPPROTO_HOMA) // but reliable
setsockopt(fd, key) // key handshake already performed, like kTLS
sendmsg(fd, msg, dst) // or io_uring_prep_sendto(sqe, fd, msg, dst)
```

- Transport-level encryption
  - Transport protocol must be aware of encryption, unlike TLS
- Opportunistic HW offload
- Optional 0-RTT handshake
- ~2800 LoC change in Homa/Linux
- ~300 LoC change in the mlx5 driver
- Support Linux 6.2 and 6.6



Two-level segmentation

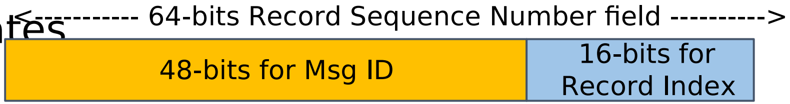
- An app message can consist of multiple TSO segments
  - Example below: one app message over two TSO segments
- A TSO segment can consist of multiple packets

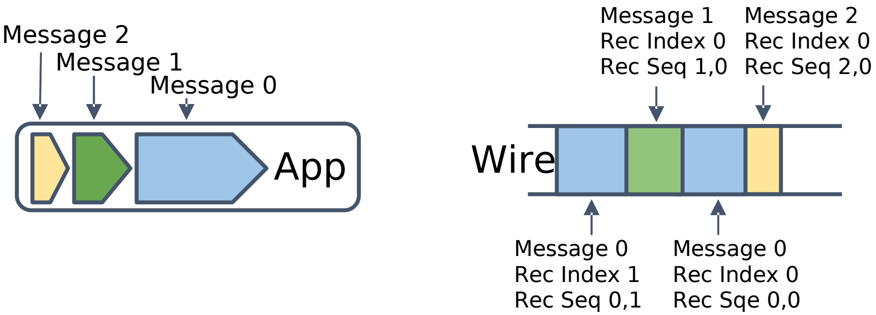


Per-message record sequence number space

- Granularity of parallelism
  - TCP (Connection-level) - strict in-order delivery
  - SDP (Message-level) - out-order delivery at both message level and segment le
    - A later message or segment in message can received earlier
    - Global incrementing record sequence number no longer work

Solution: a record sequence number that integrates a message ID with an intra-message record index

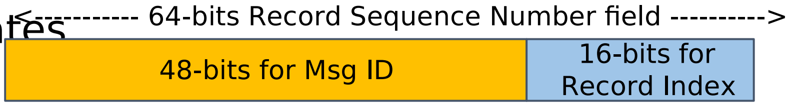


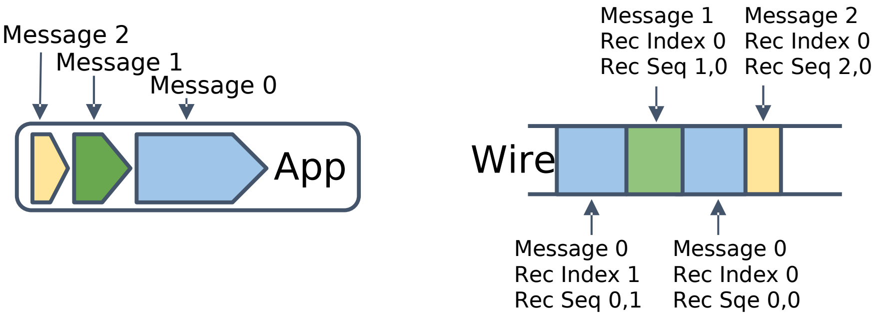


Per-message record sequence number space

- Messages can reuse one hardware crypto engine by sharing the record sequence number
- Different messages can be sent and received independently with 48-bits Message
- Unique record sequence number for all records across and inside messages to prevent replay attack

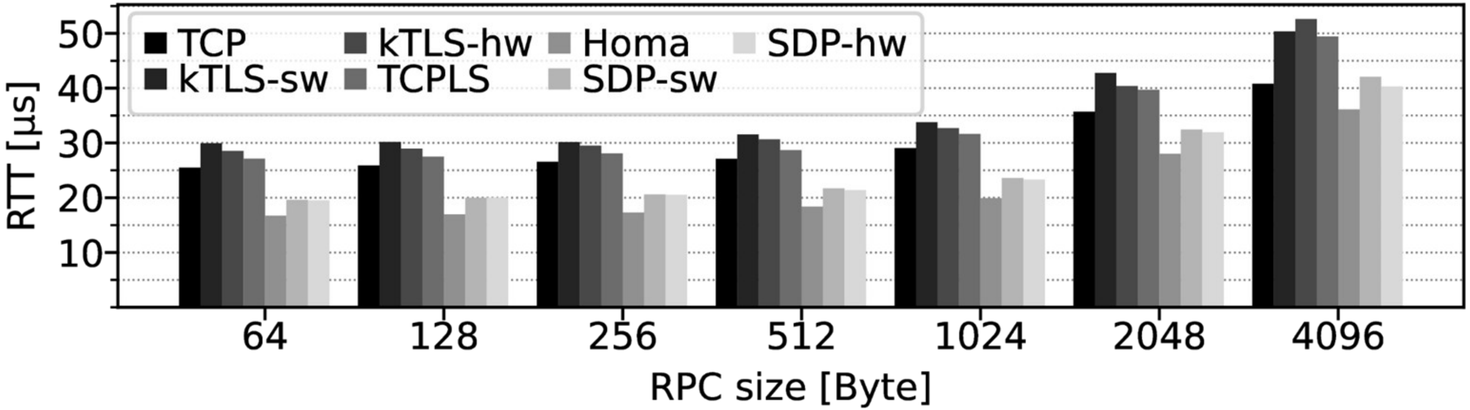
Solution: a record sequence number that integrates a message ID with an intra-message record index





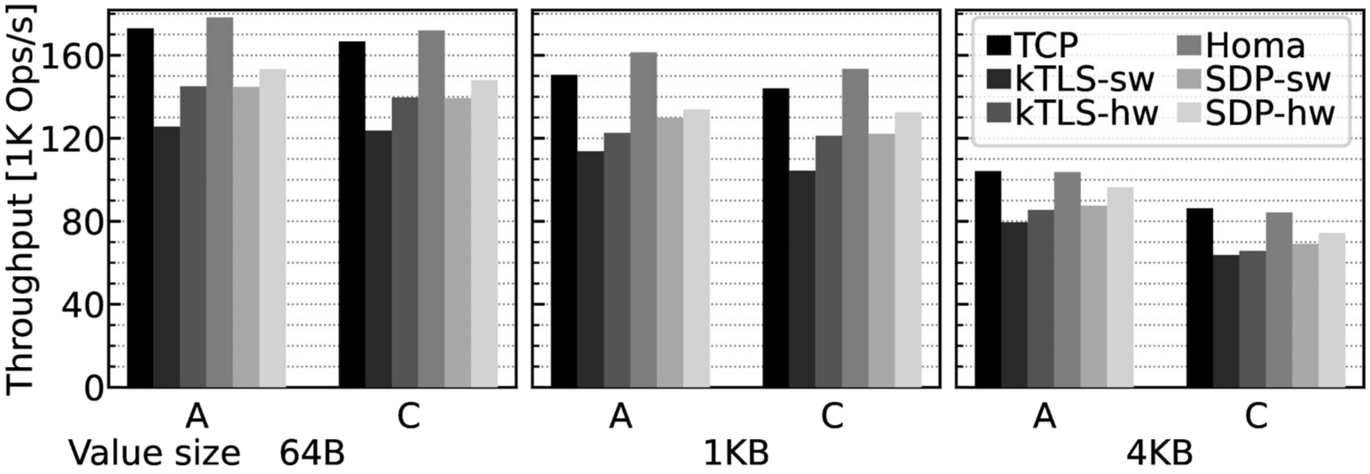
Unloaded latency

- SDP outperforms kTLS by 21–32% with hw offload and 16–35% without it
    - Homa is faster than TCP by 5–35 %



Redis throughput

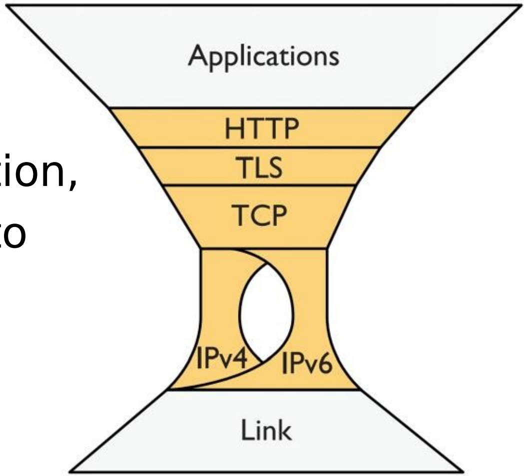
- SDP outperforms kTLS by 5–13 % with TLS offload and 8–17 % without it



Workload A: Update heavy
Workload C: Read only

Implications

- Opportunity for (proper) evolution of transport in datacenters
  - Internet: TCP-as-a-substrate philosophy for middleboxes
  - Datacenter transports can still evolve
- Is Homa/Linux a right basis?
  - Generic and documented enough for abstraction, packet format, and reasonable performance to build other receiver-driven protocols



Trammell, B. et al., (2014). Evolving transport in the Internet. IEEE Internet Computing, 18(5), 60-64.

Thank you!
Any questions?