Designing transport-level encryption for datacenter networks

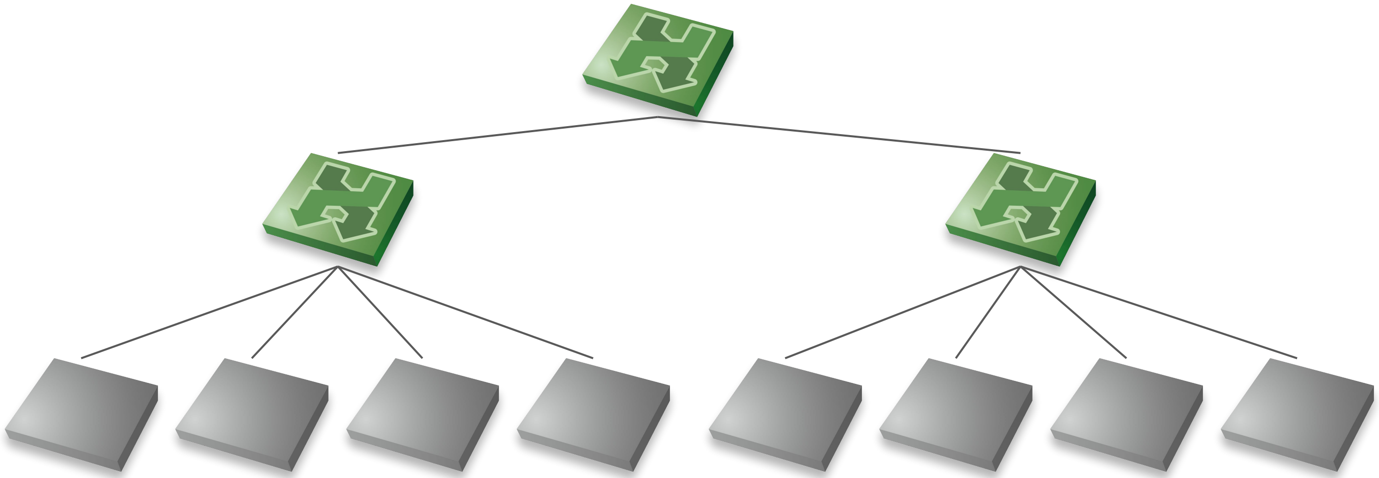
Michio Honda
University of Edinburgh

THE UNIVERSITY of EDINBURGH informatics

On behalf of my research group members.

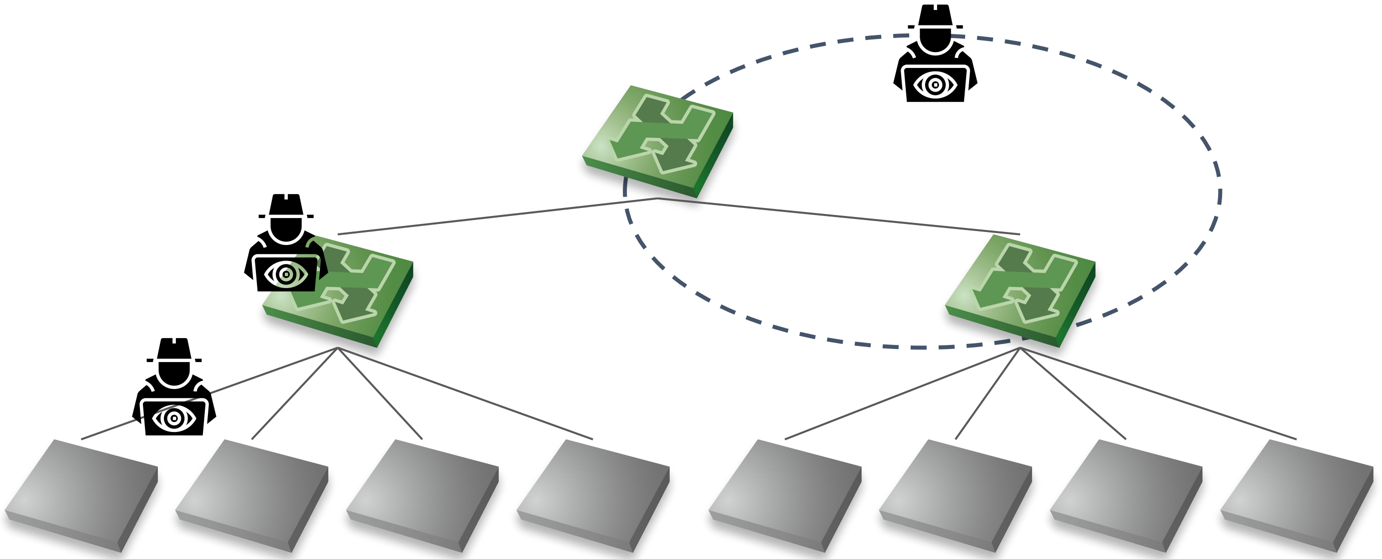
The role of datacenter transport today

- Apps want all of high throughput, low latency & many CPU cycles
- Datacenter transports need good
  - End-to-end congestion control
  - Host stack
  - Switch service



Datacenters need end-to-end encryption

- Multi tenancy
- Multi-vendor hardware/software network components



Datacenter transport requirements

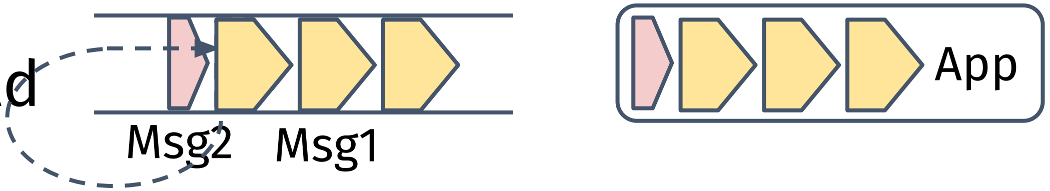
- Modern transport requirements are complex
  - ***Radically new transport beyond TCP***
    - 0-RTT data, receiver-driven congestion control, message boundaries
  - ***Hardware offload***
    - Leaving CPU cycles to the apps
  - ***In-network compute***
    - Load balancing, congestion signaling and routing

Can we design secure datacenter transport without sacrificing those properties?

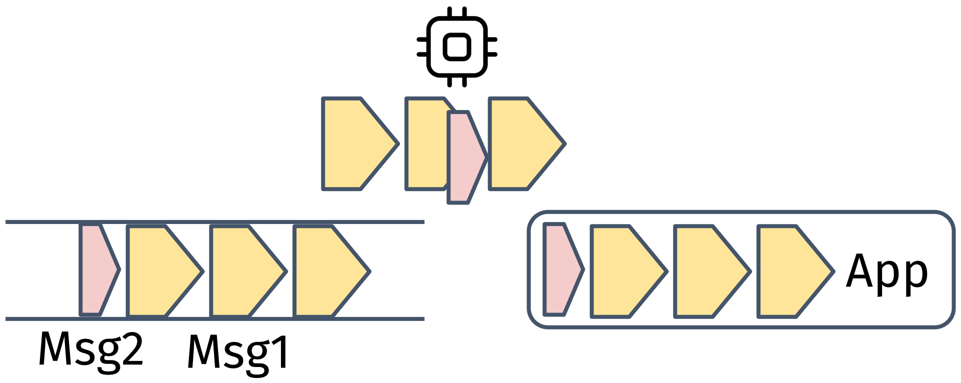
Limitation with bytestream abstractions

- TCP/QUIC
- Head of Line Blocking
  - Early-arriving small messages should be delivered first

On a packet loss



On a CPU hotspot



Problems with RDMA abstractions

- Google/Intel Falcon, AWS Scalable Reliable Datagram (SRD)
- Unordered *packet* delivery is supported
  - **NOT** unordered *message delivery*

Design space: Transport-level encryption

Encrypted transports

Message transports

	Encrypt.	Abstract.	NIC offload	Wire proto.	Host LB	In-net com.	
TcpCrypt[3]	Inline	Stream	N	TCP	Conn.	N	
QUIC[19]	TLS	Stream	N*	UDP	Conn.	N	*FPGA NIC attempt [52]
TLS/TCP[32]	TLS	Stream	Crypto+TSO	TCP	Conn.	N	
Falcon [8]	PSP	Ordered conns.*	Crypto+TSO**	UDP	Conn.	N	*RDMA verbs **Custom NIC or Intel IPU
SDT	TLS	Msg.	Crypto+TSO	New	Msg.	Y*	*With shared key for data mutation [46]
Homa[29]/NDP[14]	-	Msg.	TSO	New	Msg.	Y	
MTP[46]	-	Msg.	TSO	UDP	-	Y	
SRD[43]	-	Dgram.	Full*	Unknown	-	Y	*Custom NIC
KCM[21]/μTCP[27]	-	Msg.*	TSO	TCP	Conn.	Y*	*high overheads

Table 1: Key properties of encrypted or message-based transports.

Middleground: Unencrypted message-based transport

	Encrypt.	Abstract.	NIC offload	Wire proto.	Host LB	In-net com.	
TcpCrypt[3]	Inline	Stream	N	TCP	Conn.	N	
QUIC[19]	TLS	Stream	N*	UDP	Conn.	N	*FPGA NIC attempt [52]
TLS/TCP[32]	TLS	Stream	Crypto+TSO	TCP	Conn.	N	
Falcon [8]	PSP	Ordered conns.*	Crypto+TSO**	UDP	Conn.	N	*RDMA verbs **Custom NIC or Intel IPU
SDT	TLS	Msg.	Crypto+TSO	New	Msg.	Y*	*With shared key for data mutation [46]
Homa[29]/NDP[14]	-	Msg.	TSO	New	Msg.	Y	
MTP[46]	-	Msg.	TSO	UDP	-	Y	
SRD[43]	-	Dgram.	Full*	Unknown	-	Y	*Custom NIC
KCM[21]/μTCP[27]	-	Msg.*	TSO	TCP	Conn.	Y*	*high overheads

Table 1: Key properties of encrypted or message-based transports.

Middleground: Unencrypted message-based transport

- Homa*
  - Active development in Linux
  - General to transform to other protocols like NDP
- MTP**
  - Similar to Homa
  - Introduction of in-network compute
    - Load balancing, multipath, congestion signalling, data mutation

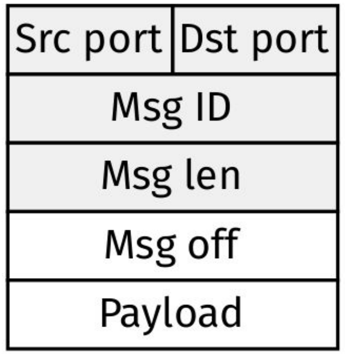


Figure 1: Generalized message-based transport packet format based on MTP [45] and Homa [29]. Shaded parts are identical between the packets that belong to the same message.

* Ousterhout et al, ATC'21
** Stephens et al, HotNets'21

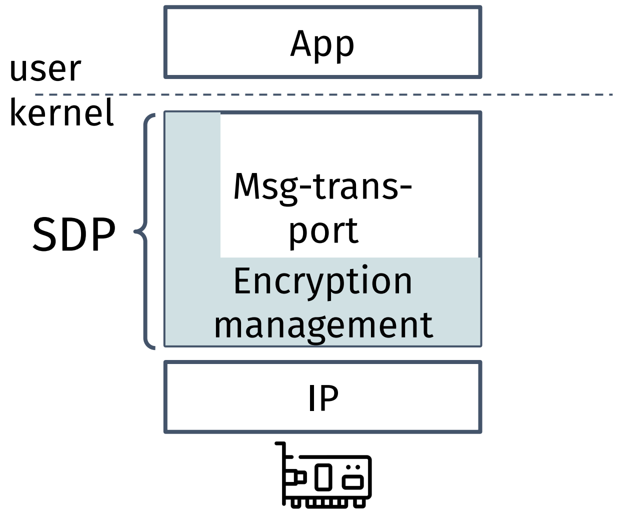
SDP Overview

	Encrypt.	Abstract.	NIC offload	Wire proto.	Host LB	In-net com.	
TcpCrypt[3]	Inline	Stream	N	TCP	Conn.	N	
QUIC[19]	TLS	Stream	N*	UDP	Conn.	N	*FPGA NIC attempt [52]
TLS/TCP[32]	TLS	Stream	Crypto+TSO	TCP	Conn.	N	
Falcon [8]	PSP	Ordered conns.*	Crypto+TSO**	UDP	Conn.	N	*RDMA verbs **Custom NIC or Intel IPU
SDT	TLS	Msg.	Crypto+TSO	New	Msg.	Y*	*With shared key for data mutation [46]
Homa[29]/NDP[14]	-	Msg.	TSO	New	Msg.	Y	
MTP[46]	-	Msg.	TSO	UDP	-	Y	
SRD[43]	-	Dgram.	Full*	Unknown	-	Y	*Custom NIC
KCM[21]/μTCP[27]	-	Msg.*	TSO	TCP	Conn.	Y*	*high overheads

Table 1: Key properties of encrypted or message-based transports.

SDP overview

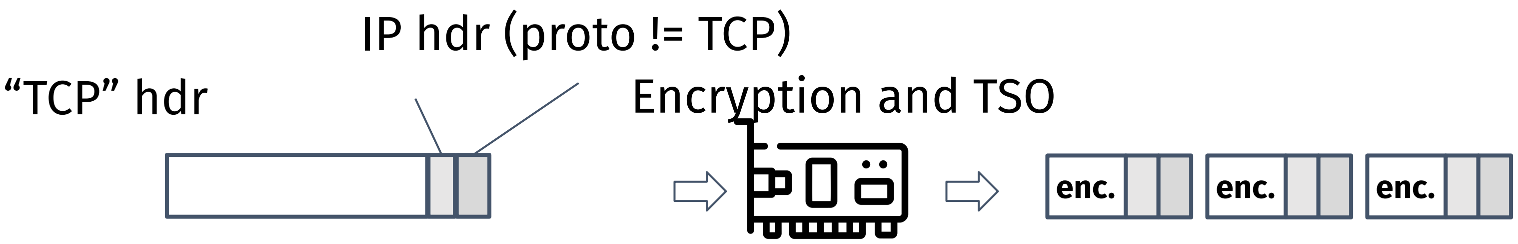
- Transport-level encryption for datacenter networks
- Message level transport
  - In-network computing support
    - Even data mutation with key sharing
- Opportunistic NIC offload
  - Commodity NVIDIA CX6/7 NICs
- Transport protocol number agnostic
  - Co-existence with existing traffic
- Optional 0-RTT handshake



- ~2800 LoC change in Homa/Linux
- ~300 LoC change in the `mlx5` driver
- Support Linux 6.2 and 6.6

TLS offload with commodity NICs

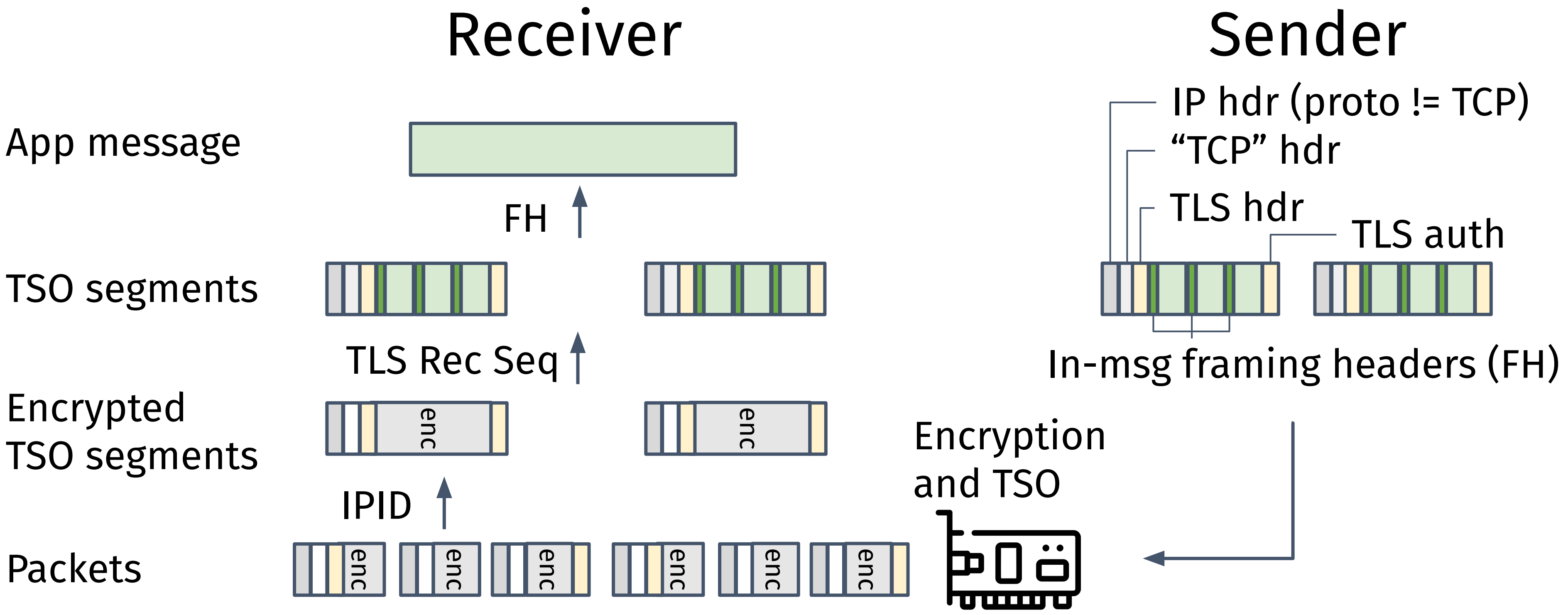
- It is a deal breaker to be able to use existing HW offload
- Full TOE-based approach (Chelsio T6)
  - Bad even for TCP (e.g., options are gone) and unfavored by operators*
- Autonomous offload* (NVIDIA ConnectX-6/7)
  - Mainstream today
  - Likely similar architecture in Fungible (Microsoft) and Netronome NICs
- **It works for non-TCP!**



* Pismenny et al, ASPLOS'21

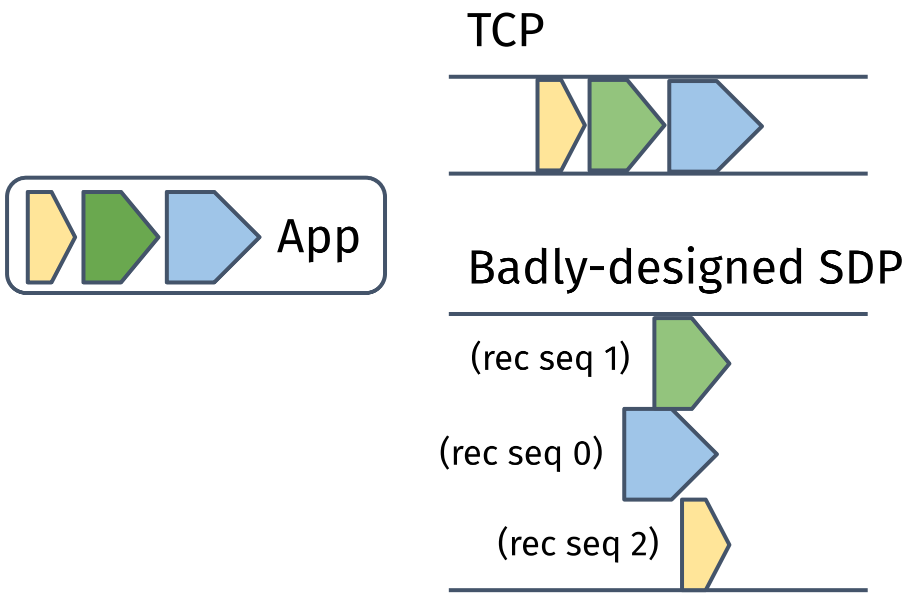
Any-size, unordered authenticated message

- 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



Message-level parallelism

- Granularity of parallelism
  - TCP (Connection-level) - strict in-order delivery
  - SDP (Message-level) - out-order delivery at message level
    - A later message can be received earlier
    - Global record sequence number (over TCP bytestream) no longer works

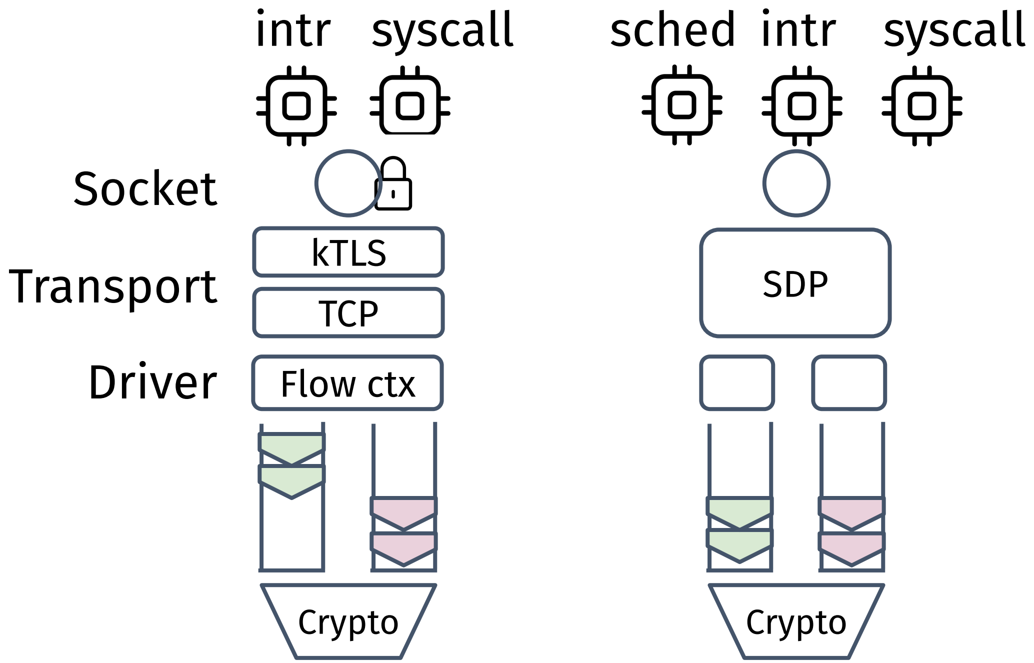


Result A:
Receiver decrypts record sequence 2 with expecting record sequence 0 **->** decrypt failure

Result B:
Receiver waits for record sequence 0 even other records are received **->** Head-of-line blocking

NIC offloading

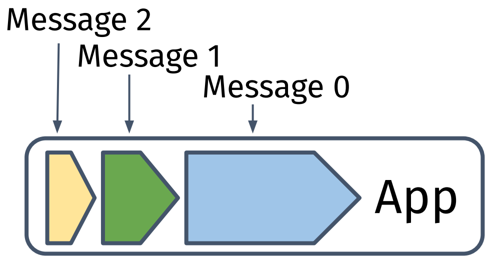
- NICs expect all the data is serialized
  - Under socket lock for TCP
- Message-based transports send multiple messages in parallel in the same flow



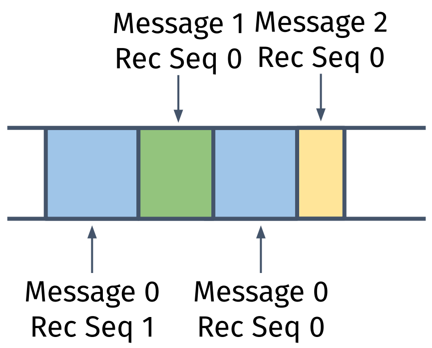
Message-level parallelism

- Granularity of parallelism
  - TCP (Connection-level) - strict in-order delivery
  - SDP (Message-level) - out-order delivery at message level
    - A later message can be received earlier
    - Global record sequence number (over bytestreawm) no longer work

Solution Assign unique record sequence space to each message

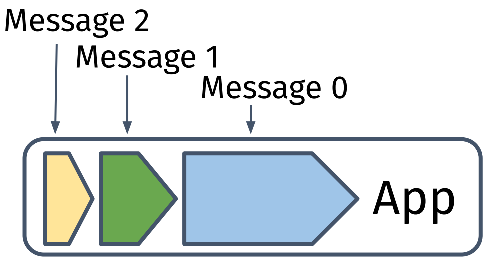


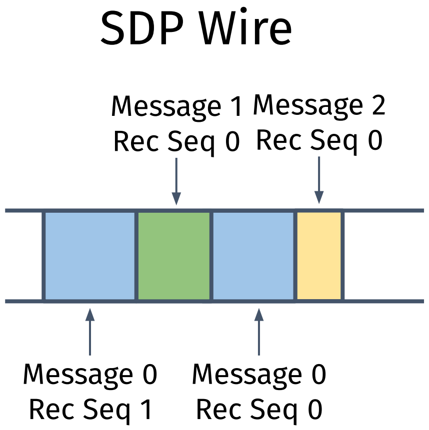
SDP Wire



Replay attack protection

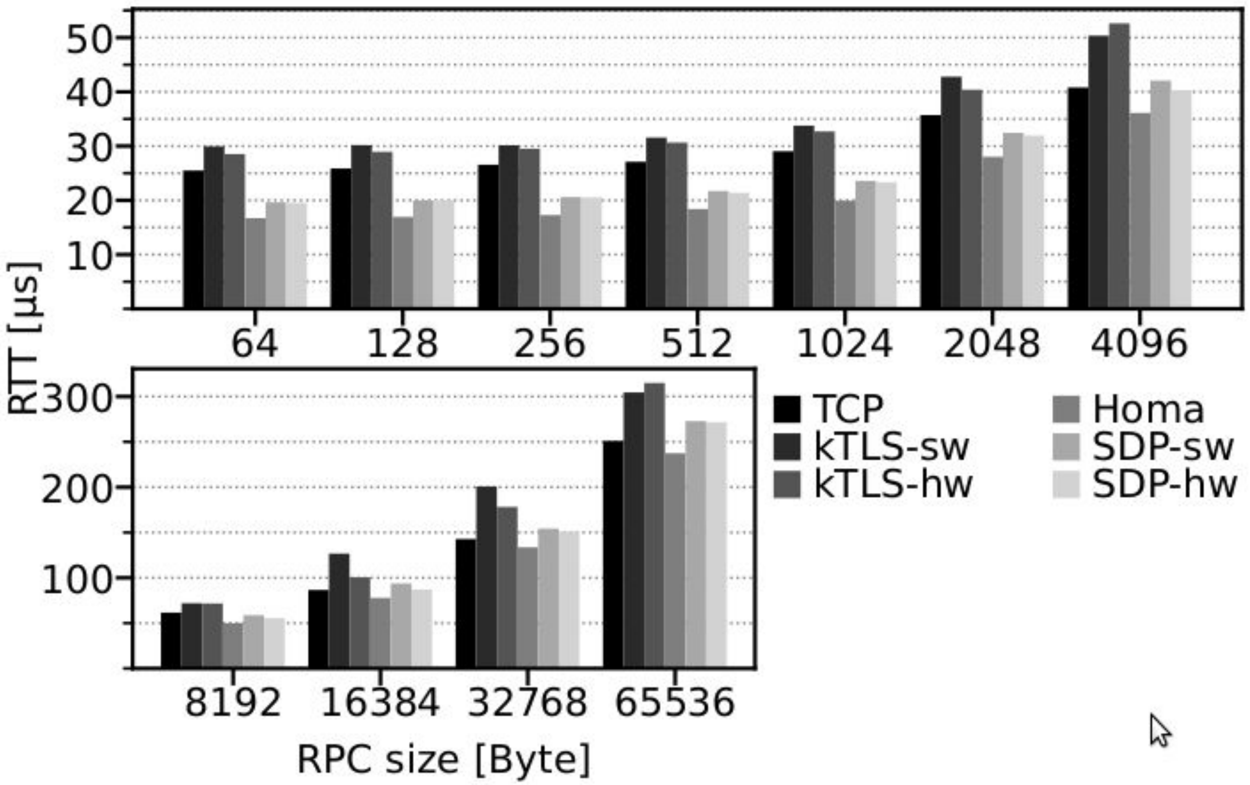
- Intra-message: Record sequence numbers increment sequentially like normal TLS
- Inter-Message: Unique message ID used only once in the authenticated session





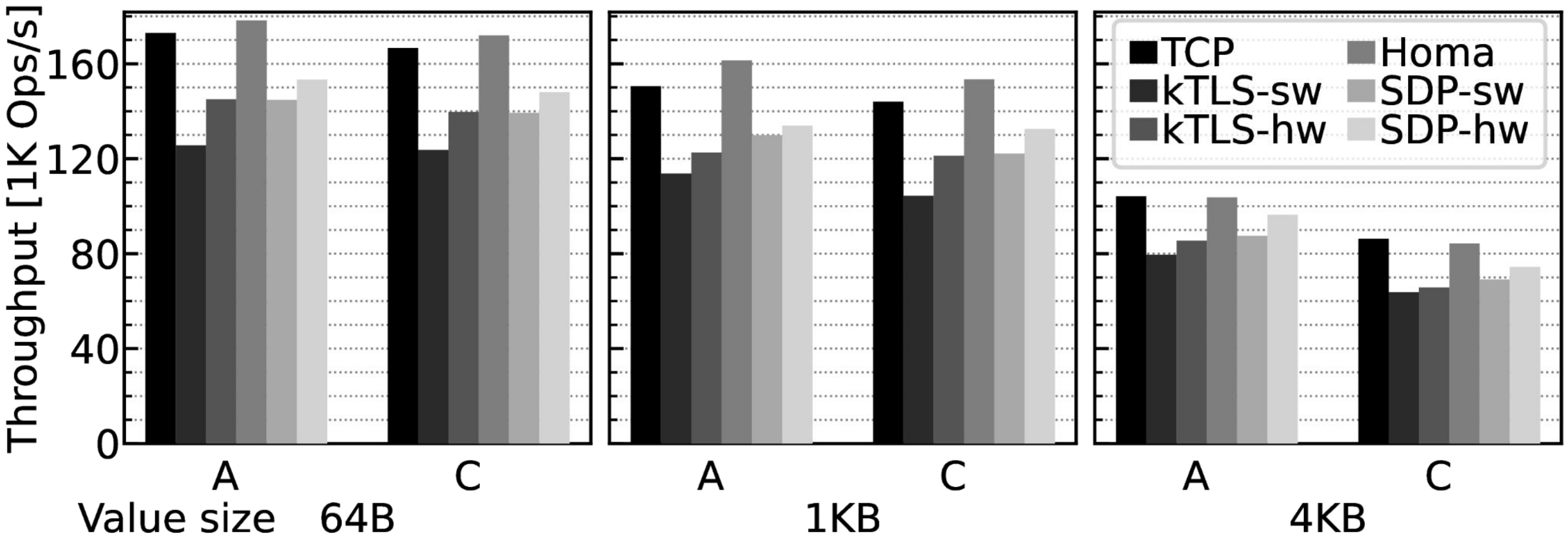
Unloaded latency

- SDP outperforms kTLS by 13–32% with hw offload and 10–35% 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

Summary

- We need security in datacenter networks
- Challenging to preserve important transport properties today:
  - NIC offloading
  - Departure from TCP
  - In-Network Computing support

  while preserving the same threat model as TLS/TCP
- SDP solves it
  - Existing TLS NIC offload
  - Arbitrary-sized, encrypted message
  - Same threat model as TLS/TCP
  - Protocol number agnostic