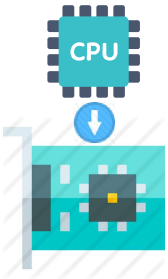
Dissecting QUIC Implementation Performance

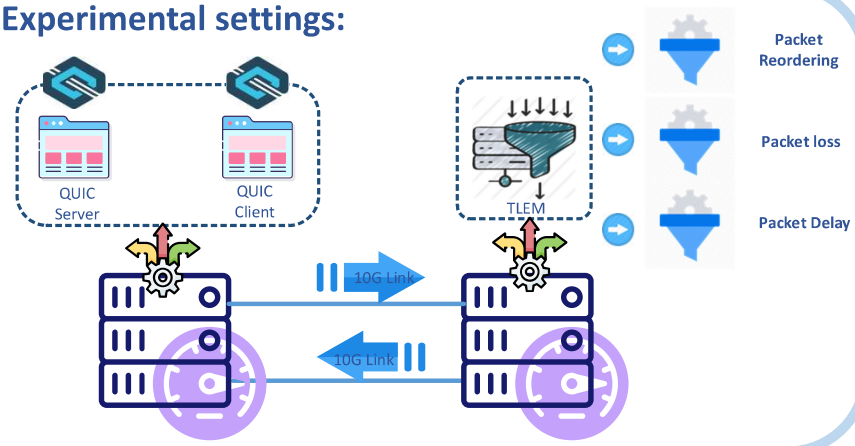
Xiangrui Yang[1], Lars Eggert[2], Jörg Ott[3], Steve Uhlig[4], Zhigang Sun[1], Gianni Antichi[4]

National University of Defense Technology[1], NetApp[2], Technical University of Munich[3], Queen Mary University of London[4]

Goal: What are the **primitives in QUIC** that should be offloaded onto **SmartNICs**?

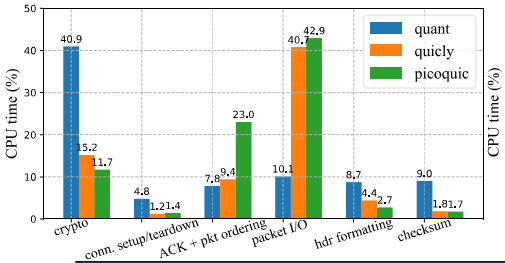


Experimental settings:

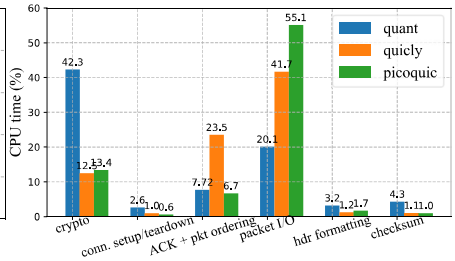


Lesson #1: use kernel-bypass and offload crypto operations.

a. CPU breakdown (client)



b. CPU breakdown (server)



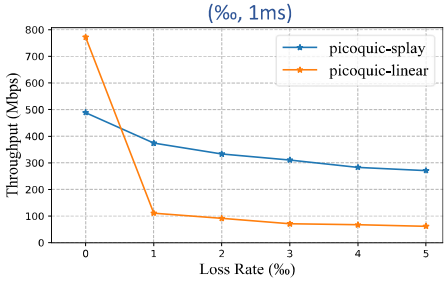
	QUANT	QUICLY	PICOQUIC
Throughput	3696Mbps	463Mbps	489Mbps
CPU usage	58%	54.8%	60.4%

With **kernel-bypass**, quant reaches **7x throughput** than quicly & picoquic.

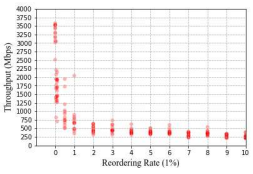
- w/o kernel bypass, packet I/O costs more than **40%** of CPU overhead.
- w/ netmap, crypto operations cost more than **40%** of CPU overhead.

Lesson #2: offload the per-packet reordering process.

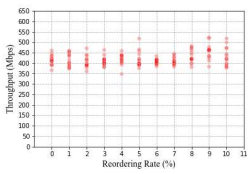
Throughput under pkt reordering **splay tree** vs. **linear searching** (‰, 1ms)



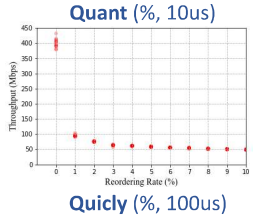
Tradeoffs on sw:
w/o pkt reordering/loss, picoquic with **linear search** performs **1.5x** better;
w/ only **1‰** pkt reordering/loss, picoquic with **splay tree** performs **3x** better.



Quant (%, 10us)



Picoquic (%, 100us)



Quicly (%, 100us)

Some more results of pkt reordering ...

Ongoing work

Measurement: multi-conn scenarios → **FPGA Architecture:** For QUIC acceleration → **packet reordering on hw:** PIFO vs PIEO?

NetApp · TECHNISCHE UNIVERSITÄT MÜNCHEN · Queen Mary University of London