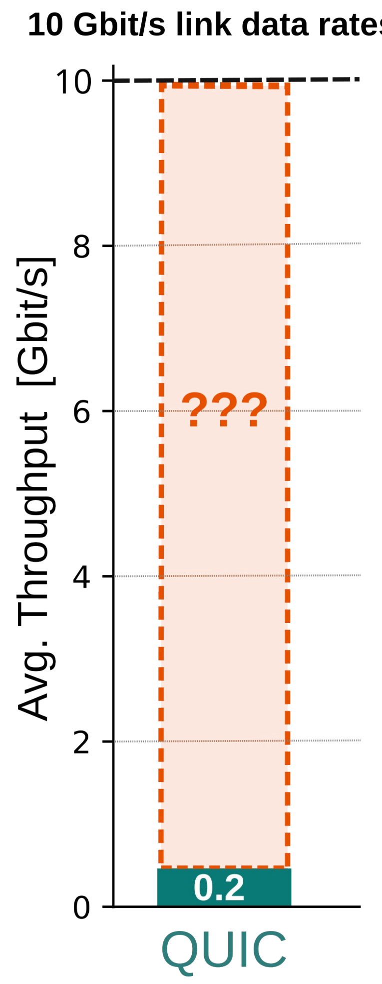
Sustained Throughput Performance of QUIC Implementations

Michael König[*], Oliver P. Waldhorst[‡], Martina Zitterbart[*]

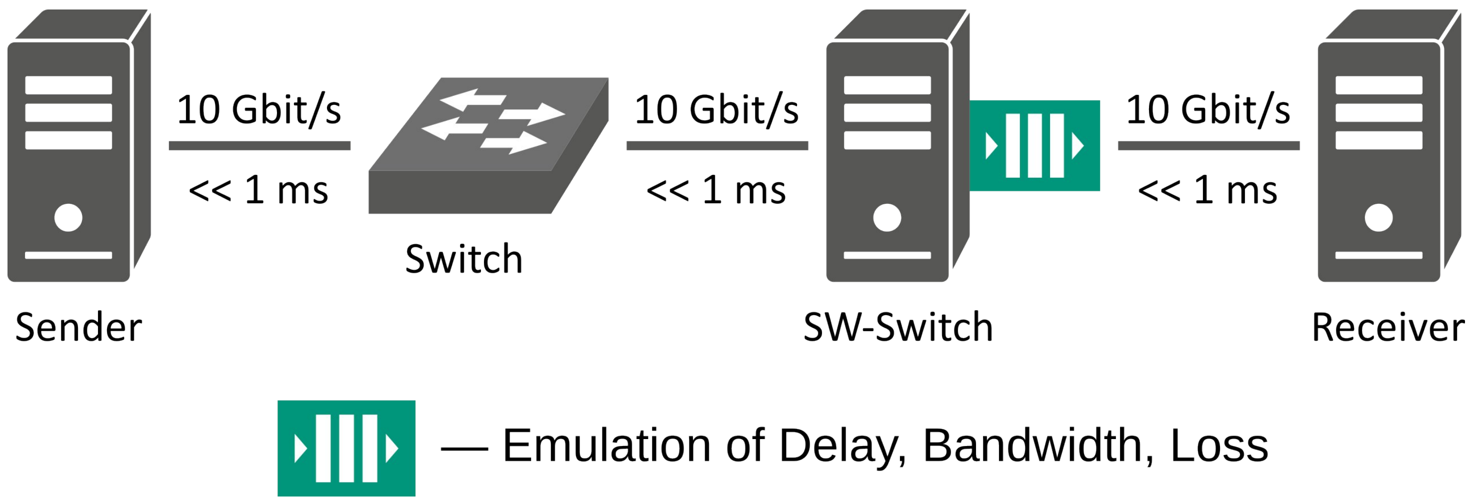
[*]Institute of Telematics, Karlsruhe Institute of Technology (KIT), {m.koenig, martina.zitterbart, roland.bless}@kit.edu
[‡]Institute of Applied Research, Karlsruhe University of Applied Sciences (H-KA), oliver.waldhorst@h-ka.de

Motivation

- "QUIC is a secure general-purpose transport protocol.“ [RFC9000]
- Our research indicated slow throughput performance:
  A QUIC-based prototype achieved ~200 Mbit/s
  on a 10 Gbit/s capable testbed…
- Related work
  - Primarily focused on latencies and flow completion times
  - Only few prior evaluations on
    sustained throughput in
    high bandwidth environments



Evaluation Setup



Setup Sender, SW-Switch, Receiver:

- CPU: Intel Xeon W-2145, 3.7–4.5 GHz, 8 Cores
- RAM: 128 GB (4x 32 GB DDR4 with 2666 MT/s)
- NIC: Intel X550-T2 (10 Gbit/s)
- OS: Linux Ubuntu 22.04.1 LTS, Kernel 5.15.0-56

Evaluated Implementations

Six popular QUIC implementations with traffic generators (perf clients) available

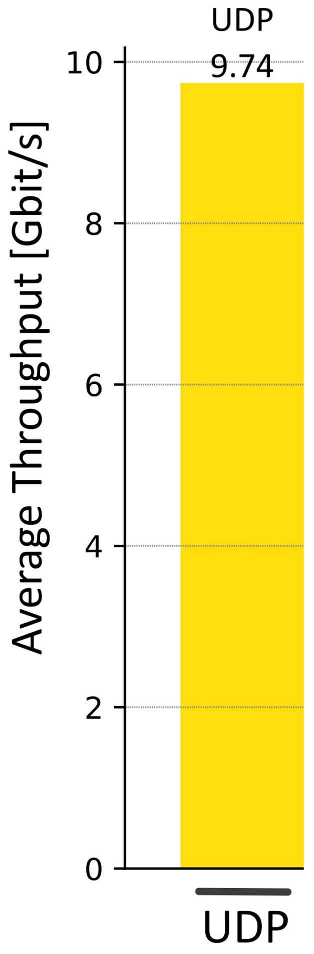
- lsquic (Litespeed)
- msquic (Microsoft)
- mvfst (Facebook)
- s2n-quic (Amazon)
- picoquic
- quinn

TCP and (pure) UDP as comparison

- iperf3
- netperf

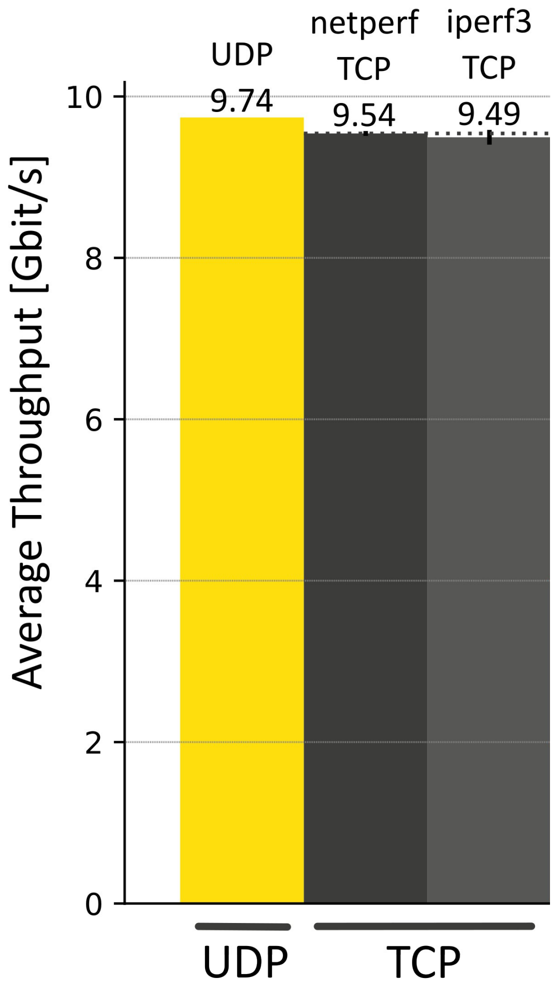
(For all TCP and QUIC traffic: Cubic as congestion control algorithm)

Results: Sustained Throughput



Average throughput of one single flow (10 runs, each 30s)

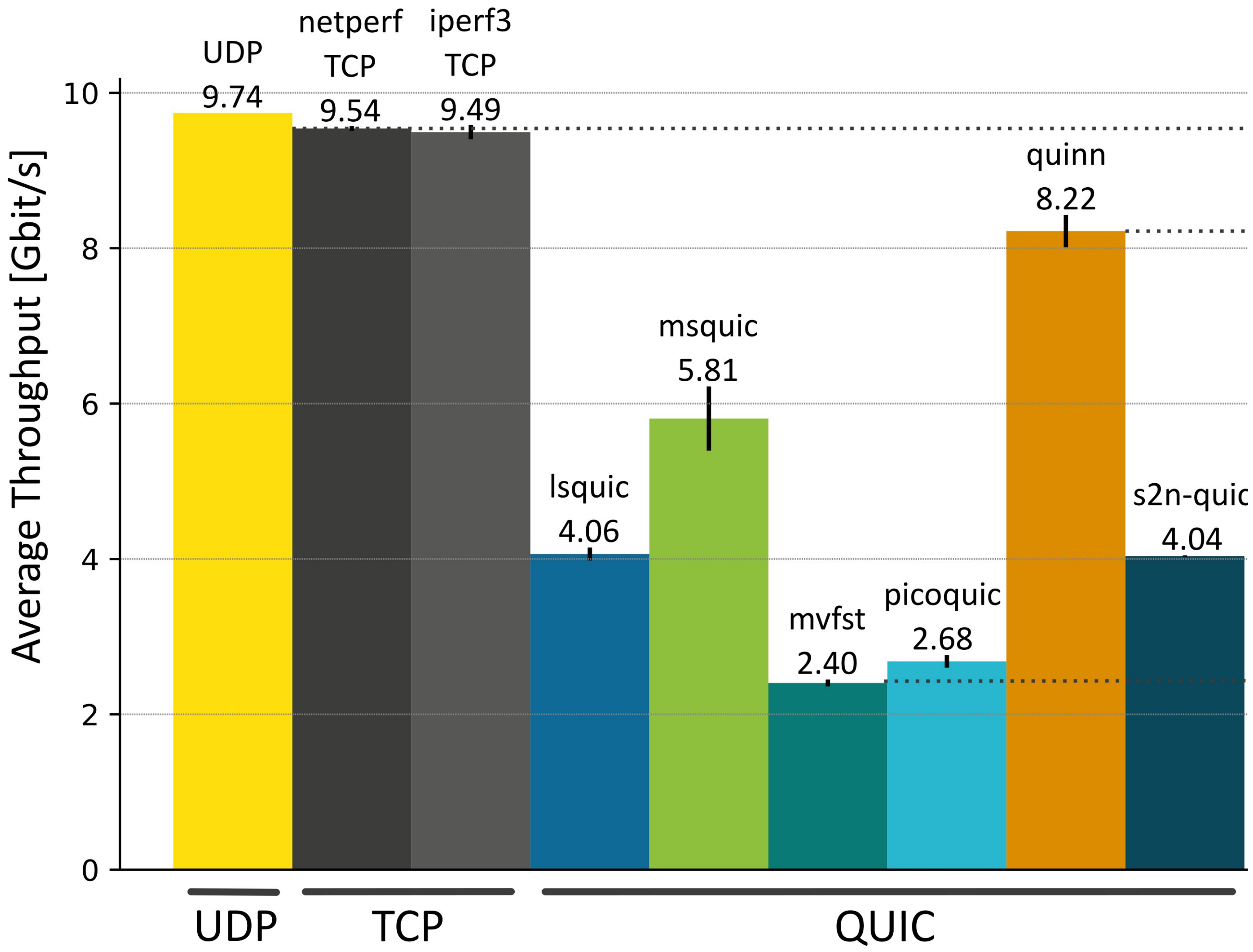
Results: Sustained Throughput



Average throughput of one single flow (10 runs, each 30s)

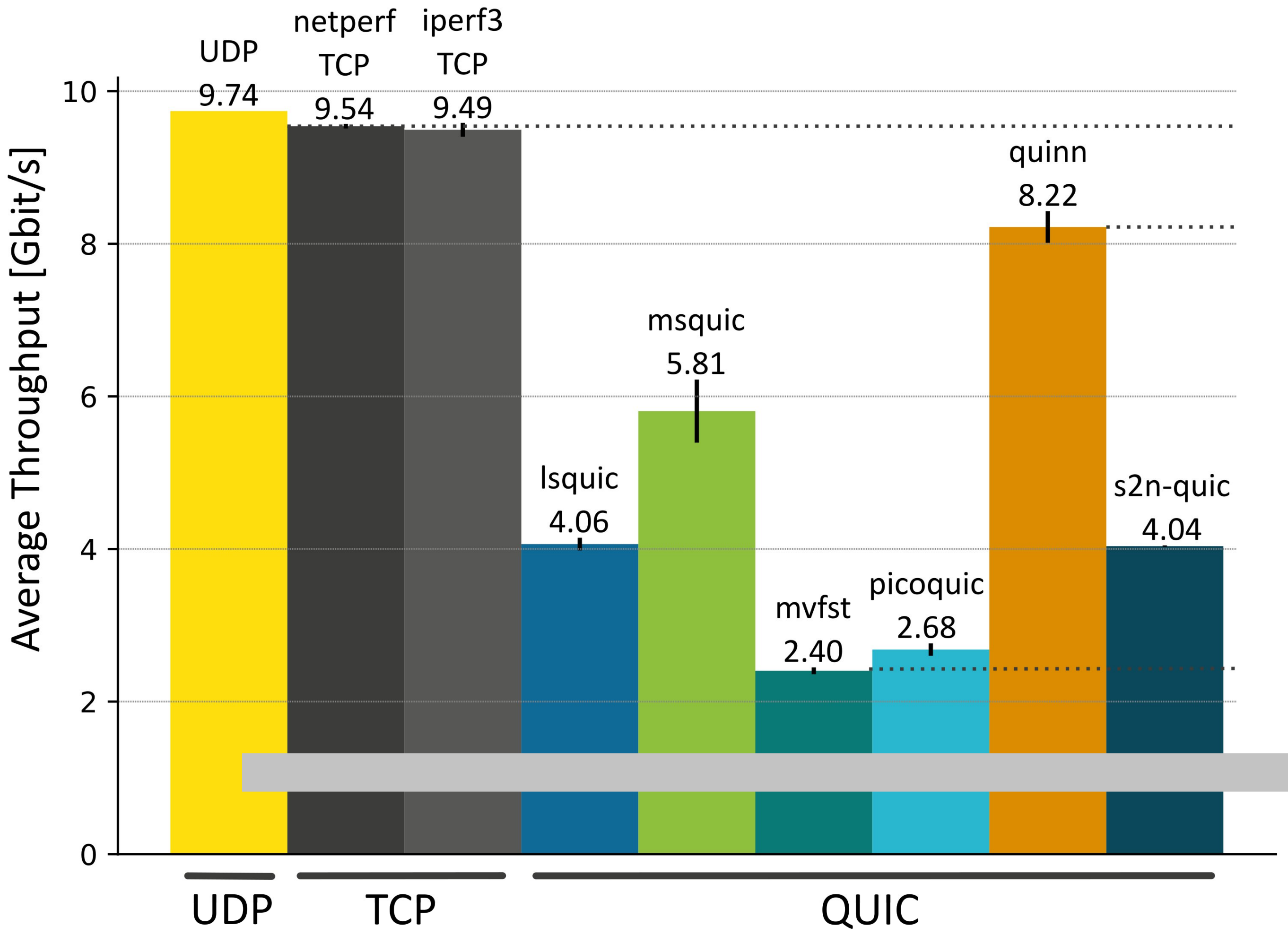
Results: Sustained Throughput

Karlsruhe Institute of Technology



Average throughput of one single flow (10 runs, each 30s)

Results: Sustained Throughput

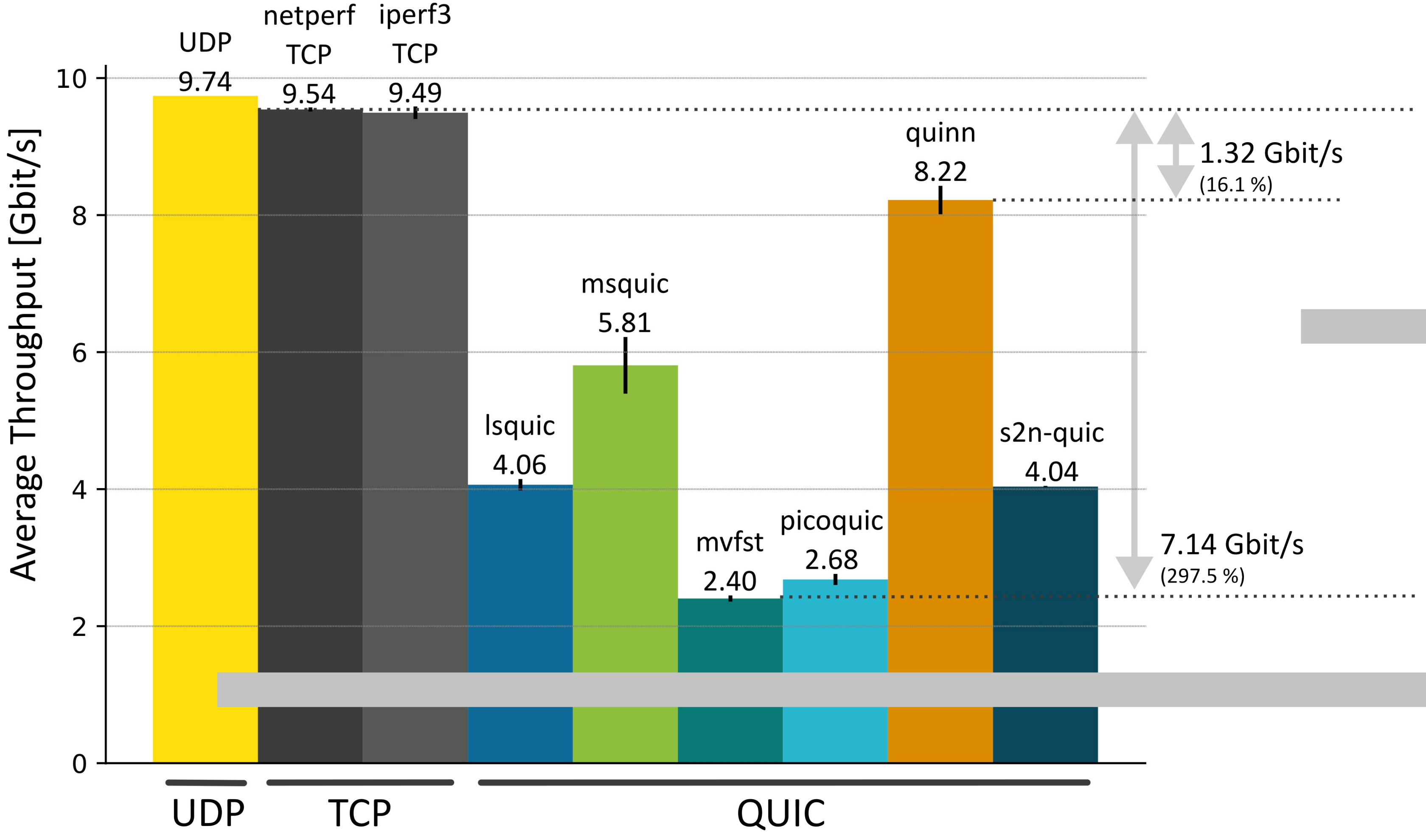


Average throughput of one single flow (10 runs, each 30s)

UDP data path through the Linux Kernel is no bottleneck for QUIC

Results: Sustained Throughput

Karlsruhe Institute of Technology



Average throughput of one single flow (10 runs, each 30s)

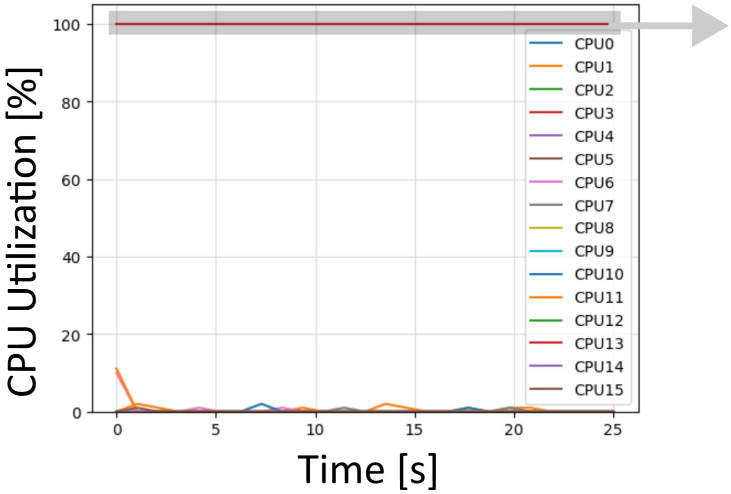
TCP* significantly outperforms QUIC implementations
(from 16.1 % up to 297.5 %)

*TCP limited by testbed –
Single TCP flow can achieve even 40+ Gbit/s [2]

UDP data path through the Linux Kernel is no bottleneck for QUIC

Potential Reasons for Limitations

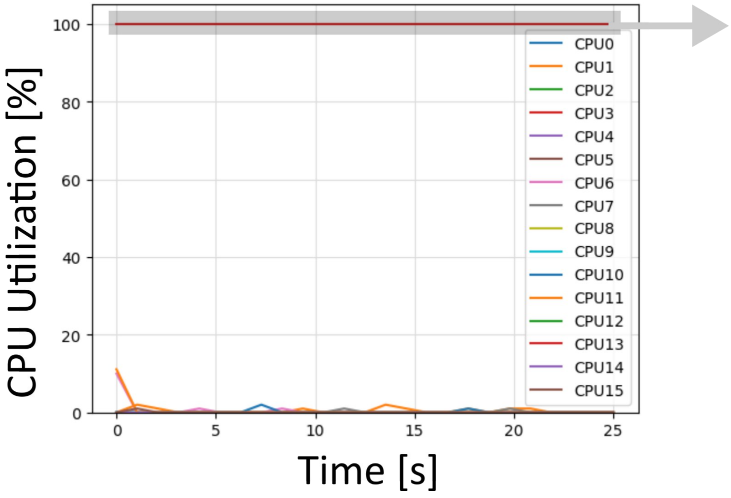
msquic



Limited by single core performance (no multi-threading)

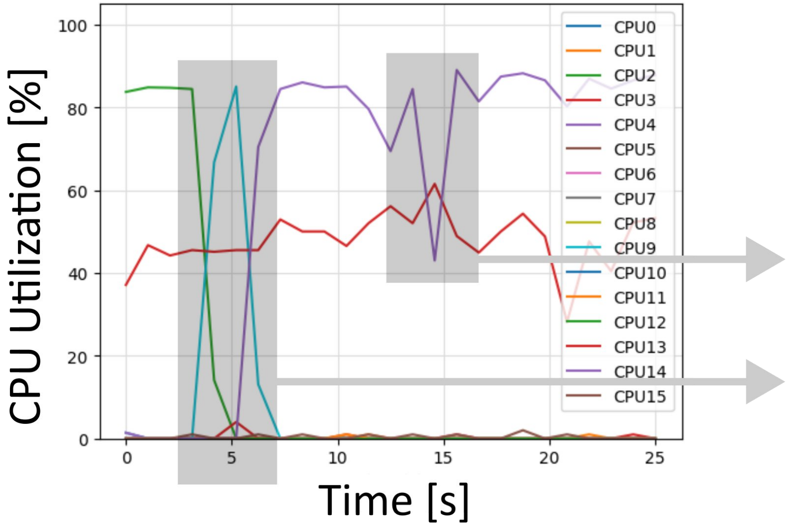
Potential Reasons for Limitations

msquic



Limited by single core performance (no multi-threading)

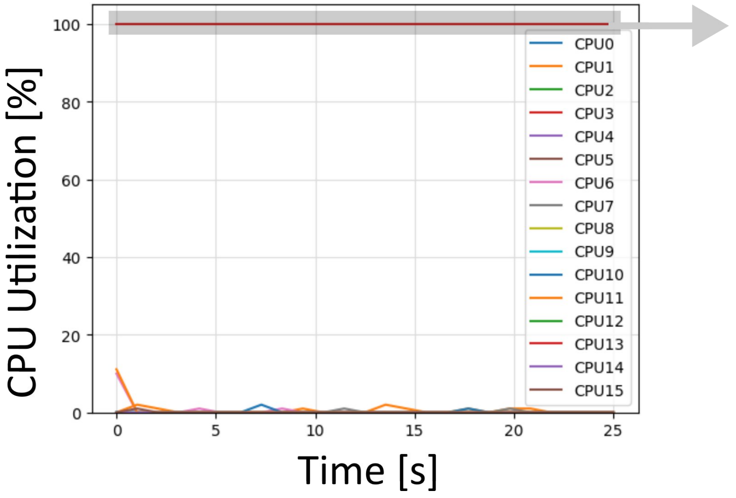
lsquic



Scheduling between CPU cores degrades throughput

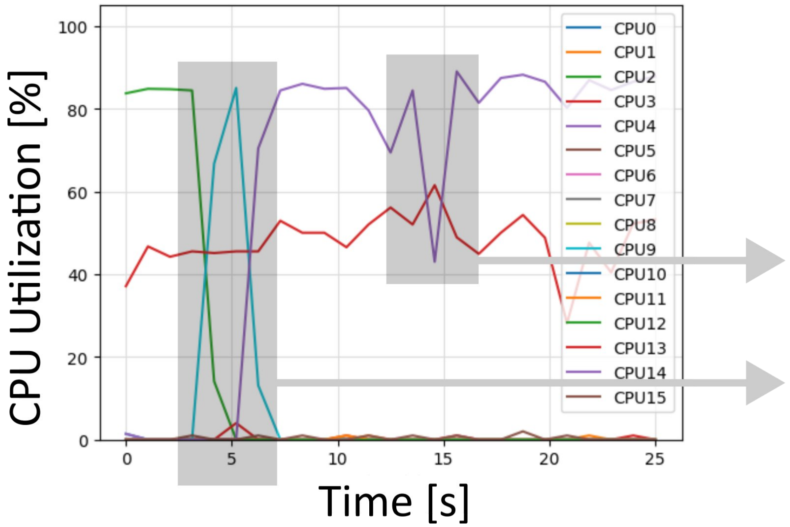
Potential Reasons for Limitations

msquic



Limited by single core performance (no multi-threading)

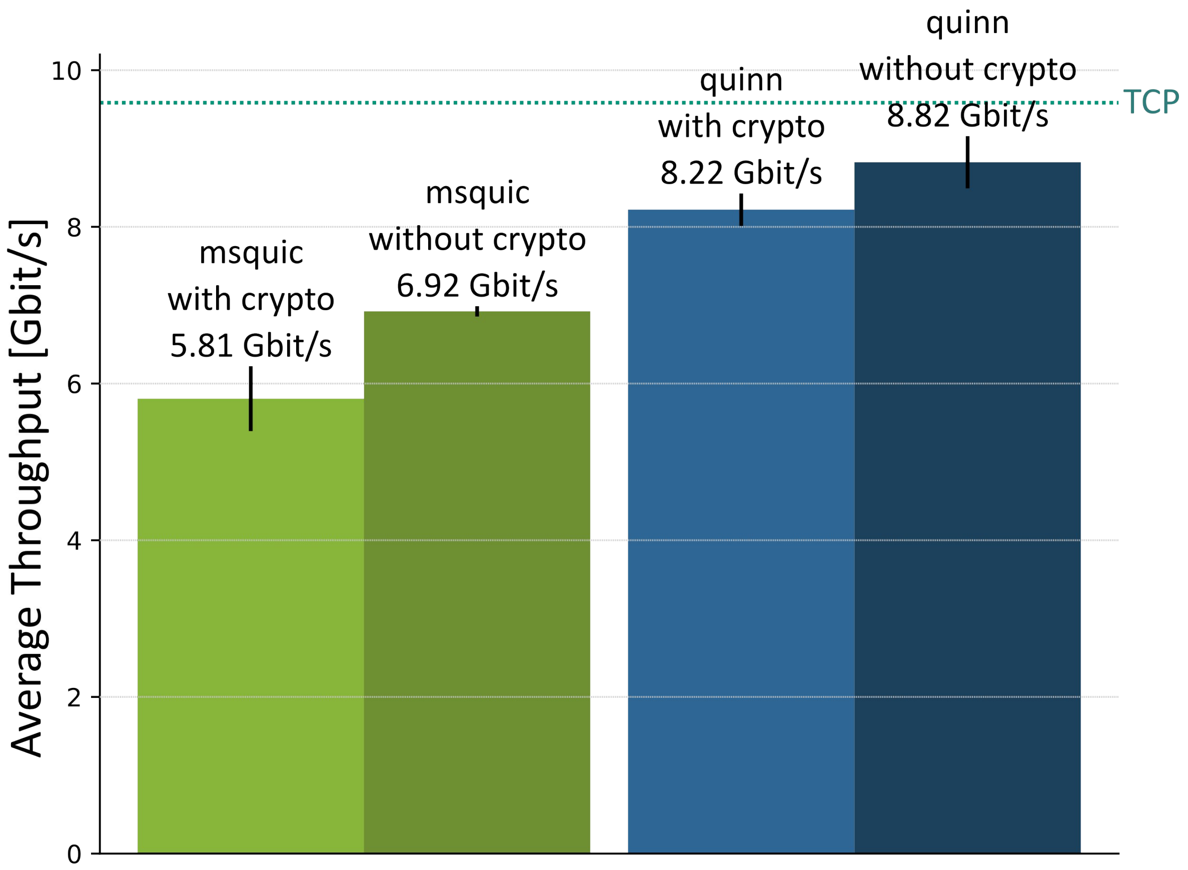
lsquic



Scheduling between CPU cores degrades throughput

➔ Inefficient Usage of CPU Resources

Impact of Cryptography



➔ QUIC's performance gap: More than overhead by cryptography

Evolution of QUIC Throughput Performance

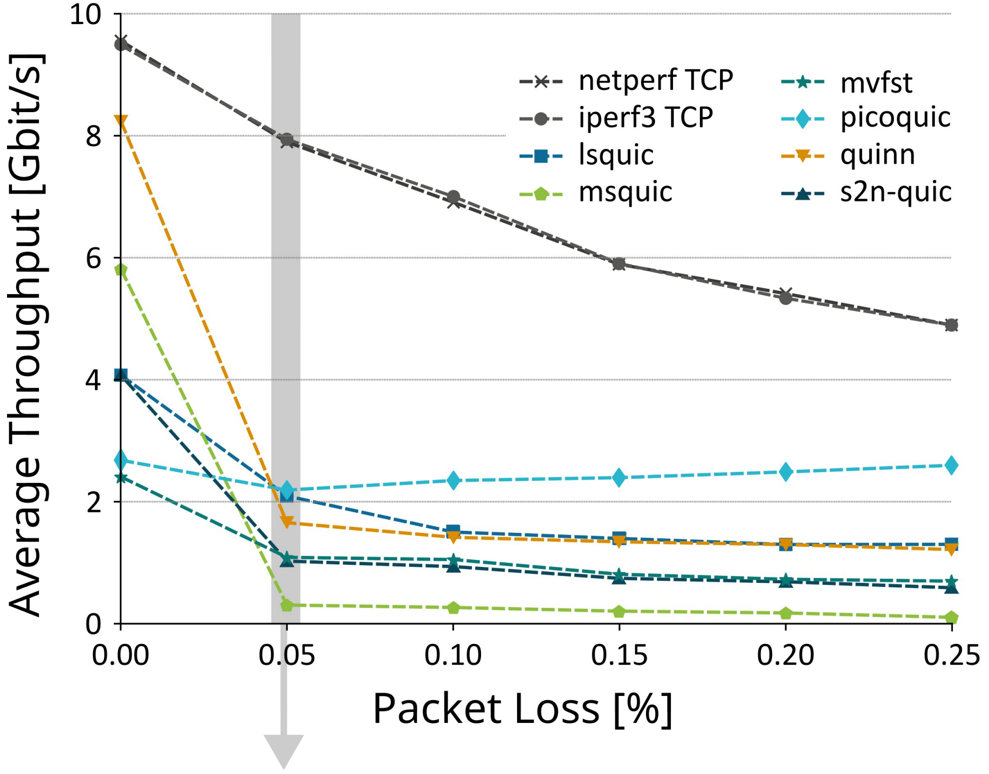
- QUIC Implementations already getting quicker

Implementation	Throughput in 2020 [3]	Throughput in 2023 [1]	Performance Increase
Picoquic	489 Mbit/s	2.68 Gbit/s	5.48x
Mvfst	325 Mbit/s	2.40 Gbit/s	7.38x

Throughput Comparison with [3] from 2020

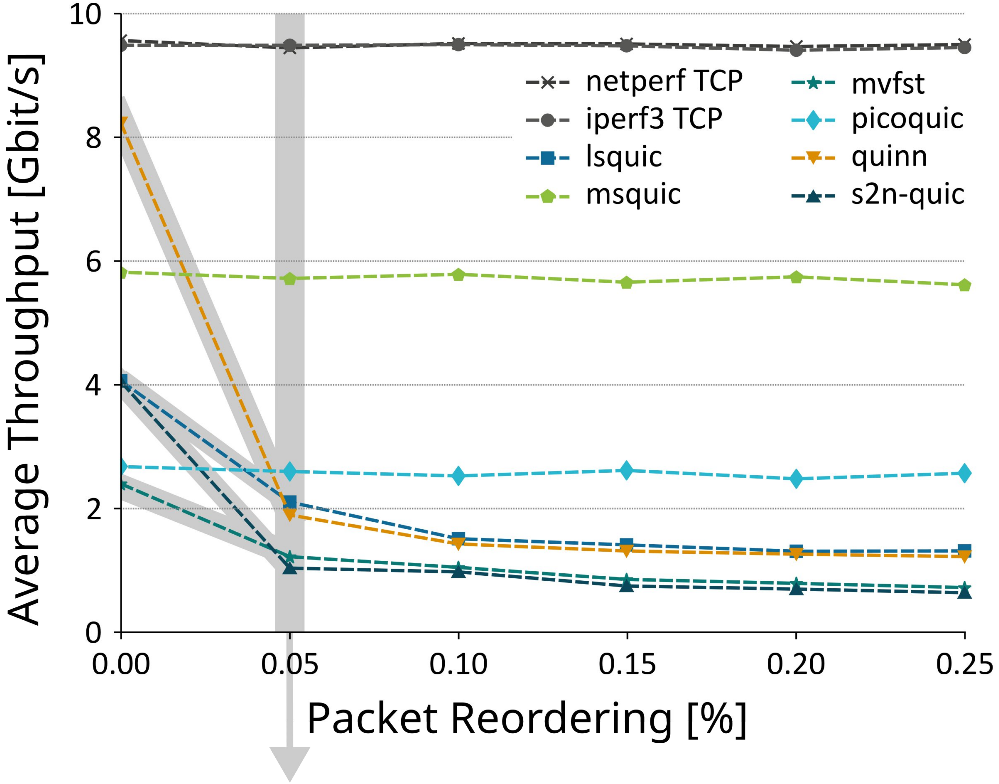
Further Issues

Packet Loss



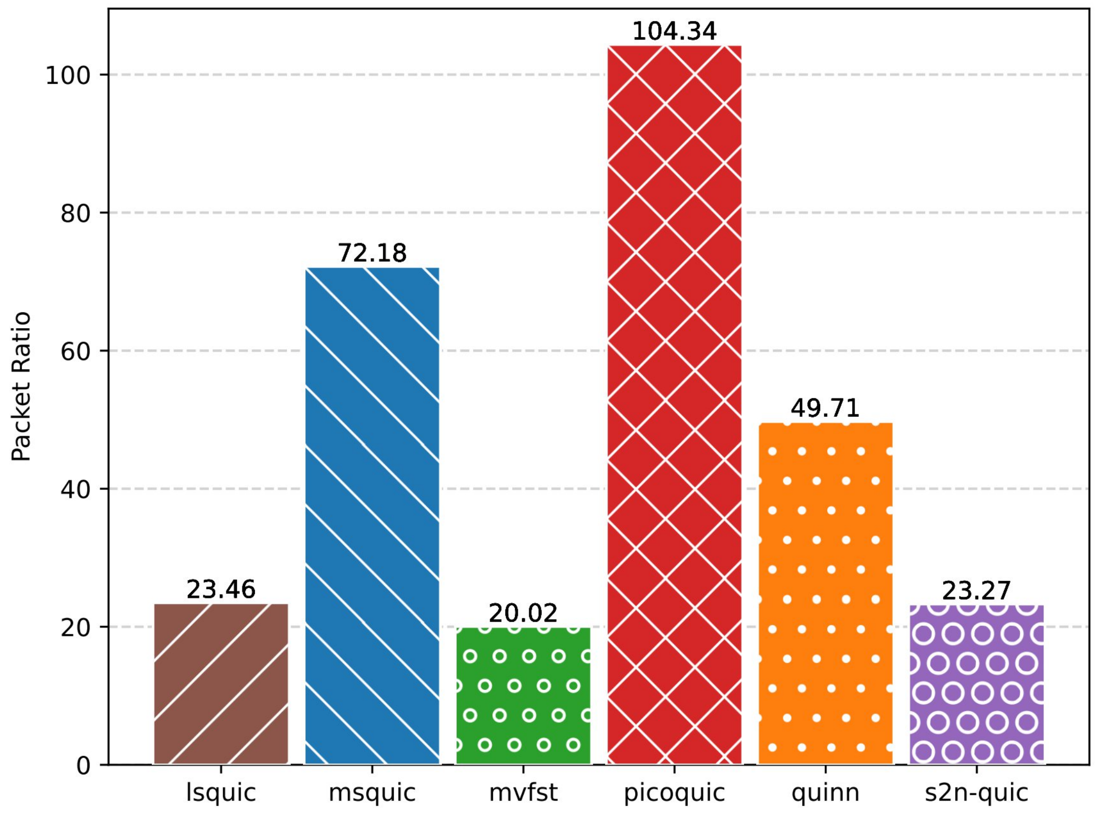
→ **QUIC implementations stronger affected by packet losses than TCP**

Packet Reordering



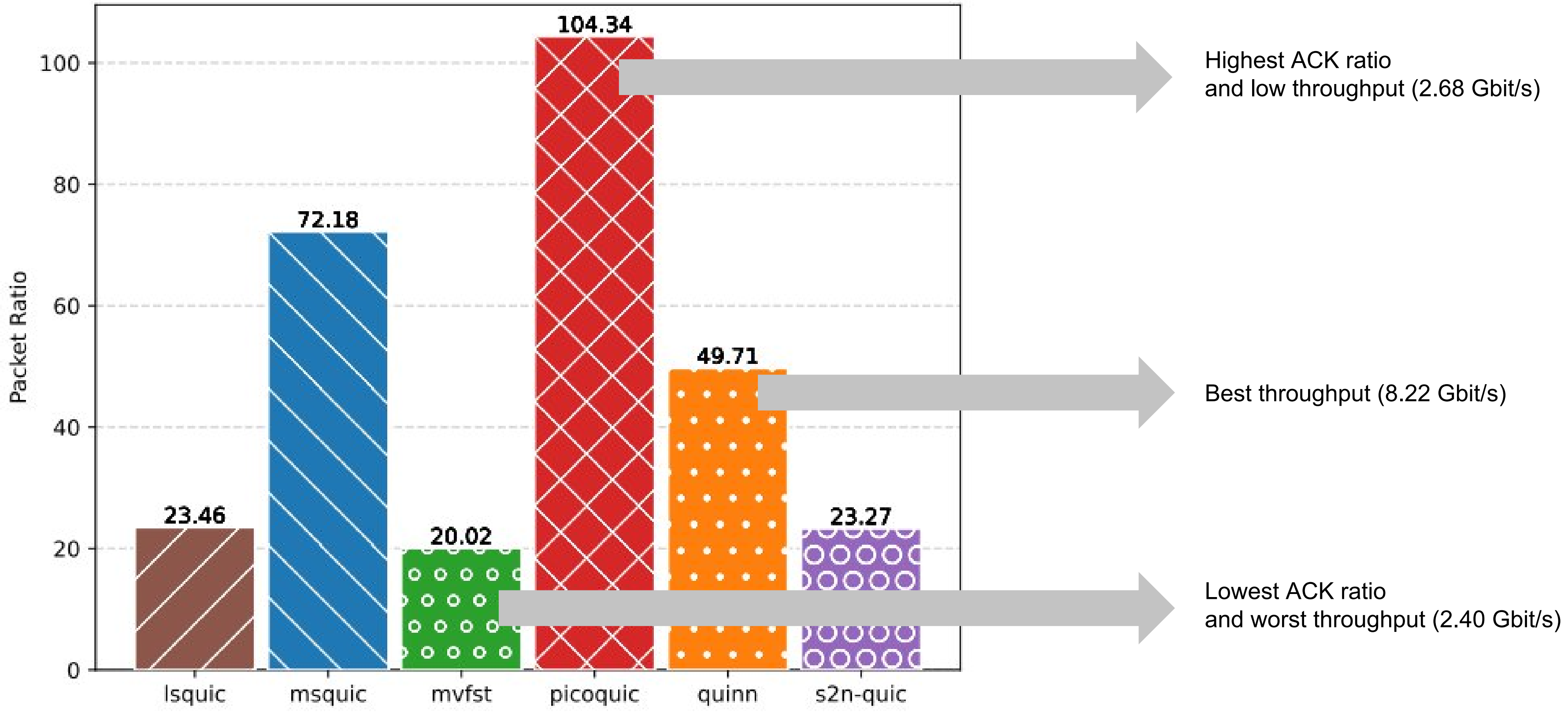
→ **mvfst, quinn, lsquic, and s2n-quic misinterpret reordered packets as losses**

ACK Ratios



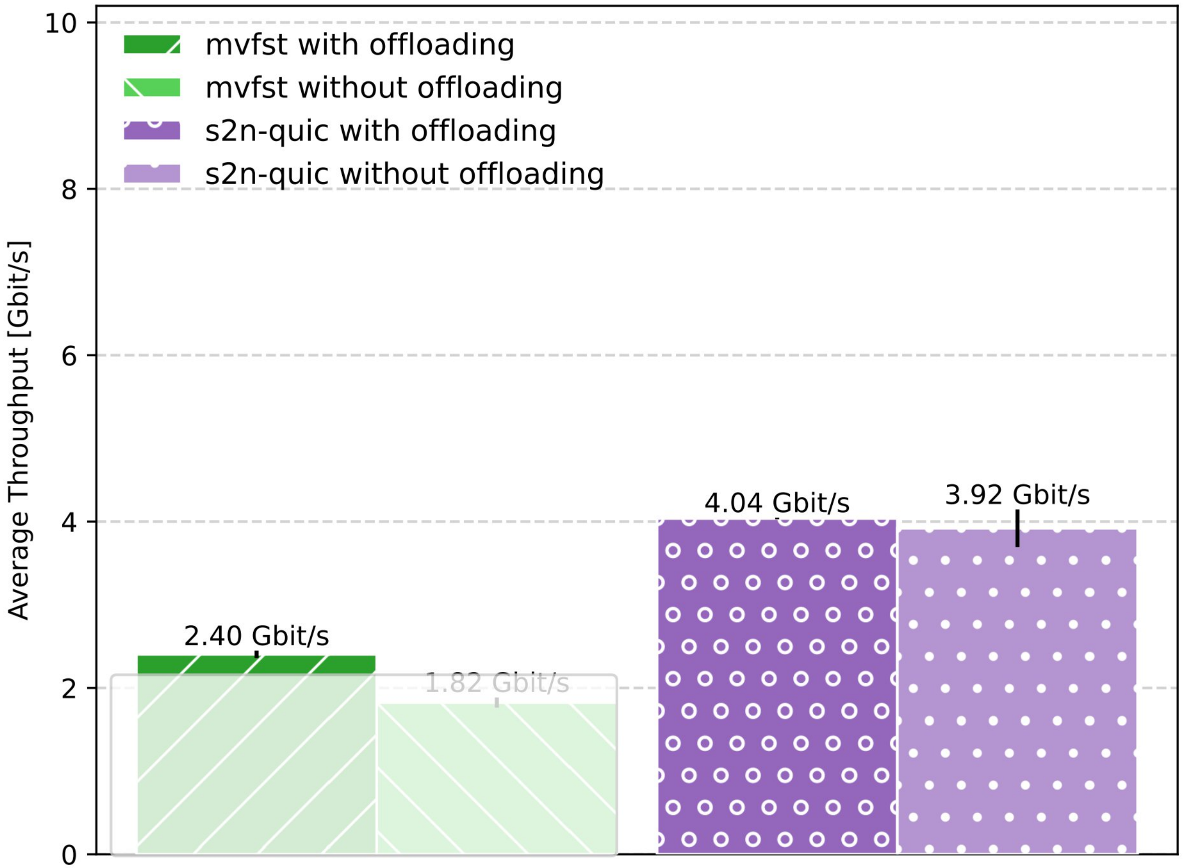
Packet Ratio
0
20
40
60
80
100
23.46
72.18
20.02
104.34
49.71
23.27
lsquic
msquic
mvfst
picoquic
quinn
s2n-quic

ACK Ratios



➔ ACK Ratio seemingly not correlated with throughput performance

Impact of Offloading



➔ Offloading can improve performance

Conclusion

- Current QUIC implementations: Not a up to par with TCP regarding sustained throughput rates
  - QUIC's performance gap: More than overhead by cryptography
  - Inefficient usage of CPU resources

- Possible solutions
  - Better usage of multiple CPU cores
  - Avoid scheduling between CPU cores
  - Offloading to (optimized) Kernel functions

References

- [1] M. König, O. P. Waldhorst and M. Zitterbart, "QUIC(k) Enough in the Long Run? Sustained Throughput Performance of QUIC Implementations," 2023 IEEE 48th Conference on Local Computer Networks (LCN), Daytona Beach, FL, USA, 2023, pp. 1-4, doi: 10.1109/LCN58197.2023.10223395.
- [2] M. Hock, M. Veit, F. Neumeister, R. Bless and M. Zitterbart, "TCP at 100 Gbit/s – Tuning, Limitations, Congestion Control," 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 2019, pp. 1-9, doi: 10.1109/LCN44214.2019.8990842.
- [3] Yang, Xiangrui, et al. "Making quic quicker with nic offload." Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC. 2020.