Dissecting QUIC Implementation Performance

Xiangrui Yang², Lars Eggert³, Jörg Ott³, Steve Uhlig³, Zhigang Sun¹, Gianni Antichi⁴

National University of Defense Technology¹, NetApp³, Technical University of Munich³, Queen Mary University of London⁴

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

Experimental settings:

Packet Reordering
Packet Loss
Packet Delay

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

a. CPU breakdown (client)

<table>
<thead>
<tr>
<th>QUANT</th>
<th>QUICKLY</th>
<th>PICOQUIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>3696Mbps</td>
<td>463Mbps</td>
</tr>
<tr>
<td>CPU usage</td>
<td>58%</td>
<td>54.8%</td>
</tr>
</tbody>
</table>

With kernel-bypass, quant reaches 7x throughput than quicky & 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 (% vs. 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.

Ongoing work

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

[Diagram showing various settings and results related to QUIC implementation performance]