STORAGE DEVELOPER CONFERENCE



BY Developers FOR Developers

# How Bad is TCP?

(And What Are the Alternatives?)

Endric Schubert, Ph.D. endric@mle.biz

# MLE Mission: "From Software to Silicon!"

High-Performance (Embedded) Compute & Connected Systems-of-Systems need "Offload Engines" for better performance, lower and deterministic latency and improved energy efficiency.

Focus on standards such as PCIe, NVMe, Ethernet, TCP/UDP/IP, TSN.



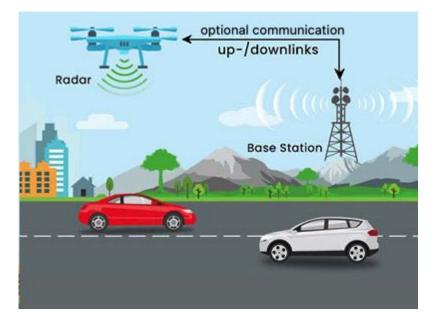


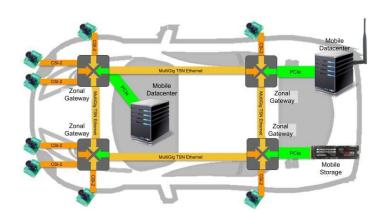


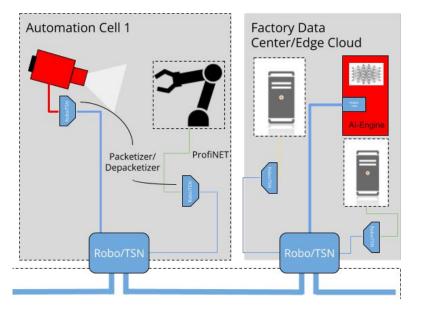
# Multi-Gigabit Real-Time Networking Market & Technology Forces

6G Radio Integrated Communication and Sensing (ICAS) Zone-Based In-Vehicle Networking (Auto/TSN)

### 100G Real-time Backbone for Virtualized PLC (Robo/TSN)









# Work Motivation

Systems-of-systems

- Tightly-coupled: i.e. distributed processing with microservices
- Loosely-connected via networks (which continuously are the bottleneck)

Need to optimize

- for power / energy efficiency
- for throughput
- for (deterministic) latency and real-time delivery

Domain-Specific Architectures:

- "Offload" (protocol) processing software onto silicon but adhere to (defacto) standards and APIs
- Make networks more deterministic and Time-Sensitive

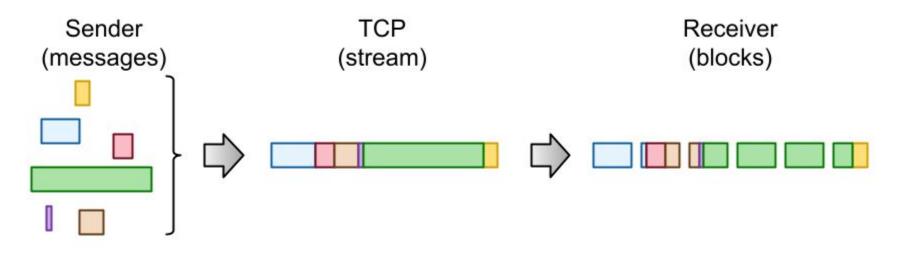


# **Presentation Outline**

- How Bad is TCP?
  - Computational Burden
  - Tail-end Latencies
- What Are the Alternatives?
  - Homa from John Ousterhout's team at Stanford University
  - QRP (Quad R P) a Hardware Accelerated Version of Homa



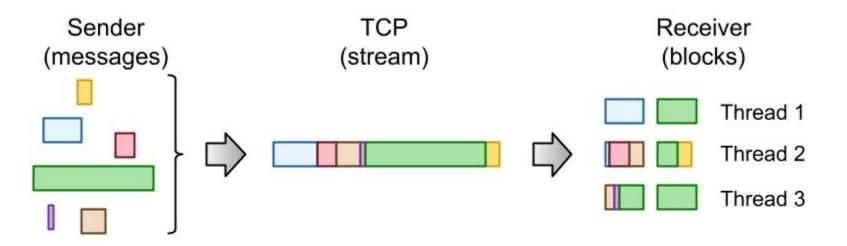
## **1. TCP Data Model: Byte Stream**



- Applications care about messages, but TCP drops boundary info
- Extra complexity/overhead for message reassembly



## 1. TCP Byte Streams, cont'd



#### Disastrous for load balancing

- Can't share one stream among multiple threads
- Can't offload dispatching to NIC

October 26, 2022

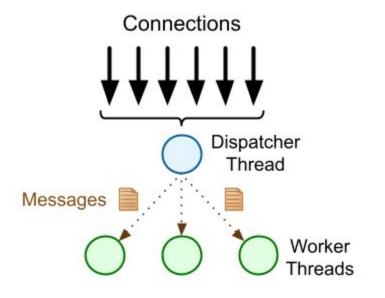




# Courtesy of John Ousterhout, Stanford University

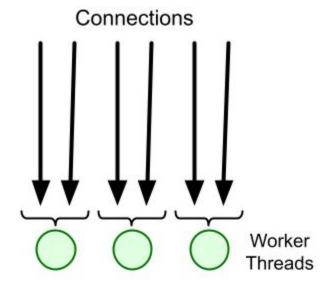
## **Load Balancing Choices**

**Choice #1: dispatcher thread** 



- Extra latency for worker handoff
- Dispatcher is throughput bottleneck (~1M msgs/sec)

**Choice #2: partition connections** 



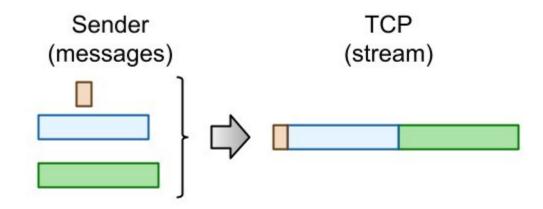
 Static load balancing: prone to hot spots

Slide 8



October 26, 2022

## 1. TCP Byte Streams, cont'd



#### Head-of-line blocking:

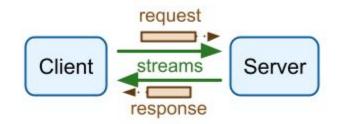
- Short messages can get stuck behind long ones
- High tail latency



## **Stream-Level Reliability Inadequate**

#### • Clients want round-trip guarantees:

- Deliver request
- Ensure it is processed
- Deliver response
- Or, notify of error
- Stream guarantees are weaker:
  - Best-effort delivery of request or response
  - No notification if server machine crashes
- Clients must implement additional timeout mechanisms
  - Even though TCP already implements timers



October 26, 2022





## **2. TCP is Connection-Oriented**

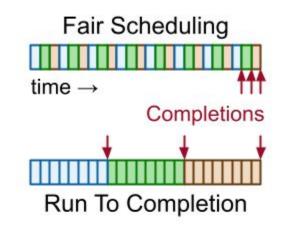
#### Requires long-lived state for each stream

- ~2000 bytes per connection in Linux, not including packet buffers
- Individual datacenter apps can have thousands of connections
- Mitigate with connection pooling/proxies (e.g. Facebook)? Adds overhead
- Challenging for NIC offloading (e.g. Infiniband): thrashing in connection caches
- Before sending any data, must pay round-trip for connection setup
  - Problematic in serverless environments: can't amortize setup cost
- Motivation for connections:
  - Enable reliable delivery, flow control, congestion control
  - But, all these can be achieved without connections



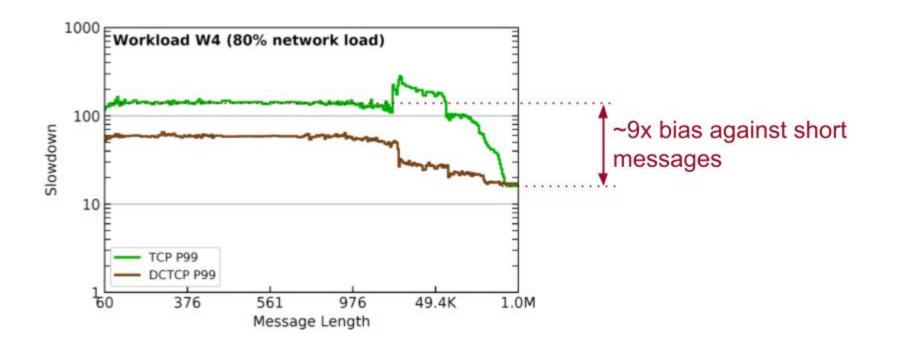
## **3. TCP Uses Fair Scheduling**

- When loaded, share bandwidth equally among active connections
- Well-known to perform poorly: everyone finishes slowly
- Run-to-completion approaches (e.g. SRPT) are better
  - But requires message sizes





## **TCP Isn't Actually Fair!**





### 4. TCP: Sender-Driven Congestion Control

#### Senders responsible for scaling back transmission rates when needed

But, they have no direct knowledge of congestion

#### Congestion signals based on buffer occupancy:

- Packets dropped if queues overflow
- Congestion notifications based on queue length

#### • Problems:

- Significant buffer occupancy when system is loaded
- Queuing causes delays, especially for short messages



# The Computational Burden of TCP

Benchmark TCP Point-to-Point with Netperf 2.6

- TCP\_STREAM (and TCP\_MAERTS) for throughput in Gbps
- CPU Load on Tx and Rx side

- TCP\_RTT for Round-Trip-Time
- Efficiency = Throughput / CPU Load

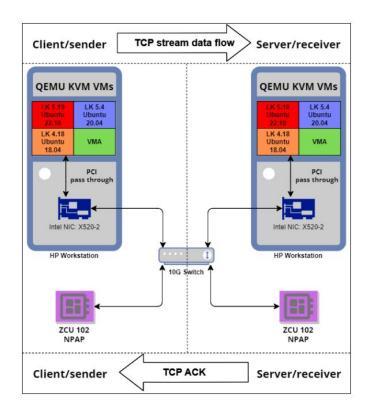
#### System Setup

- AMD FPGA w/ NPAP-25G (Fraunhofer HHI's TCP/UDP/IP Full Accelerator)
- 25 GigE NIC: Mellanox ConnectX-4 LX
- CPU: Intel(R) Xeon(R) e5-1620 v0 @ 3.60GHz
   1 socket, 4 cores, 1 thread per core
   RAM: 32G (4x8G) (Samsung, DDR3, 1600 MT/s, syn reg)

Effects of different Linux kernels

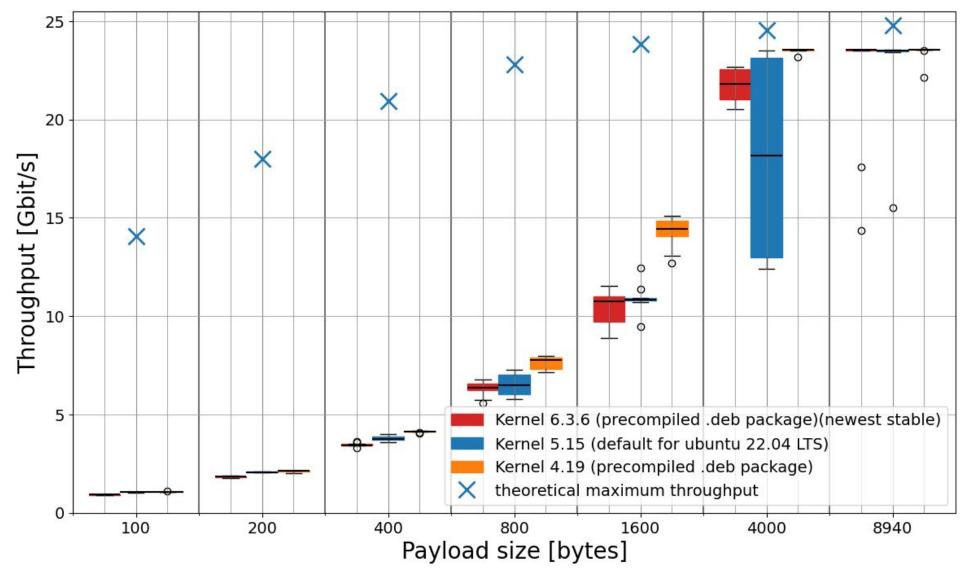
- Vanilla 4.19 vs vanilla 5.15. vs vanilla 6.3.6
- Vanilla 4.19 vs Centos 4.18

#### **Experimental Setup**



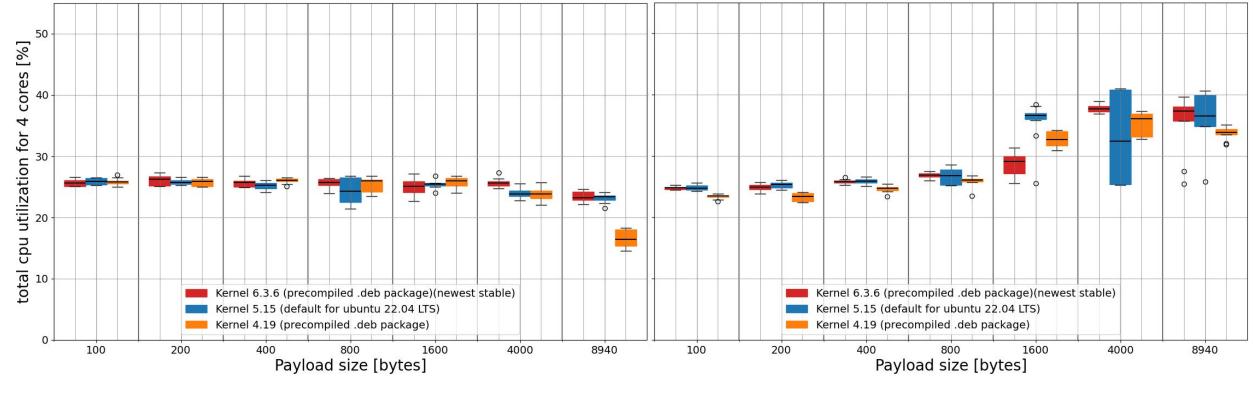


# TCP\_STREAM Results for Different Linux kernels





# TCP\_STREAM Results for Different Linux kernels



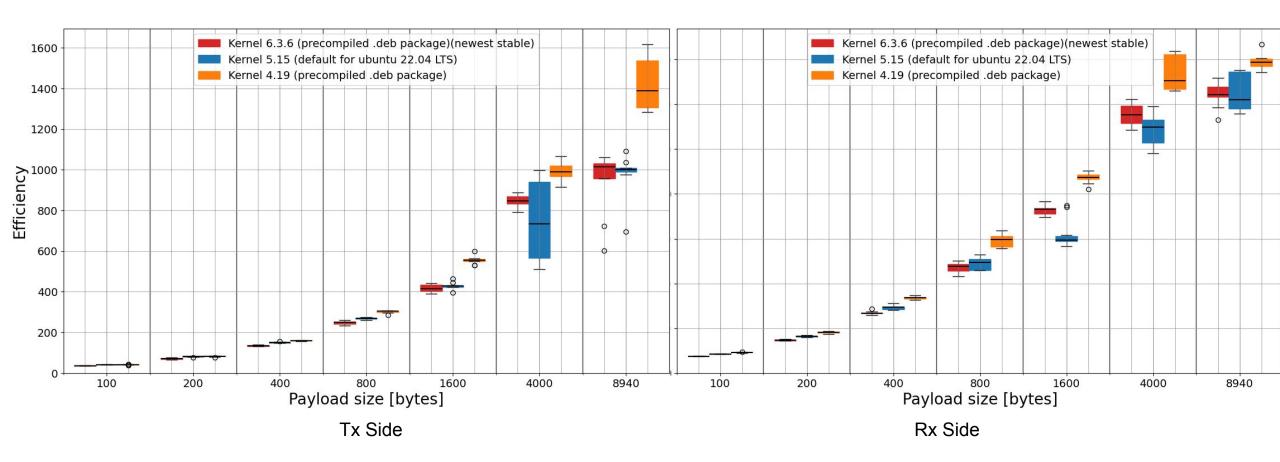
Tx Side

Rx Side



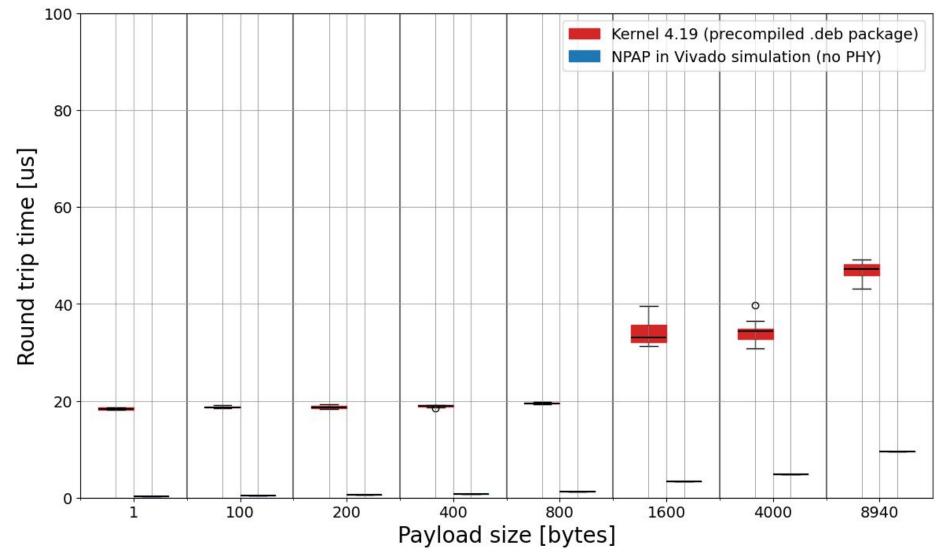
#### 

# TCP STREAM Results for Different Linux kernels





# TCP\_RR Results for Different Linux kernels



Average RTT of FPGA Full Accelerator clearly outperforms (Linux) software



## **1. Homa is Message-Based**

- Dispatchable units are explicit in the protocol
- Enables efficient load balancing
  - Multiple threads can safely read from a single socket
  - Future NICs can dispatch messages directly to threads
- Enables run-to-completion (e.g. SRPT)



# 2. Homa is Connectionless

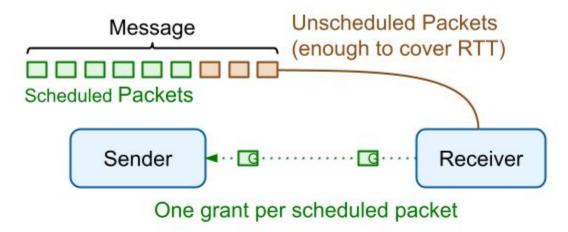
- Fundamental unit is a remote procedure call (RPC)
  - Request message
  - Response message
  - RPCs are independent

### No long-lived connection state

- (But there is long-lived per-peer state: ~200 bytes)
- No connection setup overhead
  - Use one socket to communicate with many peers
- Homa ensures end-to-end RPC reliability
  - No need for application-level timers



### 3. Homa: Receiver-Driven Congestion Control



- Receiver can delay grants to:
  - Reduce congestion in TOR
  - Prioritize shorter messages

### Message sizes allow receivers to predict the future:

Faster, more accurate response to congestion

October 26, 2022



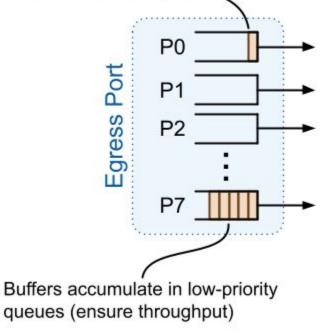


## **Homa Uses Priority Queues**

- Modern switches: 8–16 priority queues per egress port
- Homa receivers select priorities for SRPT:
  - Favor shorter messages
- Achieve both high throughput and low latency
  - Need buffering to maintain throughput (e.g. if sender doesn't respond to grant)
  - But buffers can result in delays
  - Solution: overcommitment:
    - Grant to multiple messages
    - Different priority for each message

### Overcommitment

Short messages use high priority queues (low latency)

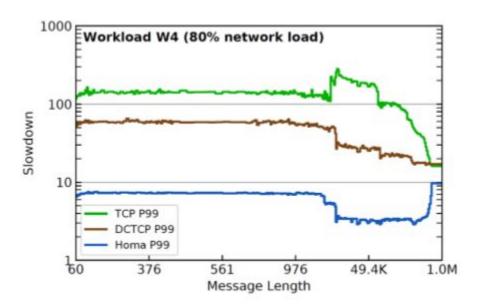


It's Time to Replace TCP in the Datacenter



## 4. Homa: SRPT

- Combination of grants, priorities
- Run-to-completion improves performance for every message length!
- Starvation risk for longest messages?
  - Use 5-10% of bandwidth for oldest message





## 5. Homa: No Order Requirement

#### Can use packet spraying in datacenter networks

- Hypothesis: will eliminate core congestion (unless core fabric systemically overloaded)
- Better load balancing across CPU cores



#### 

homa.pca

## Wireshark Dissector for Homa

Add a	basic dissector for × +				
$\rightarrow$ (	G 🏠 🔒 github.com/PlatformLab/HomaModule/pull/43			ଓ < ☆ Ł	. 🗆 🐧
<b>)</b> Pi	roduct × Solutions × Open Source × Pricing	Search or jump to		7 Sign in	Sign
📮 Pla	tformLab / HomaModule Public	다. Notifica	ations 😵 Fork	33 🟠 Star 14	45 -
<> Co	de 💿 Issues 2 🏦 Pull requests 1 💿 Actions 🖽 Projects 🔅	Security 🗠 Ins	ights		
Adc	a basic dissector for viewing Homa	•			lew issue
	onversation 6 -o- Commits 2 💽 Checks 0 🗈 Files changed	3		+506	
-	pmo73 commented last week		Reviewers		
	The dissector just shows the Homa header information and has no notion about responses. The dissector was tested with Ubuntu 18.04 and Ubuntu 22.04 with version 3.6.2	No reviews Assignees No one assigned			
	🗢 🚑 Add a basic dissector for viewing Homa packets. 🚥	fdfb2cb	Labels		
6	johnousterhout commented last week	Member ···	None yet		
	Thanks for contributing this; it looks great. My only concern is that it uses the G license, which is more restrictive than the rest of Homa. Would you be willing to restrictive license, such as BSD-1-Clause?	Projects None yet			
	-John-		Milestone No milestone		
Ţ	pmo73 commented last week • edited 🗸	Author ····	Development Successfully merging this pull request may close		
	Good day <b>@johnousterhout</b> , thank you for your feedback. We decided to use the because it is used by Wireshark and therefore the dissector can be added to the project later Would it be a colution for you to use a dual licence of GRI 2 + BCD.	these issues. None yet			
	project later. Would it be a solution for you to use a dual license, so GPL2 + BSD	tor example?	2 participants		

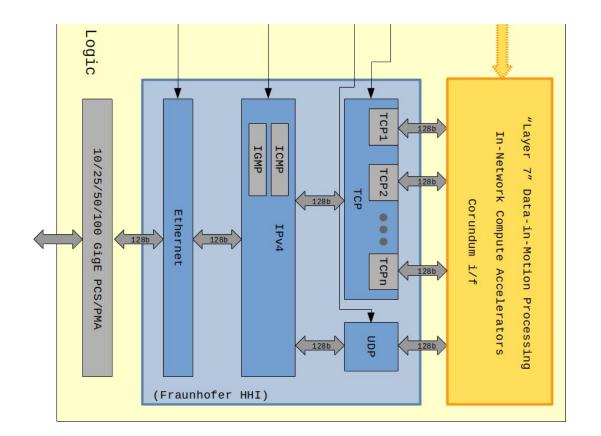
No.	zeigefilter anwenden	10.00			
	Time	Source	Destination		Length Info
	15 40.580767	192.168.123.11	192.168.123.10	HOMA HOMA	1514 Data Packet 67 Grant Packet
	16 40.580790	192.168.123.10	192.168.123.11		
	17 40.580800	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	18 40.580811	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	19 40.580819	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	20 40.580831	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	21 40.580837	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	22 40.580852	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet 67 Grant Packet
	23 40.580858	192.168.123.10	192.168.123.11	HOMA	
	24 40.580870	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	25 40.580889	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	26 40.580894	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	27 40.580910	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	28 40.580919	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	29 40.580932	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	30 40.580938	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	31 40.580951	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	32 40.580957	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	33 40.580968	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	34 40.580973	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	35 40.580994	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	36 40.581000	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
	37 40.581016	192.168.123.11	192.168.123.10	HOMA	1514 Data Packet
	38 40.581045	192.168.123.10	192.168.123.11	HOMA	67 Grant Packet
Eth	nernet II, Src: I	IntelCor_4a:d9:ac (9		t: IntelCor	bits) _3a:b2:38 (90:e2:ba:3a:b2:38)
<pre>Eth Int Hon ()</pre>	hernet II, Src: I cernet Protocol V na Protocol Common Header HOMA Source Po Homa Dest Port 1010 = Hon Homa Protocol Homa sender ID	IntelCor_4a:d9:ac (90 /ersion 4, Src: 192.: rt: 32794 : 4000 ma data offset: 10 Type: 0x10		t: IntelCor	
<pre>Eth Int Hon ()</pre>	hernet II, Src: I cernet Protocol V na Protocol Common Header HOMA Source Por Homa Dest Port 1010 = Hon Homa Protocol	IntelCor_4a:d9:ac (90 /ersion 4, Src: 192.: rt: 32794 : 4000 ma data offset: 10 Type: 0x10 : 5122874	0:e2:ba:4a:d9:ac), Dst	t: IntelCor	

Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telephonie Wireless Tools Hilfe



# NPAP - Network Protocol Accelerator Platform

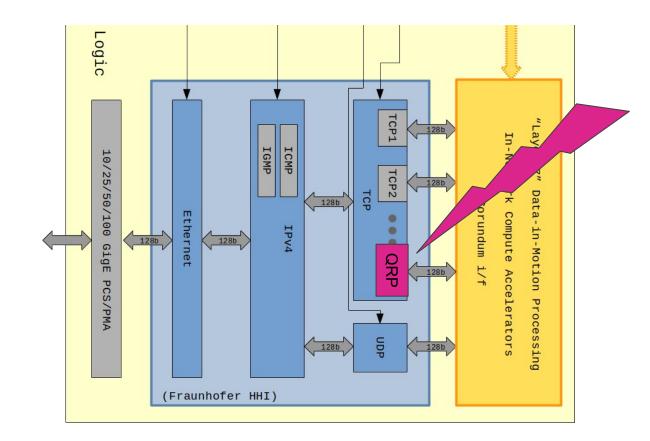
- Full Accelerator = no CPUs, no software
- Standard IEEE Ethernet PHYs with RMII, GMII, XGMII, etc via PCS/PMA via ASIC/FPGA Ethernet Subsystem
- Ethernet, ARP, IPv4, ICMPv4, IGMPv4, UDP & TCP, DHCP
- Optional TSN, optional TLS
- Datapath via AXI4-Stream 128-bit
- Complete stack uses generic VHDL code
- In production use for automotive, aero & defense, industrial test & measurement, telco applications





# QuadRP - Reliable, Rapid Request-Response Protocol

- Based on Homa
- Implemented within NPAP Tested and proven Ethernet and IPv4
- Complements TCP/IP and UDP/IP
- $\Rightarrow$  Best of both worlds:
- No CPU load
- Very low, deterministic latency
- Option for handling messages in
  - Programmable Logic, or
  - in Linux software





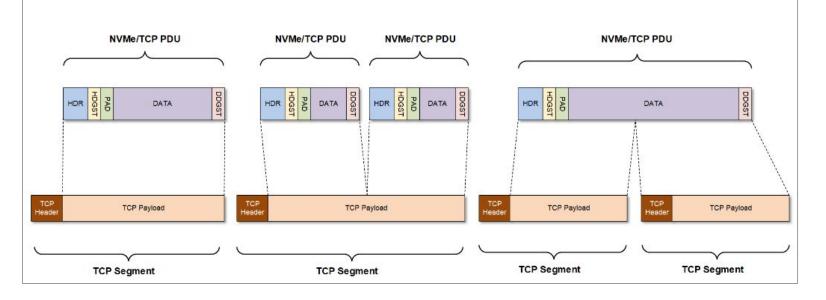
## Homa's Benefits for NVMe-over-Fabric

LAN is the bottleneck already, with now additional burden from heavy SAN traffic. Homa latencies can be 100x faster that TCP and promises to put less load on the network. NVMe eliminated the legacy software overhead and uses fast PCIe Posted Writes for better response times and IOPS.

So, is TCP then the proper foundation for NVMe-over-Fabric?

NVMe-over-Homa can be a drop-in replacement or an add-on, achieving storage latencies close to DAS performance, with less overhead on network and servers.

### Mapping of NVMe<sup>™</sup>/TCP PDUs to TCP





# HOMA References

[1] John Ousterhout, Stanford University: <u>https://web.stanford.edu/~ouster/cgi-bin/papers/replaceTcp.pdf</u>

[2] John Ousterhout's presentation at USENIX ATC'21 (15 minutes) https://www.usenix.org/conference/atc21/presentation/ousterhout

[3] Montazeri's presentation at SIGCOMM18 (starts at 1:22) <u>https://www.youtube.com/watch?v=o\_sg1nnN2bQ&t=4927s</u> <u>https://conferences.sigcomm.org/sigcomm/2018/files/slides/paper\_4.4.pptx</u>

[4] Homa Linux kernel module implementation <u>https://github.com/PlatformLab/HomaModule</u>

[5] Montazeri's PhD dissertation http://purl.stanford.edu/sp122ms2496





# Please take a moment to rate this session.

Your feedback is important to us.

