Can Homa and TCP Get Along?

John Ousterhout
Stanford University

Overview

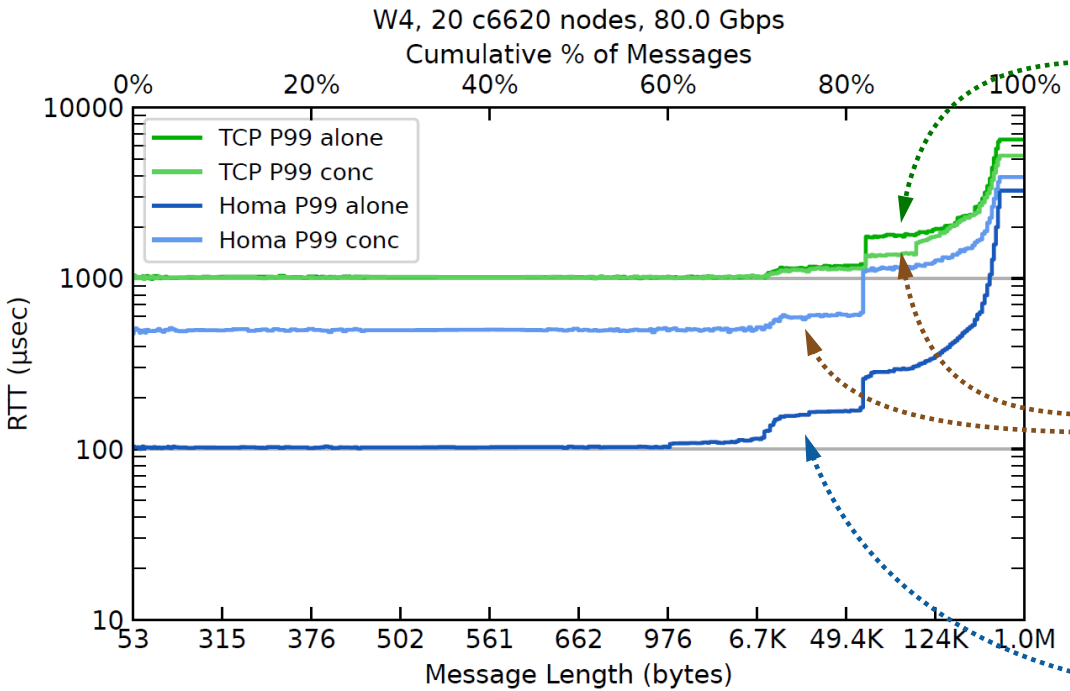
- **Homa transport protocol is quite different from TCP**
- **What happens when TCP and Homa are used simultaneously?**
  - Will they interfere with each other?
- **Originally:**
  - TCP's performance is a bit better with Homa than without
  - But TCP degrades Homa performance significantly
- **Developed new homa_qdisc to handle interactions between Homa and TCP**
- **Now:**
  - Homa performance only degrades slightly with TCP
  - TCP's performance is still better with Homa than without
  - homa_qdisc improves TCP latency even when there is no Homa traffic!

Homa Review

- **Clean-slate redesign of network transport for datacenters**
- **Reduces tail latency for short messages by 10x compared to TCP**
- **Homa differs from TCP in every major design decision:**

Homa	TCP
Message-based	Streams
Connectionless	Connection-oriented
Receiver-driven congestion control	Sender driven
SRPT priorities	Fair sharing
Uses switch priority queues	Doesn't know about queues

Initial Experiment



2. Run TCP stand-alone (80% load)

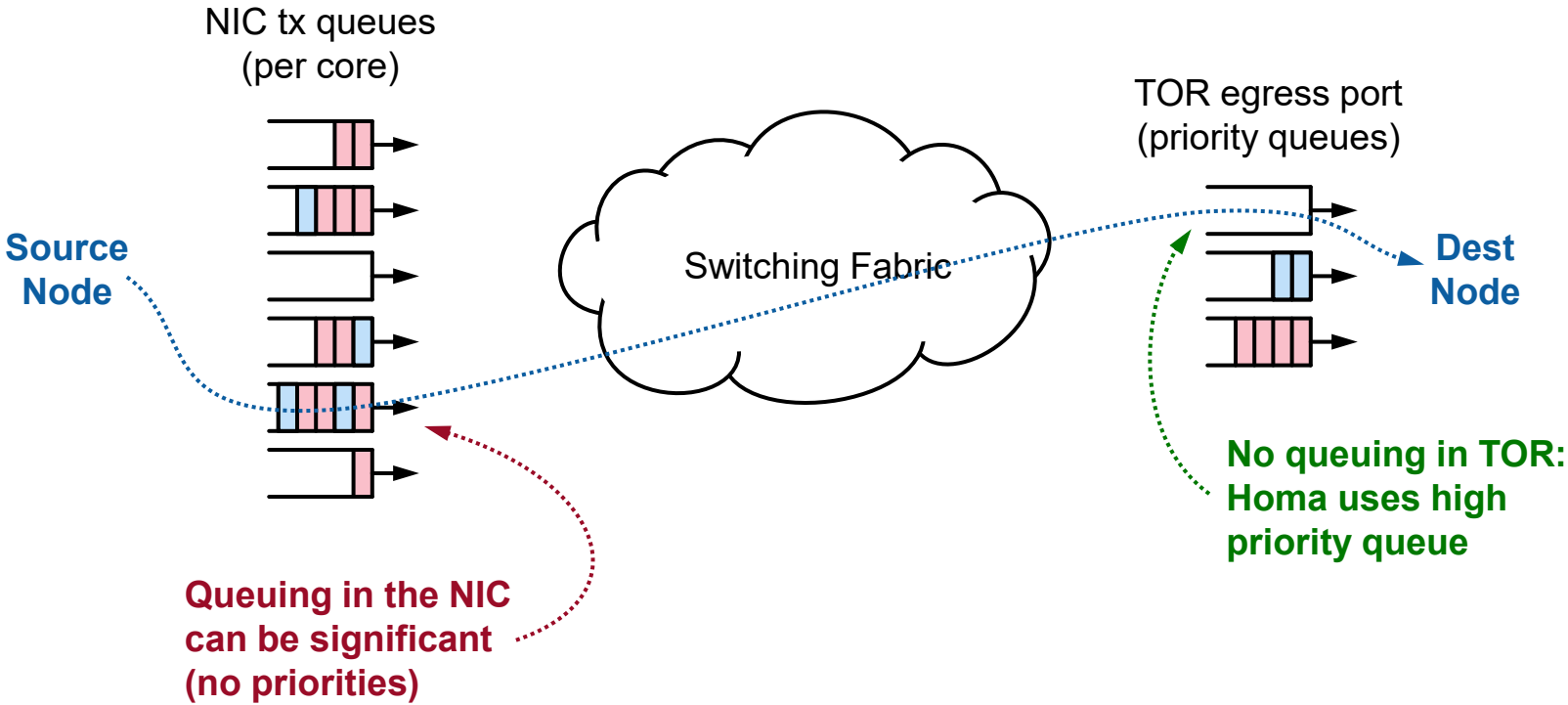
3. Run Homa and TCP together (40% load each):
   - TCP slightly faster
   - Homa 5x worse

1. Run Homa stand-alone (80% load)

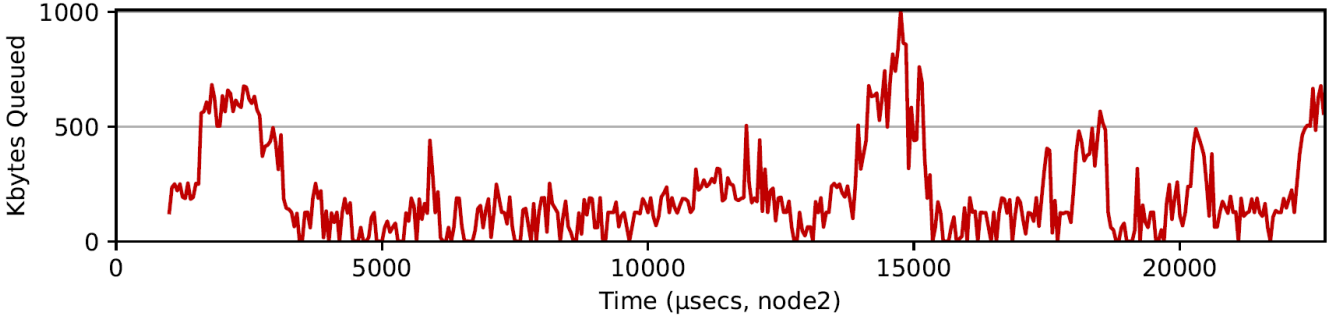
Analysis

- **Why did TCP get faster?**
  - Homa is a good citizen: doesn't generate much queuing
- **Why doesn't Homa's use of priorities hurt TCP?**
  - Homa uses priorities carefully
  - Only a small amount of data in top priority level
  - A lot of data is sent at P0 (same as TCP)
- **Why did Homa get slower?**
  - TCP is not a good citizen: generates long queues

Queuing for Small Homa Messages

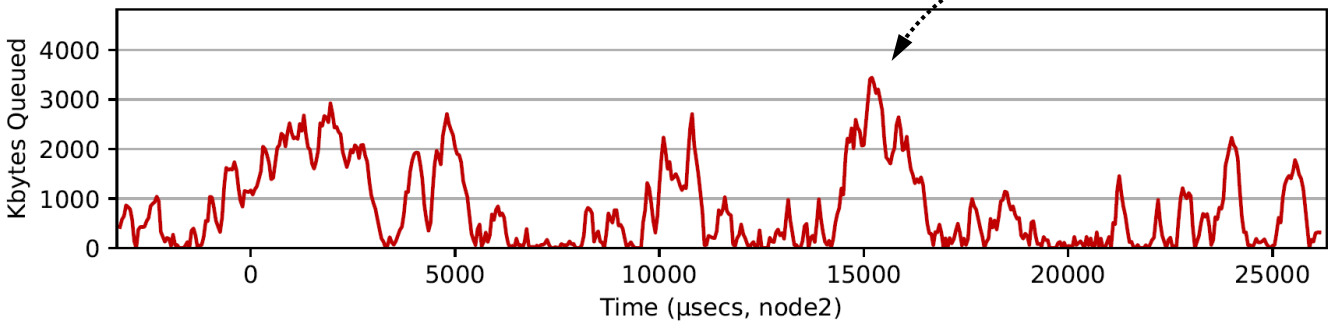


NIC Queues



Homa standalone: pacer limits queuing

3 Mbytes → 240 µsecs delay



Homa and TCP together: TCP loads queues

homa_qdisc

New queuing discipline that is part of Homa:

- **Replaces existing Homa pacer**
- **Limit growth of NIC queues**
  - Defer both Homa and TCP packets if queues get too long
- **Implement SRPT policy for Homa output**
- **When NIC congested, balance traffic fairly between Homa and TCP**

Estimating NIC Queue Length

- **Approach #1: real-time NIC simulation**
  - Assume NIC transmits at line rate
  - Maintain variable `link_idle_time`:
    - Estimate of when the NIC queue will become empty.
    - Update on each packet handoff
- **If `link_idle_time` too far in the future (typically ~5 µsecs):**
  - Defer outbound packets, wake up pacer thread
  - Pacer spins until `link_idle_time` within limit, then transmit (SRPT)
- **Disadvantage:**
  - NICs can't always transmit at line rate (especially with fast
  - NIC queues occasionally become 10x too long

Too large?

Hurts SRPT: small messages delayed

Too small?

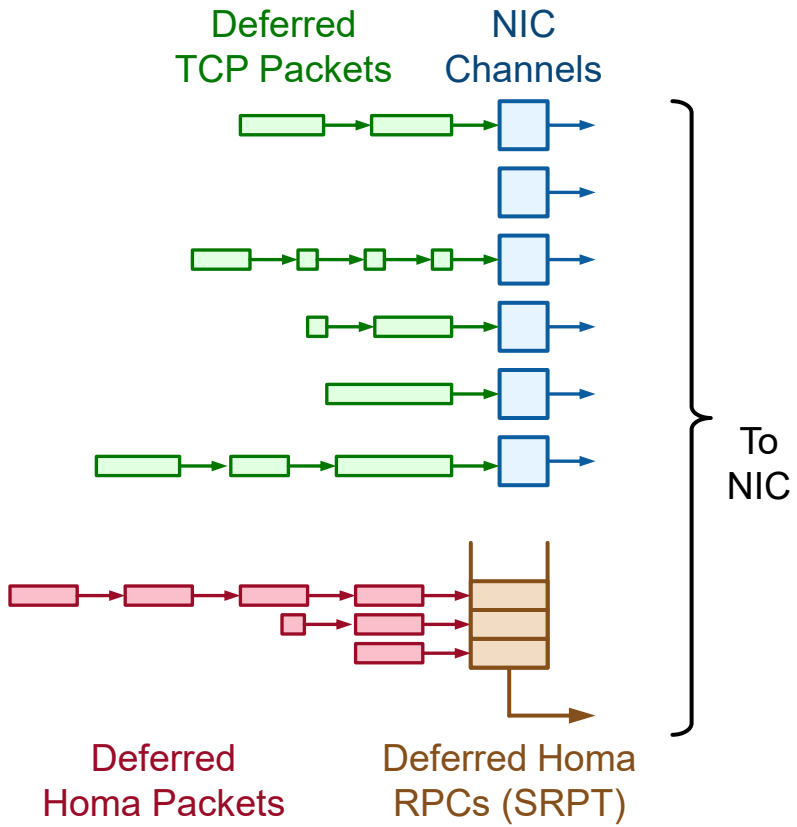
Pacer interruptions result in queue underflow, lost bandwidth

Estimating NIC Queue Length, cont'd

- **Approach #2: use DQL statistics**
  - DQL keeps statistics for each NIC channel:
    - Cumulative bytes in sk_buffs passed to NIC
    - Cumulative bytes in sk_buffs returned after transmission
  - Pacer doesn't transmit if bytes owned by NIC exceed limit
  - Limit specified in units of time (typically ~40 µsec)
- **With DQL, why is simulator also needed?**
  - Delays in returning packets after transmission: need large time constant (40 µsec vs. 5 µsec)
- **Doesn't DQL already do what Homa's pacer does?**
  - Allows much longer queues
  - Doesn't support SRPT priorities

Queuing in homa_qdisc

- **Per-NIC-channel queues of deferred TCP packets**
  - Maintain packet order within flows
  - Some SRPT for shorter packets
- **Global queue of Homa RPCs with deferred packets**
  - SRPT order
  - Implemented with red-black tree
- **Both Homa and TCP packets deferred?**
  - Split bandwidth between TCP and Homa

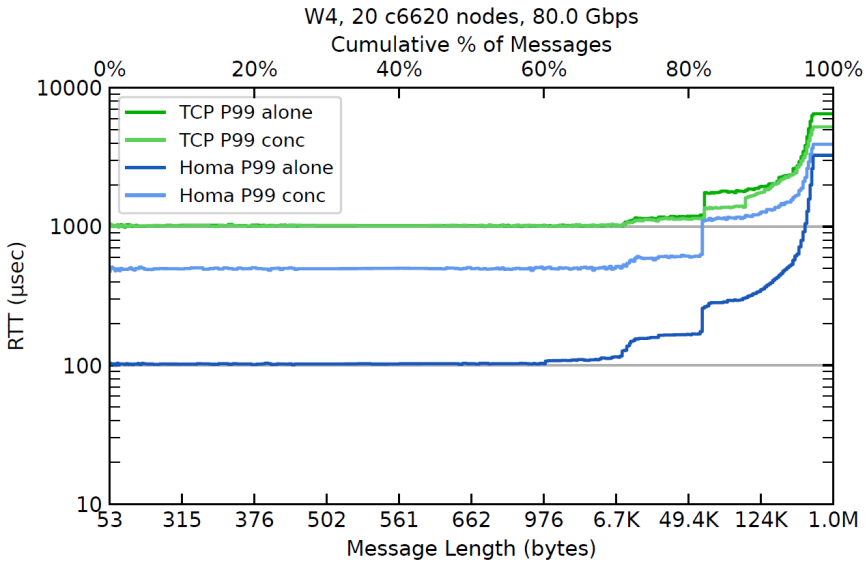


Bypassing

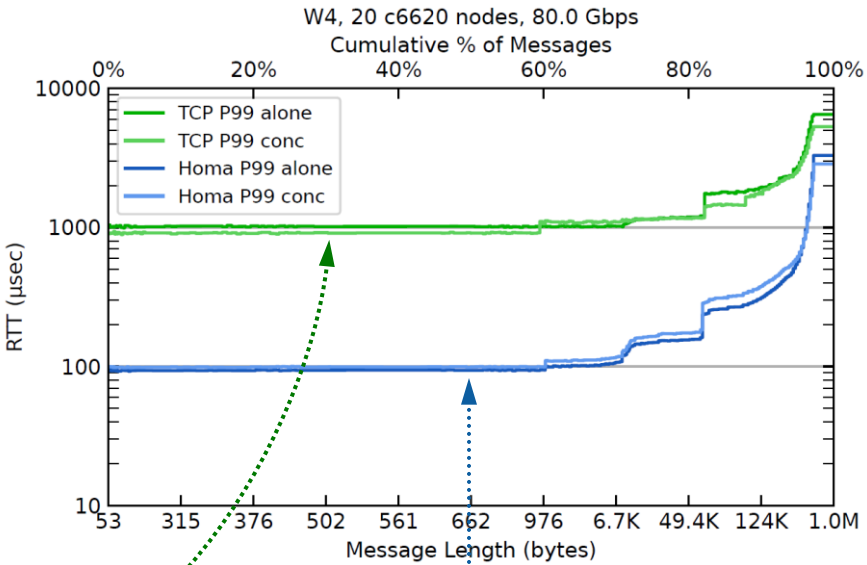
- **Short packets/messages bypass queuing entirely:**
  - Transmit immediately, even if NIC congested
  - Threshold set with sysctl parameter (~1000 bytes)
- **Necessary for efficient pacing:**
  - Single pacer thread can’t keep up with short packets
  - Instead, use multiple cores/channels
  - Avoid overhead of synchronizing on global queues
- **In practice, can’t generate enough short packets to congest NIC**
  - Limited by software
- **An extra SRPT boost for short messages**

Results

Before



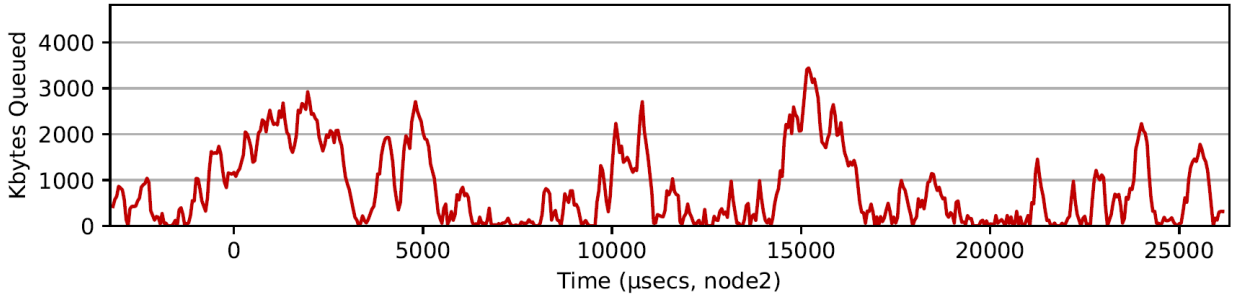
homa_qdisc



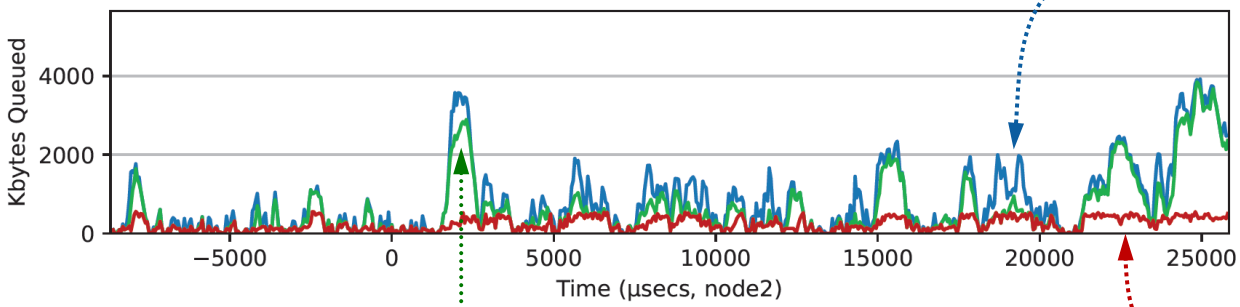
TCP still benefits from running with Homa

Almost no penalty for Homa

NIC Queue Length



Before



Total queuing
(NIC and homa_qdisc)

homa_qdisc

NIC + queued TCP

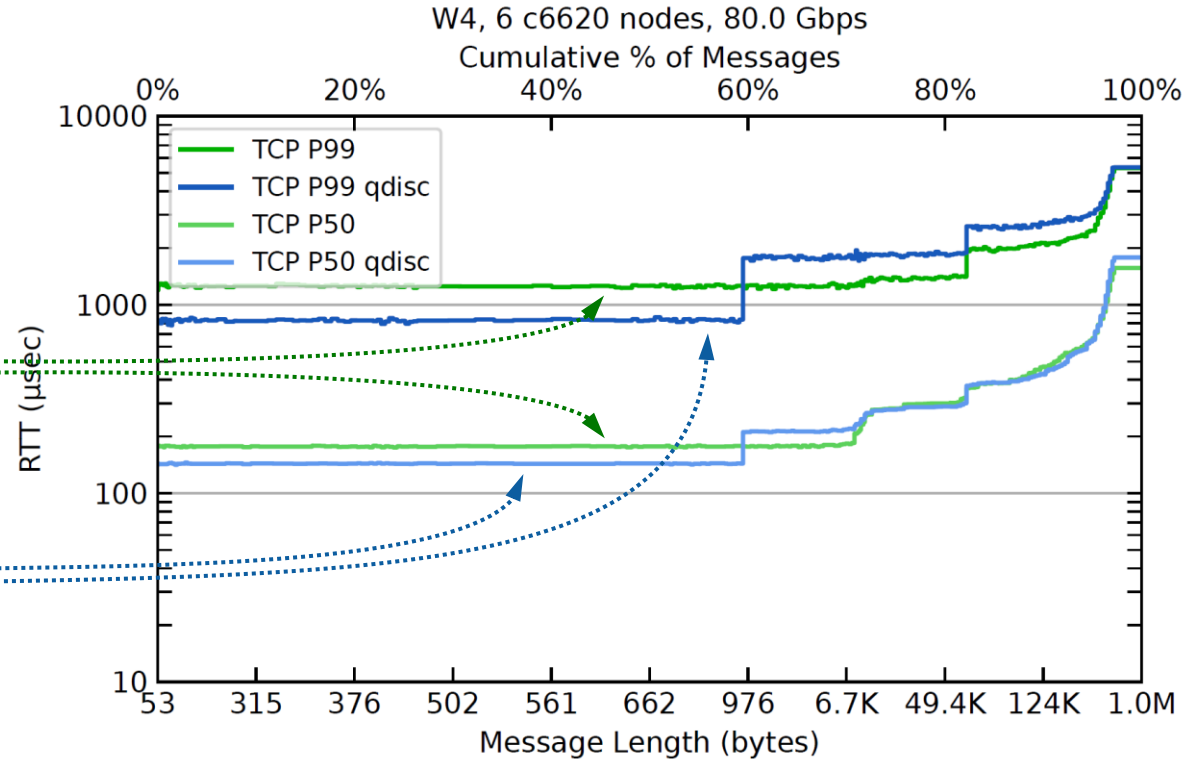
Queued in NIC

TCP Without Homa

- **homa_qdisc provides a bit of SRPT for TCP**
- **Tail latency for short messages drops by 1/3**
- **Overall slowdown drops from 11.6 to 10.8**

TCP normal

TCP with homa_qdisc



Conclusions

- **Homa is not a threat to TCP performance**
  - Your TCP will perform better if other TCP apps convert to Homa
- **Without homa_qdisc, TCP damages Homa performance**
- **With homa_qdisc, TCP doesn't significantly degrade Homa**
- **homa_qdisc helps TCP even without Homa running**
  - homa_qdisc provides a bit of SRPT for TCP

Congestion in the NIC is a significant factor in tail latency