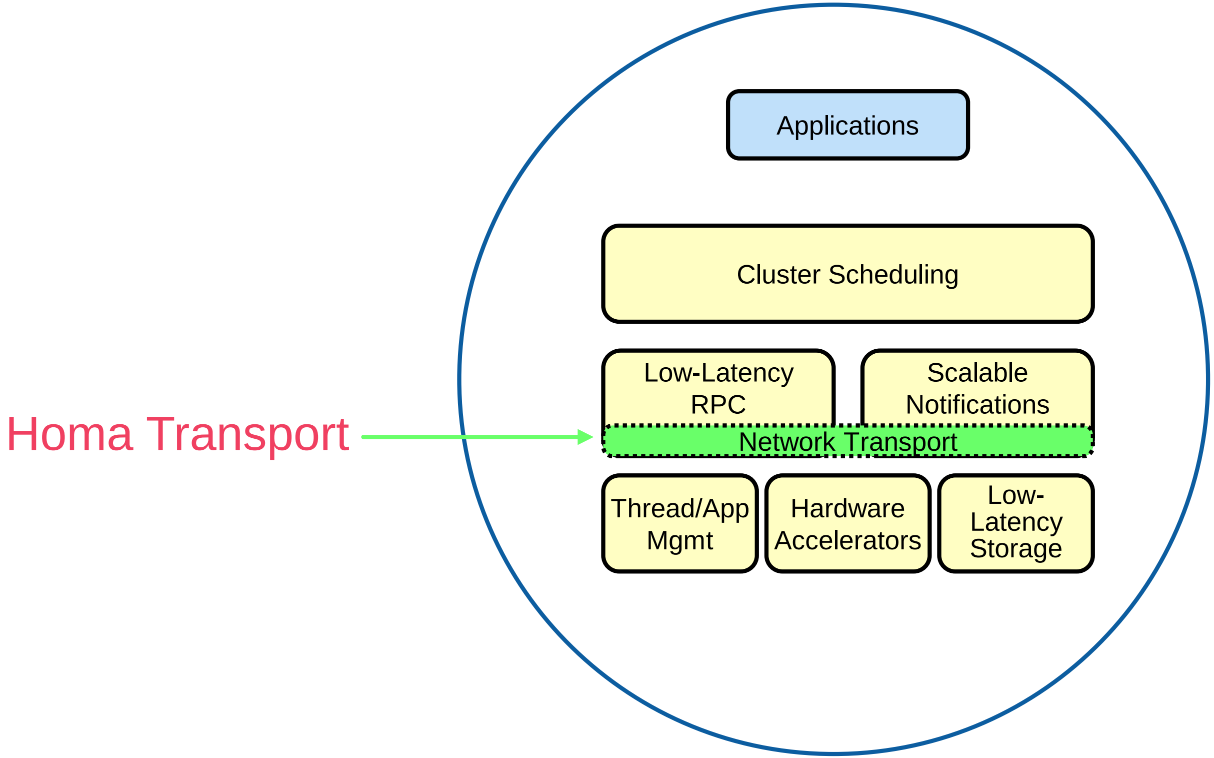
Homa: A Receiver-Driven Low-Latency Transport for Datacenters

Behnam Montazeri
with John Ousterhout, Yilong Li, Mohammad Alizadeh*
Stanford University, MIT*

Granular Computing Platform



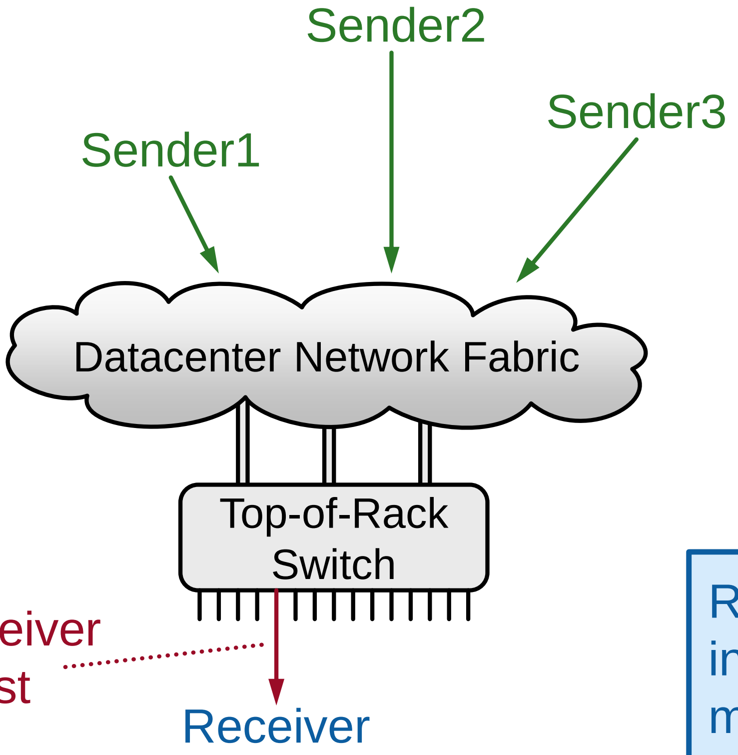
Replace TCP in the Datacenter?

- **TCP bad for datacenters: poor latency at high load**
  - Congestion managed from sender
  - Uses buffer exhaustion to detect congestion
  - At high load, short messages get buffered behind long messages
- **Homa: new approach to datacenter transport**
  - Manage congestion from the receiver
    - Reduce buffer occupancy
  - Use network priority levels for preemption, high bandwidth utilization
- **Homa: New Transport Protocol**
  - **Low-latency** for short messages at high network loads
    - 99%tile tail latency, at 80% network load, is within factor of 2.2 of minimum possible latency
  - **High network bandwidth utilization**
    - More than 90% network bandwidth utilization over various workloads

Outline

- Receiver-side scheduling
- Unscheduled packets to avoid latency overhead
- Preemption using receiver grants
- Priorities for preemption
- Scheduled priorities for bipartite matching
- Evaluation

Congestion: at the Edge



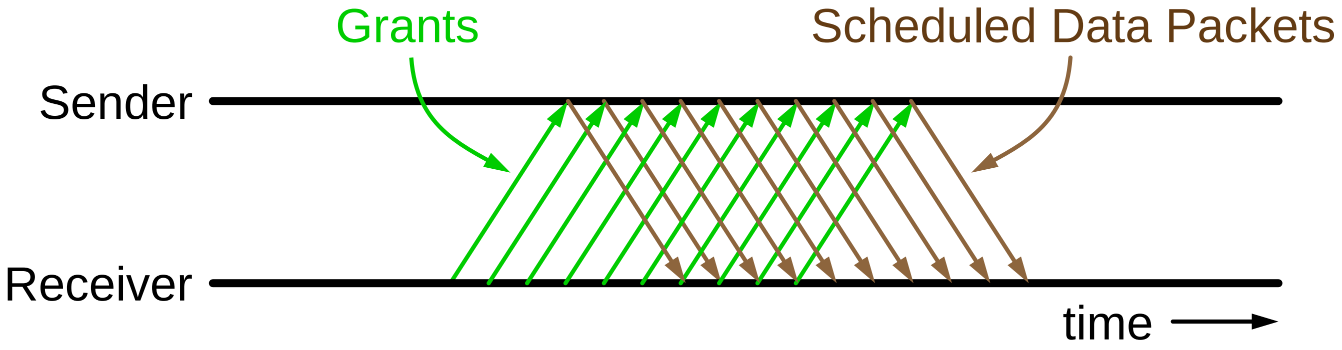
Multi-path core fabric: congestion rare

Congestion at receiver downlink: incast

Receiver has best information to manage congestion

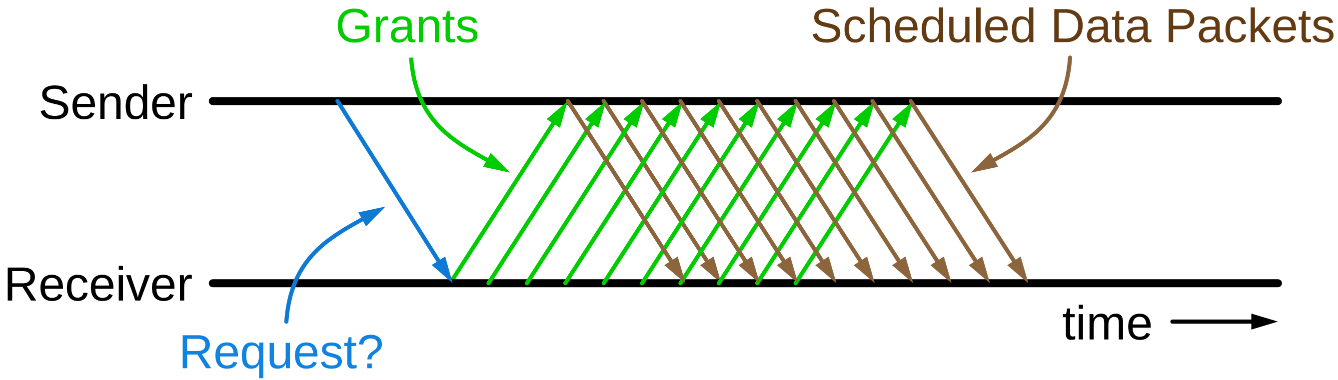
Scheduled Packets

- **Homa receivers schedule incoming packets: one grant per data packet**



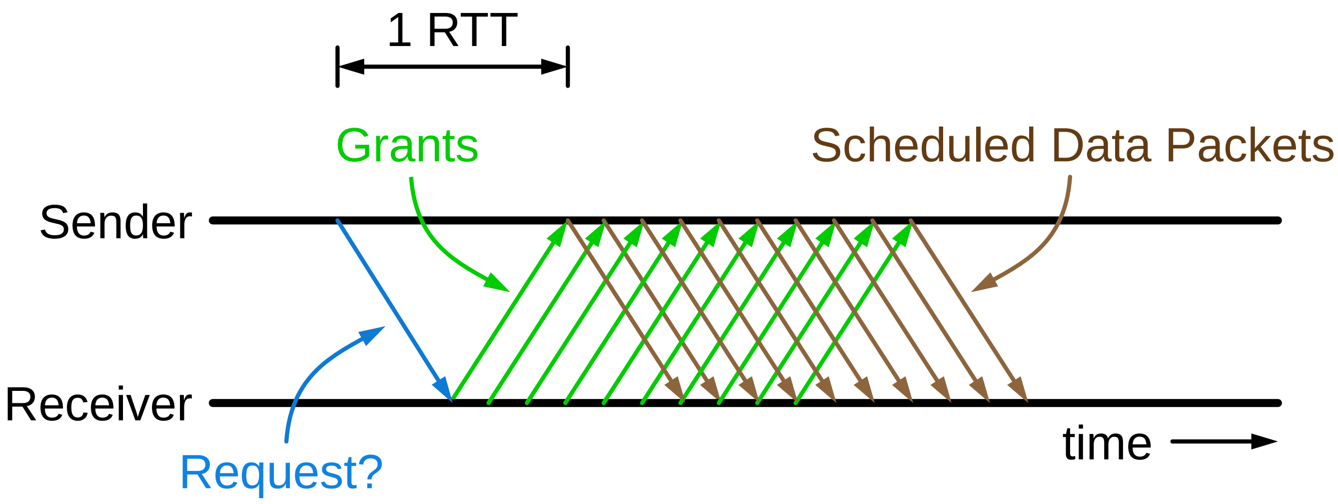
Scheduled Packets

- **Homa receivers schedule incoming packets: one grant per data packet**



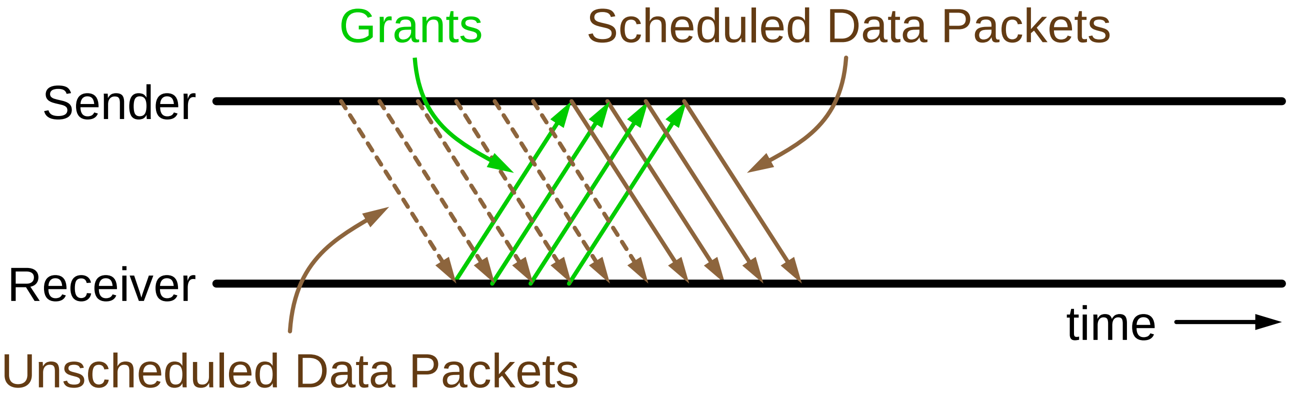
Scheduled Packets

- **Homa receivers schedule incoming packets: one grant per data packet**
- **Problem: 1 RTT additional latency for scheduling**



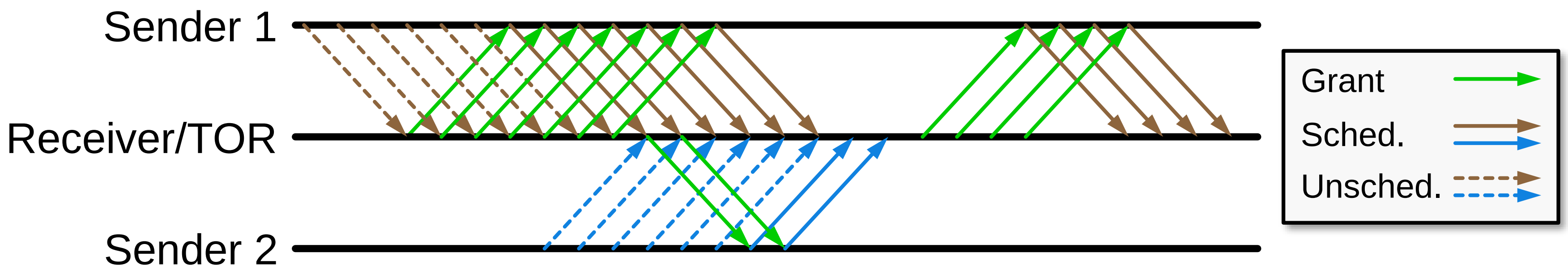
Unscheduled Packets

- **Homa receivers schedule incoming packets: one grant per data packet**
- **Problem: 1 RTT additional latency for scheduling**
- **Solution: 1 RTT of unscheduled packets**
- **Optimal latency when unloaded**



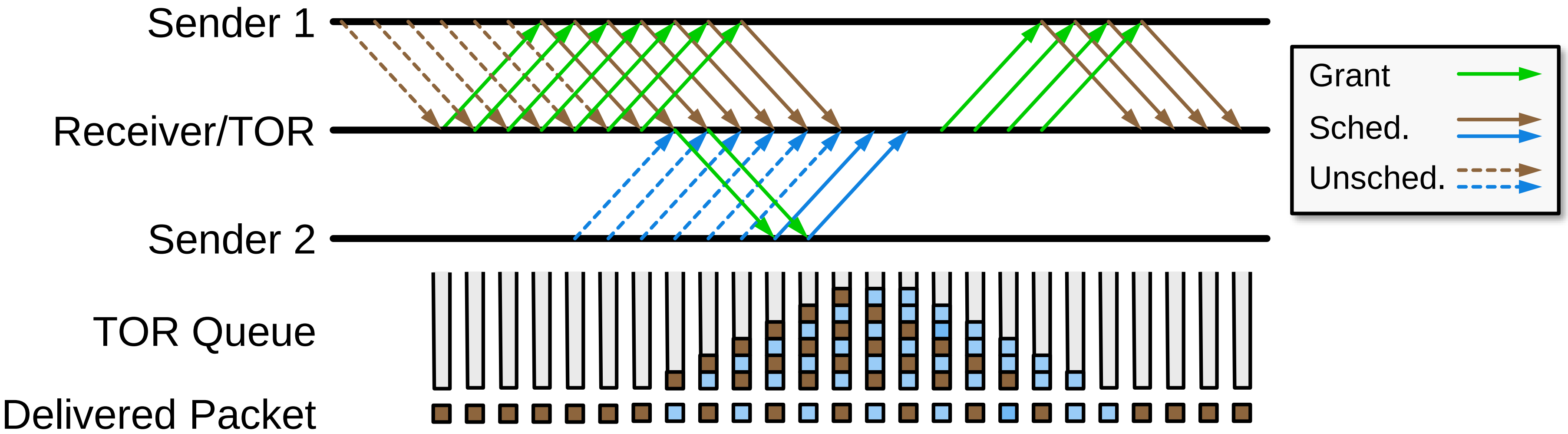
Use Grants for Preemption

- Homa is a message based protocol
- If competing senders, grant only to shortest message
- Policy: Shortest Remaining Processing Time (SRPT)
- Sender includes message size in packets



Preemption Lag

- **One RTT of data from long message already committed when shorter message arrives**
- **Full preemption is delayed by 1 RTT**

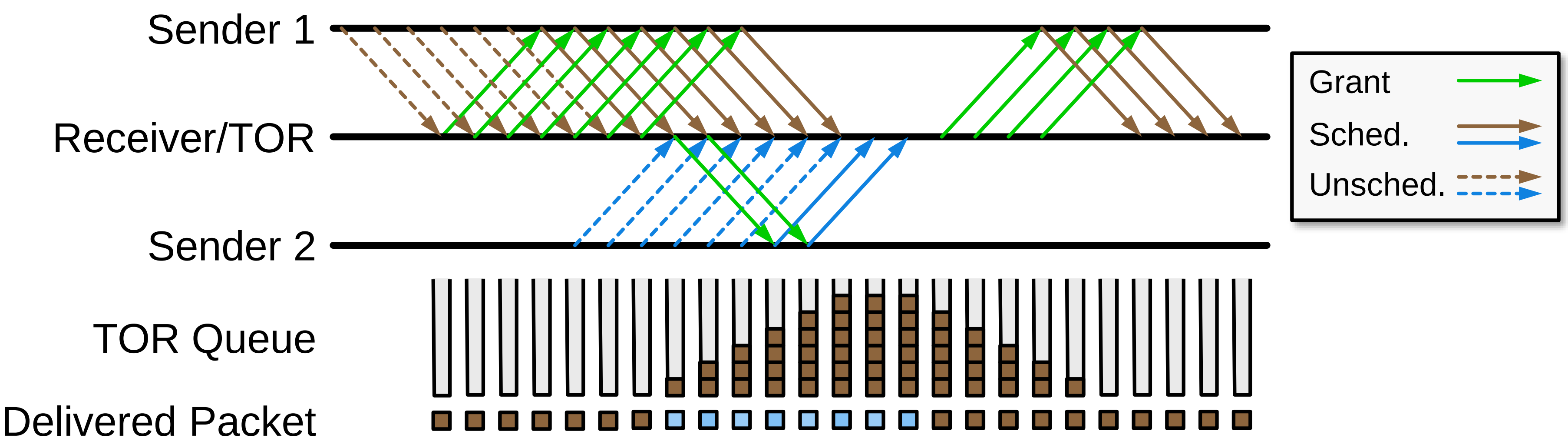


Priorities for Preemption

- **Most network switches support 8+ priority levels:**
  - Each packet header indicates priority level
  - Switch transmits highest-priority packet for each link
- **Priorities for scheduled packets:**
  - Receiver selects priority, indicates in grant
  - When shorter message arrives, use next higher priority level
- **Priorities for unscheduled packets:**
  - Receiver determines cutoff points based on traffic (equal bytes per level)
  - Receiver piggybacks cutoffs on existing packets
  - Sender uses most recent cutoffs for new messages

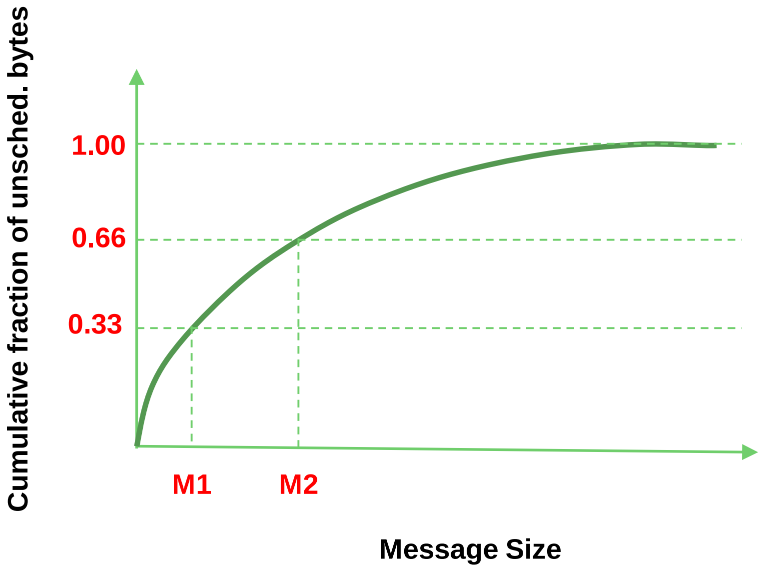
Priorities For Preemption

- **Priorities for scheduled packets:**
  - Receiver selects priority, indicates in grant
  - Higher priority for short message of Sender 2



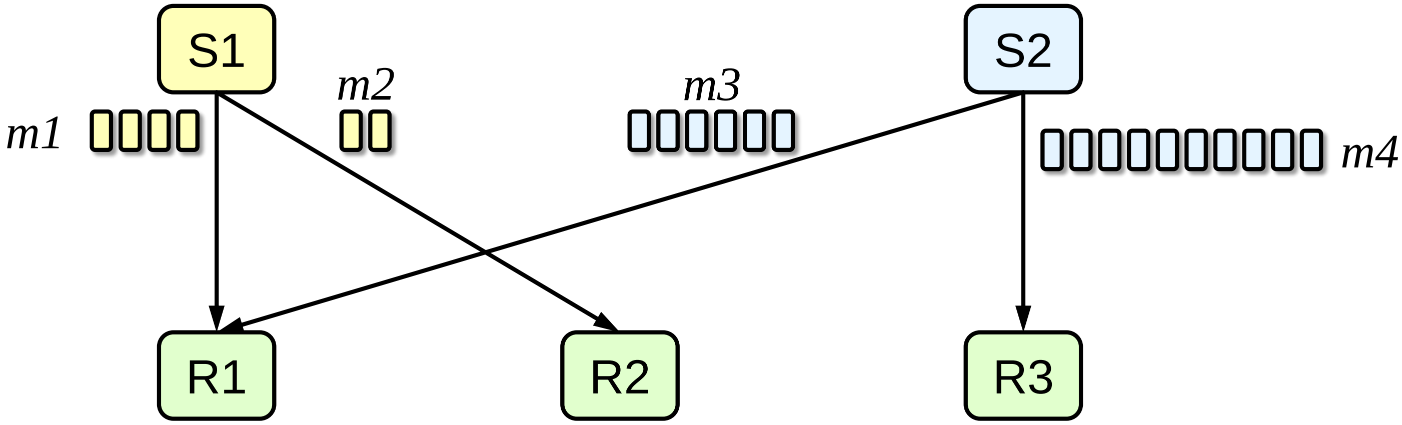
Priorities For Preemption

- **Priorities for unsched. packets:**
  - Shorter messages get higher unsched priority
  - Unsched. packets priorities are statistically assigned by the receiver
- **Priority cutoffs: To receive equal unsched. bytes on priority levels**



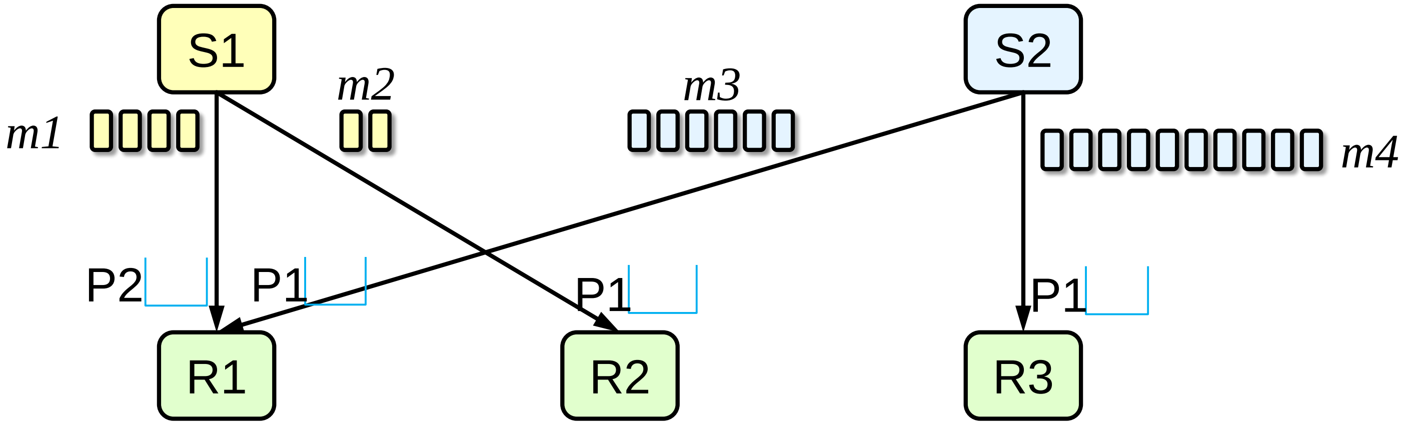
Bipartite Matching Problem

- **S1 uses SRPT to prioritize outgoing messages**
  - Doesn't respond immediately to grant for m1
- **Result: wasted bandwidth at R1**
- **Goals: prioritize shorter messages, keep links busy**



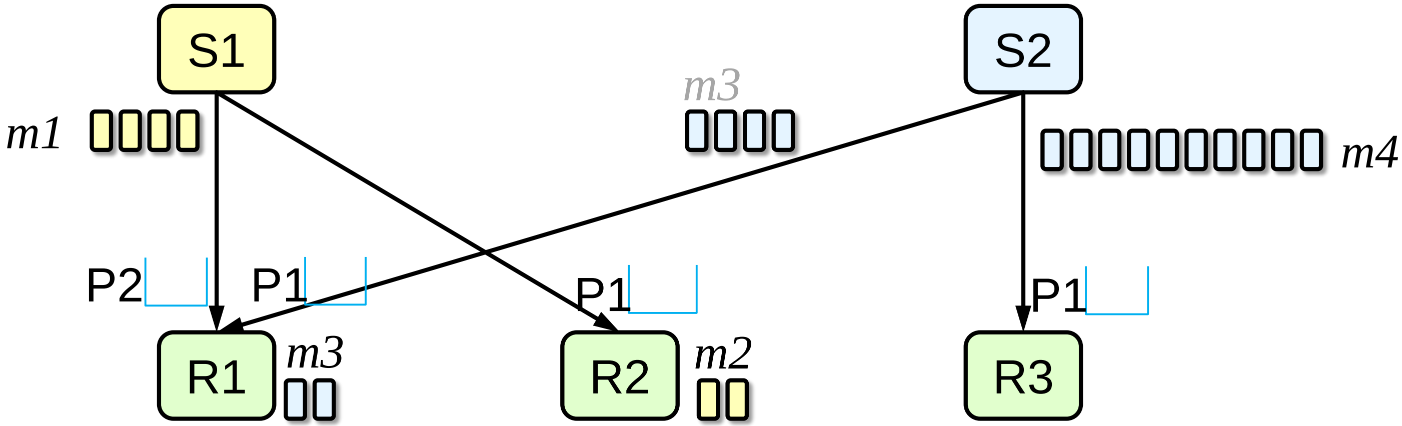
Bipartite Matching Problem

- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



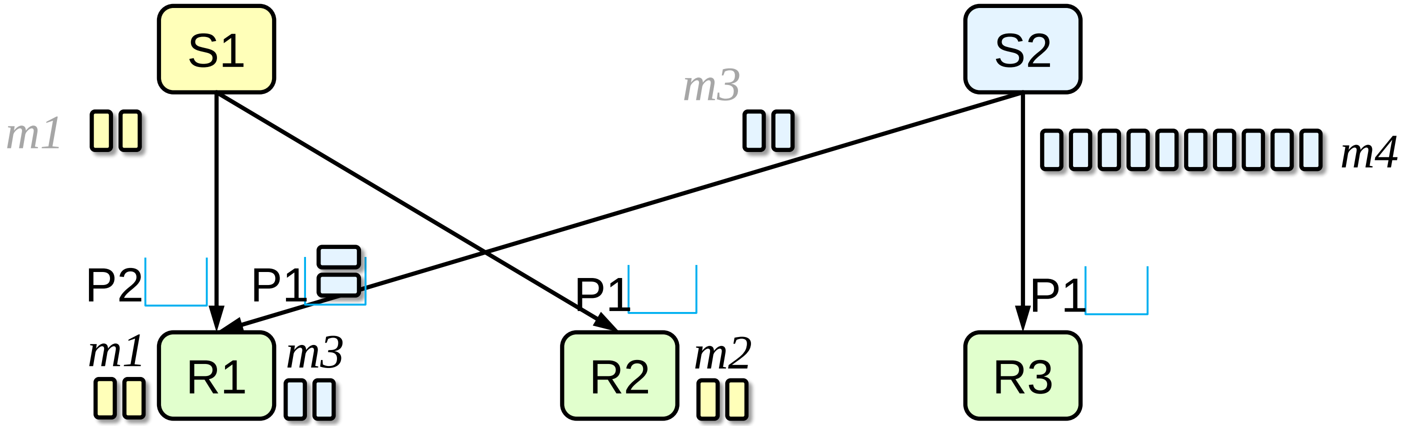
Bipartite Matching Problem

- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



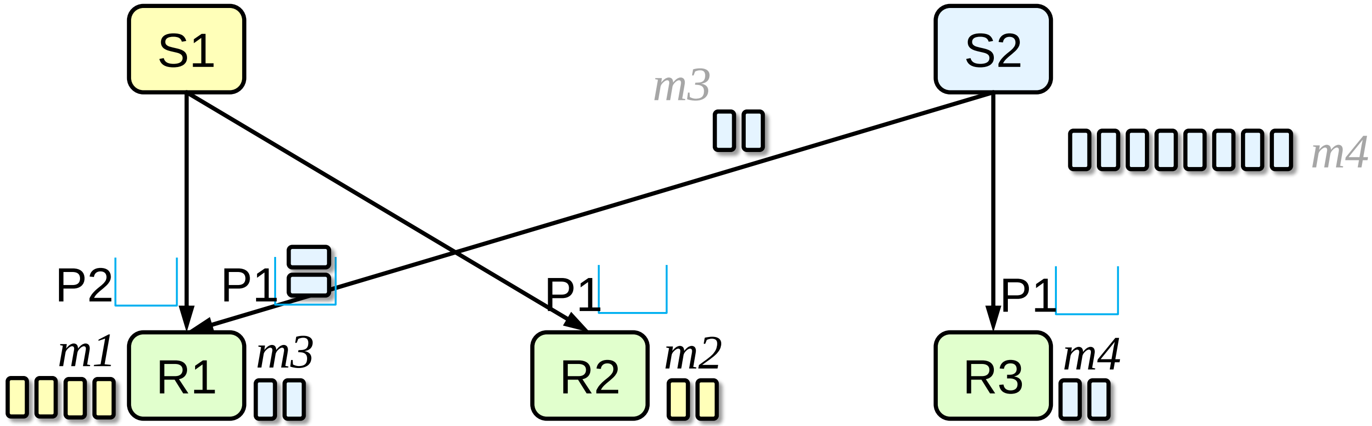
Bipartite Matching Problem

- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



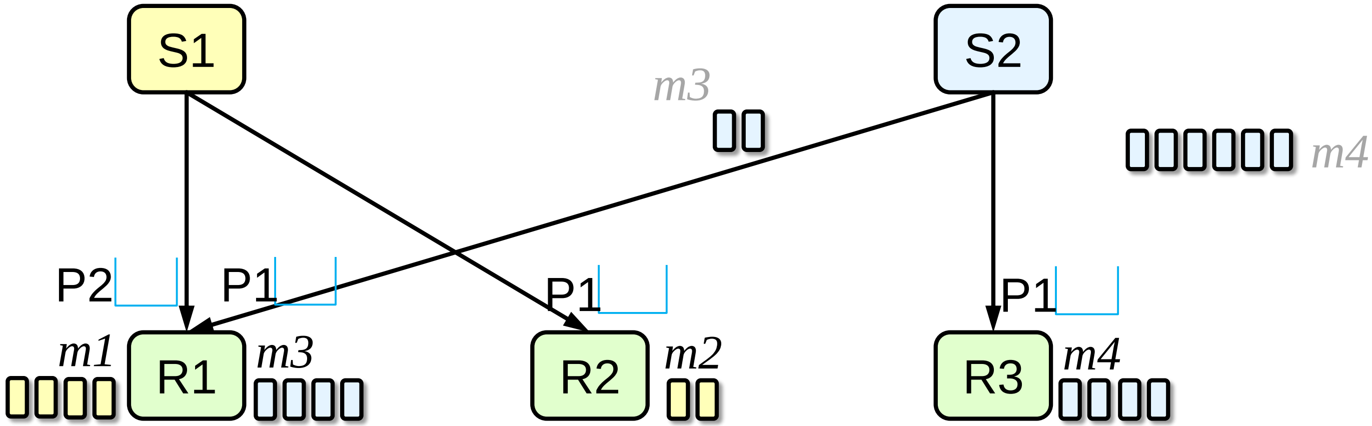
Bipartite Matching Problem

- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



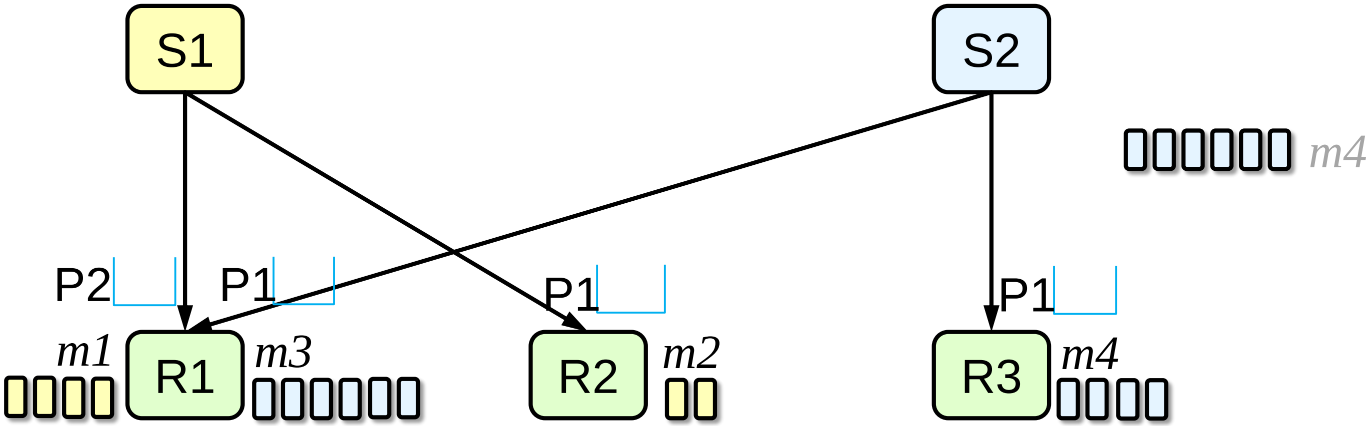
Bipartite Matching Problem

- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



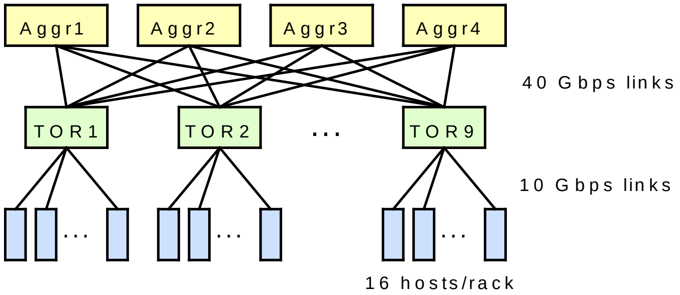
Bipartite Matching Problem

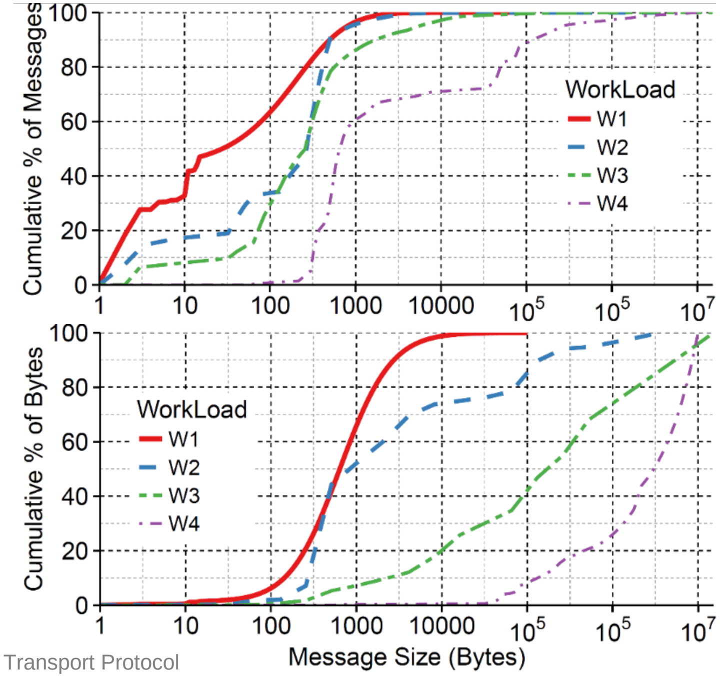
- **Receiver sends grants to multiple incoming messages simultaneously**
  - Different priority level for each message
  - Shortest message gets highest priority
- **If both senders respond, lower priority packets get queued**
- **If highest priority sender doesn't respond, lower priority packets are delivered**



Evaluation Setup

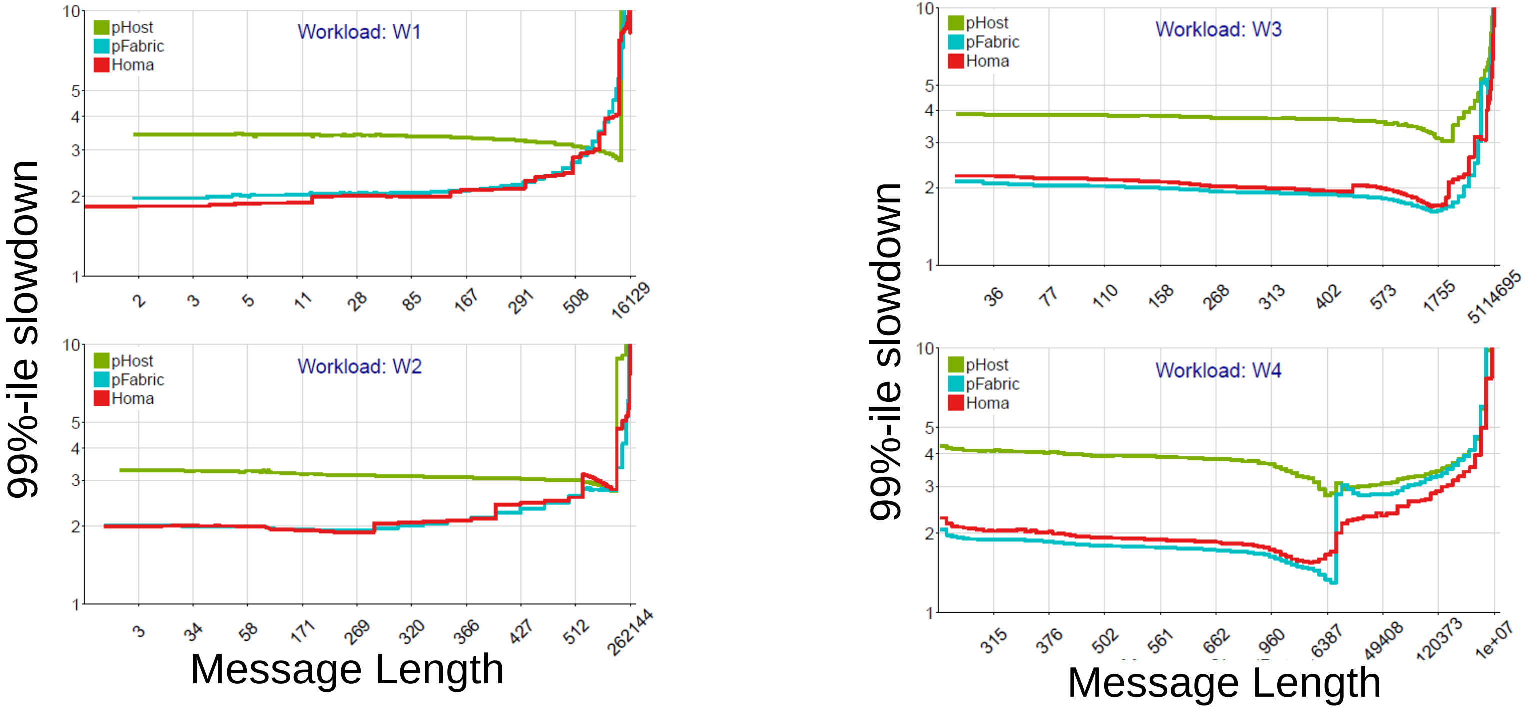
- **Workloads from heavy-tailed to heavy-headed**
  - W1 and W4 from Facebook datacenters, W2 and W3 from Google datacenters
- **Topology similar to what's been used in the prior works**





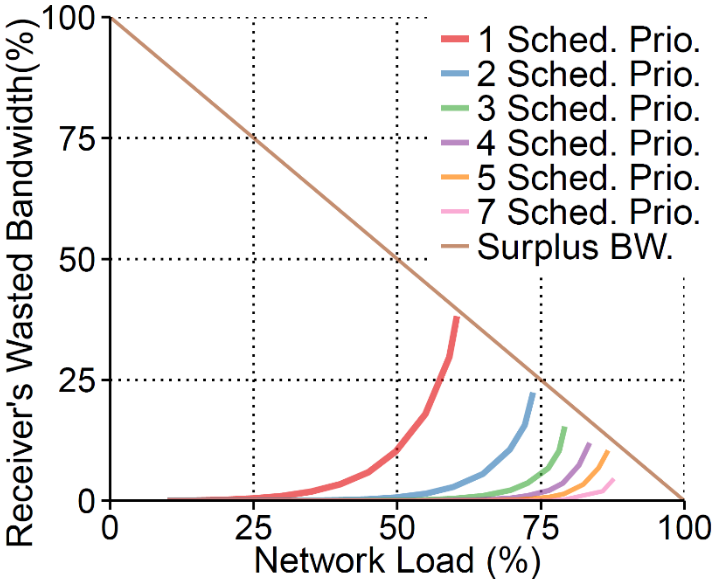
Simulated Performance

- **99%-ile slowdown < 2.5x at 80% network load**



Varying Scheduled Priorities

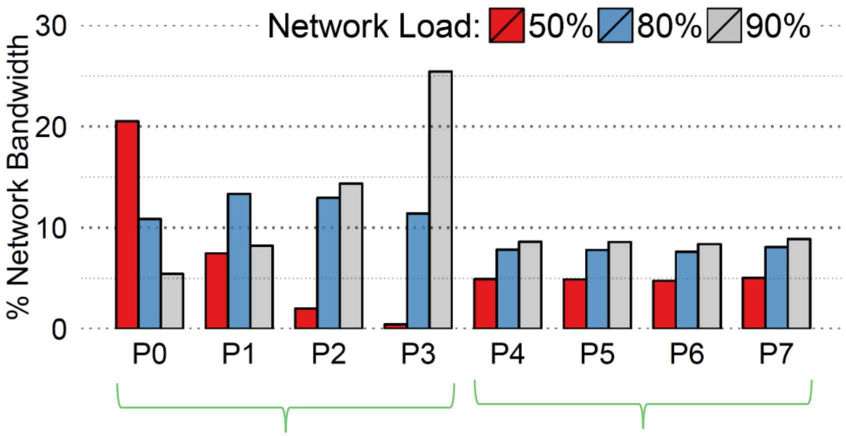
- **Wasted bandwidth: receiver has incoming messages, but its link is idle**
- **Bandwidth wastes if receiver sends grants but sender doesn't transmit immediately**
- **Receiver wastes less bandwidth if more priorities are available**



Priority Utilization

- Equal bytes per unscheduled priority level
- At low loads, sched priorities mainly used for preemption
- At high loads, sched priorities are used for achieving high bandwidth utilization

Workload W3



Sched. priorities Unsched. priorities

Wrap up

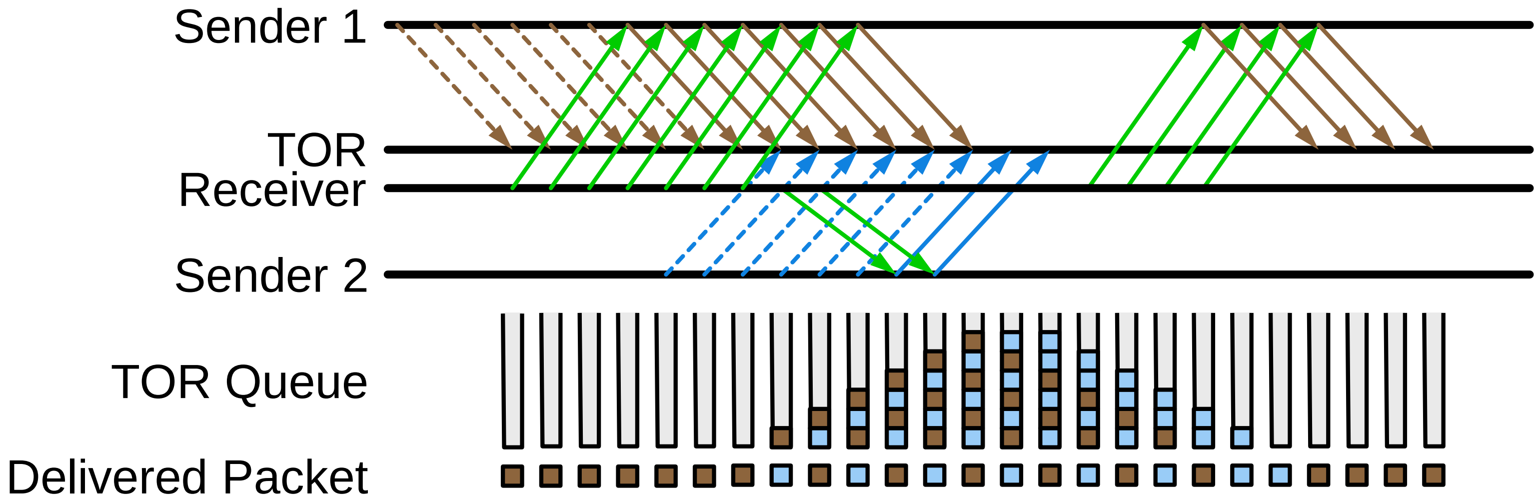
- **Homa:**
  - Low latency and high throughput
  - Supports high network loads
  - Minimizes buffer usage
- **Next step: try to really replace TCP**
- **One possibility:**
  - Linux kernel implementation
  - Support sockets interface (guess message sizes)
  - Automatic switchover (Homa for local, TCP for long-distance)
- **Your good idea goes here**

Questions/Discussion

PLATFORMLAB

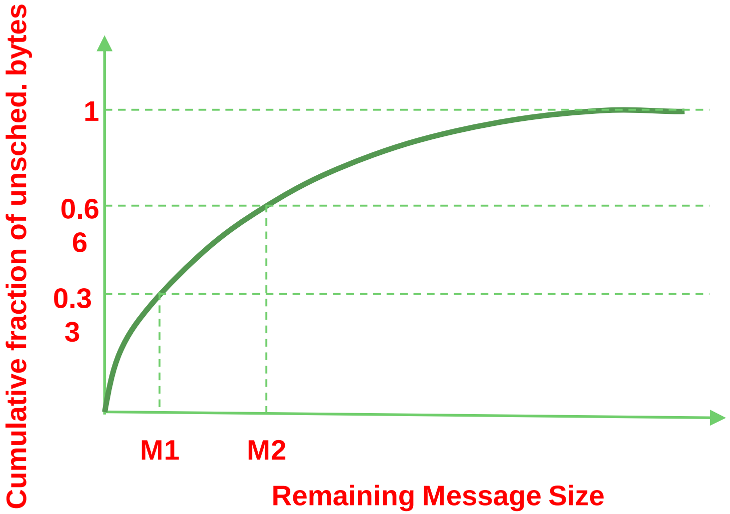
Preemption Lag

- **One RTT of data from long message already committed when shorter message arrives**
- **Full preemption is delayed by 1 RTT**



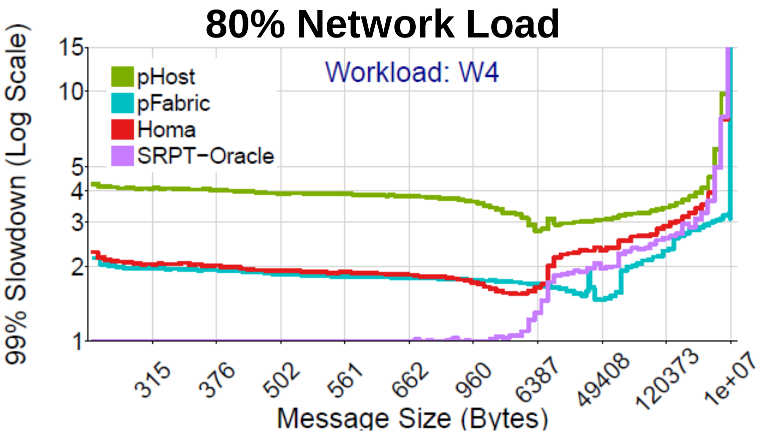
Unscheduled Priorities

- Receiver measures message size distribution
- Shorter messages get higher unsched priority
- Unsched. packets priorities are statistically assigned by the receiver
- Priority cutoffs: To receive equal unsched. bytes on priority levels



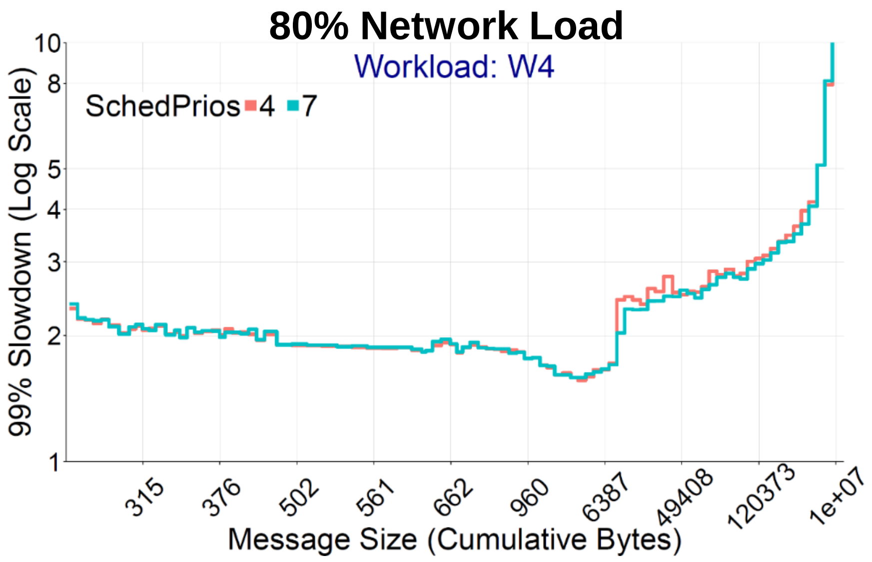
Slowdown vs. Message-Size

- **Slowdown: Message completion time divided by min completion time in an unloaded network**
- **X axis scaled by message cumulative message size distribution**
  - Vertical grid lines at 10% steps of size distribution



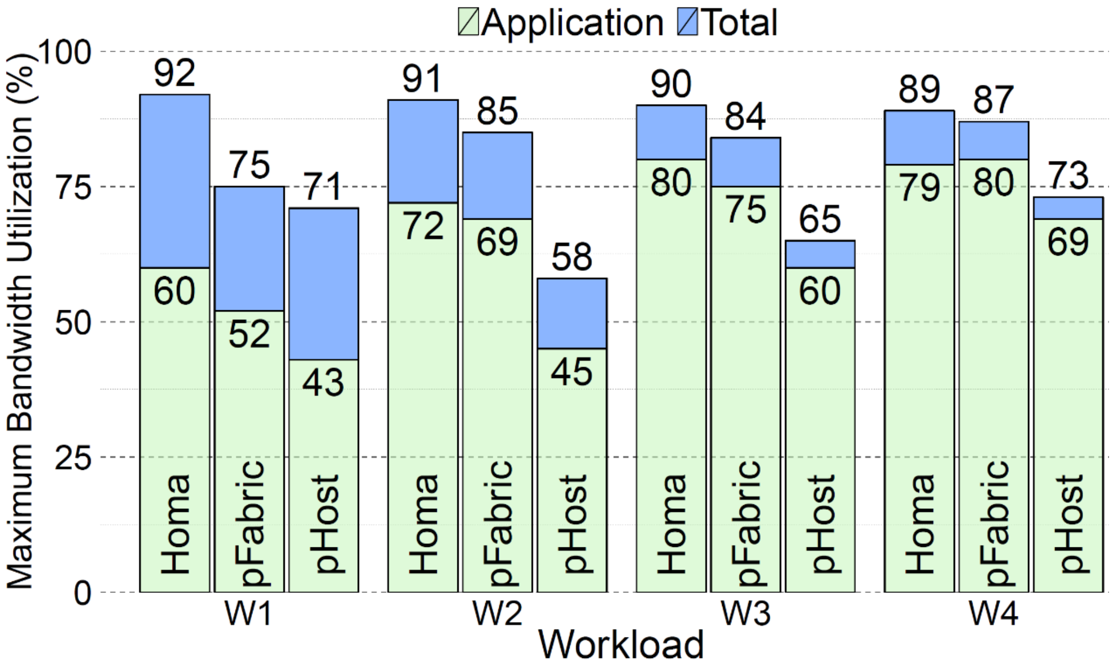
Varying Scheduled Priorities

- We only need 4 priority to achieve perfect preemption
- The remaining 3 priorities are mainly used for increasing bw utilization



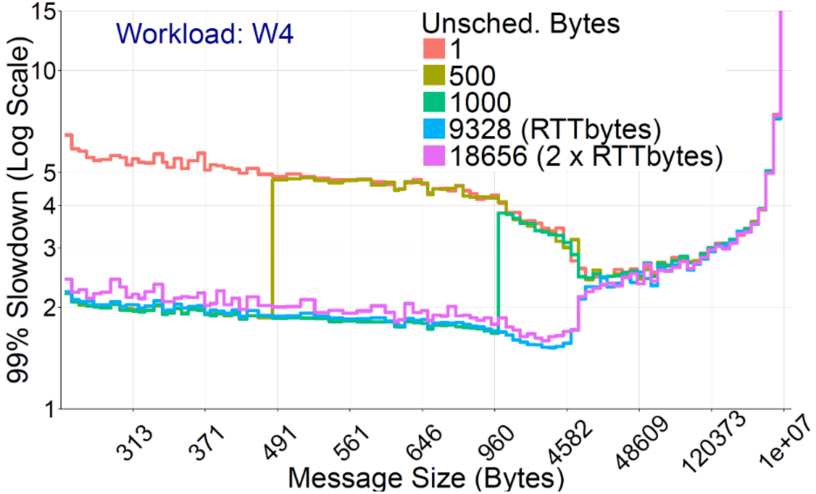
Bandwidth Utilization

- Much higher bandwidth utilization for Homa
- Total = application + overhead header bytes + retransmissions



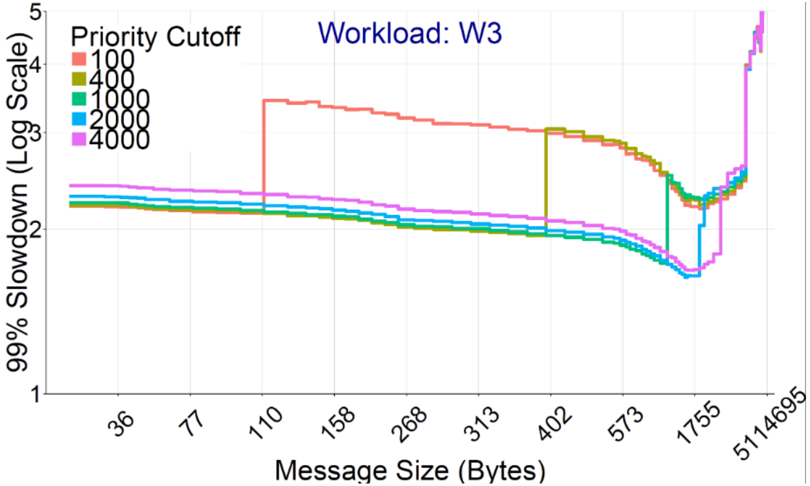
Varying Unscheduled Bytes

- **To avoid one RTT latency overhead, it is optimal to send at least RTTbytes unscheduled bytes**
- **Sending more than RTTbytes is not necessary**



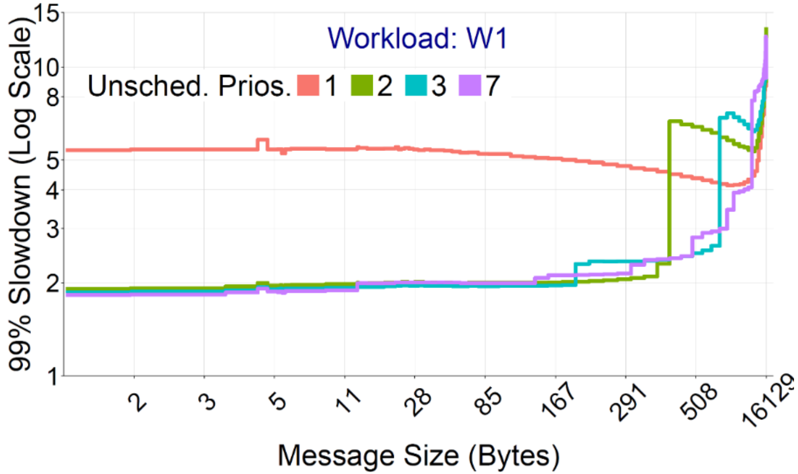
Unscheduled Priority Cut-Off

- **Total of 2 unsched. Priorities**
- **Increase cutoff, reduce latency of larger messages but increases latency of short messages**
- **Homa chooses 1930 bytes for cutoff value**



Varying Unscheduled Priorities

- Total of 8 priorities available
- 95% of total traffic is transmitted in unsched. bytes in workload W2
- Homa chooses 7 priorities for unscheduled packets for this workload which minimizes latency over the message size spectrum



Priority Utilization

- **Equal bytes per unscheduled priority level**
- **At low loads, sched priorities mainly used for preemption**
- **At high loads, sched priorities are all used for achieving high bw utilization**

Workload W3