Leverage Homa: Enhancing Homa Linux for Efficient RPC Transportation

Presenter: Xiaochun Lu, Zijian Zhang

Agenda

- Homa Congestion Control introduction
- Limitation of Homa in RPC context
- Homa Congestion Control enhancements
- Homa RPC streaming enhancements
- Performance evaluation
- Future improvements
- Conclusion
- Q&A

RPC Transport Protocols in Data Centers: TCP, RDMA, and Homa

TCP (Transmission Control Protocol)

- Widely supported
- High Latency on short messages in mixed workloads

RDMA (Remote Direct Memory Access)

- High throughput and Low latency
- Limitation on number of concurrent connections
- Requires RDMA-capable network and switch

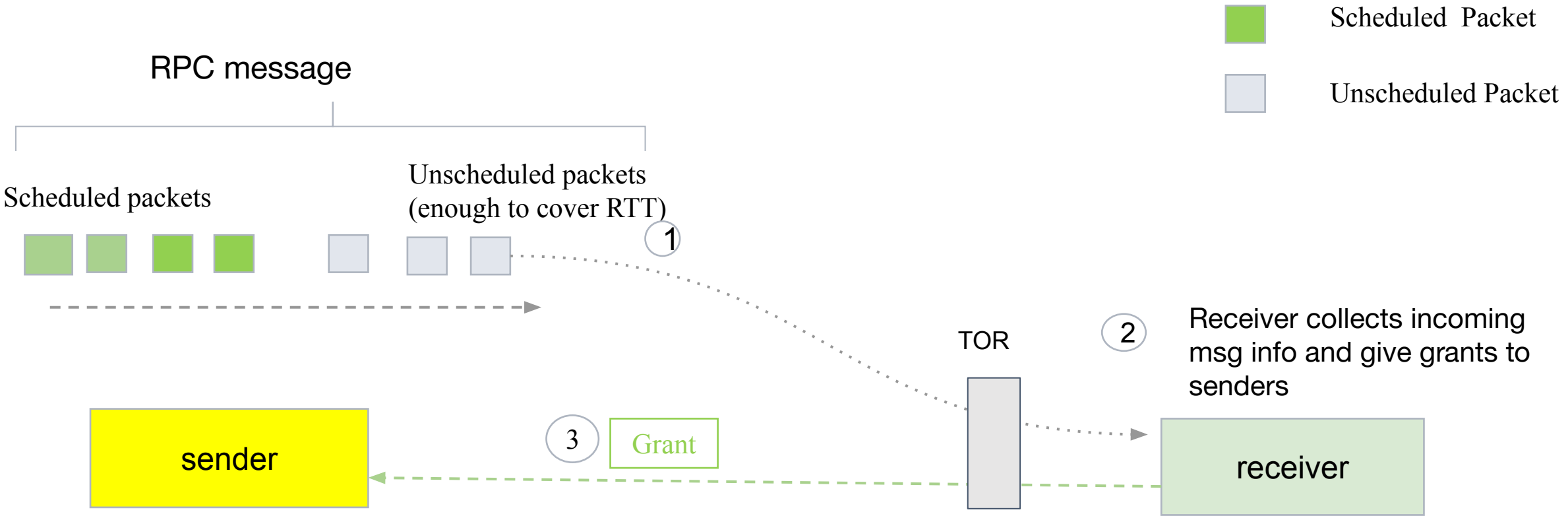
Homa:Specifically designed for Data center RPC framework

- High scalability
- Low latencies
- High throughputs

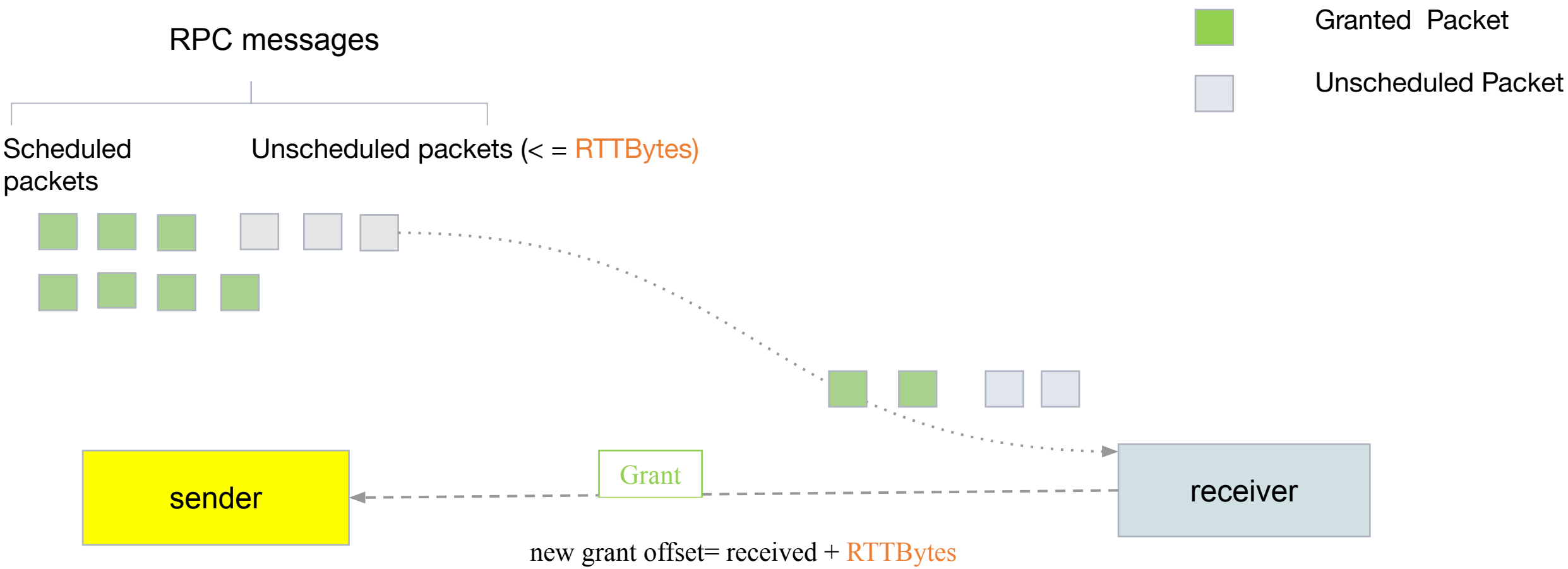
Homa protocol introduction

- **Primarily designed for datacenter networks characterized by extremely low latencies.**
- **Message based protocol.**
- **Connectionless: no connection cost, no long life connection state.**
- **Ensure reliable transmission through retransmission mechanism.**
- **Homa implements unary RPC.**

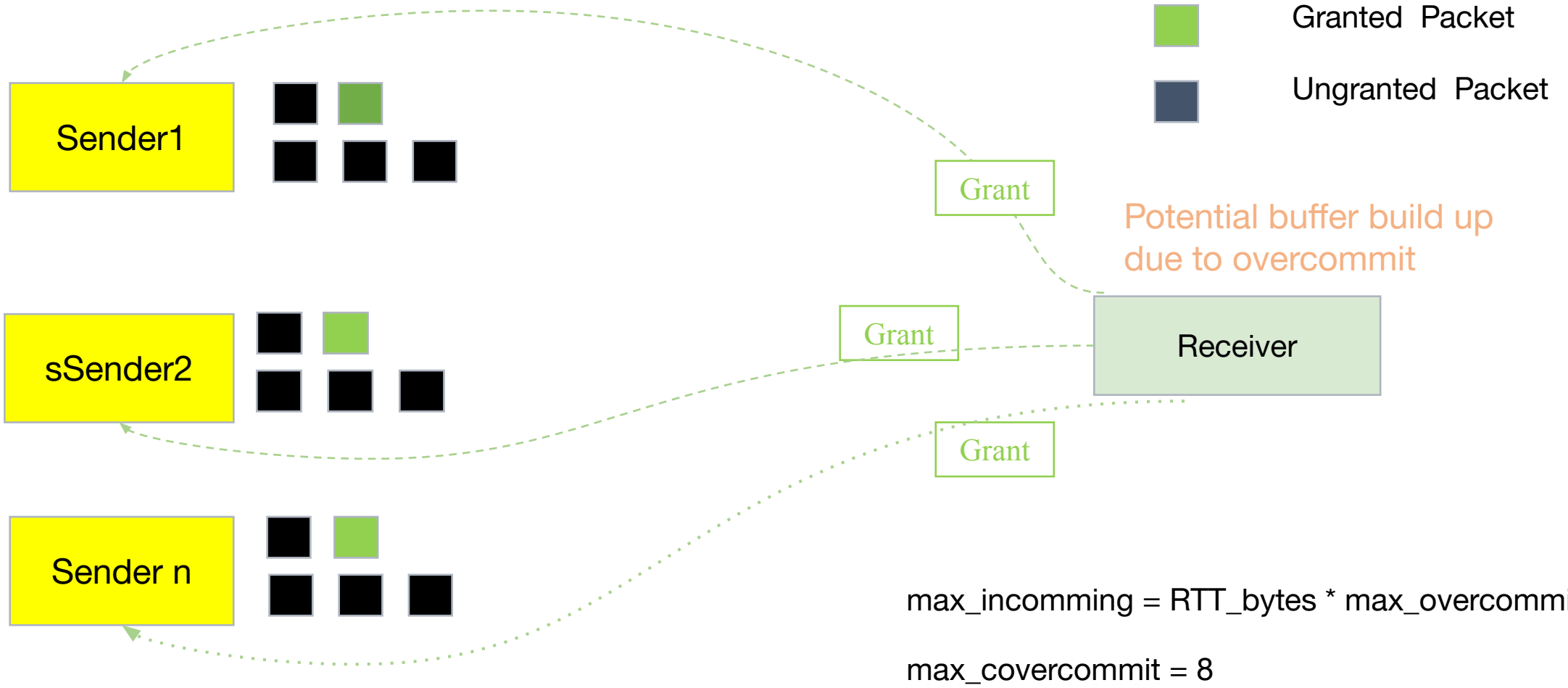
Receiver Driven Congestion Control



Preset Fixed Window (RTTBytes)



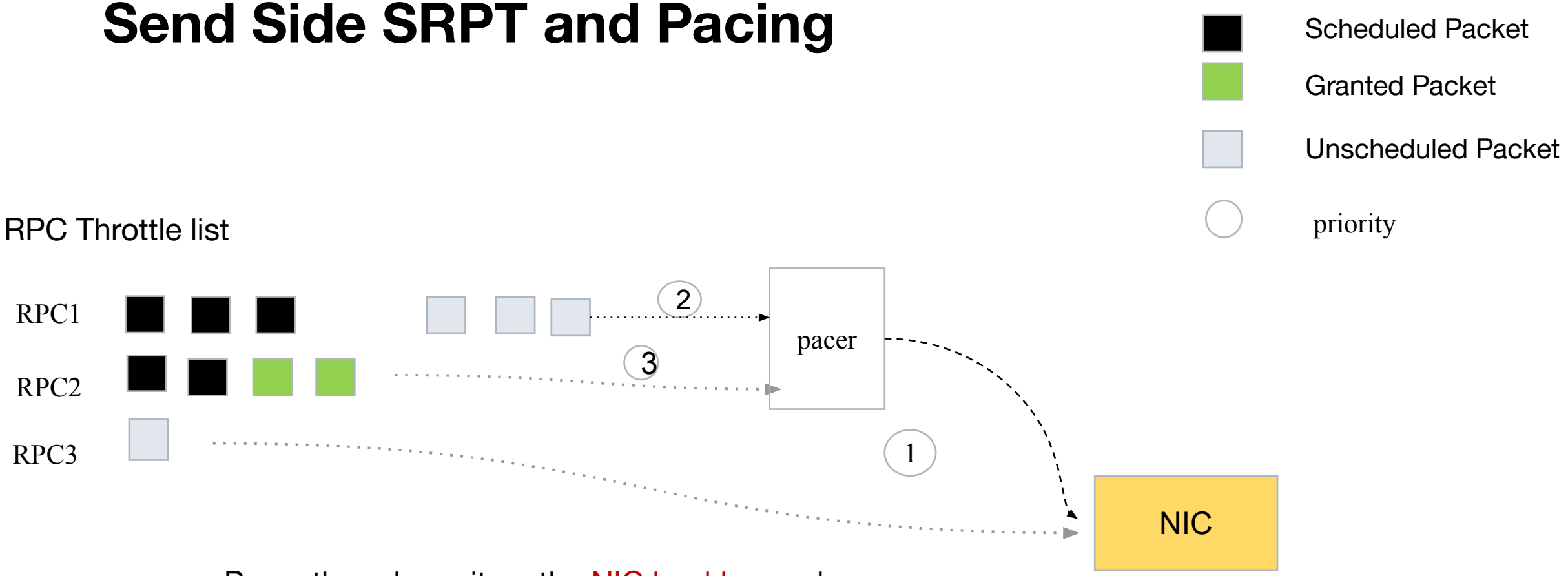
Overcommit Grants for Maximum Throughput



max_incomming = RTT_bytes * max_overcommits .

max_covercommit = 8

Send Side SRPT and Pacing



Pacer thread monitors the NIC backlog and transmits Packets from the throttled list to nic in SRPT order.

Limitations of Homa as RPC transport protocol

Inefficient Pipelining for Large Message

- Homa's message-based interface, while ensuring complete message delivery, hinders efficient pipelining, resulting in decreased throughput for larger RPC messages(size >50k) compared to TCP

Non-standard Socket API interface

- It is not easy to map Homa RPC ID to existing RPC framework.
- No long lived RPC: A RPC stream RPC is consisted of many Homa RPCs, which incurs the overhead of creating and reclaiming them.

RTTbytes Challenge

Performance sensitive to RTTBytes config

- Single preset value is not enough for diverse RTT and receiver downlink bandwidth.
- Real-time per-peer RTT detection is essential

Weak Congestion Control when RTT is large than 20 us

- Large RTTBytes inviting incast congestion
- Low RTTBytes is not able to cover RTT

Single , Static Congestion Window is Insufficient

- **Unscheduled Window and Scheduled Window(for granting) need different values**
  - Serves different purpose.
  - Unscheduled window is based on peer side bandwidth. It defines maximum bytes of packets allow to be sent out without permission.
  - Grant window is used for giving grant to scheduled bytes, which works in more predictable way.

- **RTT changes with network condition, which demands dynamic adjustable window**

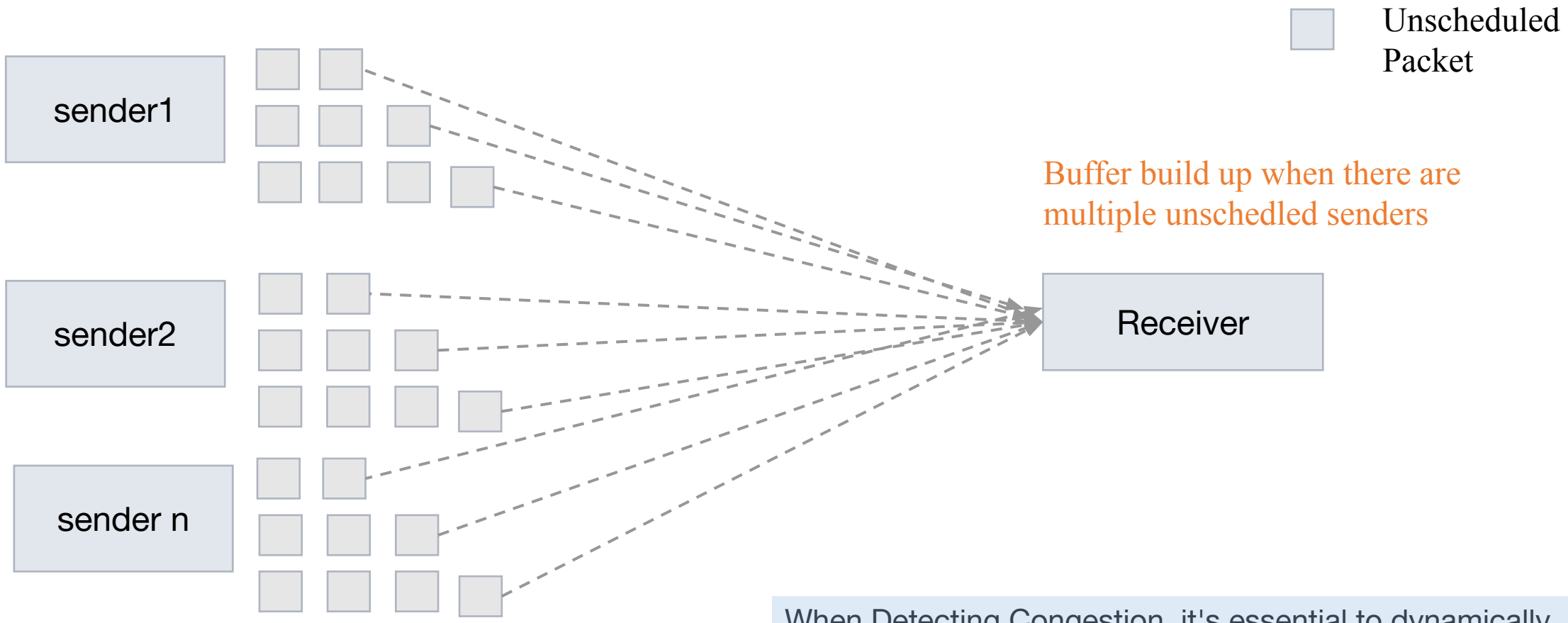
Can Homa Traffic Coexisting with TCP?

Not intentionally designed to share downlink with other protocol

In practice , network resource needs be shared with other protocols like TCP, need explore whether two traffics can coexist**.**

How Homa detect congestion due to coexisting of other type of traffic ?

Unscheduled Packet Incasting



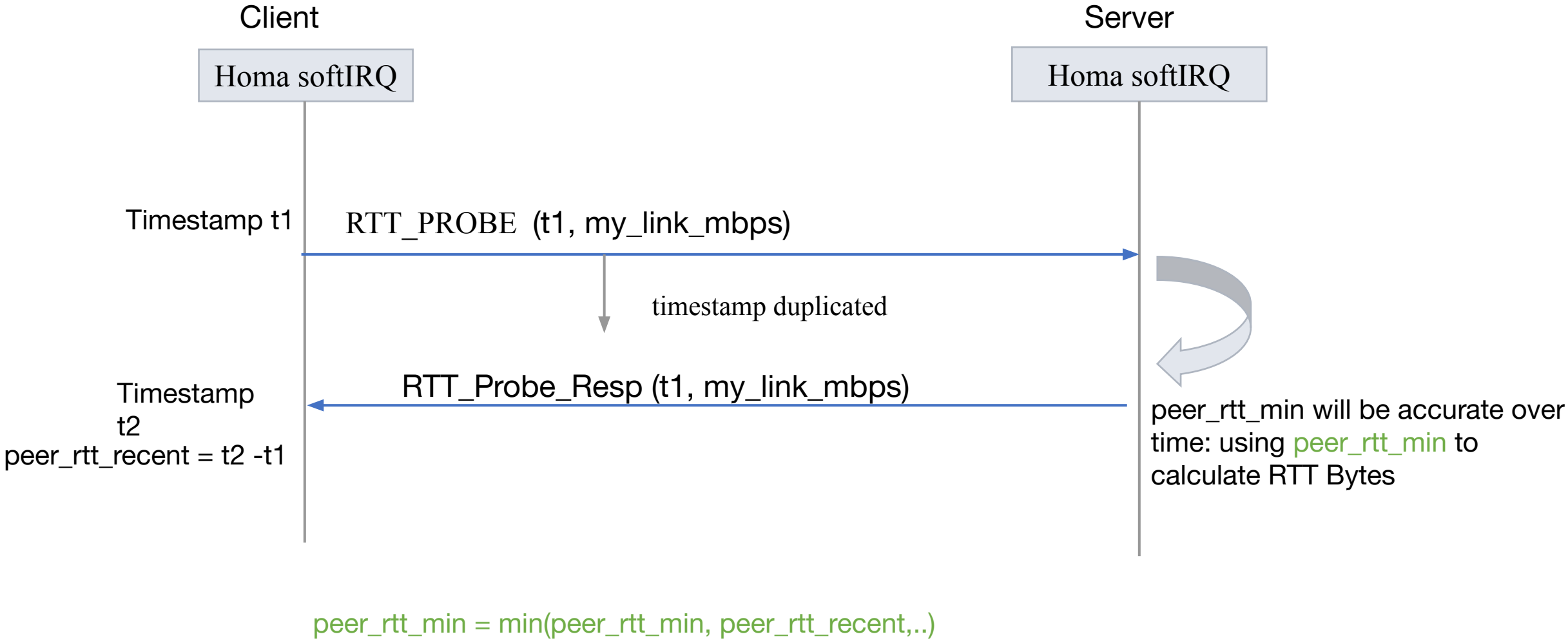
When Detecting Congestion, it's essential to dynamically reduce unscheduled Window to prevent congestion

Homa Congestion Control Enhancements

Dynamic Per Peer Adjustable Window

- Dynamic per-peer RTT detection.
- RTT-informed congestion detection.
- Adaptive per-peer adjustable window based on congestion

Dynamic Peer RTT Detection



peer_rtt_min = min(peer_rtt_min, peer_rtt_recent,..)

RTT Informed Congestion Detection

peer_rtt_min: The minimum RTT value detected over time for this peer.

peer_rtt_high: The high threshold of RTT.

*set peer_rtt_high to 8 * peer_rtt_min*

IF peer_rtt_recent > peer_rtt_high
*   set congestion to true*
ENDIF

Receiver - Scheduled Window

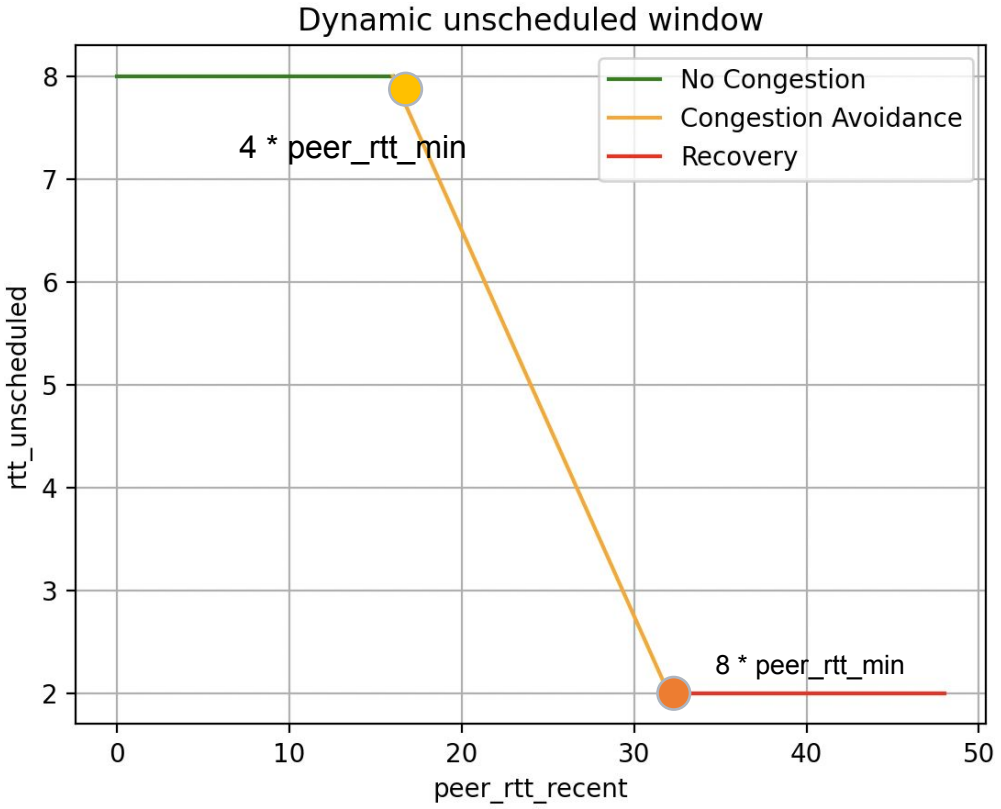
Using rtt_grant as reference RTT to calculate scheduled window for grant:

*peer_rtt_grant = 3 * peer_rtt_min;*

*scheduled_window = peer_rtt_grant * local_link_mps / 8 ;*

*max_incomming = scheduled_window *max_overcommit;*

Sender - Dynamic adjusting unscheduled window

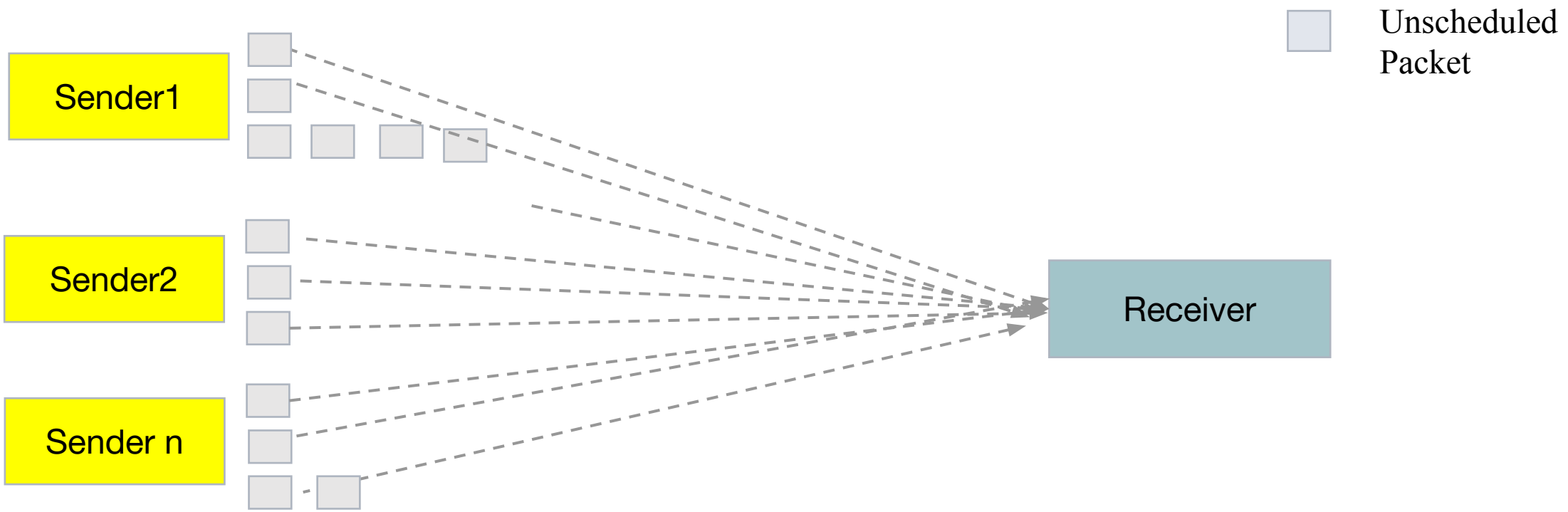


Unscheduled Ratio

Unscheduled_ratio = Total unscheduled packets / Total length of all messages

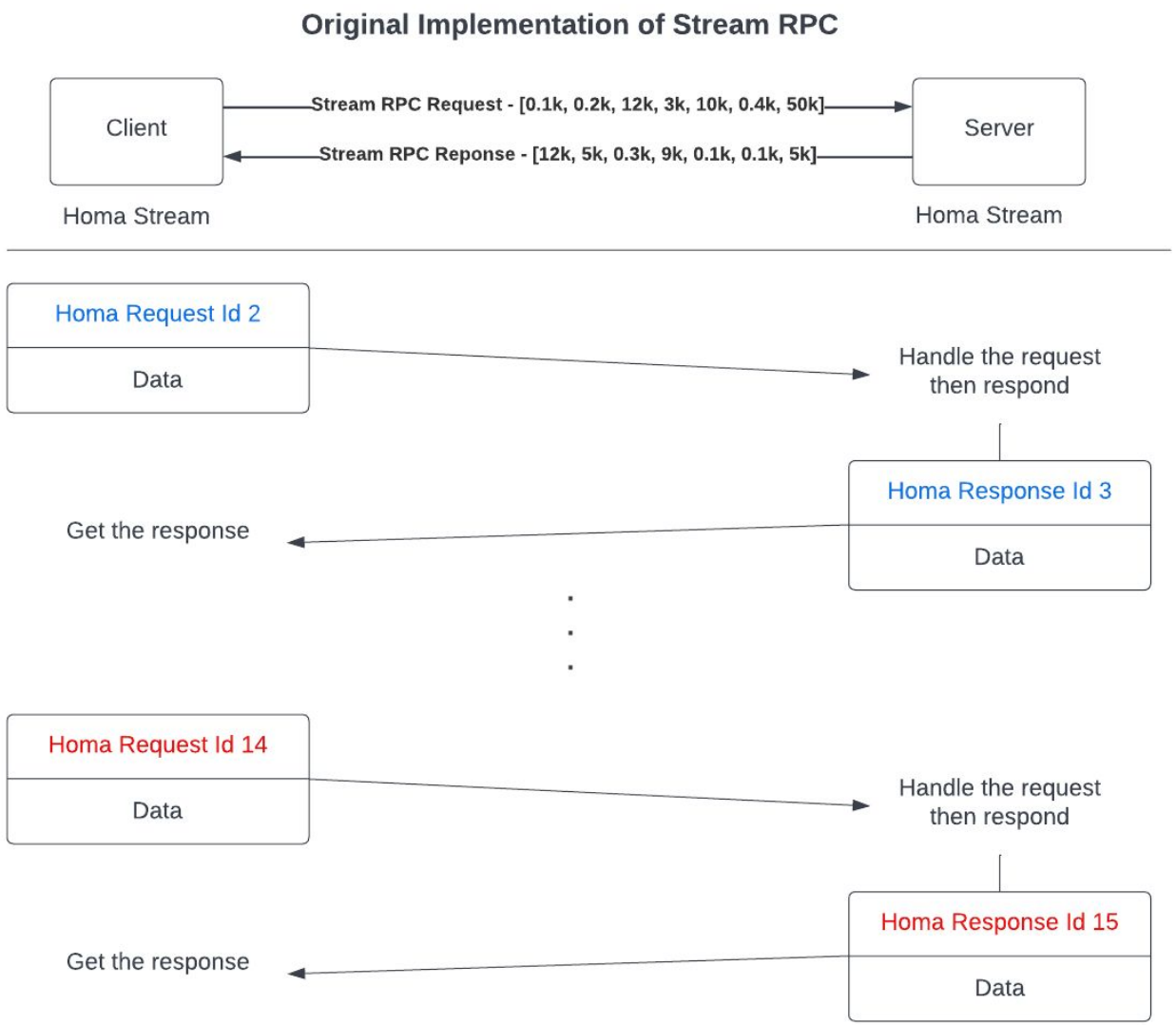
IF unscheduled_ratio is less than %40 && rpc throttle list is not empty
Set rtt_unscheduled to half of current value
ENDIF

Incasting with Congest Control

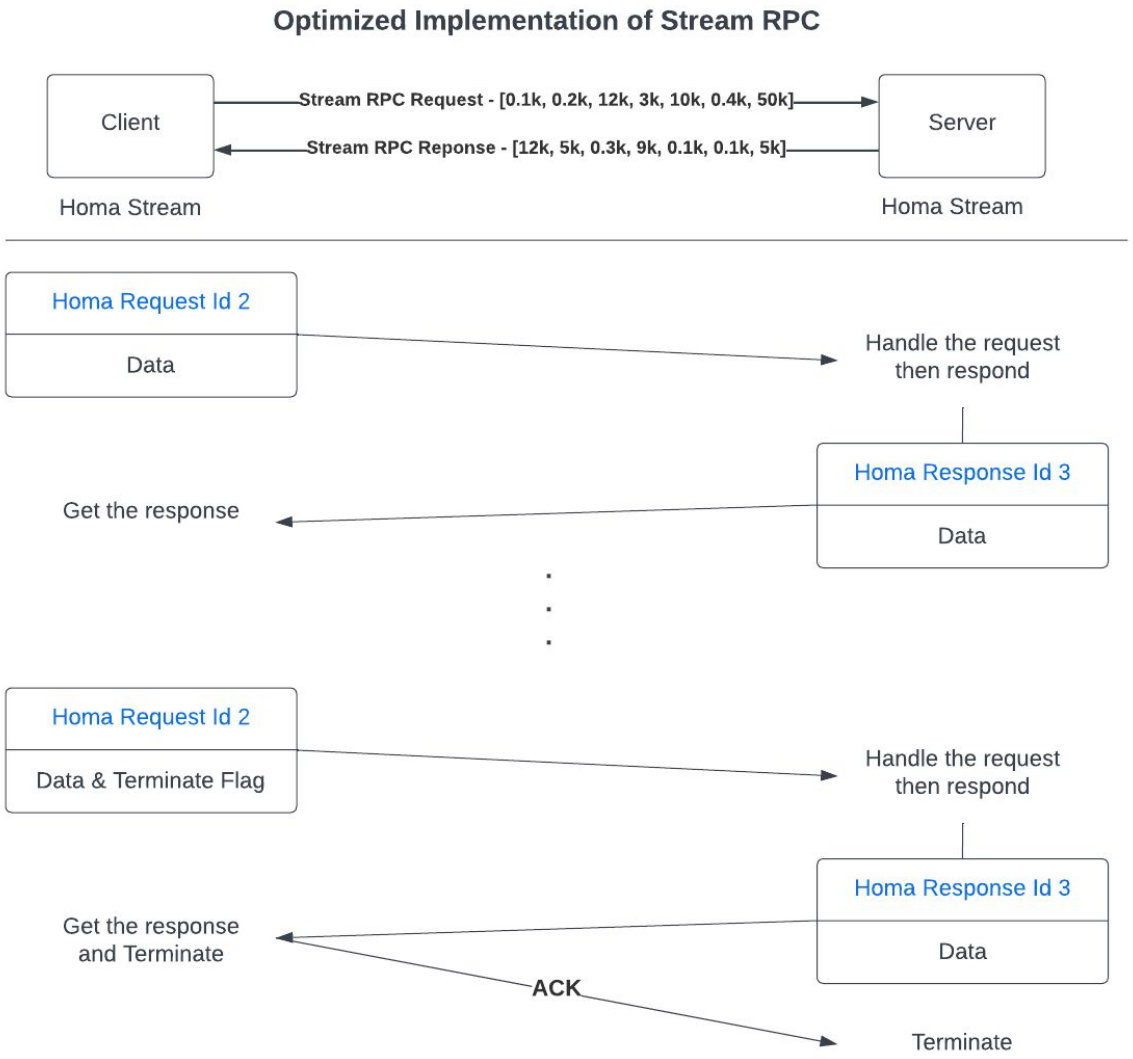


When Detecting Congestion, it will send less unscheduled packet

Homa RPC Streaming Enhancements



Homa RPC Streaming Enhancements



Testbed Setup

25G network:

CPU: Intel(R) Xeon(R) Platinum 8163 (96 core,2.50GHz) RAM: 400G DIMM DDR4 NIC: Mellanox ConnectX–4 Lx 25 Gbp

TOR Switch; Arista DCS-7050SX3-48YC12-F 25G ports

100G network:

CPU: Intel(R) Xeon(R) Silver 4314 (64 cores, 2.4 GHz) RAM: 400 GB DIMM DDR4 NIC: Mellanox Technologies MT28841 dual-port 100Gb/s

TOR Switch: Ruijie Networks RG-S6580-48CQ8QC 100G ports

Testbed Setup

Name	Mean	Description
W2	433	Search application at Google [33].
W3	2423	Aggregated workload from all applications running in a Google datacenter [33].
W4	60175	Hadoop cluster at Facebook [32].
W5	385315	Web search workload used for DCTCP [1].

We use the same workload in [2] to do the test. Since Homa congestion control enhancements only have a trivial effect when message size is small, we focus on workload W4, W5 and other fixed-size long messages.

Test Tool

Test application cp_node is a program to test the performance(including throughput, latency, etc) of Homa or TCP. In our test, we mainly tweak some parameters for clients to adjust the behavior of the client node.

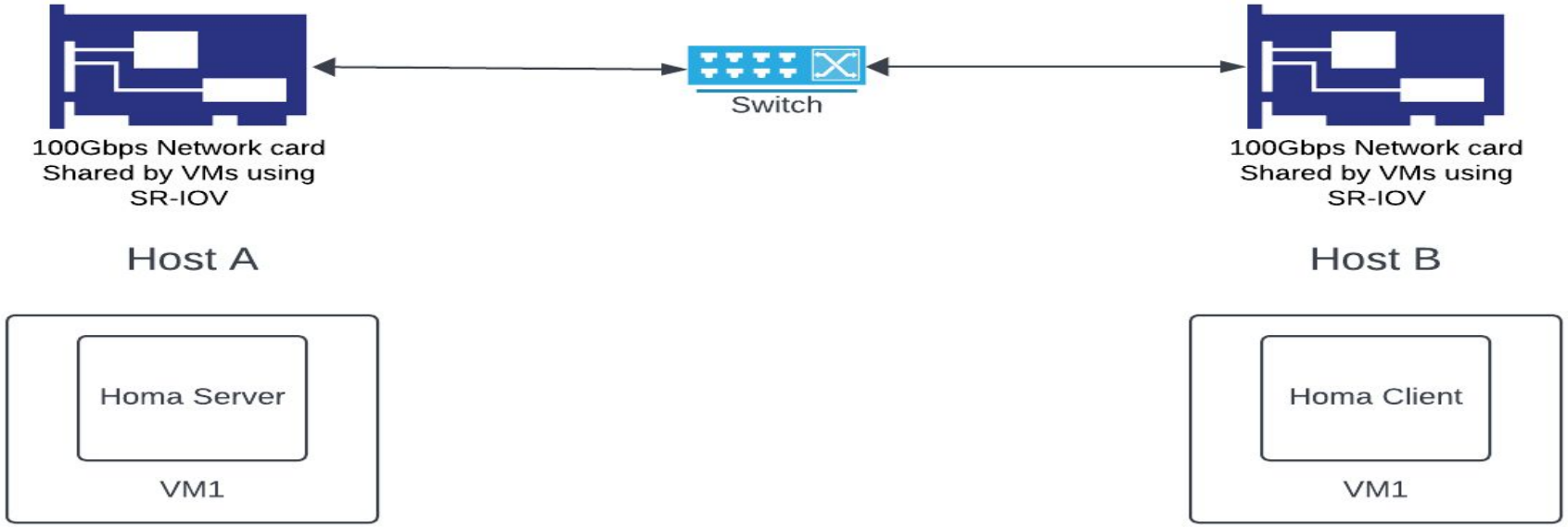
- workload, workload to run the test, could be fixed-size or workload type.

- client-max, maximum number of outstanding requests from a single client machine (divided equally among client ports).

- ports, for clients, the number of ports on which to send requests.

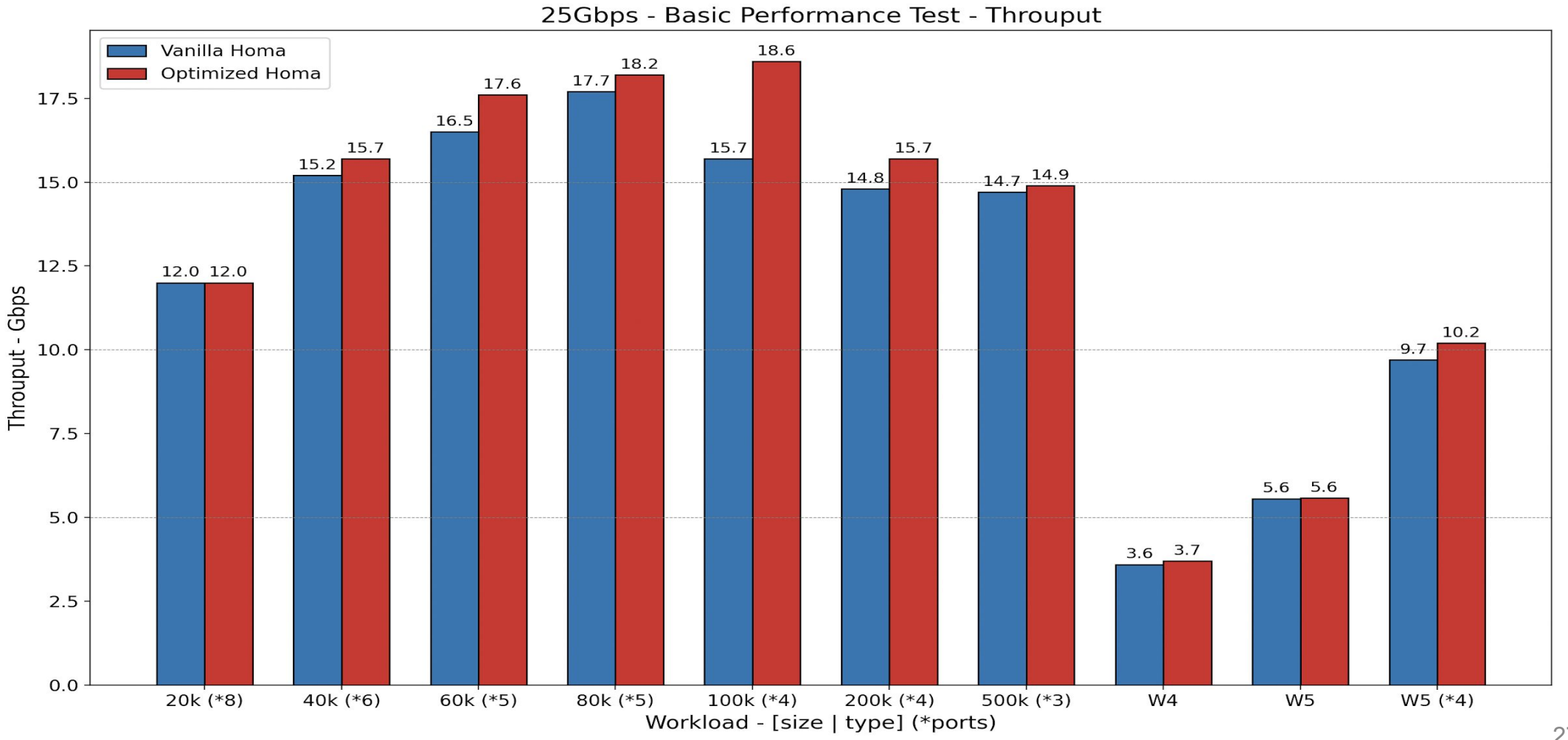
[Image]

Basic performance test setup



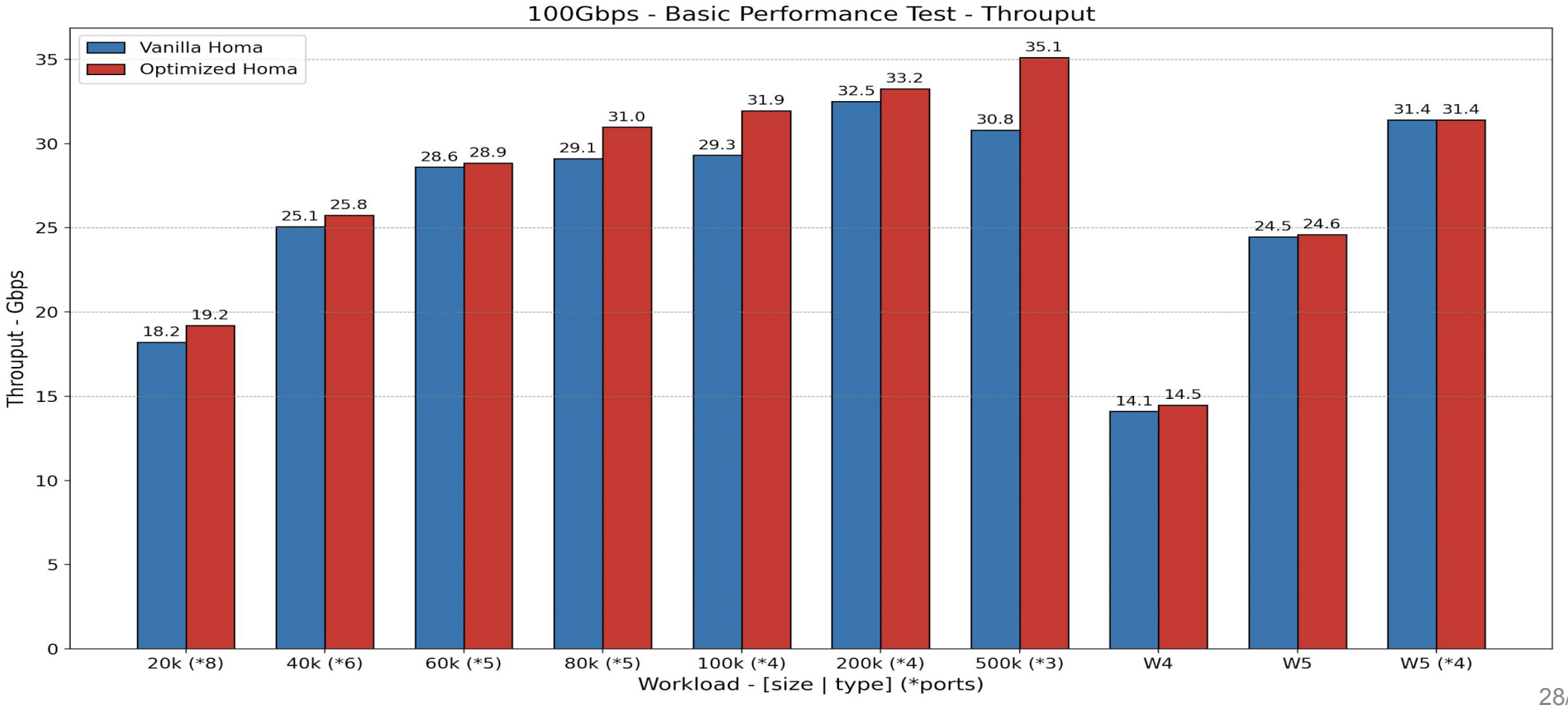
[Image]

Basic Performance Evaluation -Throughput -25G



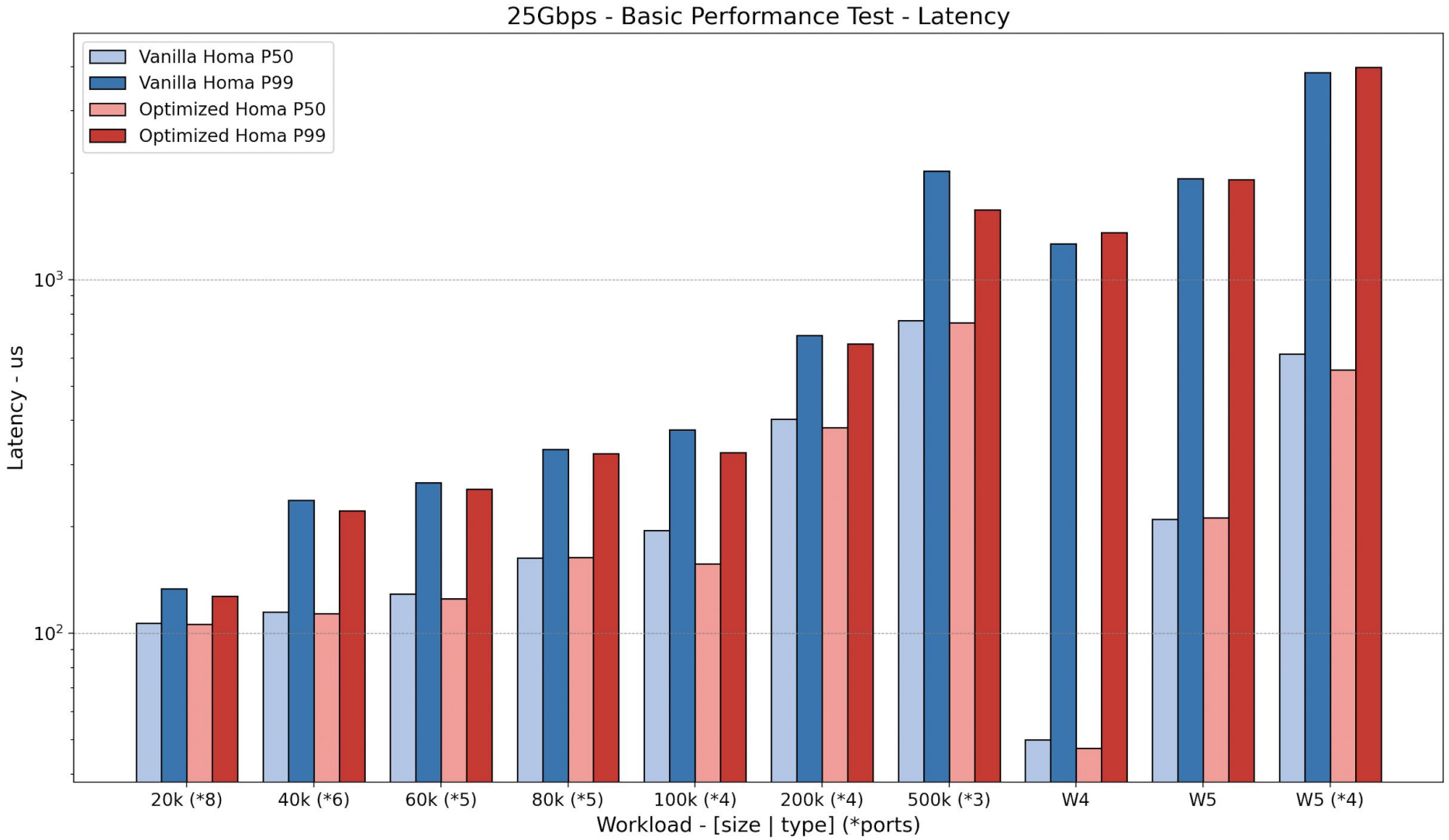
[Image]

Basic Performance Evaluation -Throughput -100G



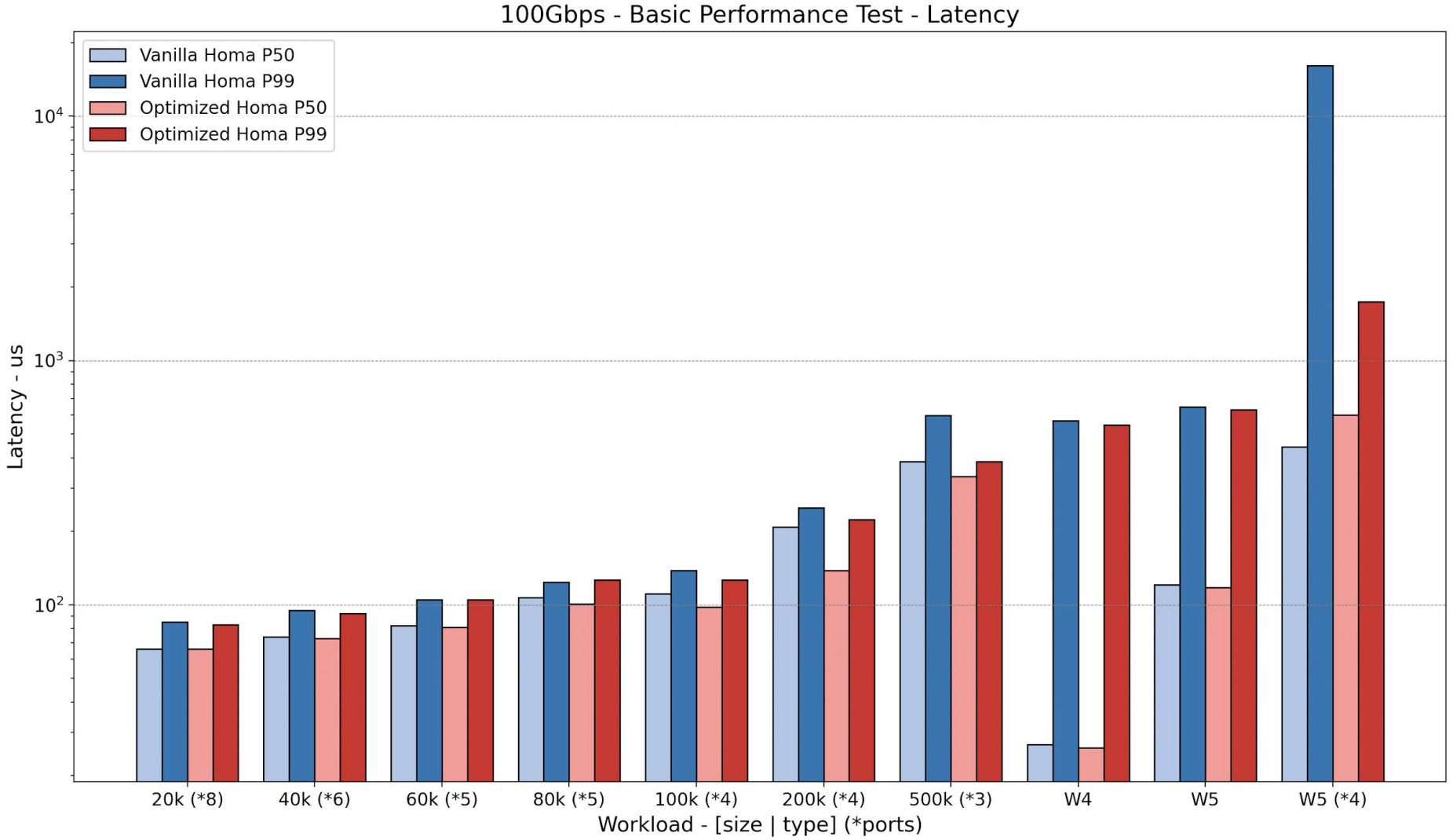
[Image]

Basic Performance Evaluation -Latency -25G

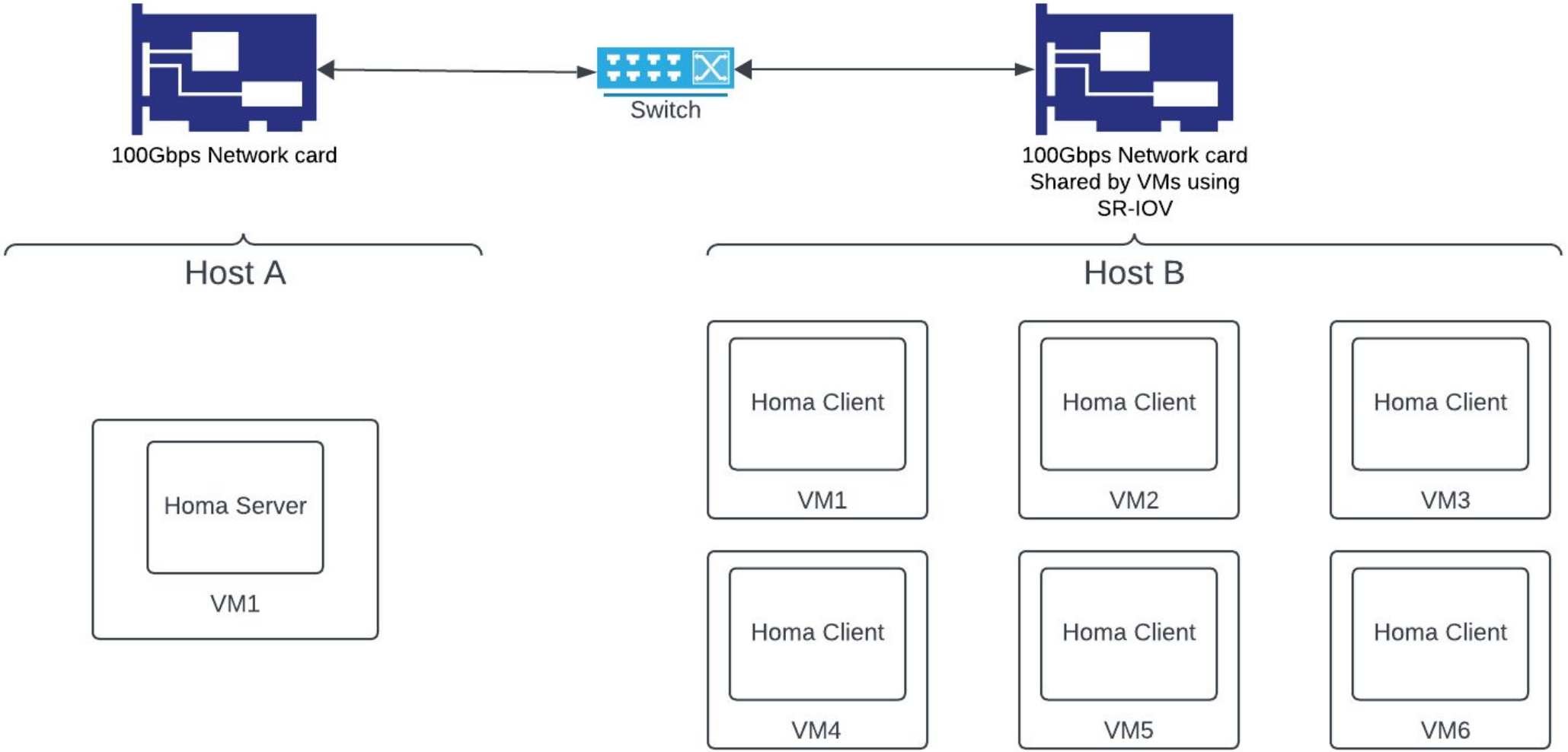


[Image]

Basic Performance Evaluation -Latency -100G

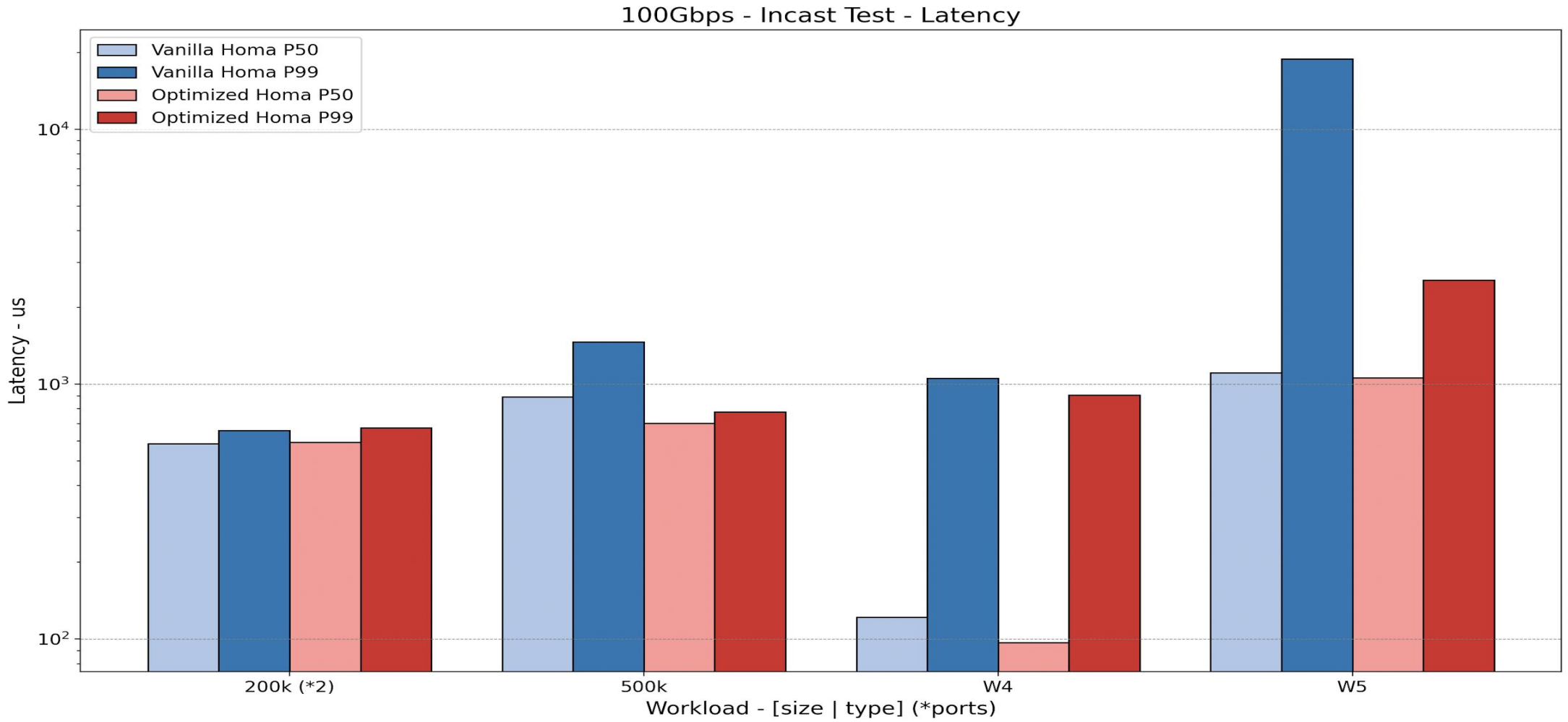


Performance Evaluation - Incast

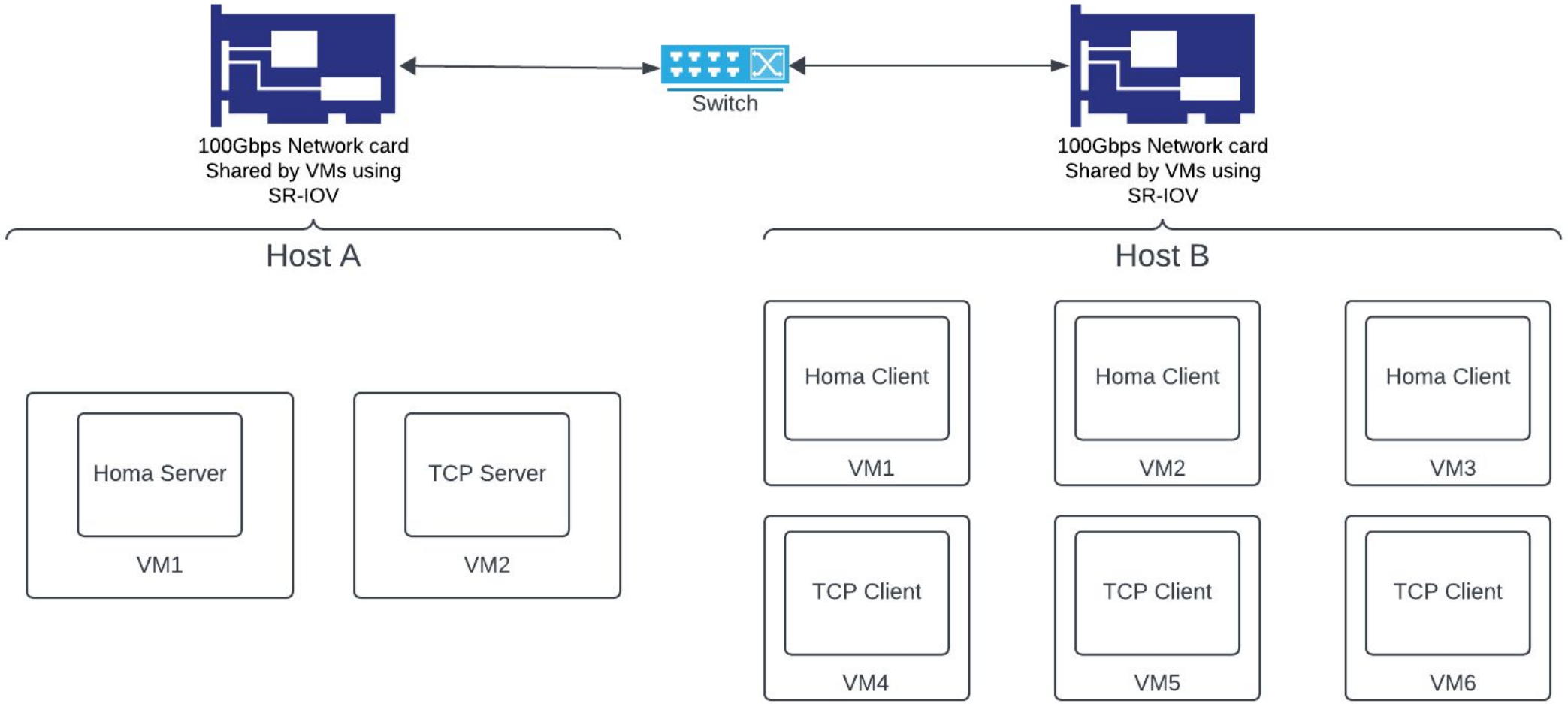


[Image]

Long Message Incast Latency



Performance Evaluation - Split Traffic

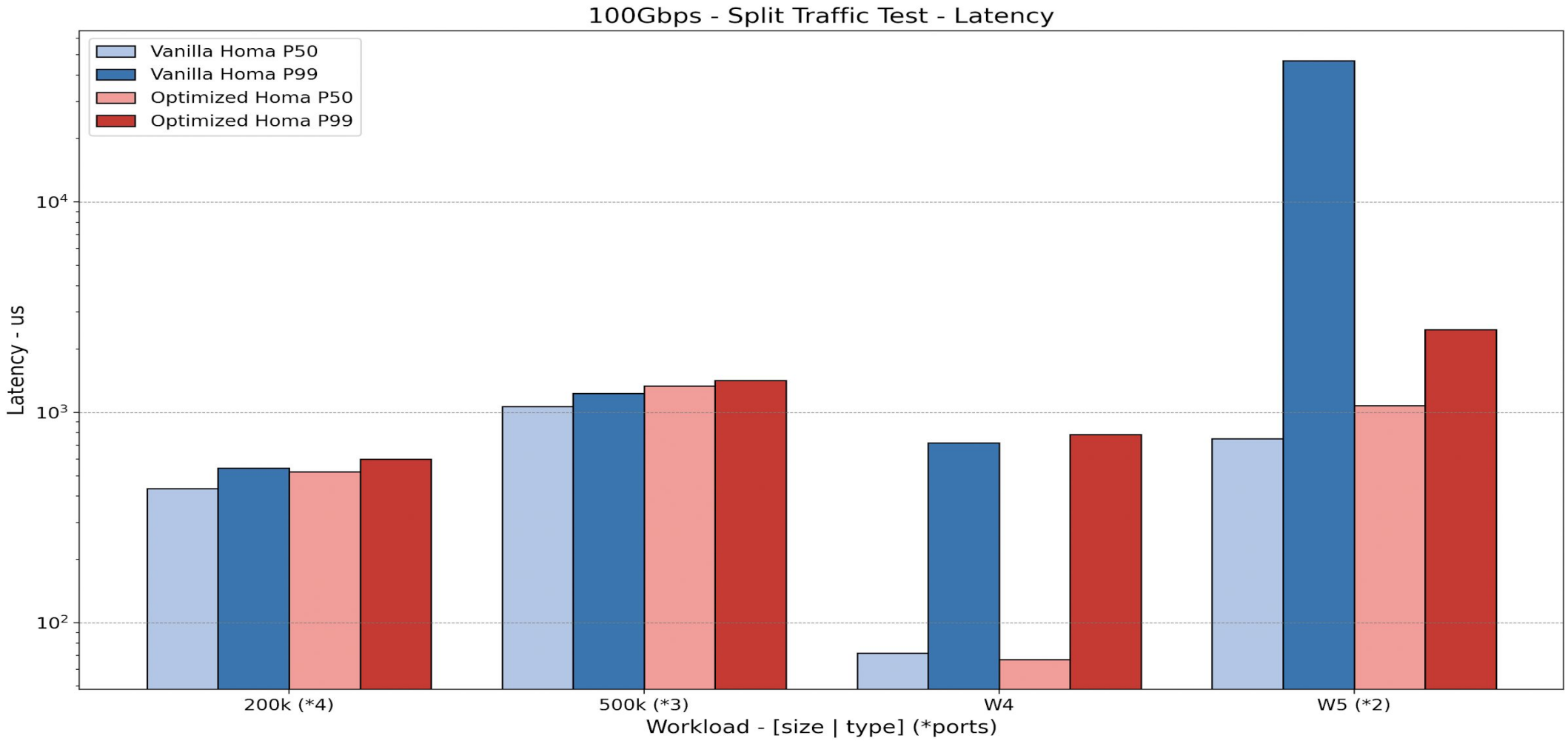


[Image]

Performance Evaluation - Split Traffic -Throughput

[Image]

Performance Evaluation - Split Traffic - Latency



Conclusion

RTT Detection Algorithm Effectiveness

Our developed RTT detection algorithm effectively identifies minimum RTT for network links. Proving its reliabibility in heterogeneous networks.This serves as a basis for defining optimal RTT_bytes .

Dynamic per peer adjustable window

Experimental results demonstrates 5% to 14% throughput improvement for large size message between 40k to 500k compared to self-tuned RTT_bytes configurations.

Buffer Overflow Resilience

The algorithm demonstrates resilience against buffer overflow situations caused by large messages during incast scenarios.

Compatibility with TCP Traffic

Optimized Homa protocol intelligently coexists with TCP traffic, detecting congestion caused by TCP and adjusting its packet transmissions accordingly.The Optimized Homa adeptly circumvent buffer overflow challenge under w5,whereas the vanilla homa manifests huge p99 latency due to this anomaly.

Future Improvements

- **More accurate RTT measurement**

  Distinguishing between fabric (network) delay and host software delay in RTT.

- **Congestion detection base on average RTT deviation**

  Deviation of average RTT (rtt_avg) from RTT_mid (three times RTT_min) for Congestion detection

- **Optimize pacer**

  Pacer needs to be aware of network congestion based on recent RTT. When RTT is low , although GRANT is not received, try to send some packets at a relatively low rate. When RTT is high, transmit less packets to NIC.

- **Zero-copy**

References

[1]https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p537.pdf

[2] https://www.usenix.org/system/files/atc21-ousterhout.pdf

[3]https://people.csail.mit.edu/alizadeh/papers/homa-sigcomm18.pdf

Q & A