Leverage Homa: Enhancing Homa Linux for Efficient RPC Transportation

Presenter: Xiaochun Lu, Zijian Zhang

System Technologies and Engineering Team

ByteDance 字节跳动

Agenda

- Homa Introduction
- Limitation of Homa in RPC context
- Homa Congestion Control Enhancements
- Homa RPC Streaming Enhancements
- Future Improvements
- Conclusion
- Q&A

Homa Introduction

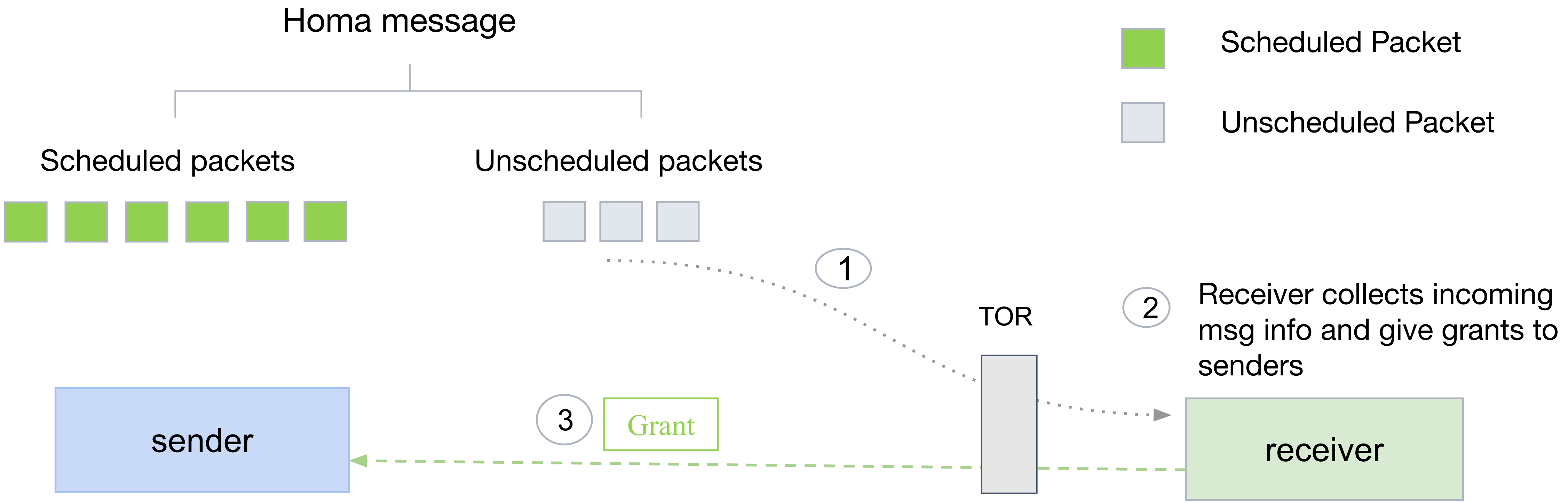
Why TCP is Wrong for Data Center?

- Designed for wide-area networks
- Connections
- Stream orientation
- Fair scheduling
- Sender-driven congestion control
- It doesn’t take advantage of in-network priority queues.
- In-order delivery, restricting opportunities for load balancing

Homa Protocol Introduction

- Designed for datacenter networks with extremely low latencies
- Connectionless, no connection cost, no long life connection state
- Message based protocol
- SRPT(Shortest Remaining Process Time first)
- Use in-network priorities
- Receiver-driven packet scheduling
- Overcommitment

Overview



Homa message

Scheduled packets

Unscheduled packets

🟩 Scheduled Packet

⬜ Unscheduled Packet

1

TOR

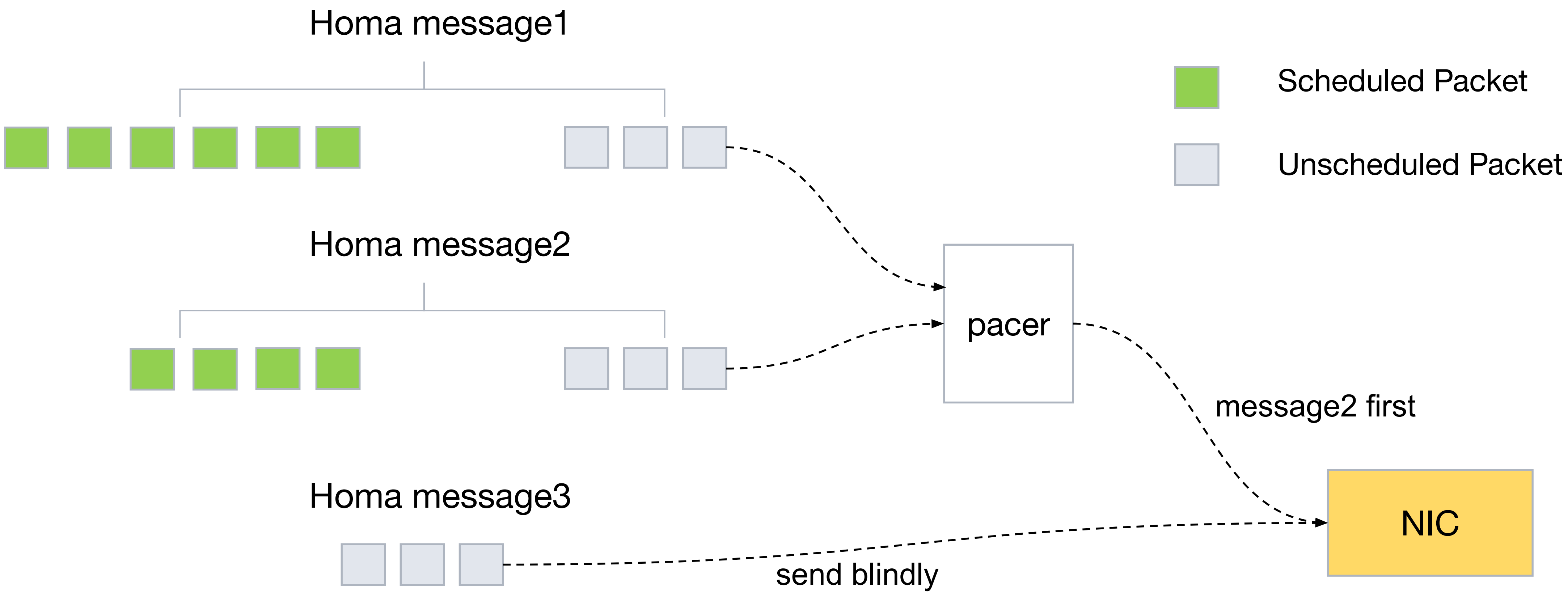
2 Receiver collects incoming msg info and give grants to senders

3 Grant

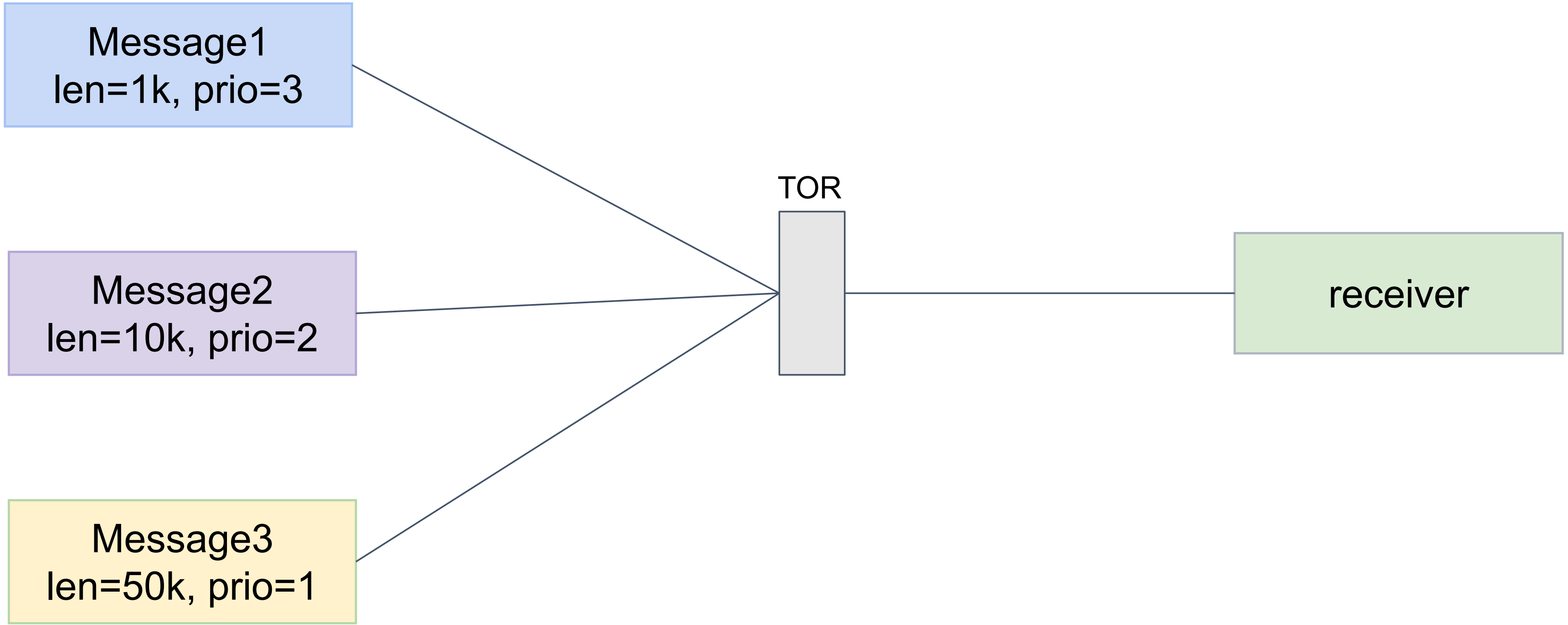
sender

receiver

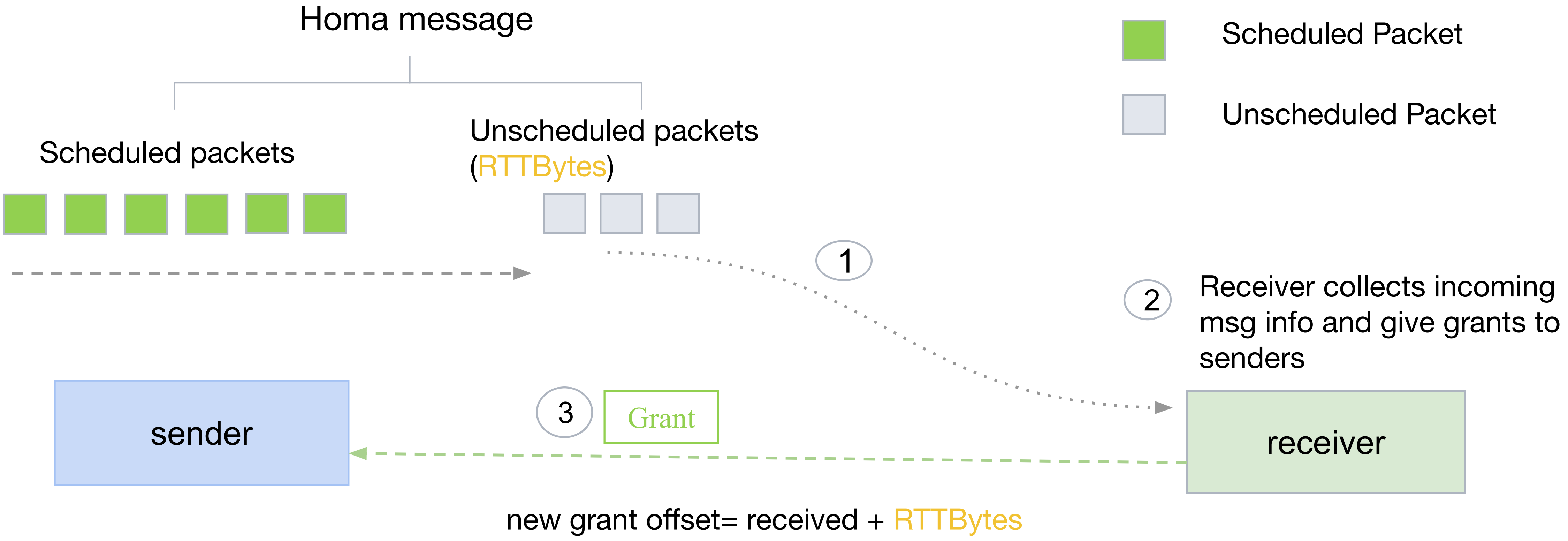
Sender – SRPT and Pacer



In-network priorities



Server – Packet Scheduling



Homa message

Scheduled packets

Unscheduled packets (RTTBytes)

🟩 Scheduled Packet

⬜ Unscheduled Packet

1

2 Receiver collects incoming msg info and give grants to senders

3 Grant

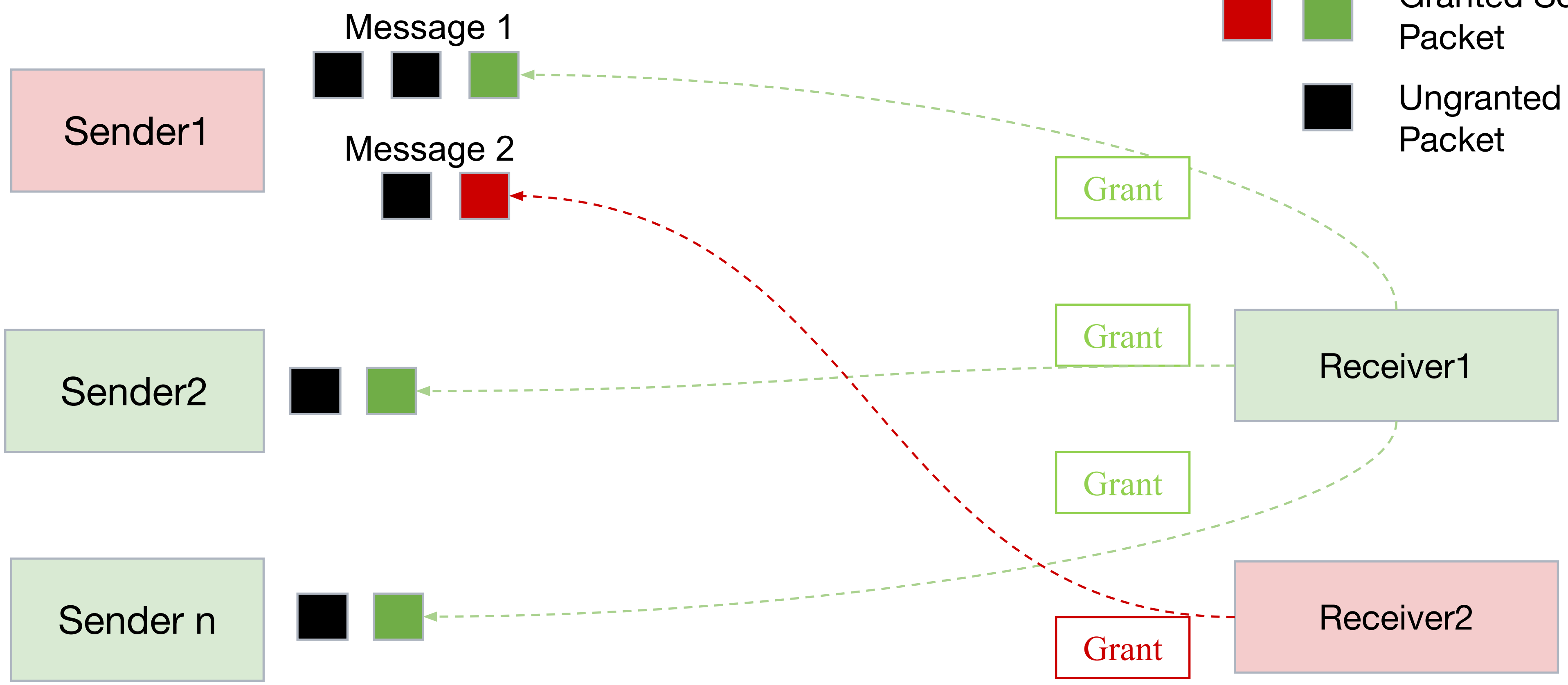
sender

receiver

new grant offset= received + RTTBytes

RTTBytes is a pre-set fixed value

Server – Controlled Overcommitment



🟥 🟩 Granted Scheduled Packet

⬛ Ungranted Scheduled Packet

Potential buffer build up due to overcommitment

Limitations of Homa as RPC transport protocol

Limitations of Homa as RPC transport protocol

Inefficient Pipelining for Large Message

- Homa is message-based protocol, while ensuring complete message delivery, it hinders efficient pipelining. As a result, for large RPC messages(size > 50k), Homa is not as good as TCP.

Non-standard Socket API interface

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

Limitations of Homa as RPC transport protocol

Performance is sensitive to RTTBytes Config

- Manually config RTTBytes can be inconvenient
- Single preset value is not enough for diverse RTT and receiver downlink bandwidth.

Weak Congestion Control when RTT is larger than 20 us

- High RTTBytes can more easily lead to incast congestion
- Low RTTBytes is not able to cover RTT

Limitations of Homa as RPC transport protocol

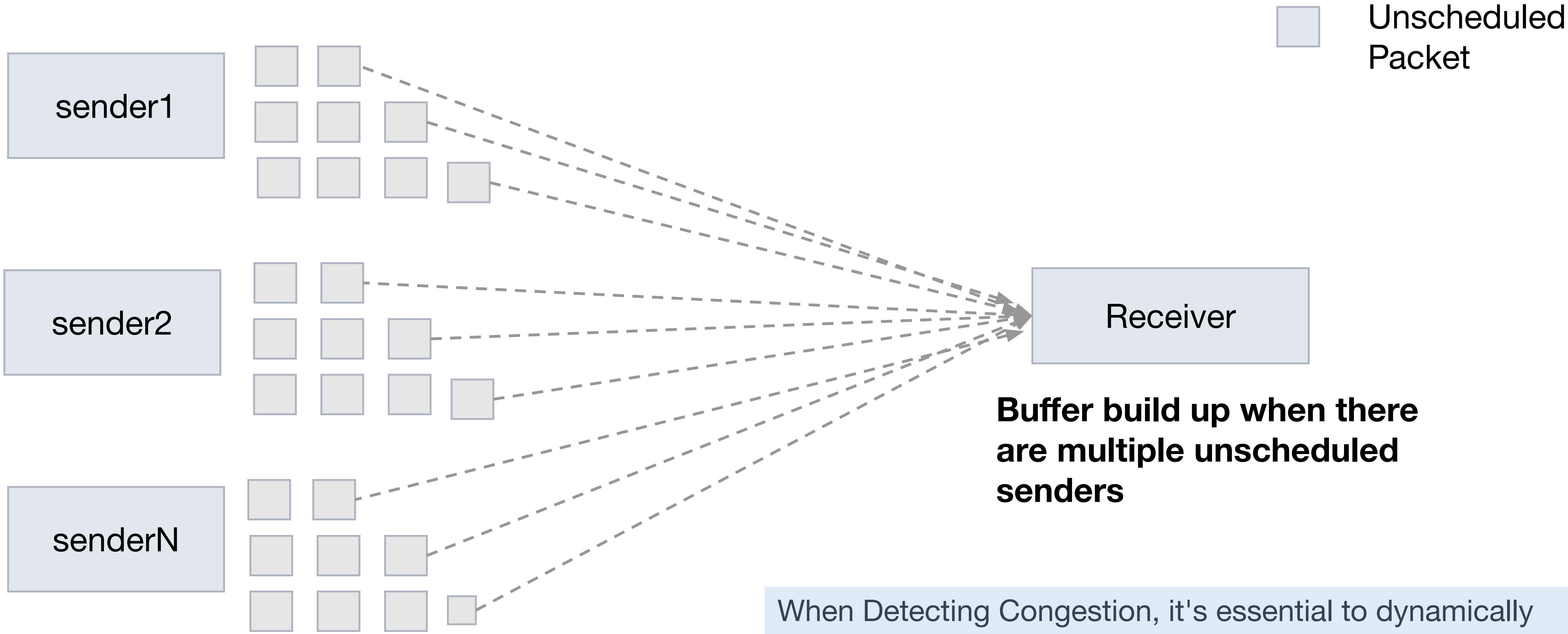
Homa cannot coexist harmoniously with TCP

- In practice, network resources need to be shared among protocols like TCP
- Homa assumes the bandwidth is all used by itself. If the bandwidth is shared by TCP, Homa maybe over-generous on unscheduled bytes, and overgrant scheduled bytes.

Homa inactively handle incast

- Homa assumes that the most severe forms of incast are predictable because they are self-inflicted by outgoing RPCs.
- Homa assumes the incast where several machines simultaneously send requests to one server is rare.

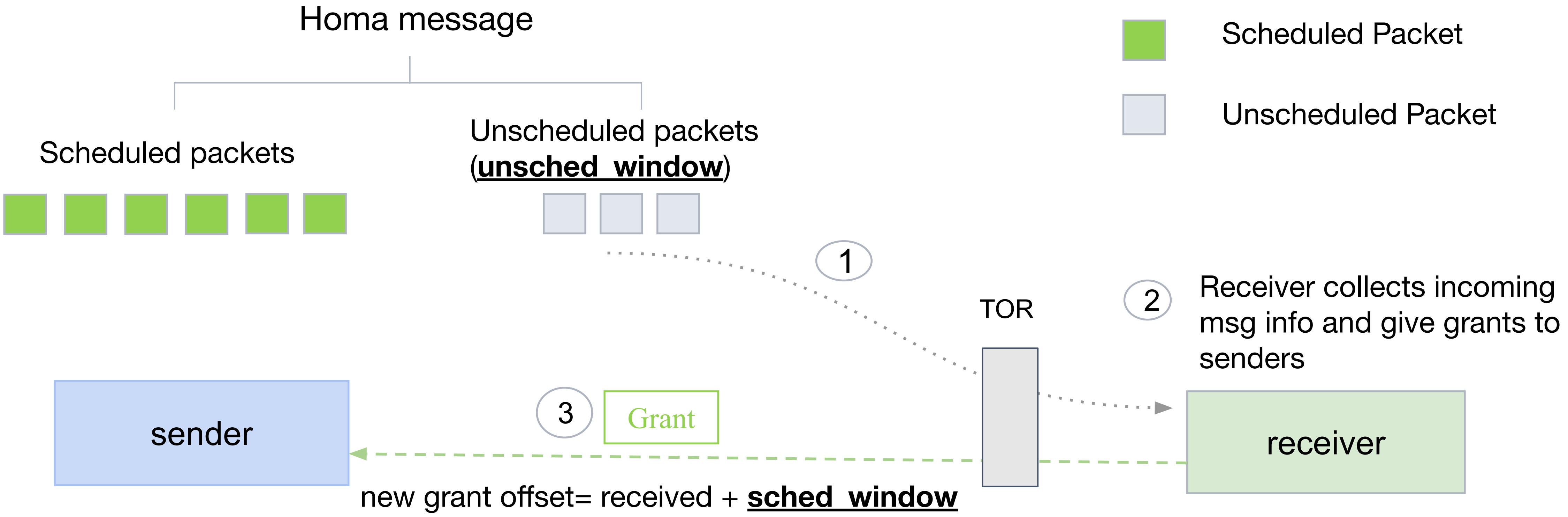
Unscheduled Packet Incasting



Buffer build up when there are multiple unscheduled senders

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

Static Congestion Window is Insufficient



Homa message

Scheduled packets

Unscheduled packets (**unsched_window**)

🟩 Scheduled Packet

⬜ Unscheduled Packet

1

TOR

2 Receiver collects incoming msg info and give grants to senders

3 Grant

sender

receiver

new grant offset= received + **sched_window**

Unscheduled Window and Scheduled Window need different values!

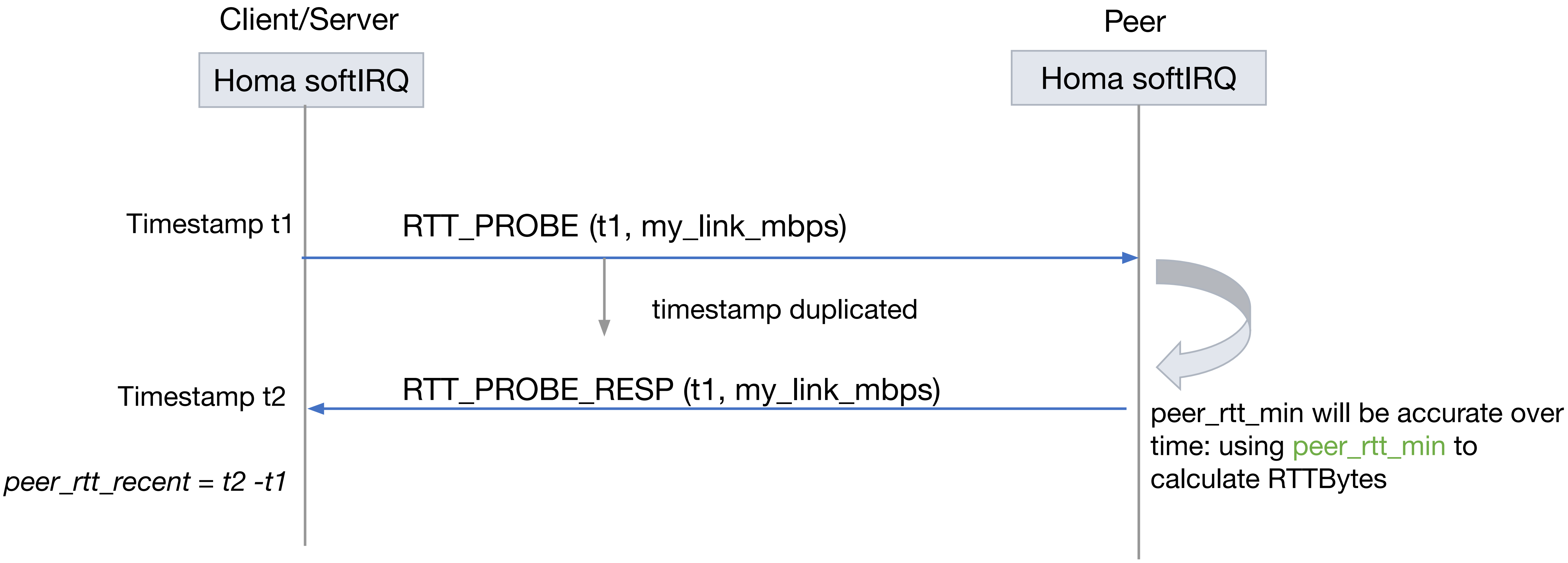
Homa Congestion Control Enhancements

Homa Congestion Control Enhancements

Dynamic Per Peer Adjustable Window

- Real-time peer RTT detection
- RTT-informed congestion detection
- Adaptive per-peer adjustable unscheduled/scheduled window based on congestion

Real-time Peer RTT Detection



Client/Server

Homa softIRQ

Peer

Homa softIRQ

Timestamp t1

RTT_PROBE (t1, my_link_mbps)

timestamp duplicated

Timestamp t2

RTT_PROBE_RESP (t1, my_link_mbps)

peer_rtt_recent = t2 -t1

peer_rtt_min will be accurate over time: using peer_rtt_min to calculate RTTBytes

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_low: The low threshold of RTT

peer_rtt_mid: The middle point of RTT

peer_rtt_high: The high threshold of RTT

```
peer_rtt_low = peer_rtt_min * 2
peer_rtt_mid = peer_rtt_low * 2
peer_rtt_high = peer_rtt_mid * 2
IF  peer_rtt_recent >  peer_rtt_high
    Set congestion To TRUE
ENDIF
```

Sender - Dynamic adjusting unscheduled window

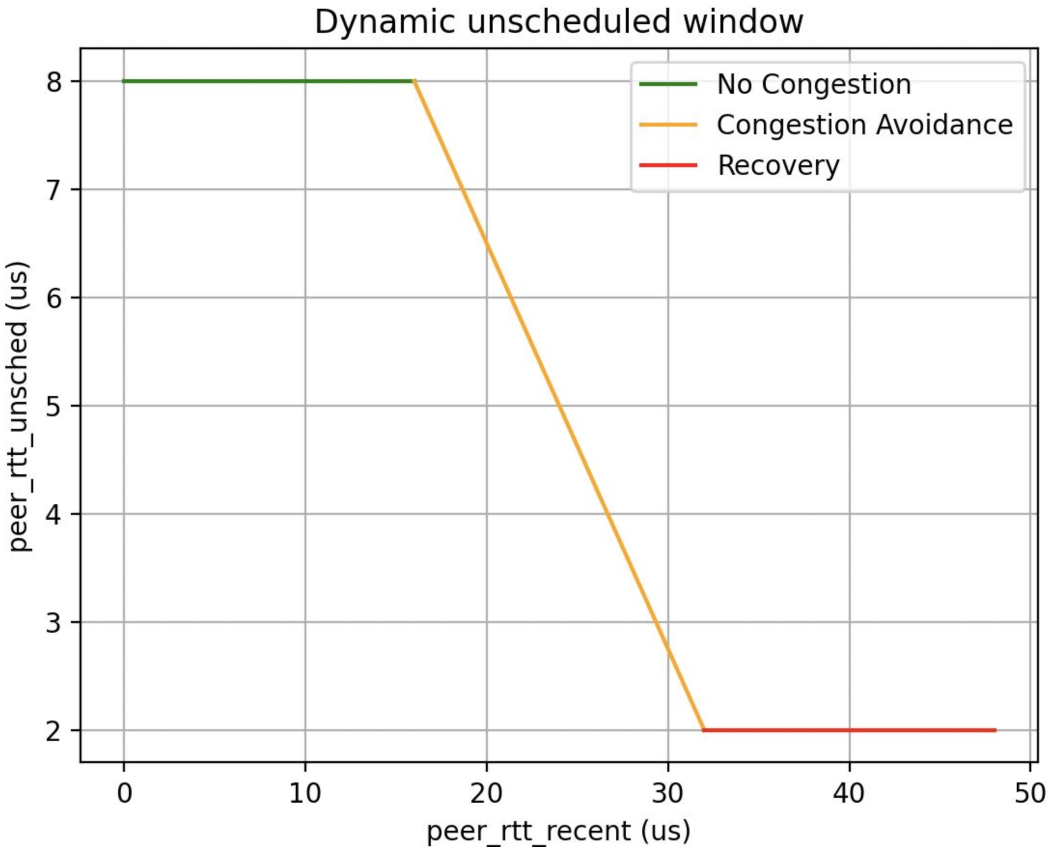
*unsched_window = peer_rtt_recent * peer_link_mbps / 8*

Sender - Dynamic adjusting unscheduled window

~~*unsched_window = peer_rtt_recent * peer_link_mbps / 8*~~

```
IF  peer_rtt_recent < peer_rtt_mid
        peer_rtt_unsched = peer_rtt_low
ELIF peer_rtt_recent < peer_rtt_high
        peer_rtt_unsched = peer_rtt_low -
                (peer_rtt_recent - peer_rtt_mid) * 3 / 8
ELSE
        peer_rtt_unsched = rtt_min / 2
ENDIF
```

unsched_window = peer_rtt_unsched * peer_link_mbps / 8



Sender - Unscheduled Ratio

```
Unscheduled_ratio = Total unscheduled packets / Total length of all messages

IF `unscheduled_ratio <= 40%` AND `pacer throttle list is not empty`
    SET unsched_window To unsched_window / 2
ENDIF
```

Receiver - Scheduled Window

Use *peer_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_mbps / 8;

max_incomming = scheduled_window * max_overcommit;
```

Homa Congestion Control Enhancements

Performance is nearly independent to RTTBytes Config

- Instead of fixed and static RTT, we now use real-time RTT.

Homa harmoniously coexist with TCP

- Homa can be aware of the existence of other protocols' traffic by feeling the turbulence of RTT.

Homa actively handle incast

- Incast can be reflected by high RTT, then senders can adjust the unscheduled window dynamically.

Performance Evaluation

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

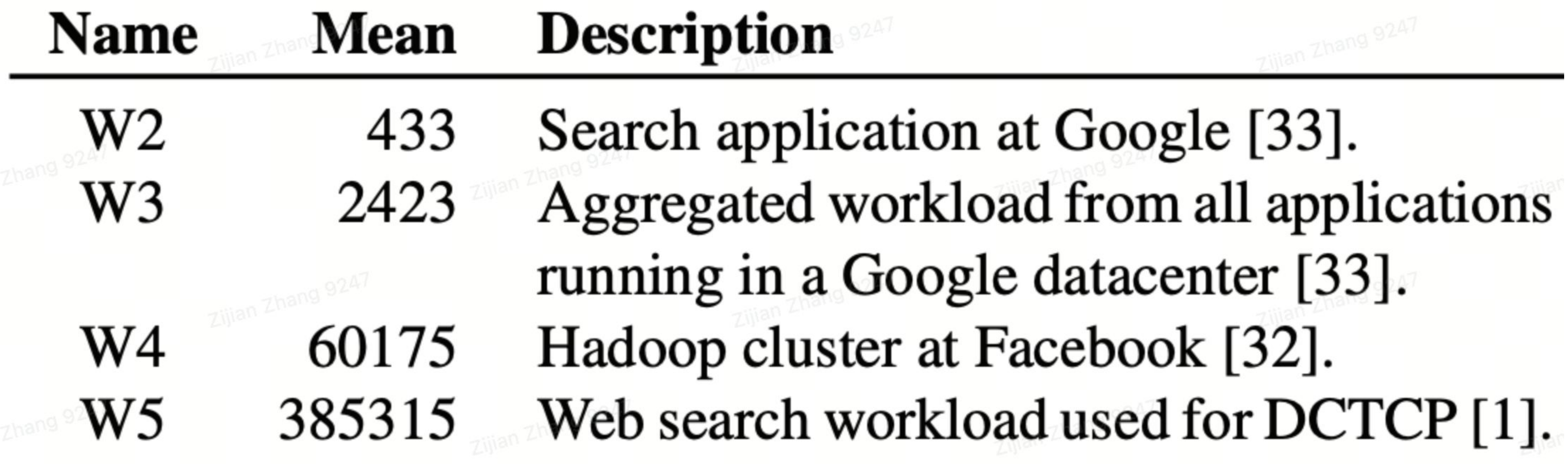
CPU: Intel(R) Xeon(R) Silver 4314 (64 cores, 2.4 GHz)

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 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.

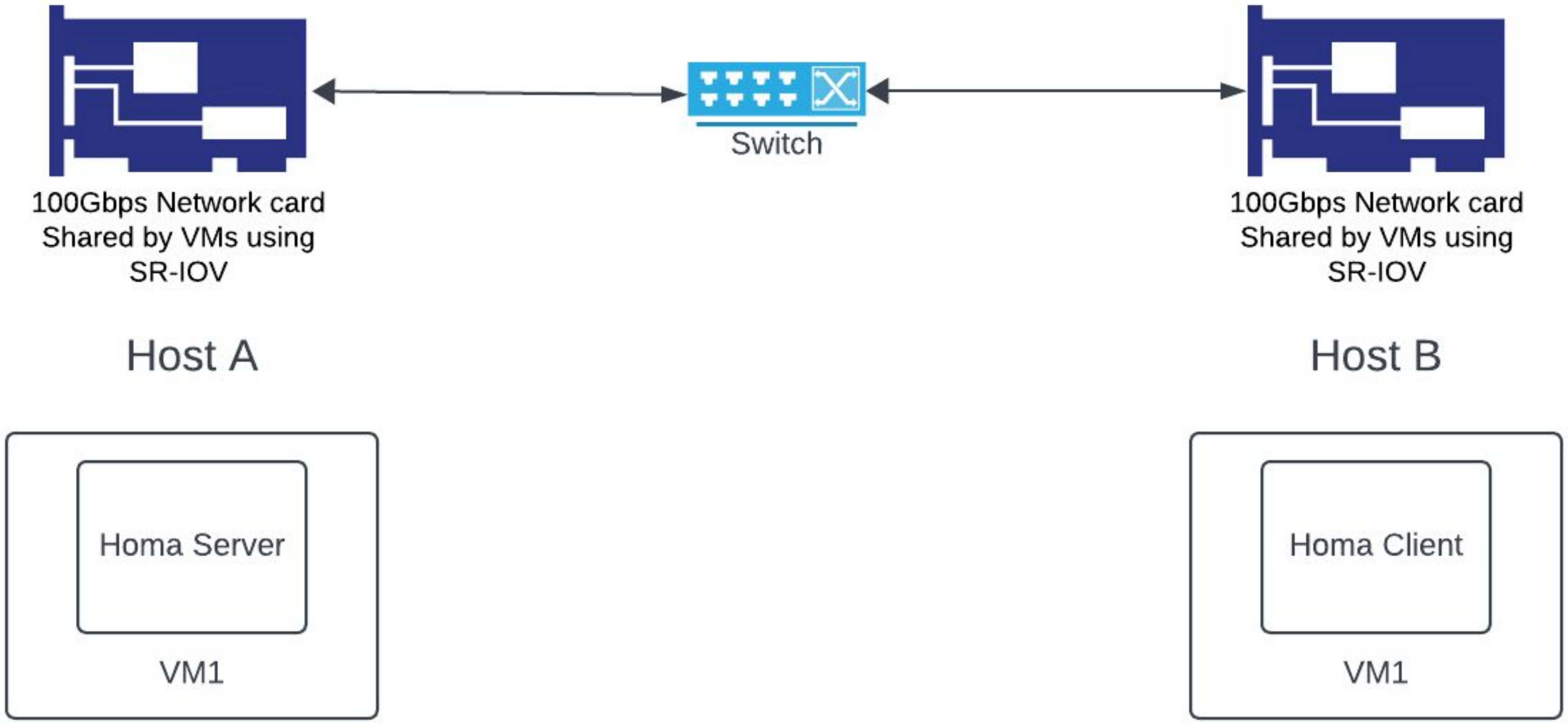
Test Tool

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.

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

Basic Performance Setup



Switch

100Gbps Network card
Shared by VMs using
SR-IOV

Host A

Homa Server

VM1

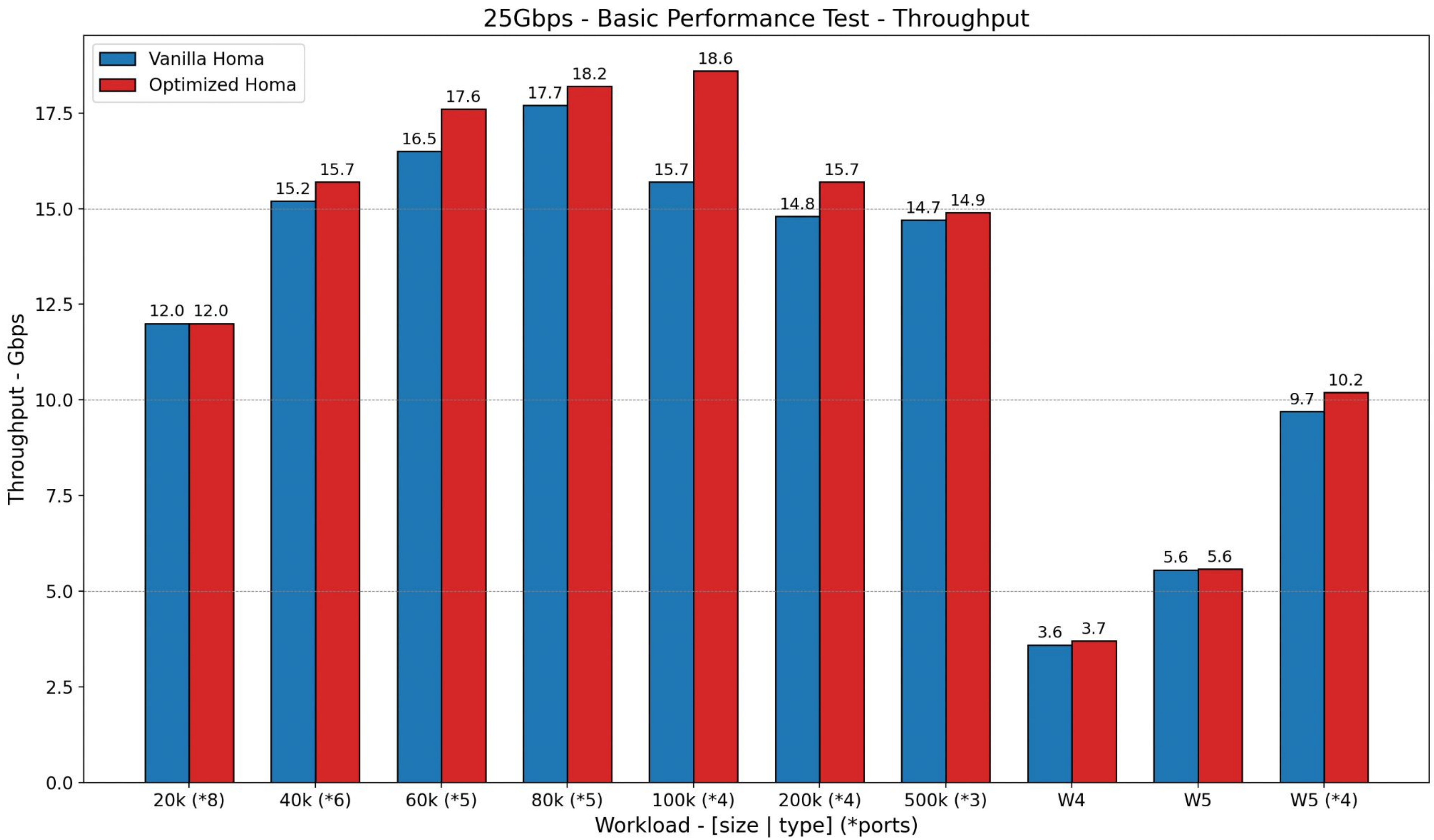
100Gbps Network card
Shared by VMs using
SR-IOV

Host B

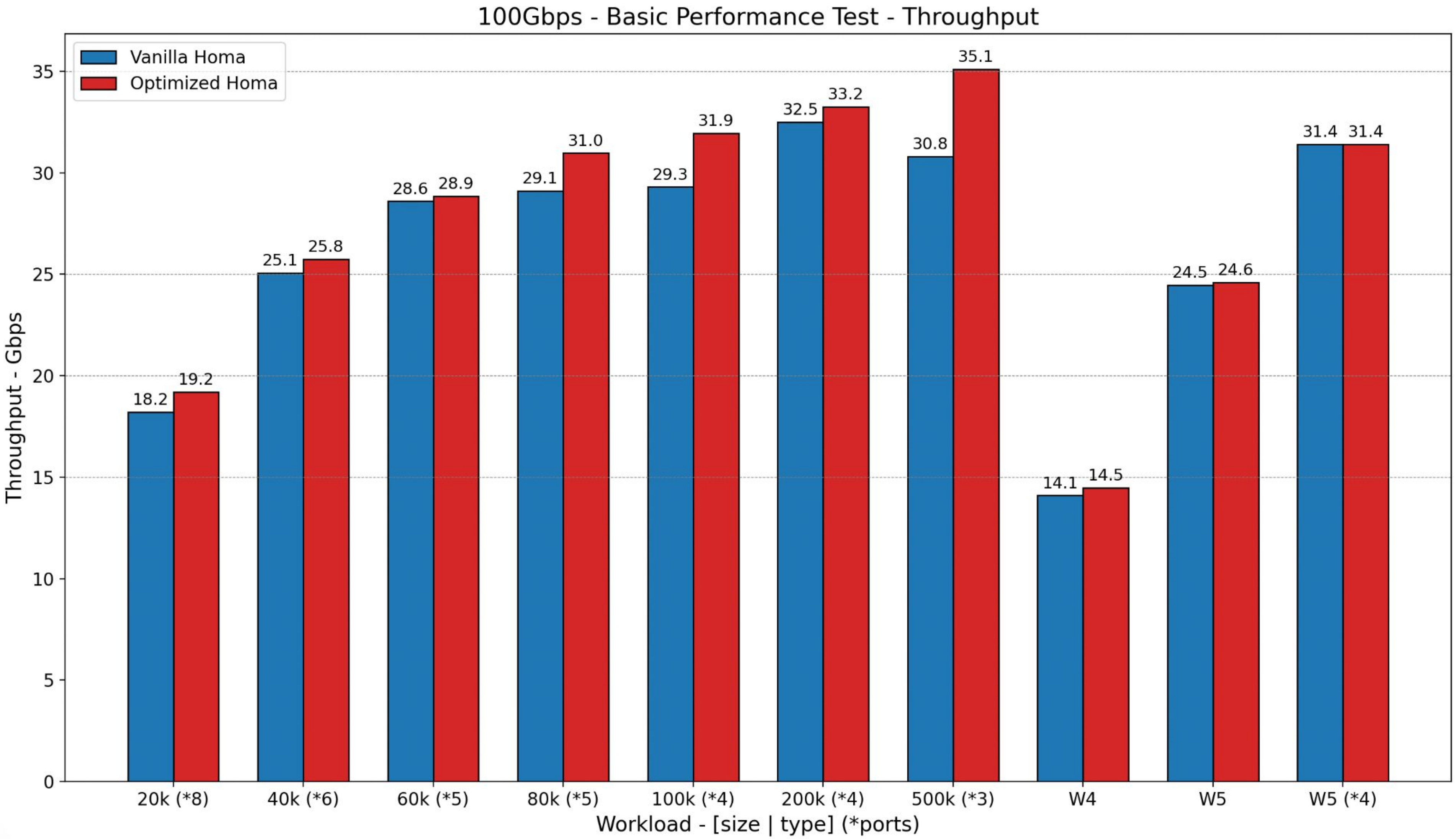
Homa Client

VM1

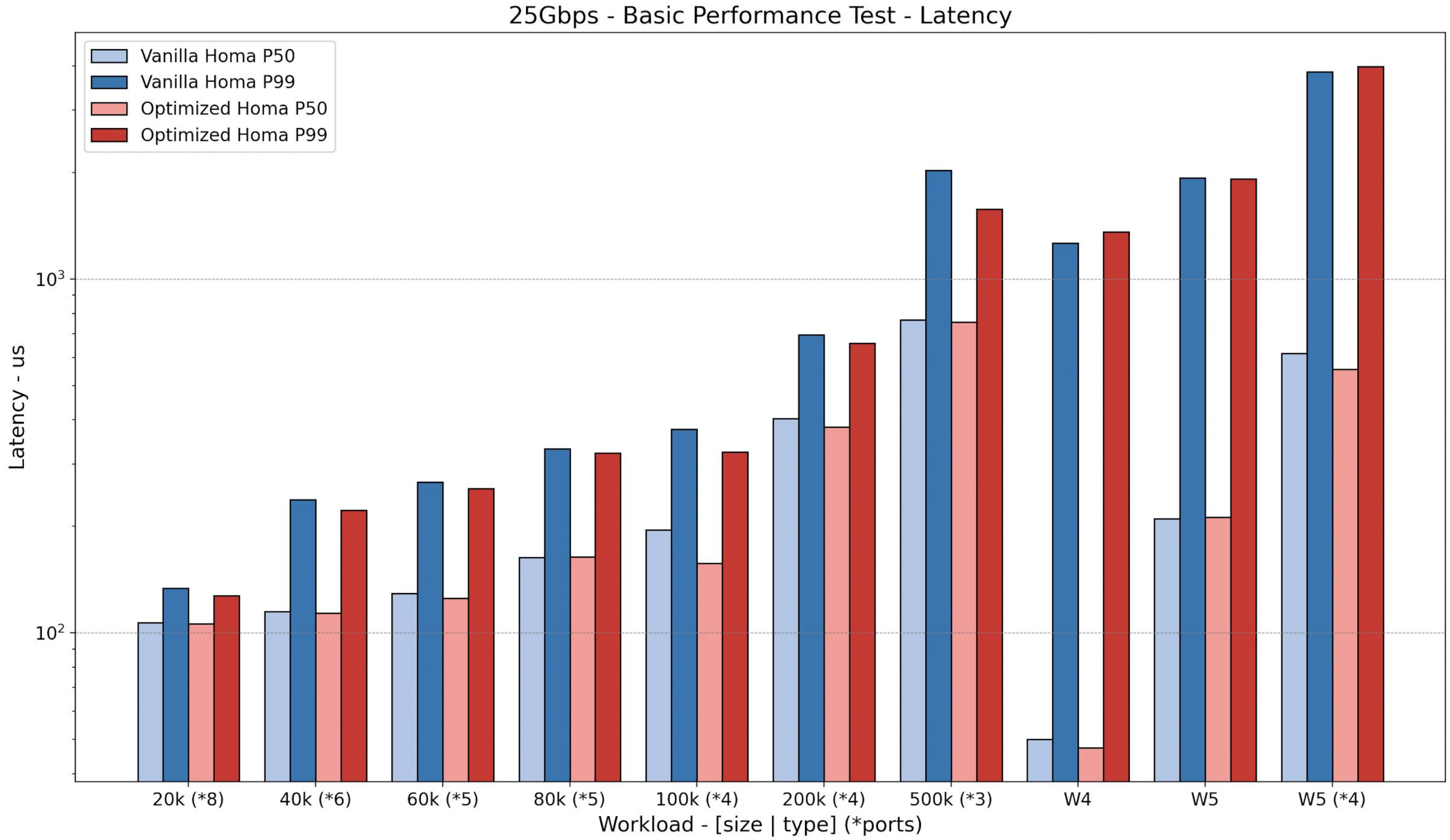
Basic Performance - 25Gbps - Throughput



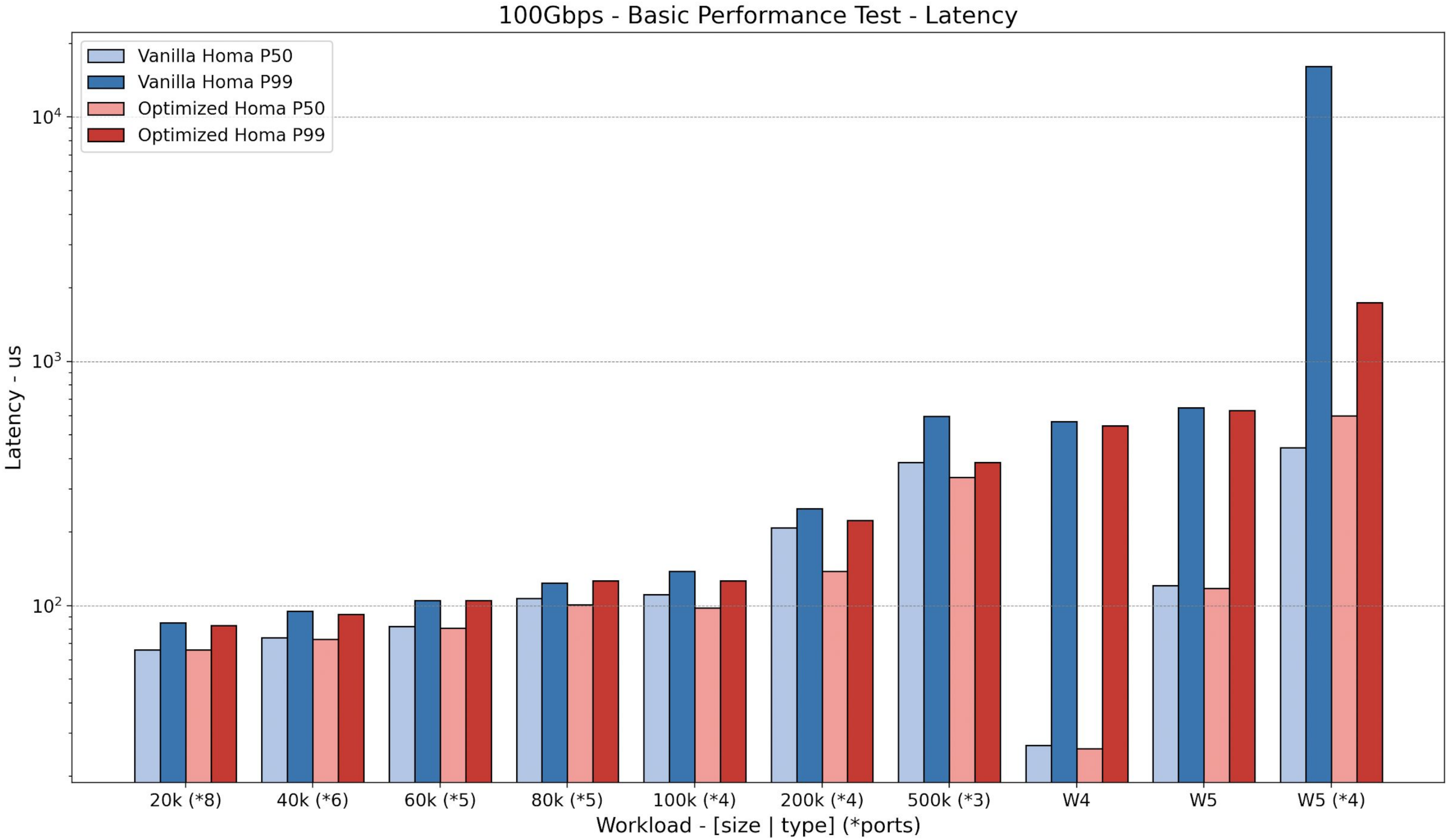
Basic Performance - 100Gbps - Throughput



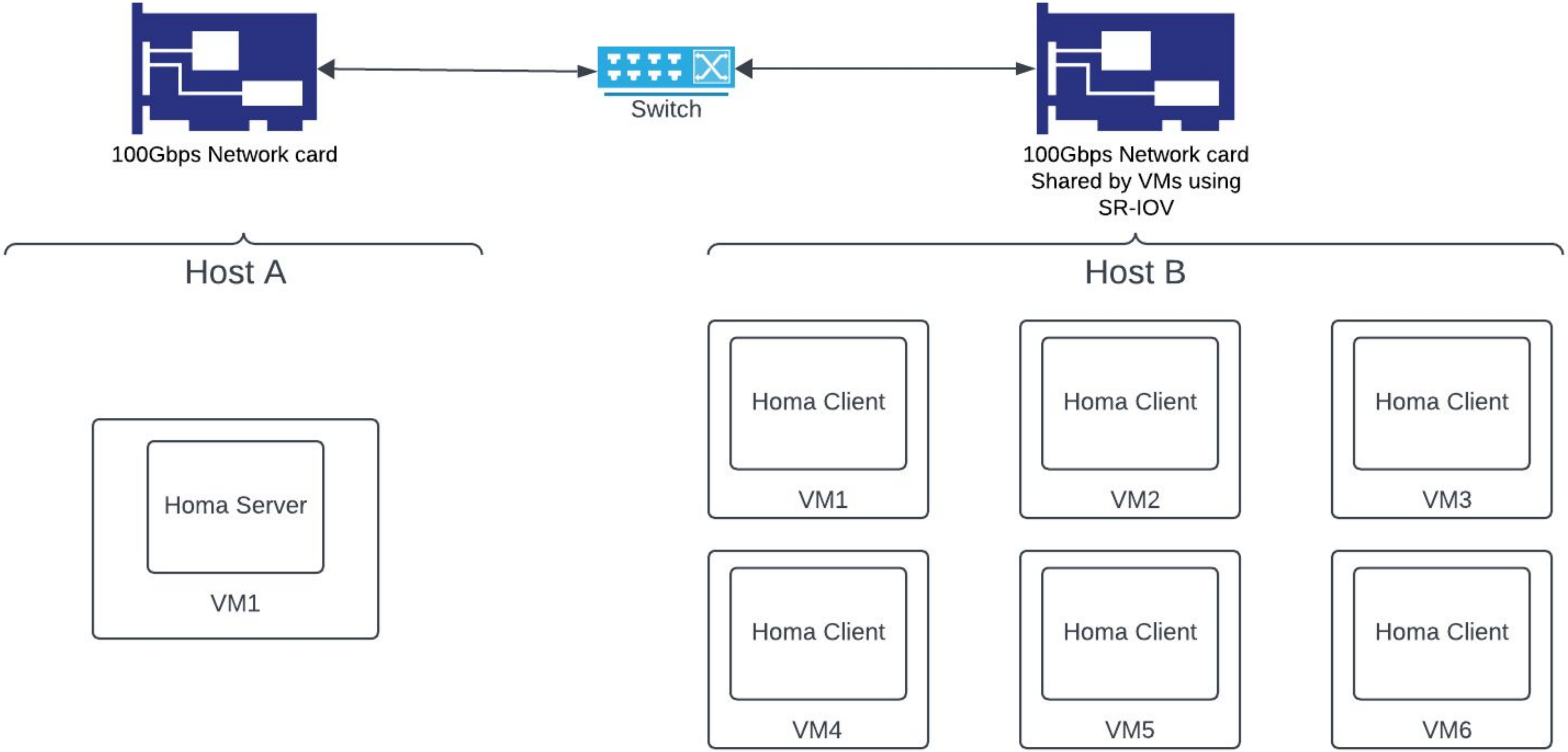
Basic Performance - 25Gbps - Latency



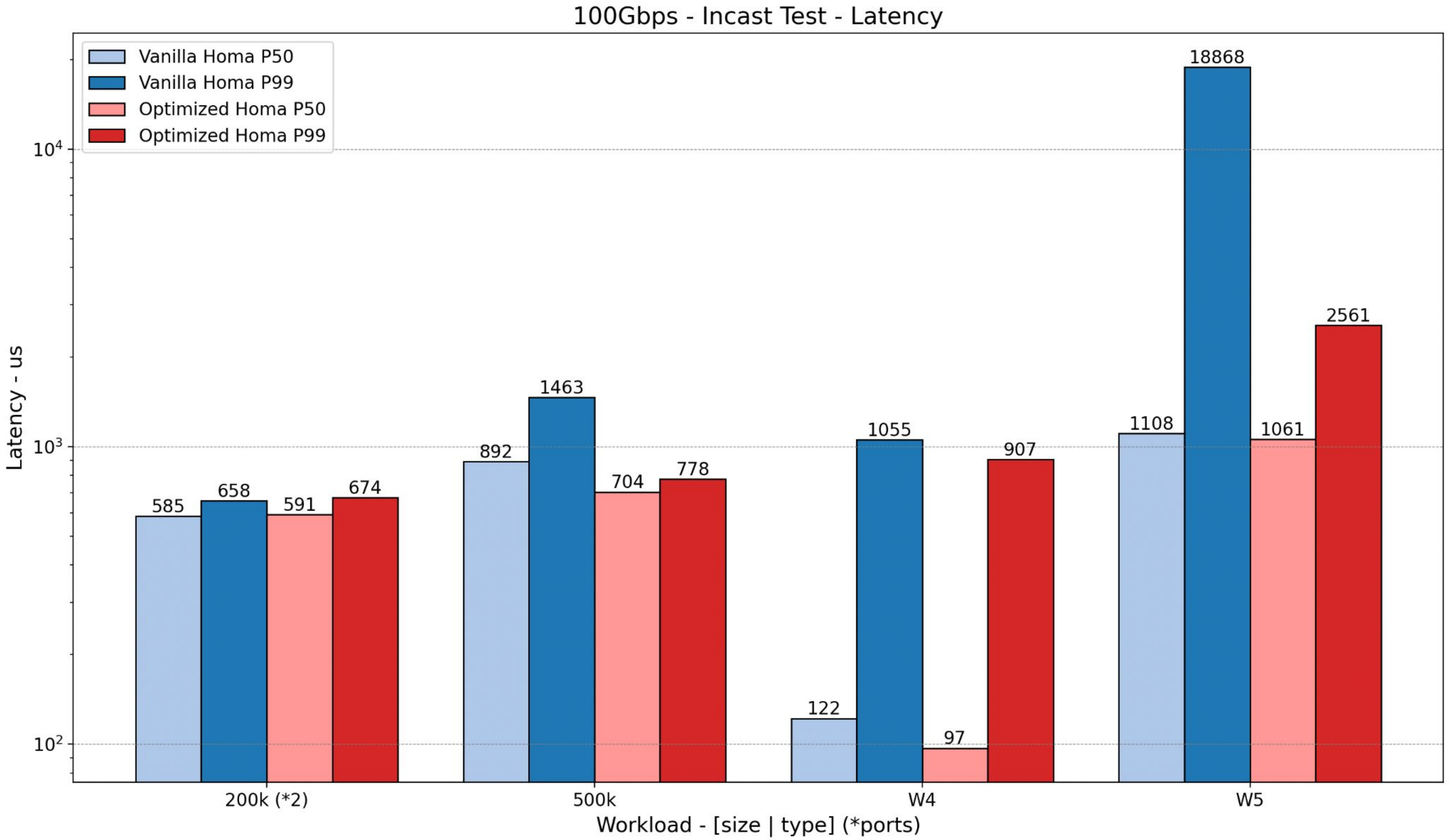
Basic Performance - 100Gbps - Latency



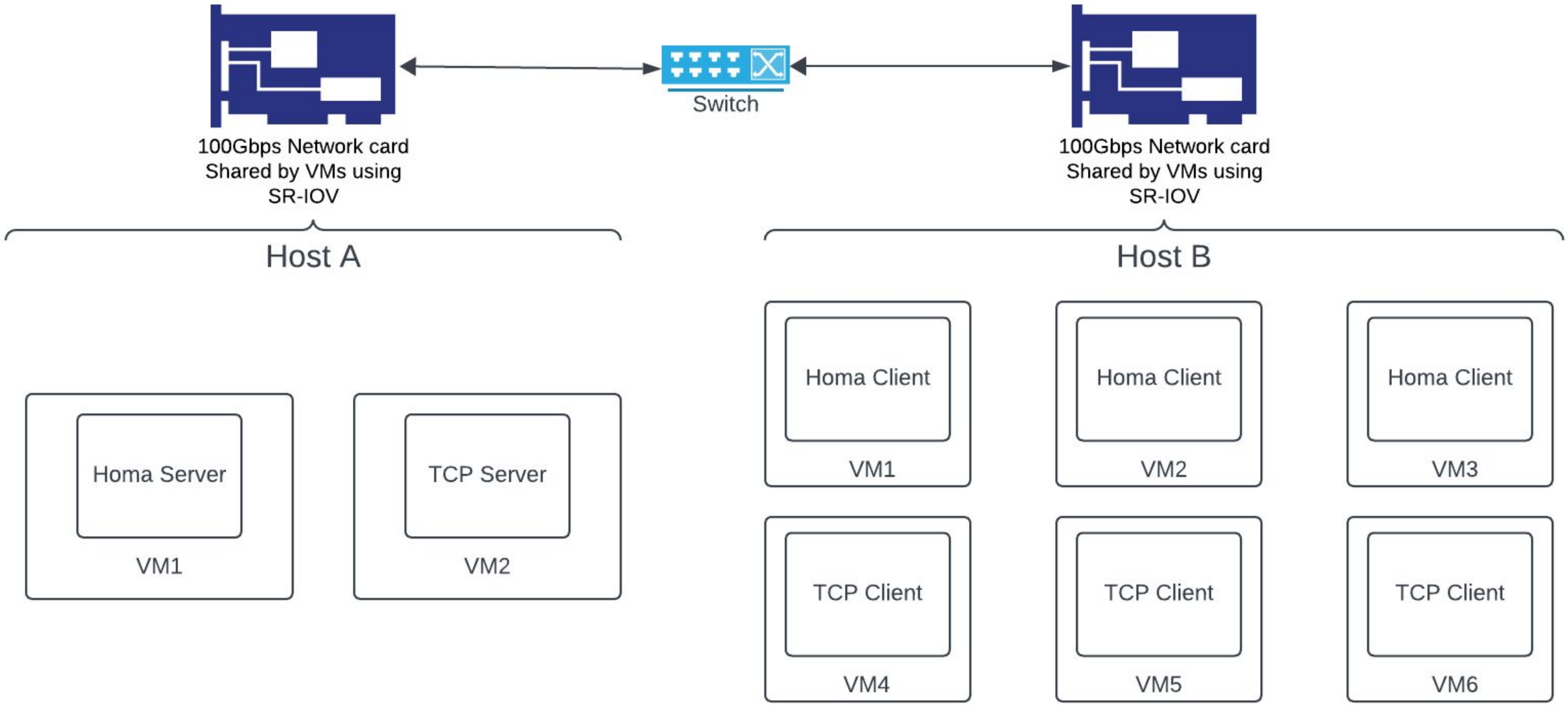
Incast - Setup



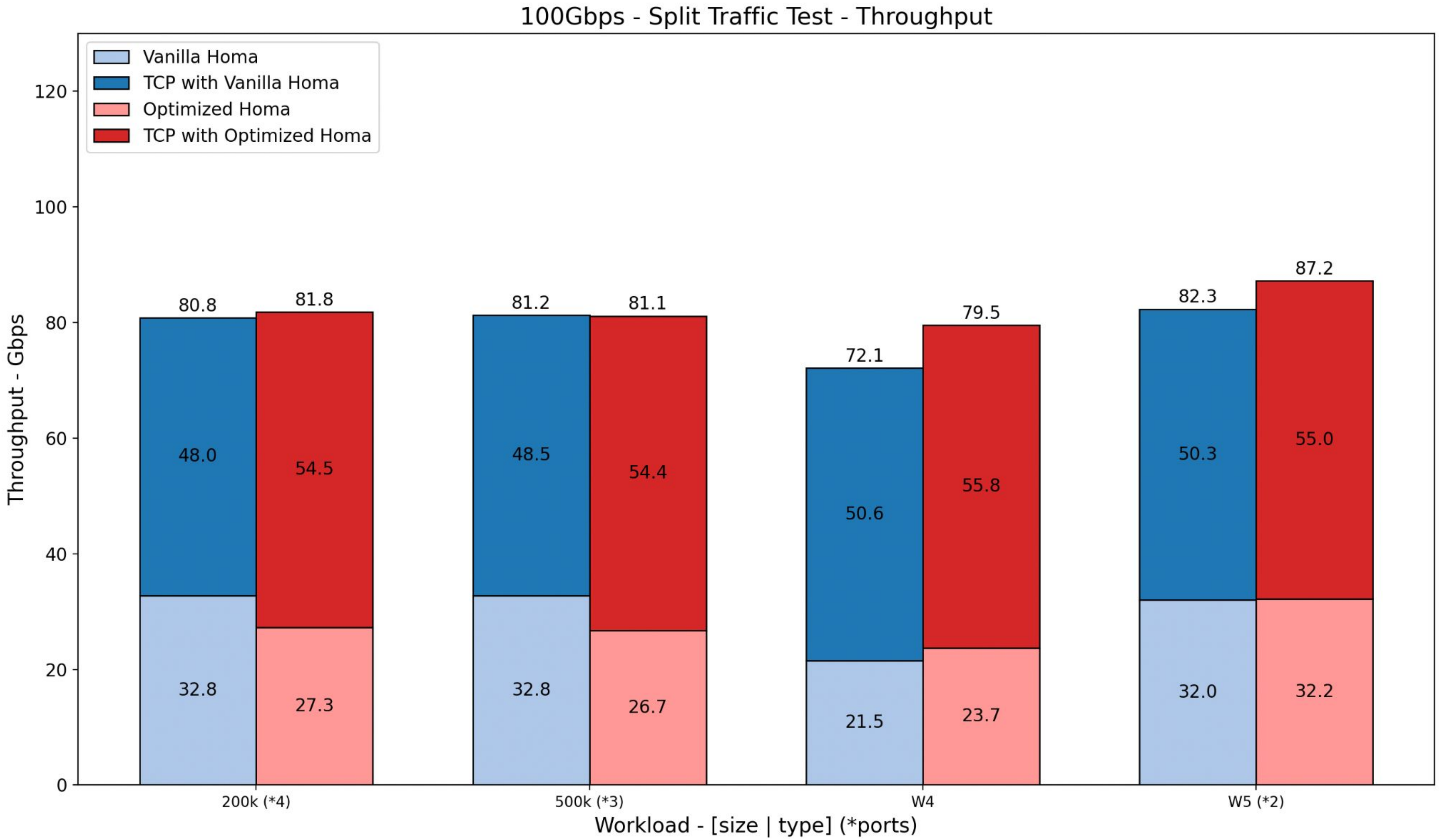
Long Message Incast - Latency



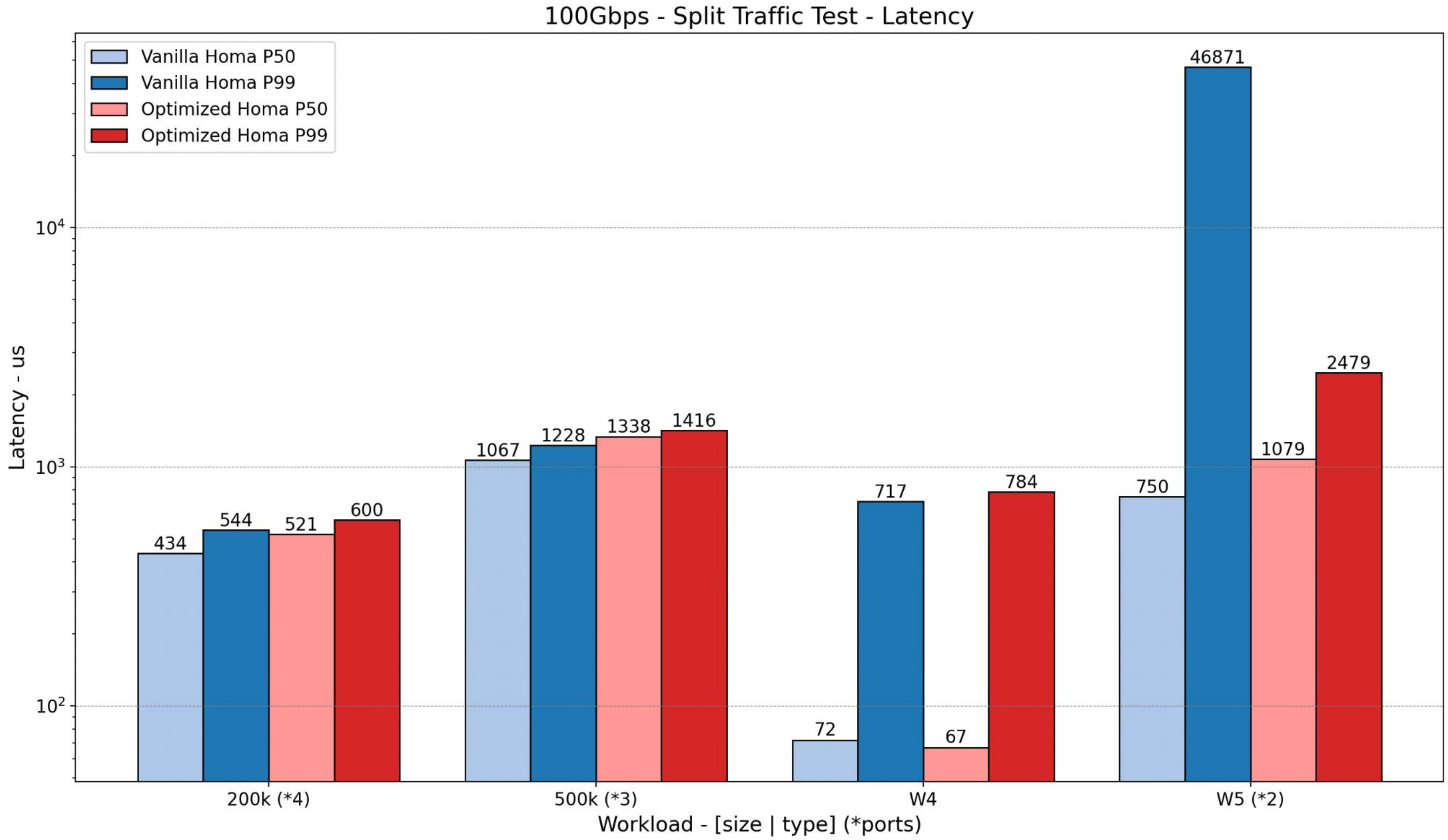
Split Traffic - Setup



Split Traffic - Throughput

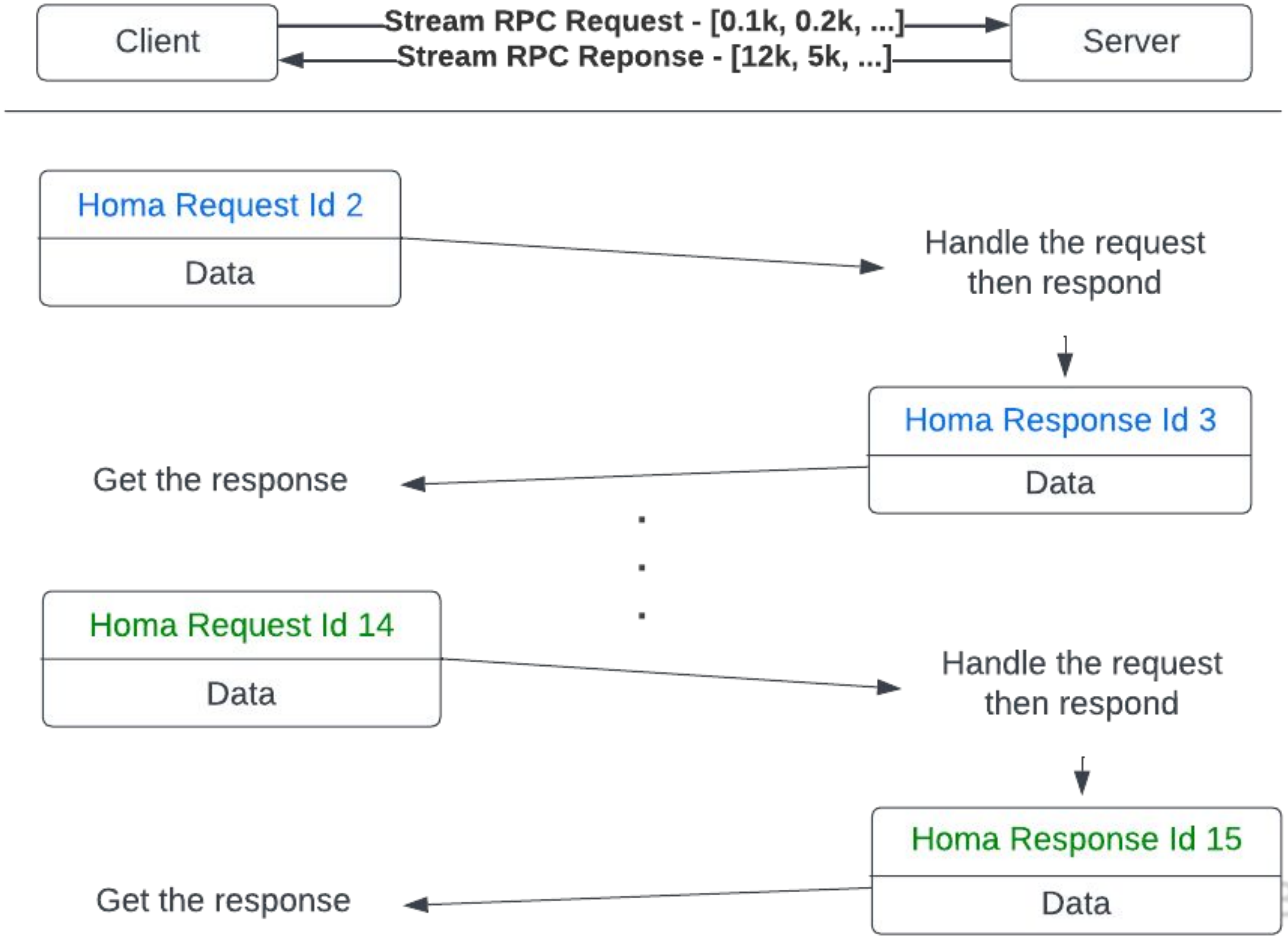


Split Traffic - Latency



Homa RPC Streaming Enhancements

Homa RPC Streaming



Client

Stream RPC Request - [0.1k, 0.2k, ...]

Stream RPC Reponse - [12k, 5k, ...]

Server

Homa Request Id 2

Data

Handle the request then respond

Homa Response Id 3

Data

Get the response

Homa Request Id 14

Data

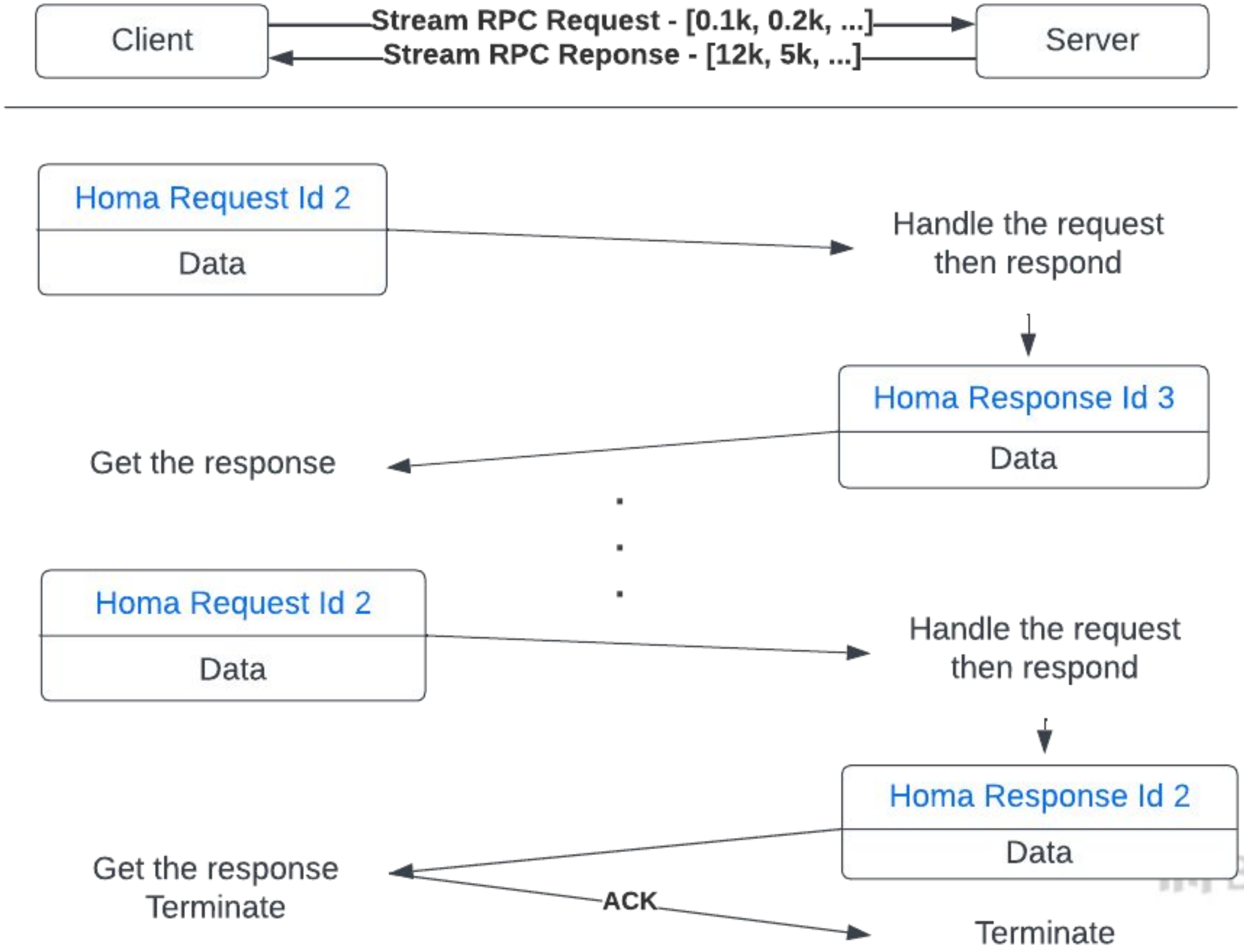
Handle the request then respond

Homa Response Id 15

Data

Get the response

Homa RPC Streaming Enhancements


Client
Stream RPC Request - [0.1k, 0.2k, ...]
Stream RPC Reponse - [12k, 5k, ...]
Server
Homa Request Id 2
Data
Handle the request then respond
Homa Response Id 3
Data
Get the response
Homa Request Id 2
Data
Handle the request then respond
Homa Response Id 2
Data
Get the response Terminate
ACK
Terminate

Future Improvements

Future Improvements

- More accurate RTT measurement (Fabric + NIC + software delay)
- Optimize the dynamic window algorithm (Any thoughts?)
- Optimize pacer
- Zero-copy
- More tests for stream RPC enhancements

Conclusion

1. Homa is a very promising protocol in RPC context

2. Automatic and Dynamic RTTBytes is a better choice

3. Dynamic per peer adjustable window

- Improve performance on throughput and latency
- Buffer Overlimit Resilience in Incast
- Compatibility with TCP Traffic

Q & A

References

1. John Ousterhout Stanford University, “A Linux Kernel Implementation of the Homa Transport Protocol”, 2021 USENIX Annual Technical Conference.
2. Radhika Mittal∗ (UC Berkeley), Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi∗ (Microsoft), Amin Vahdat, Yaogong Wang, David Wetherall, David Zats, “TIMELY: RTT-based Congestion Control for the Datacenter” SIGCOMM ’15 August 17-21, 2015, London, United Kingdom
3. Behnam Montazeri, Yilong Li, Mohammad Alizadeh† , and John Ousterhout Stanford University, +MIT, “Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities”,SIGCOMM ’18, August 20-25, 2018, Budapest, Hungary
4. John Ousterhout https://homa-transport.atlassian.net/wiki/spaces/HOMA/pages/262178/Homa+Projects