Multi-Path Transport for RDMA in Datacenters

Yuanwei Lu[1,2], Guo Chen[3], Bojie Li[1,2], Kun Tan[4], Yongqiang Xiong[2], Peng Cheng[2], Jiansong Zhang[2], Enhong Chen[1], Thomas Moscibroda[5]

[1]USTC, [2]Microsoft Research, [3]Hunan University,
[4]Huawei Technologies, [5]Microsoft Azure

Microsoft

Remote Direct Memory Access

Kernel bypass

- Low and stable base latency: <10µs

Transport hardware offloading

- High throughput with low CPU overhead

Remote Direct Memory Access

RDMA (RoCEv2) is deployed at scale in Microsoft Datacenter

- e.g. Half of the traffic in a online service datacenter is RDMA [1]

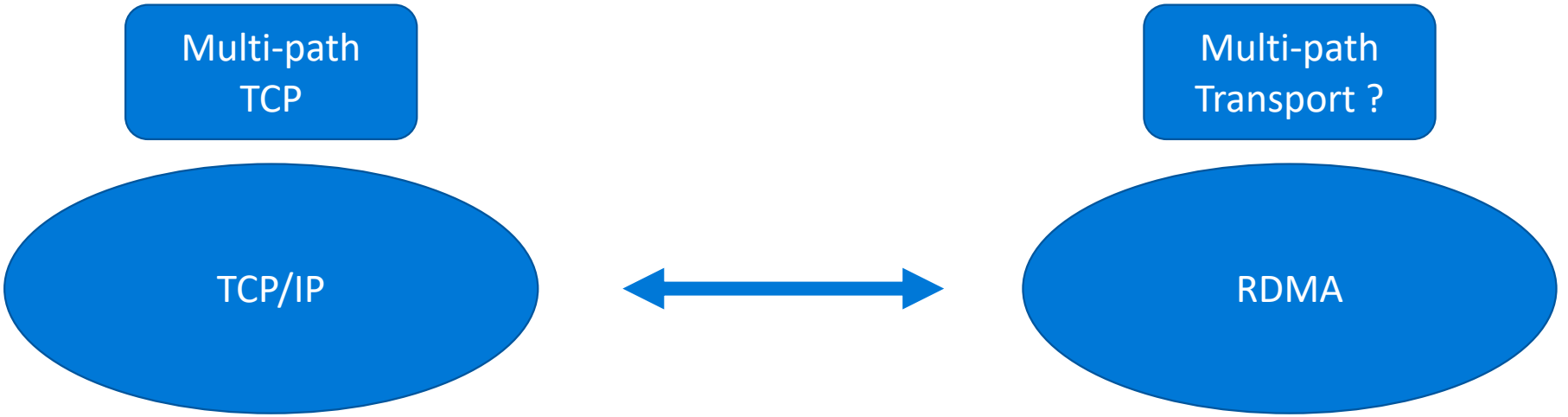
Datacenter topologies: FatTree, BCube, VL2

- Lots of parellel paths between any pair of nodes

Current RDMA is single-path with ECMP

- Hot spot caused high latency and low throughput
- Not robust to failure

[1] Guo, Chuanxiong, et al. "RDMA over Commodity Ethernet at Scale. "*Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016.



Goal: design a high performance multi-path transport for RDMA. (MP-RDMA)

Uniqueness of RDMA Environment

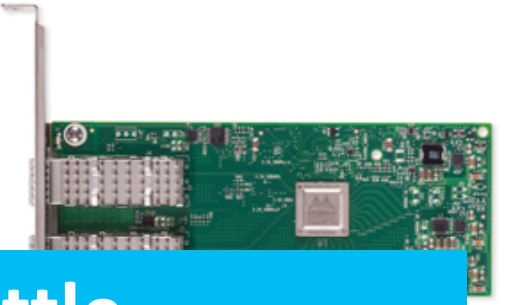
RDMA must be implemented in hardware

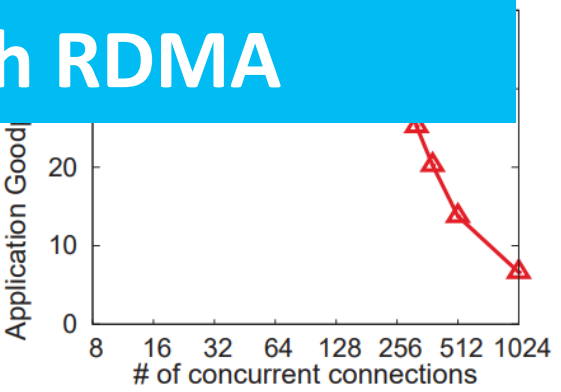
NIC on-chip memory is scarce

- C
- Large and unstable latency

Frequent cache miss results in poor performance[1]

MP-RDMA design needs to add as little hardware memory footprint as possible compared with existing single-path RDMA





[1] Kalia, Anuj, Michael Kaminsky, and David G. Andersen. "Design Guidelines for High Performance RDMA Systems." *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 2016.

Challenges

#1: How to achieve congestion-aware traffic split without per-path states at NIC?

#2: How to track out-of-order packets with small on-chip memory footprint?

#3: How to provide message ordering without hurting performance?

Challenges

#1: How to achieve congestion-aware traffic split without per-path states at NIC?

#2: How to track out-of-order packets with small on-chip memory footprint?

#3: How to provide message ordering without hurting performance?

Challenge #1

Congestion-aware load distribution

- Tracking the transmission and congestion states on every paths

Challenge #1:
How to achieve congestion-aware traffic split without per-path states at NIC?

- RoCEv2: 248B per connection

Multi-Path ACK Clocking

A transport without per-path states

- Insight: window-based protocol with ACK clocking

Algorithm:

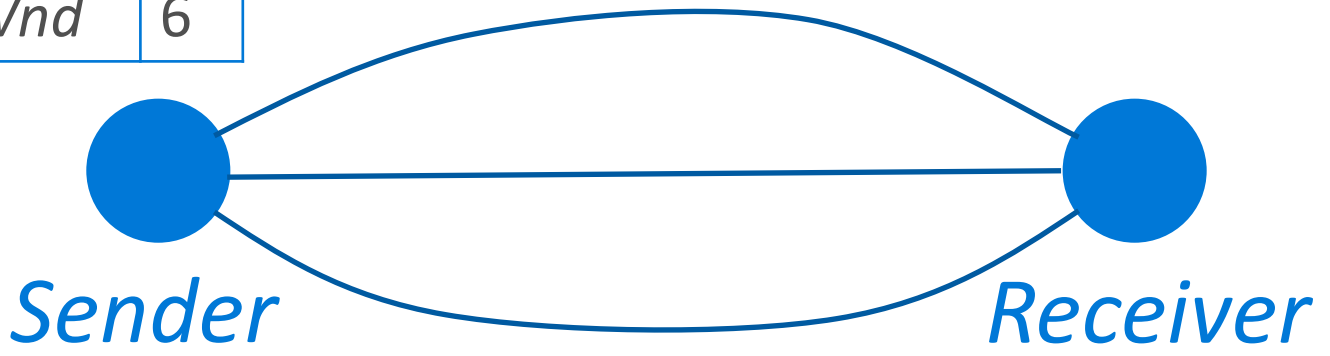
- At beginning, spread N packets randomly to N paths
- One $cwnd$ for all paths, for each received ACK:

$$cwnd \leftarrow \begin{cases} cwnd + \frac{1}{cwnd}; & if\ ECN = 0 \\ cwnd\ - \frac{1}{2}; & if\ ECN = 1 \end{cases}$$

- Clock one packet to the path ACK returns when $cwnd$ allows

An Example

cWnd	6
Inflight	0
aWnd	6

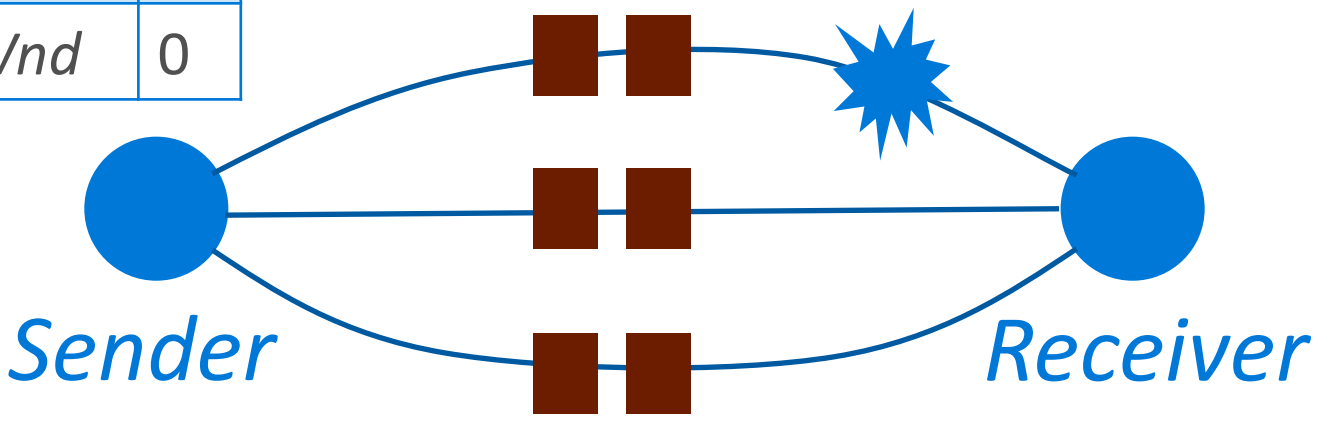


Data Packet

ACK Packet

An Example

cWnd	6
Inflight	6
aWnd	0

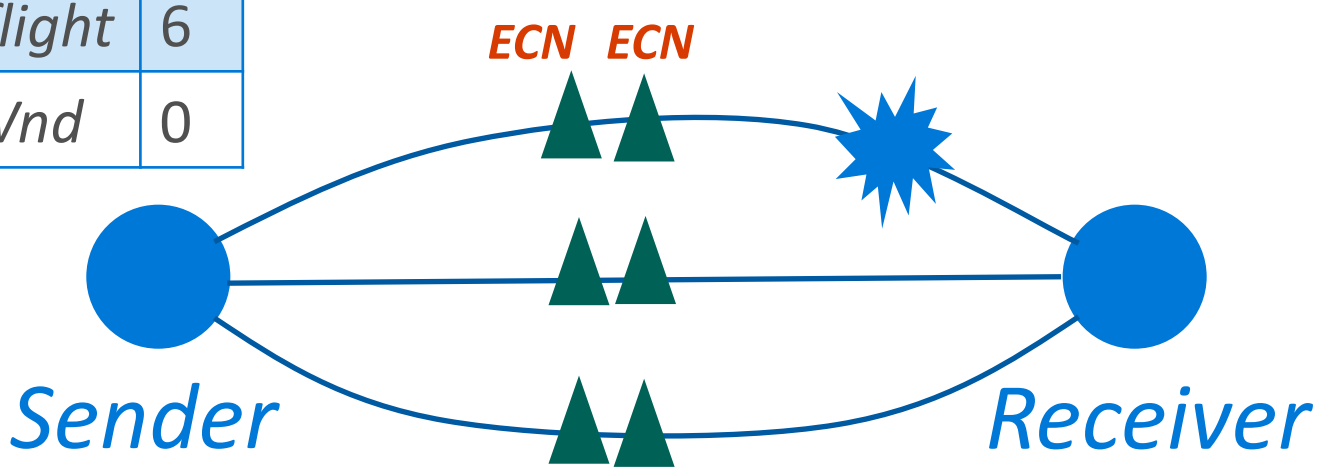


Data Packet

ACK Packet

An Example

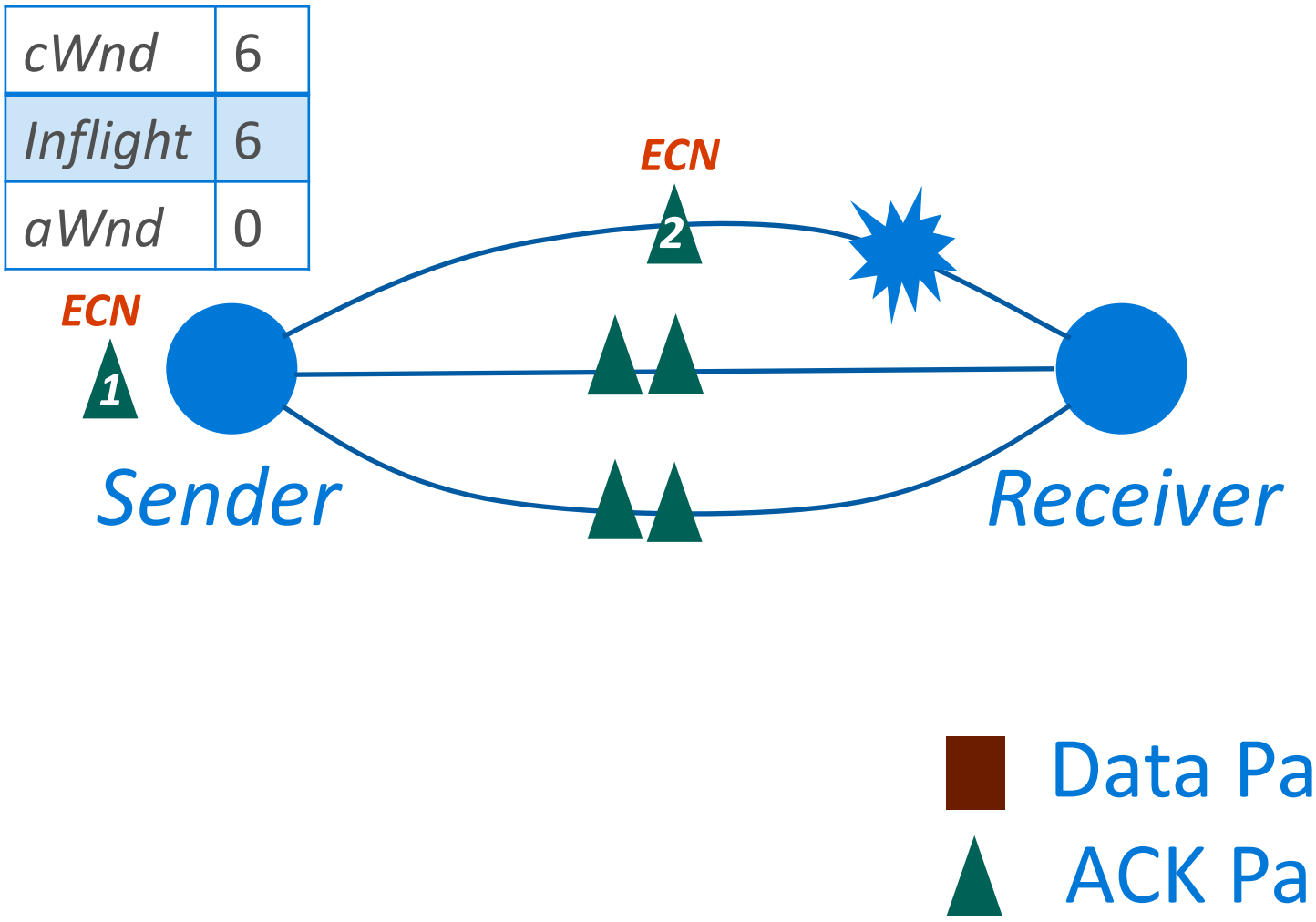
cWnd	6
Inflight	6
aWnd	0



Data Packet

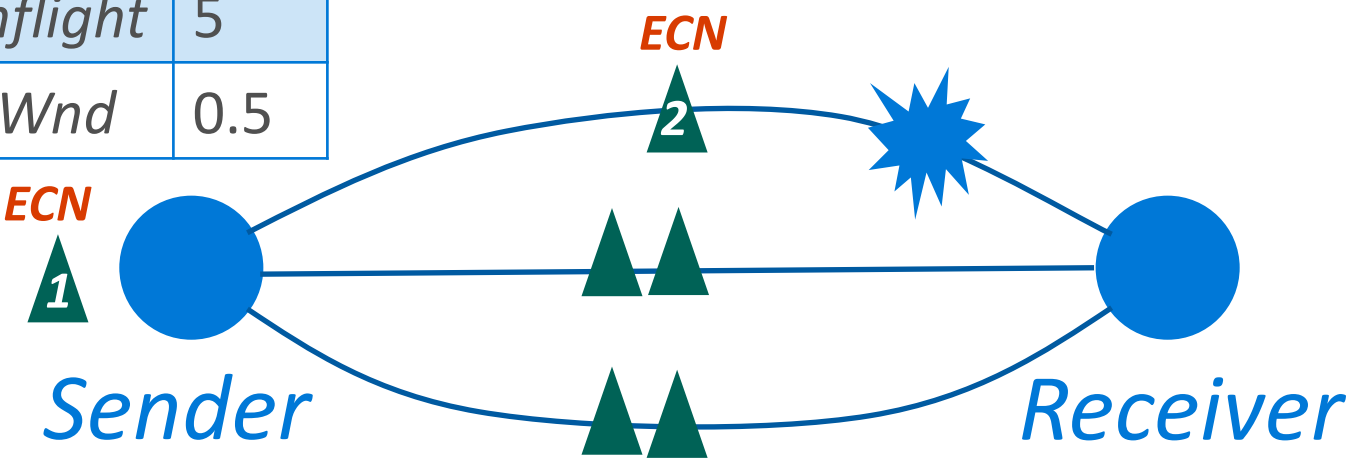
ACK Packet

An Example


cWnd 6
Inflight 6
aWnd 0
ECN
2
ECN
1
Sender
Receiver
Data Packet
ACK Packet

An Example

cWnd	5.5
Inflight	5
aWnd	0.5

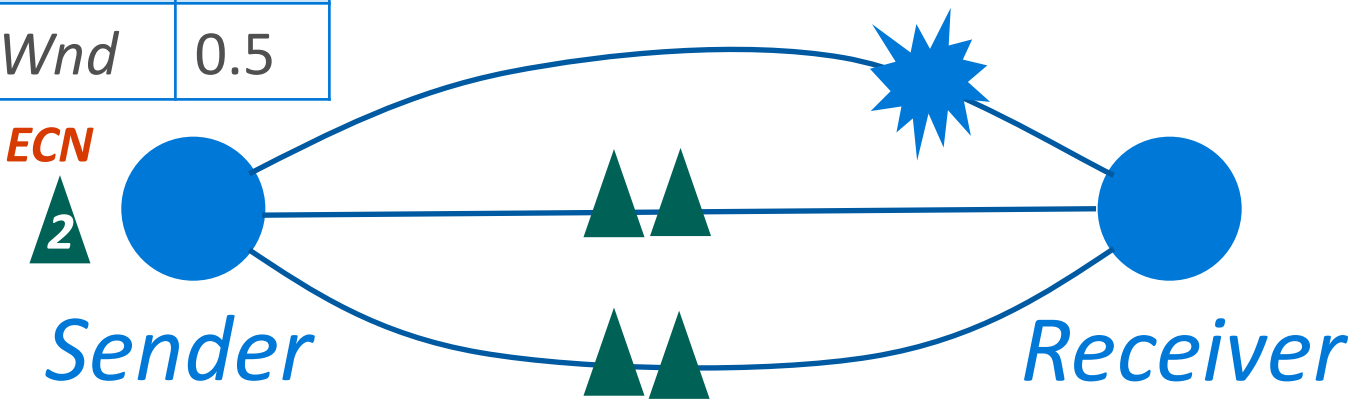


Data Packet

ACK Packet

An Example

cWnd	5.5
Inflight	5
aWnd	0.5

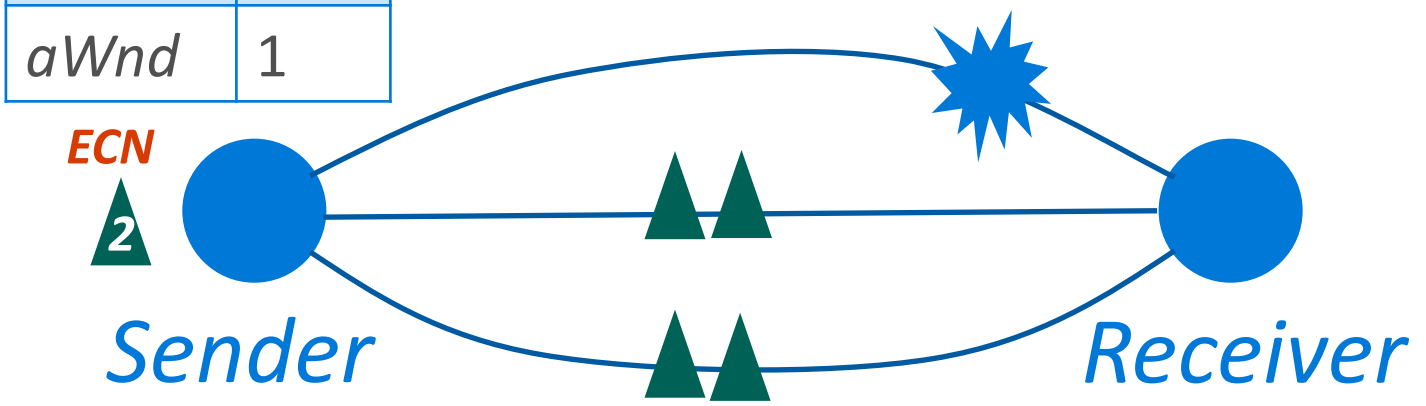


Data Packet

ACK Packet

An Example

cWnd	5
Inflight	4
aWnd	1

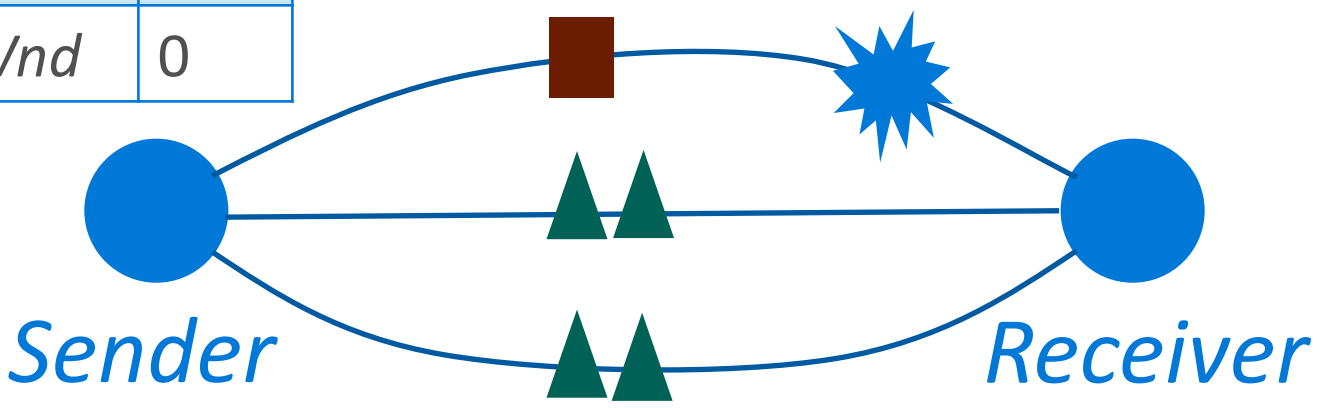


Data Packet

ACK Packet

An Example

cWnd	5
Inflight	5
aWnd	0

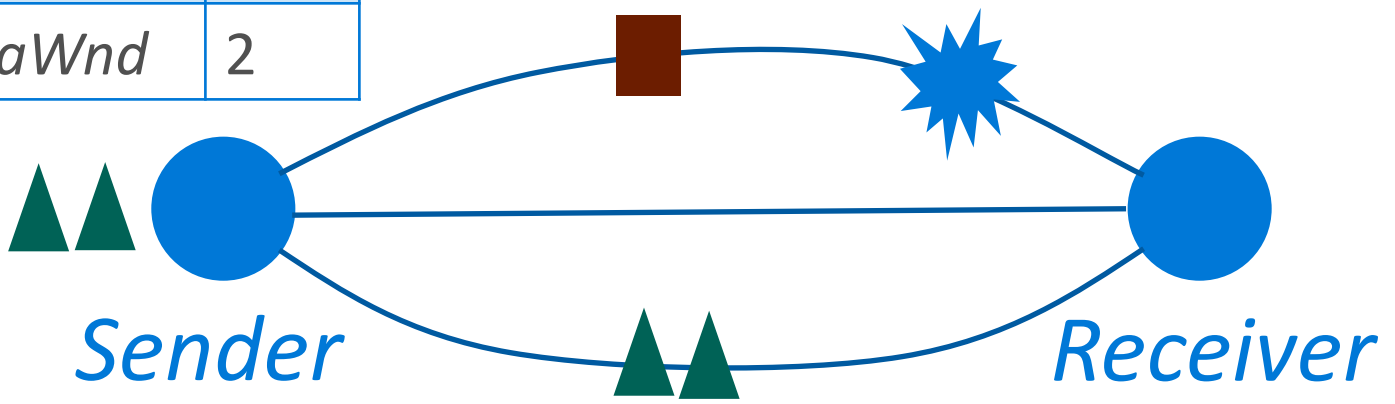


Data Packet

ACK Packet

An Example

cWnd	5
Inflight	3
aWnd	2

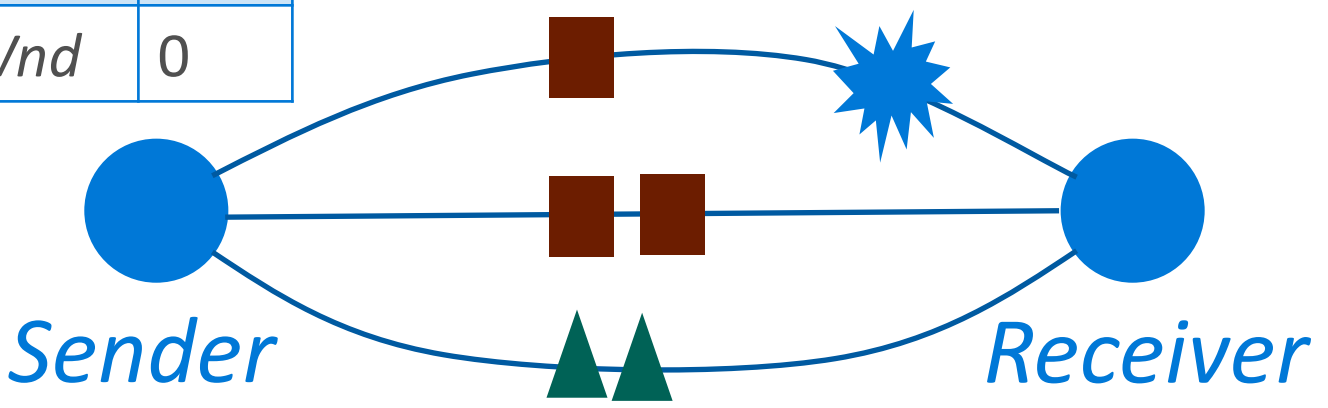


Data Packet

ACK Packet

An Example

cWnd	5
Inflight	5
aWnd	0

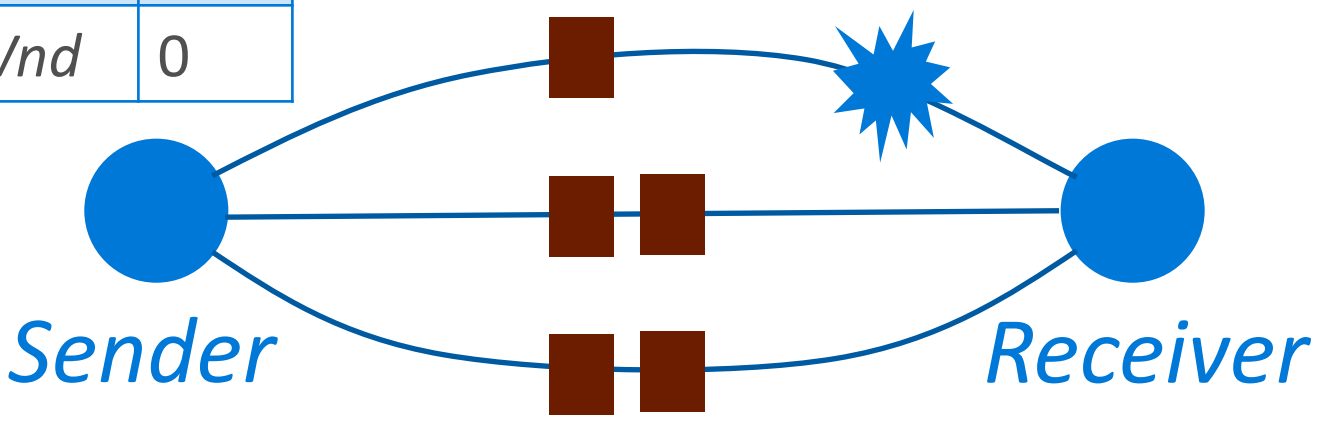


Data Packet

ACK Packet

An Example

cWnd	5
Inflight	5
aWnd	0

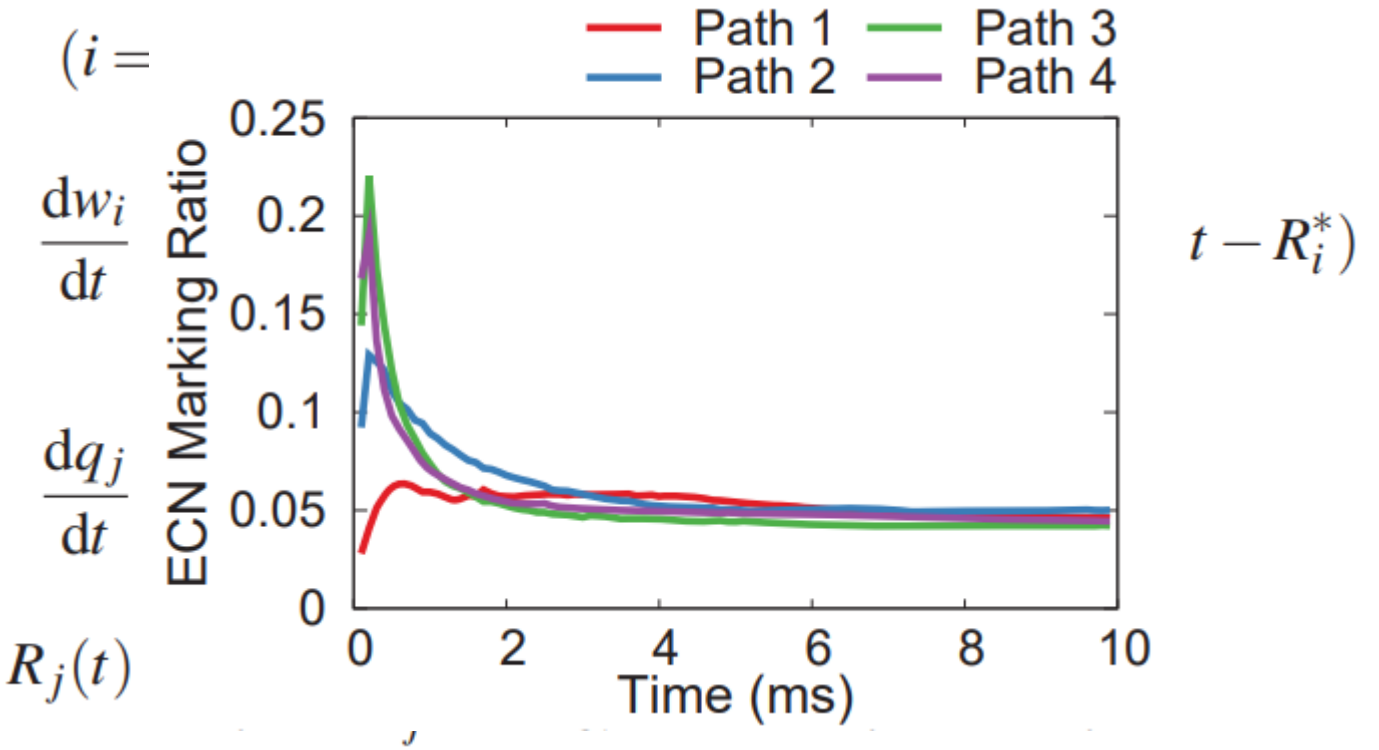


Data Packet

ACK Packet

Fluid Model

N flows, M_v virtual paths, M_p physical paths



MP-RDMA can balance the ECN marking ratio among all sending paths

Challenges

#1: How to achieve congestion-aware traffic split without per-path states at NIC?

#2: How to track out-of-order packets with small on-chip memory footprint?

#3: How to provide message ordering without hurting performance?

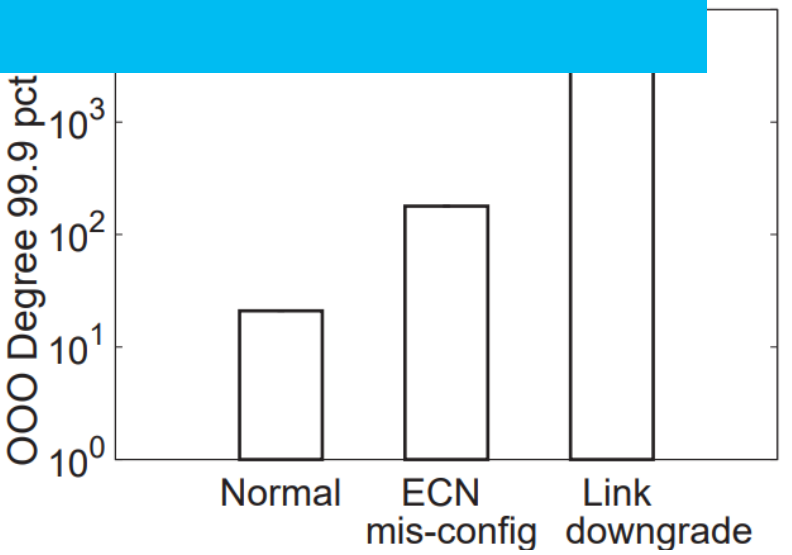
Challenge #2

Multi-path leads to packets out-of-order

- Need meta data to track out-of-order

Challenge #2:
How to track out-of-order packets with small on-chip memory footprint?

- e.g. Link downgrade: 1.2KB bitmap
- e.g. PFC pause: infinite

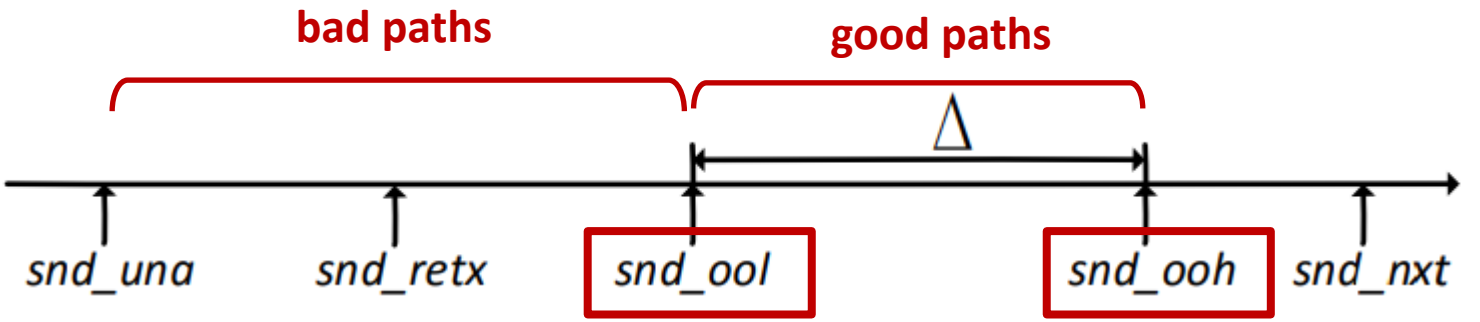


Out-of-order Aware Path Selection

MP-RDMA chooses paths with similar delay

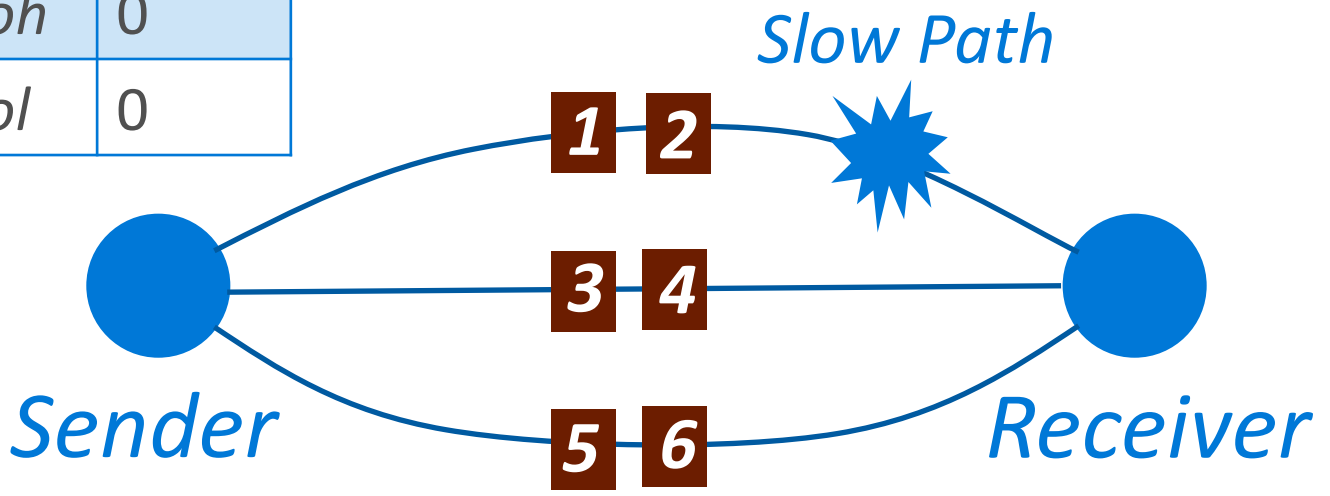
Sender maintains two pointers:

- snd_ooh: the highest PSN successfully received at receiver (fast paths)
- snd_ool: the smallest ACK PSN that can clock out packet
- $snd\_ool = snd\_ooh - \Delta$



An Example

Δ	3
snd_ooh	0
snd_ool	0

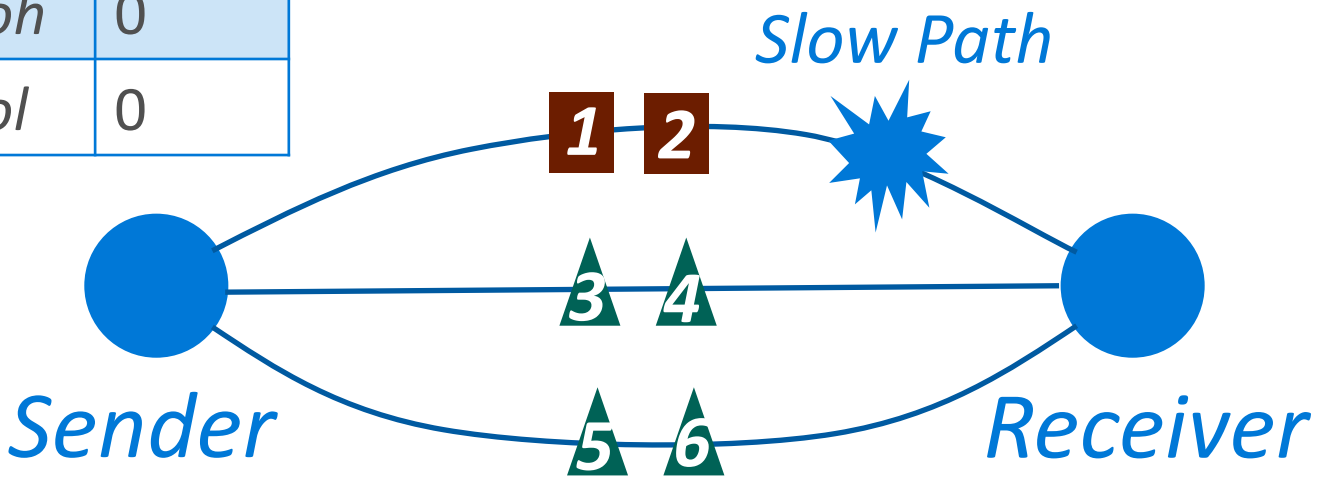


Data Packet

ACK Packet

An Example

Δ	3
snd_ooh	0
snd_ool	0

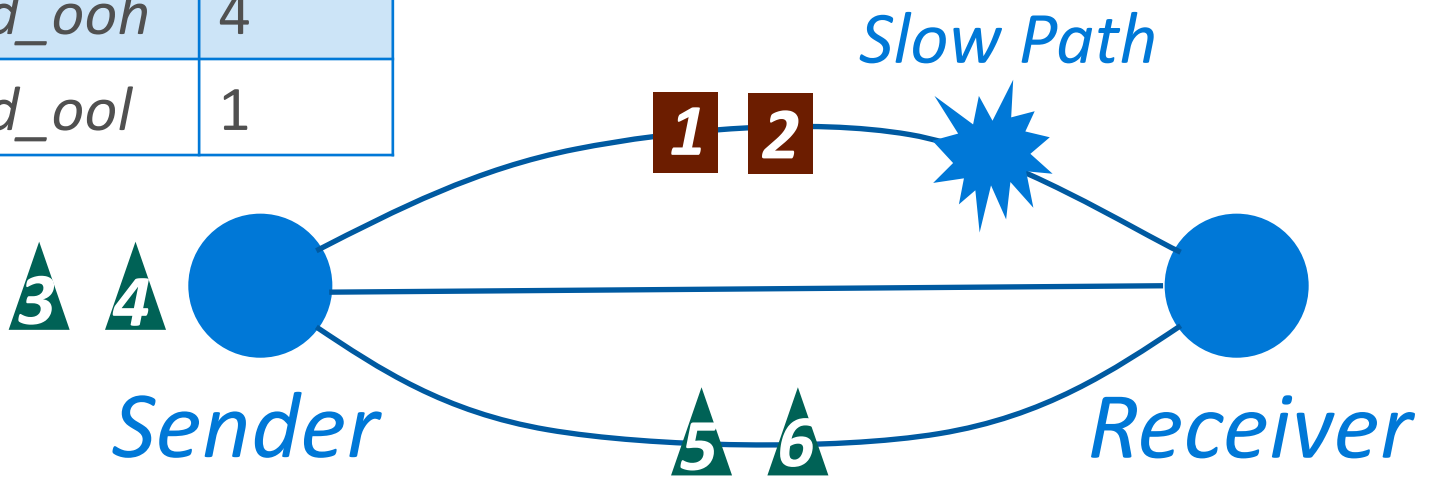


Data Packet

ACK Packet

An Example

Δ	3
snd_ooh	4
snd_ool	1

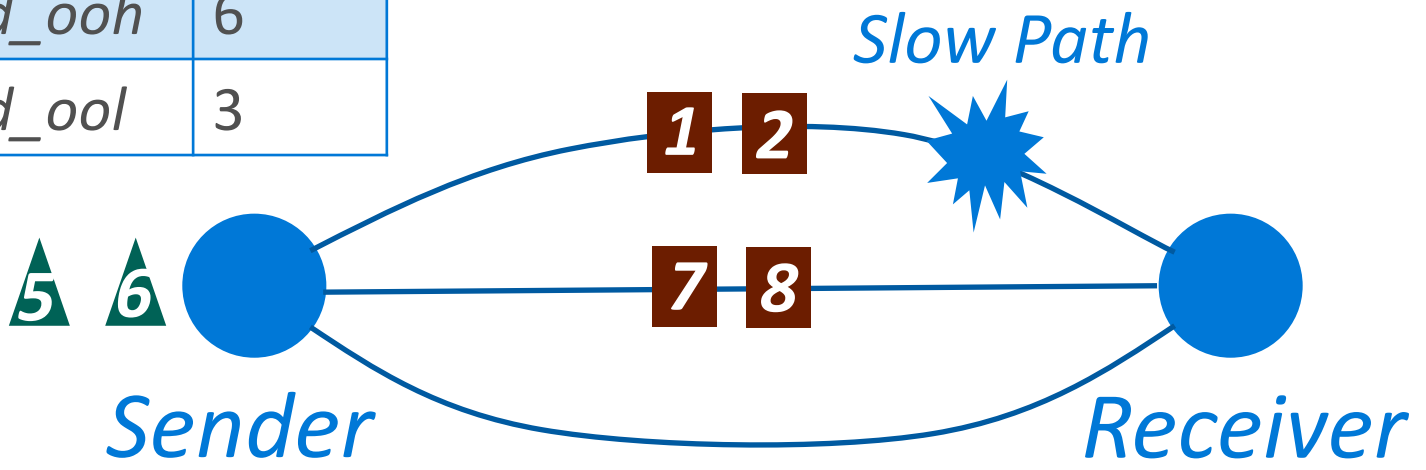


Data Packet

ACK Packet

An Example

Δ	3
snd_ooh	6
snd_ool	3

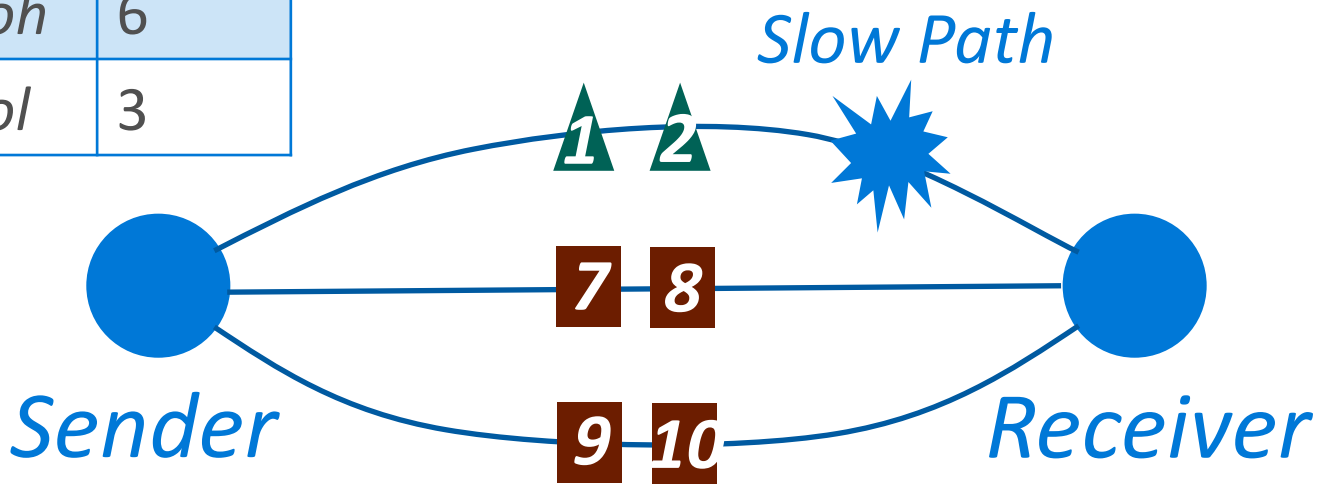


Data Packet

ACK Packet

An Example

Δ	3
snd_ooh	6
snd_ool	3

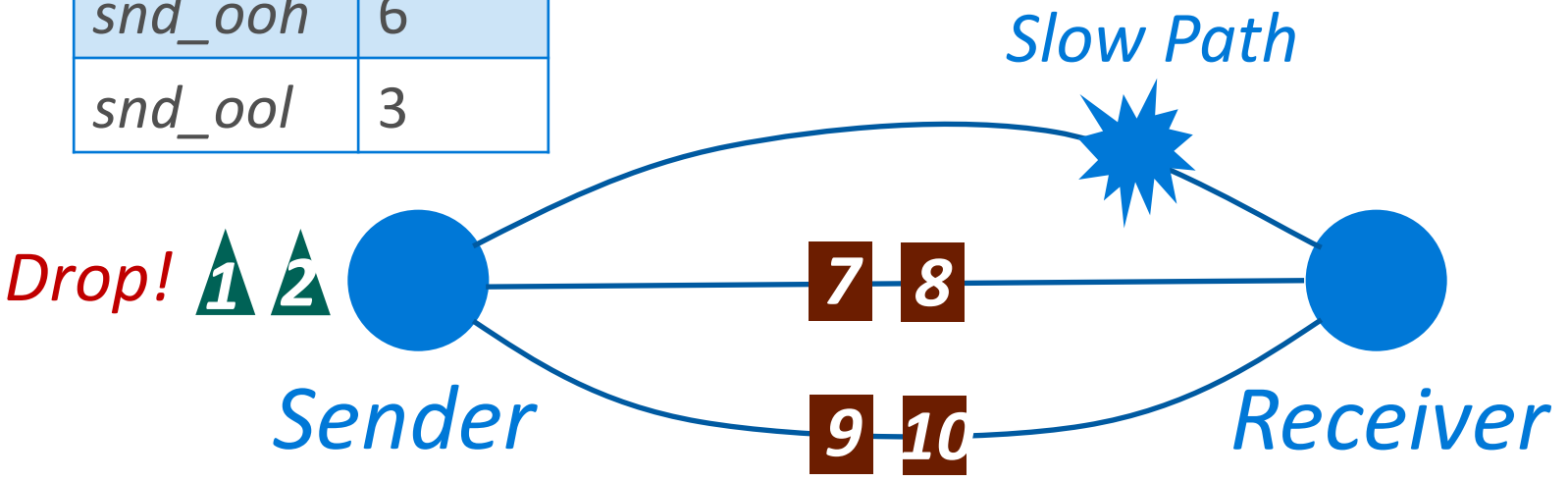


Data Packet

ACK Packet

An Example

Δ	3
snd_ooh	6
snd_ool	3



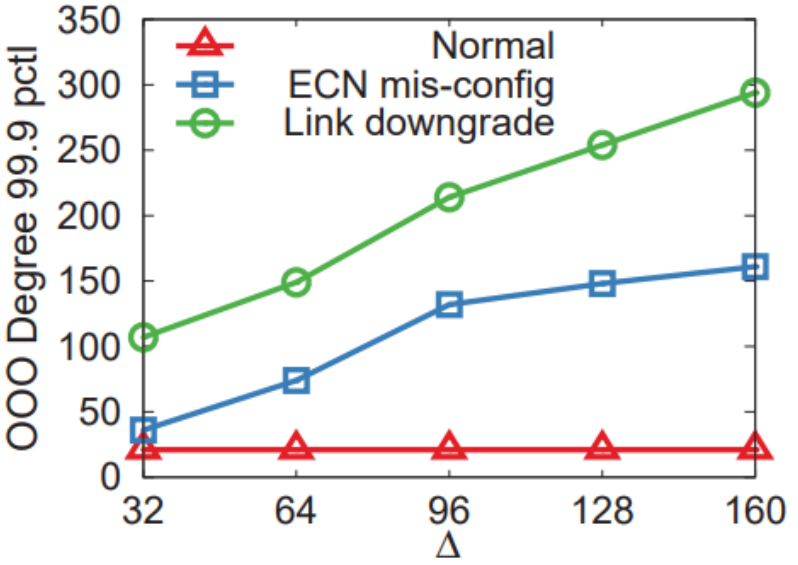
Data Packet

ACK Packet

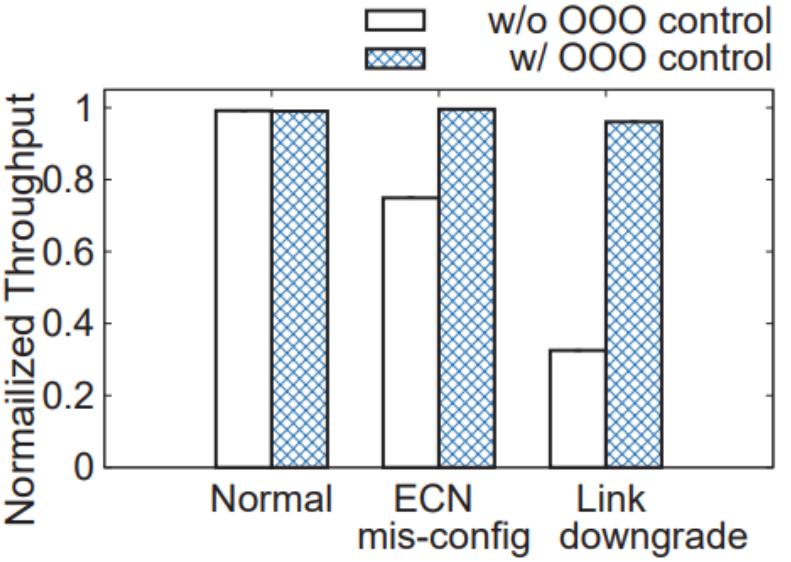
Experiment Results

Adjust Δ can control out-of-order degree

Small out-of-order improves throughput



(a) OOO control.



(b) Goodput.

MP-RDMA can track OOO with a tiny bitmap

Challenges

#1: How to achieve congestion-aware traffic split without per-path states at NIC?

#2: How to track out-of-order packets with small on-chip memory footprint?

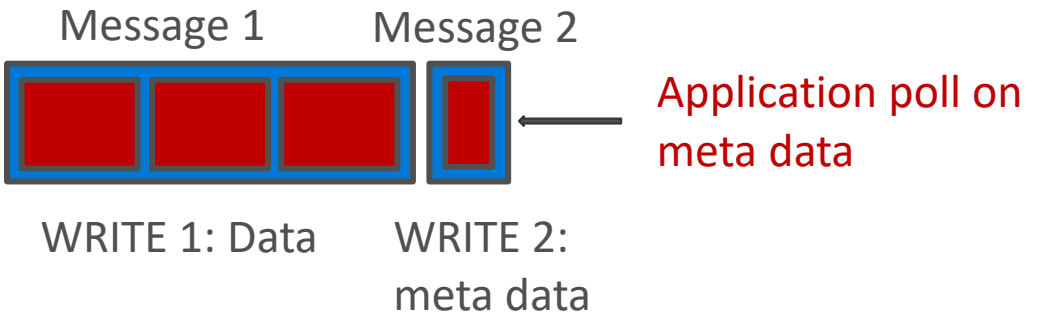
#3: How to provide message ordering without hurting performance?

Challenge #3

Out-of-order data is directly placed to application buffer

- Problem: application may assume in-order messages delivery

Challenge #3:
How to provide message ordering without hurting the application performance?



Synchronise Mechanism

MP-RDMA provides a "*Synchronise*" interface

- A synchronise message is guaranteed to arrive after all previous messages

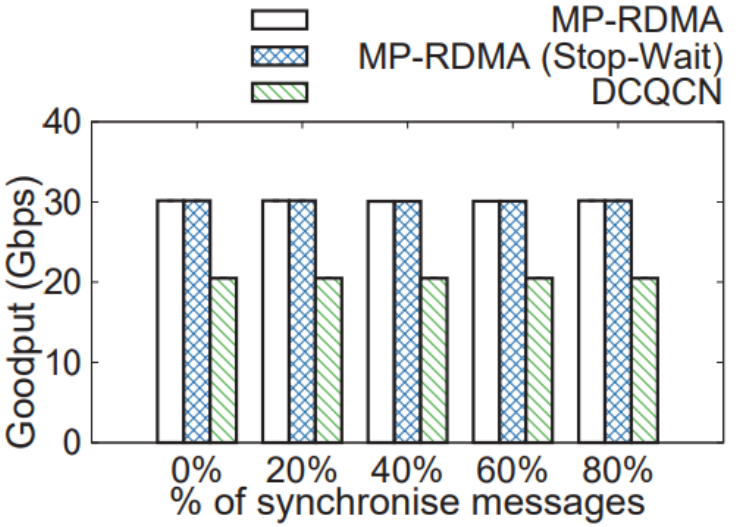
Optimistically transmit a *synchronise* message

- A synchronise message just wait Δt before previous messages
- $\Delta t = \alpha * \Delta / (cwnd/RTT)$, large enough to avoid out-of-order

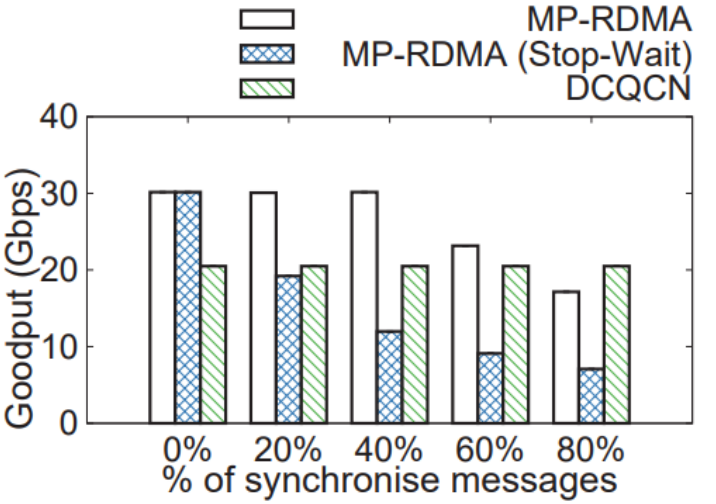
Experiment Results

No performance sacrifice for large messages

Little downgrade for small messages



(a) 512KB message.



(b) 32KB message.

Syncronise mechanism provides message ordering without sacrificing throughput

Implementation

ClickNP[1] FPGA platform

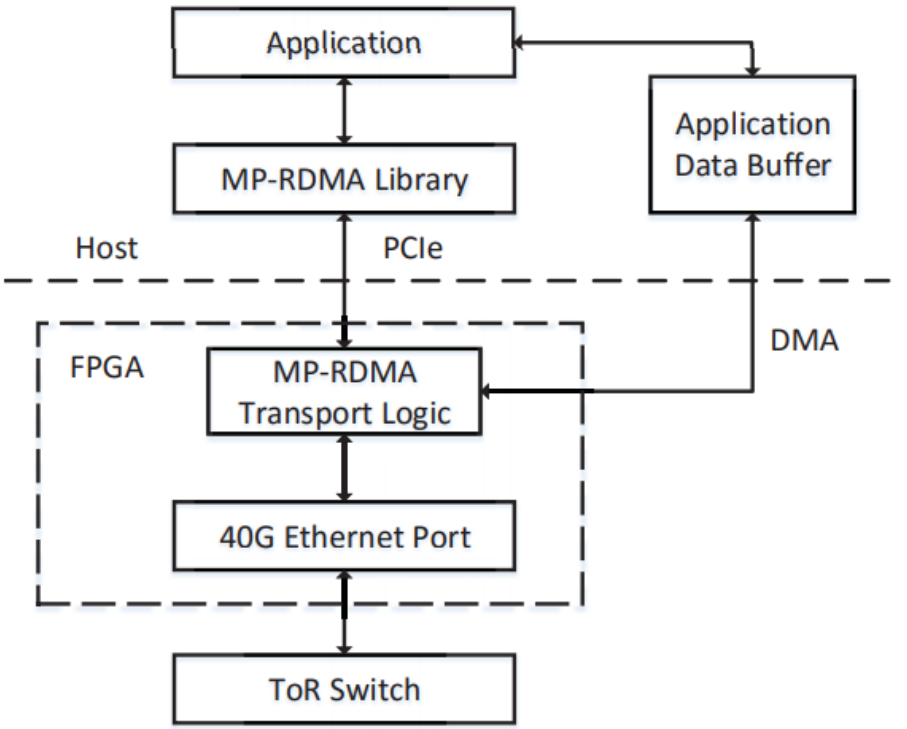
- Altera Stratix V D5 FPGA board with two 40Gb Ethernet ports
- 14 elements, ~2K lines of OpenCL code

Send logic:

- Dependency: 9 cycles
- Clock Frequency: ~206MHz

Receive logic:

- Dependency: 3 cycles
- Clock Frequency: ~196MHz



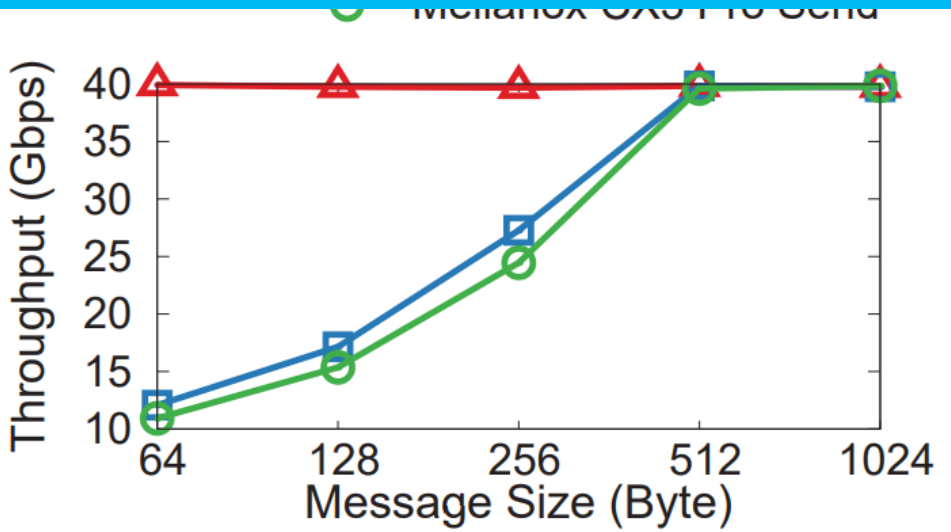
[1] Li, Bojie, et al. "ClickNP: Highly flexible and High-performance Network Processing with Reconfigurable Hardware." ACM SIGCOMM 2016 Conference.

Performance

Send and receive at line rates

Latency:

- Send: 0.54µs
- **Conclusion:**
- **MP-RDMA can be implemented in hardware with high performance**

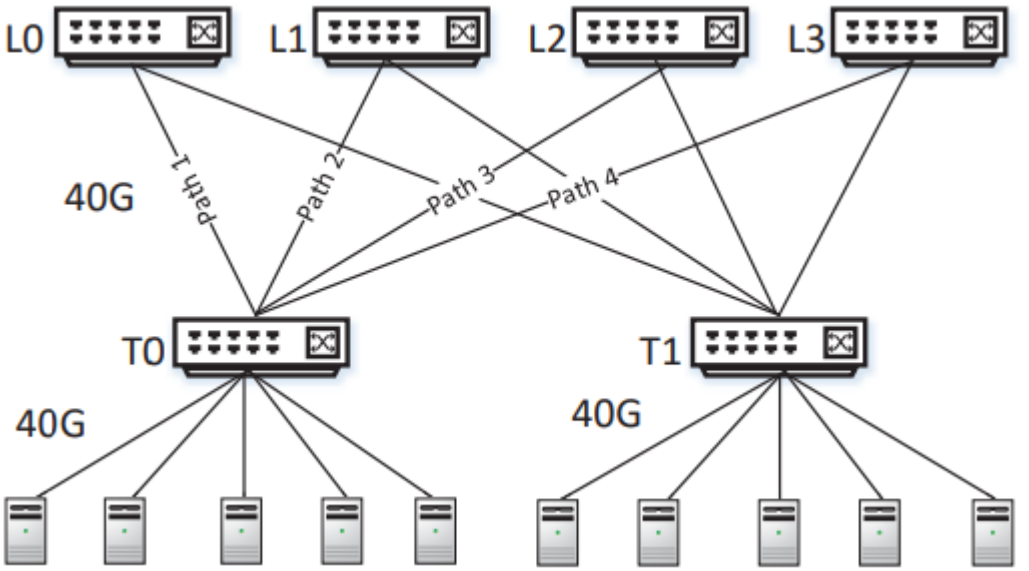


Testbed

10 FPGA boards

Leaf-Spine topology: 4 Paths

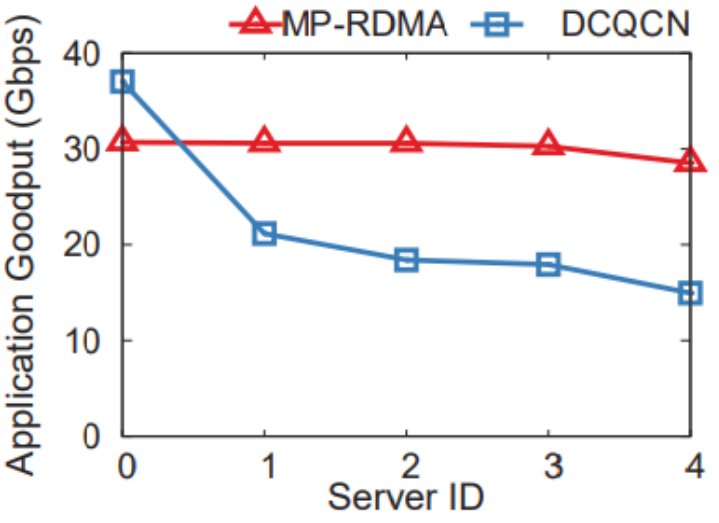
Base RTT: 16 µs



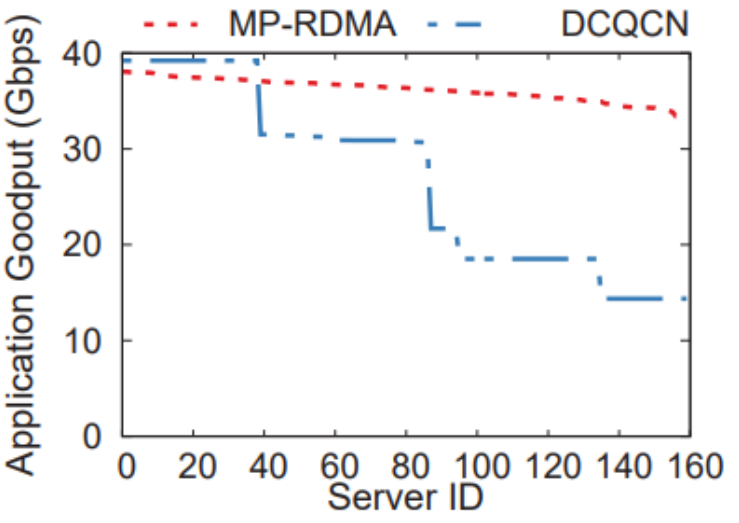
Key Results

Throughput

- 47.05% higher application throughput v.s. DCQCN



(a) Small-scale testbed.

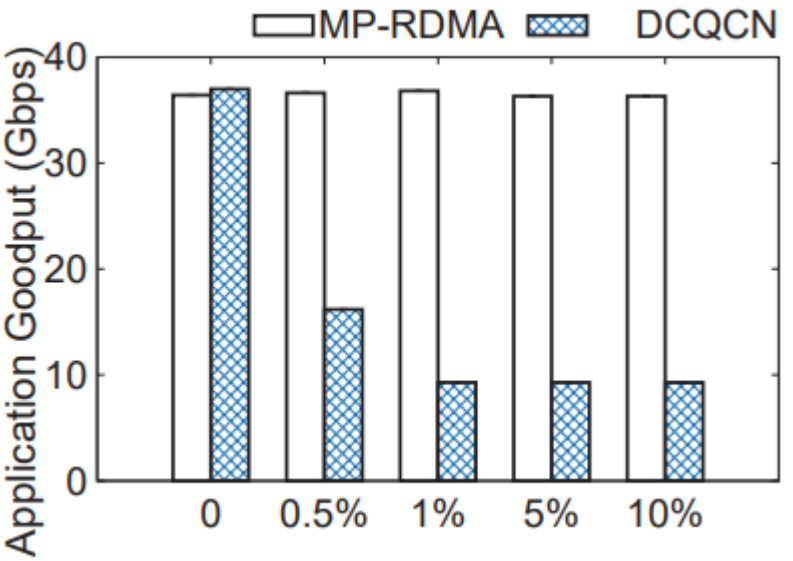


(b) Large-scale simulation.

Key Results

Robustness

- Under loss: > 3x throughput improvement



Summary

First effort to design a multi-path transport for RDMA environment

Minimize on-chip memory footprint

- No per-path states
- Out-of-order aware path selection
- Synchronise mechanism to ensure message ordering

Improve throughput, reduce latency and is robust to failure

Thank you !

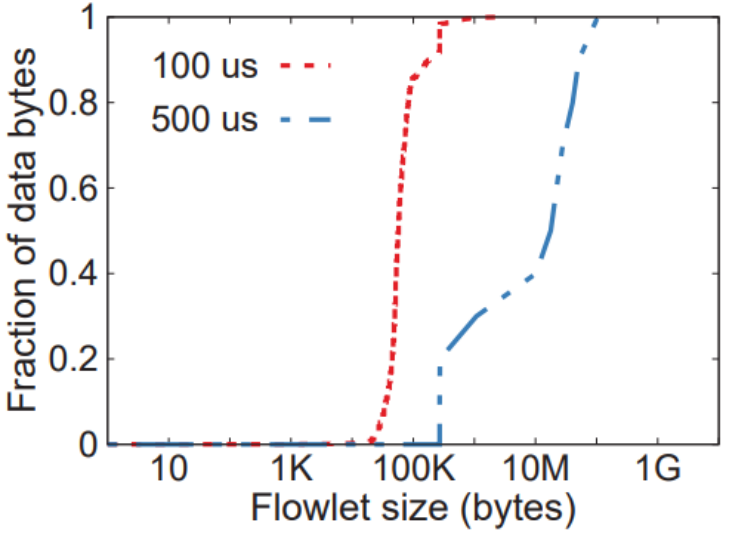
Q& A

Backup slides

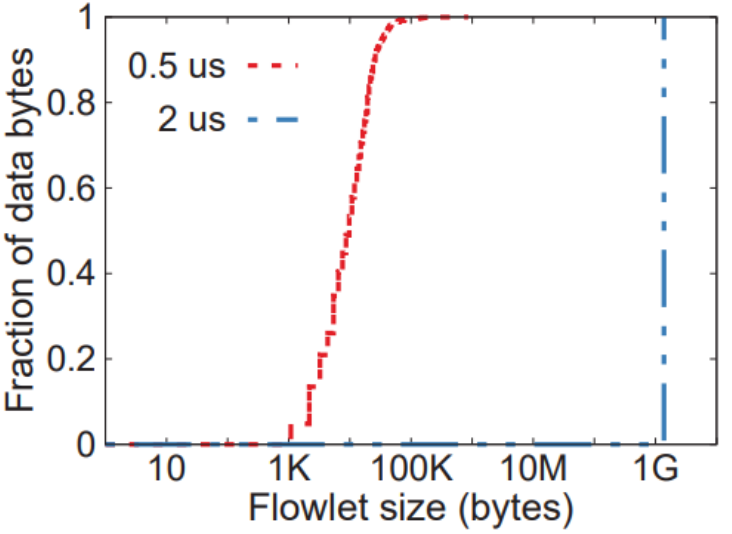
Flowlet in RDMA

TCP: inactive gap 100µs, flowlet size ~60KB

RDMA: inactive gap 2µs, flowlet size ~2GB



(a) TCP.



(b) RDMA.