Network Load Balancing with In-network Reordering Support for RDMA

ACM SIGCOMM 2023

202035303 고현철

Content

01. Introduction

What is main topic?

02. Reordering out-of-order packets

Existing reordering method

03. Conweave

What is Conweave?

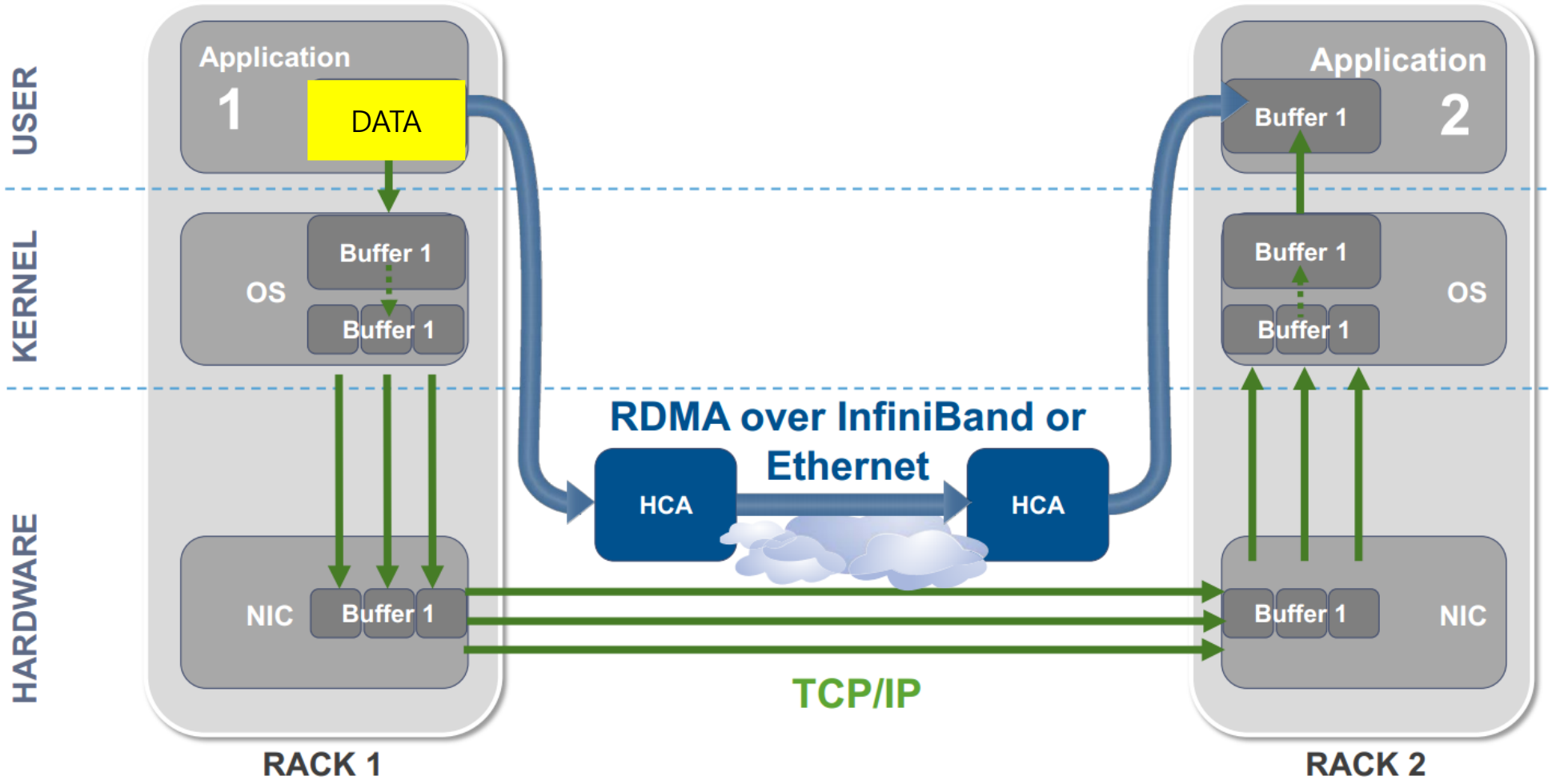
04. Evaluation

How much does it improve performance?

01.
Introduction

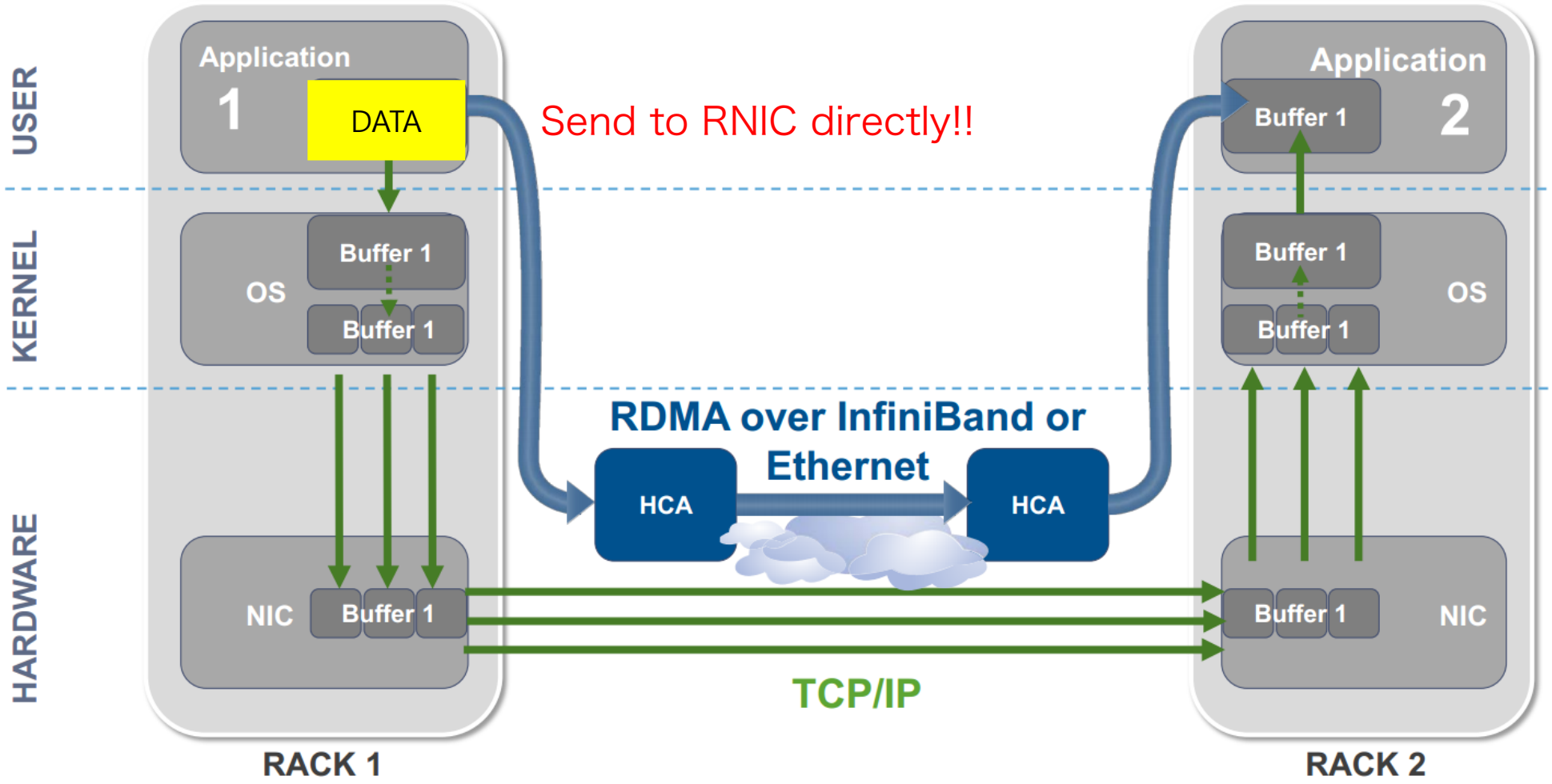
Introduction

The emergence of RDMA – Existing Method



Introduction

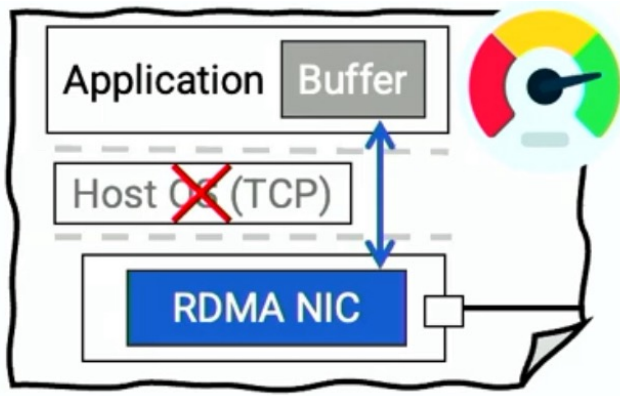
The emergence of RDMA



Introduction

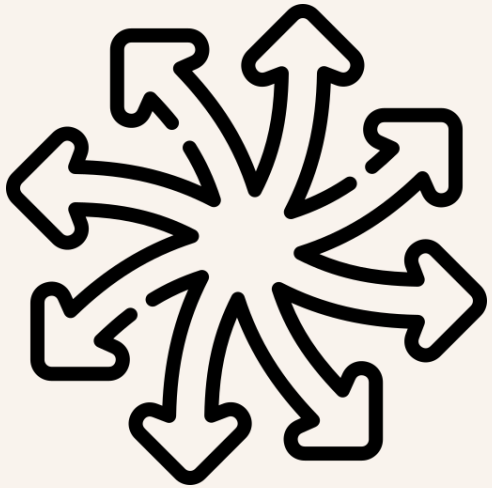
Advantages of RDMA

- RDMA is the emerging standard in modern DataCenters



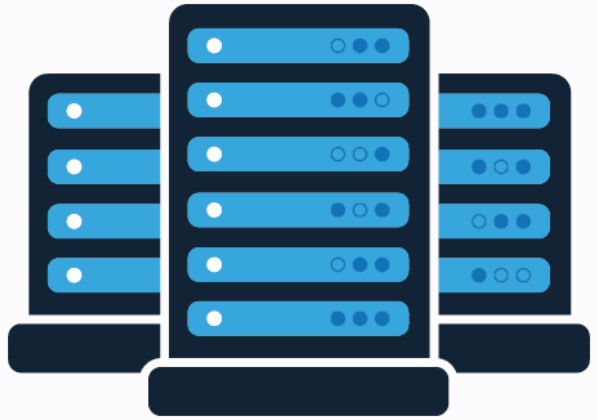
Performance

- High throughput
- Low Latency
- Low CPU overhead



Versatility

- RDMA + RPC
- RDMA + Storage
- RDMA + AI/ML



Practicality

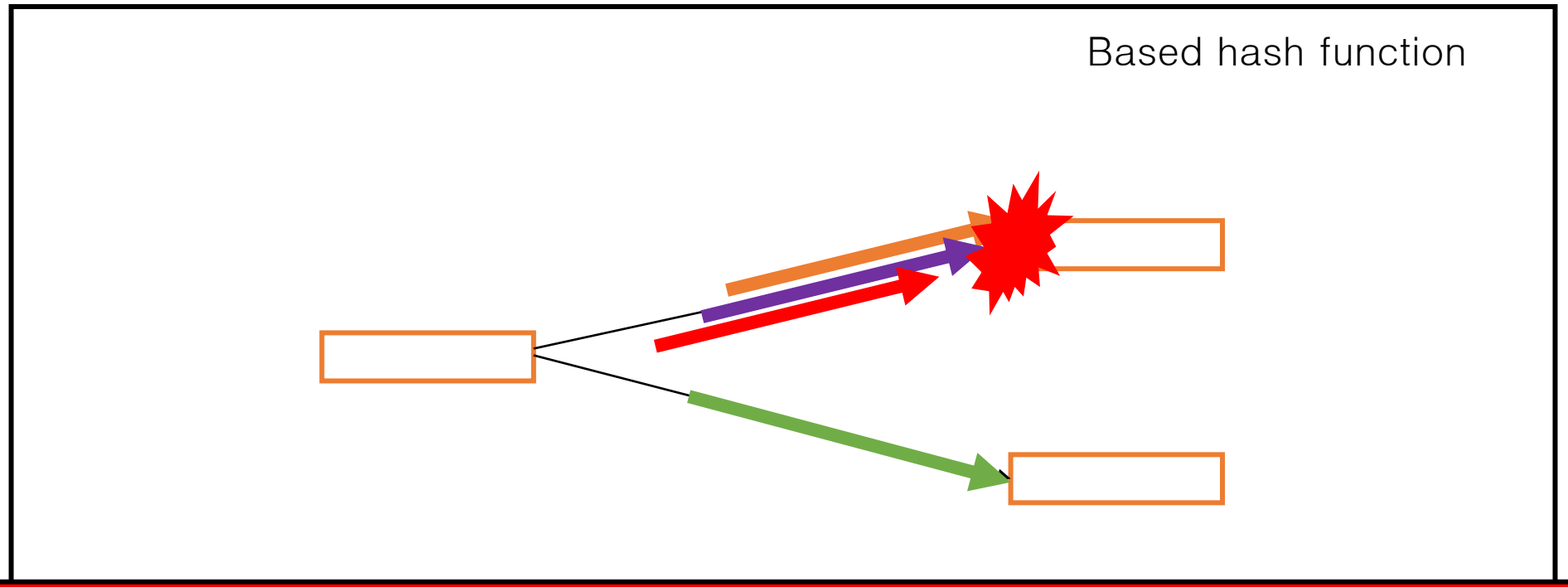
- Production-level deployment in DCs

Introduction

RDMA has these advantages, but existing load balancing algorithms do not work well!

Per-flow switching

- ECMP (Equal Cost Multi Path)

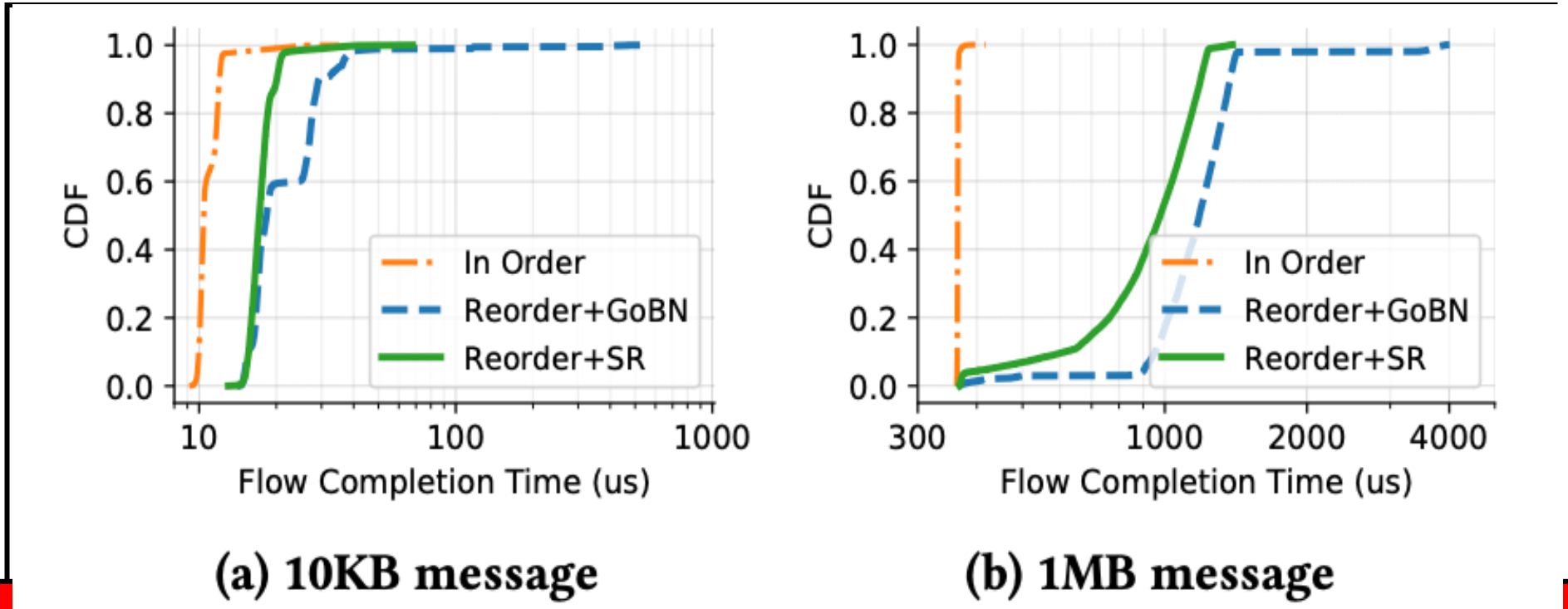


Cannot evenly distribute flows based on their sizes

Introduction

Per-packet switching

- Spraying, DRILL
- Provide a near-optimal load balance

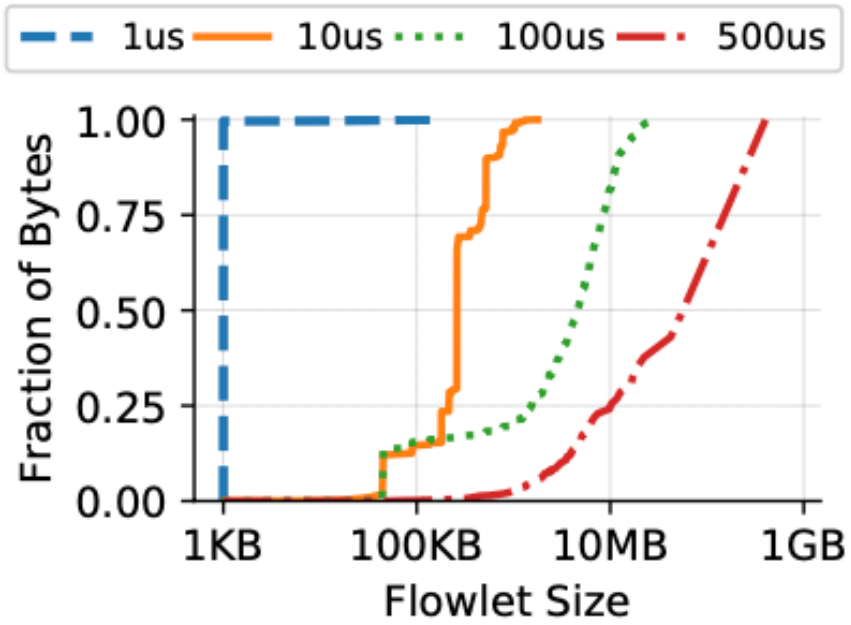


RDMA is highly sensitive out-of-order packets

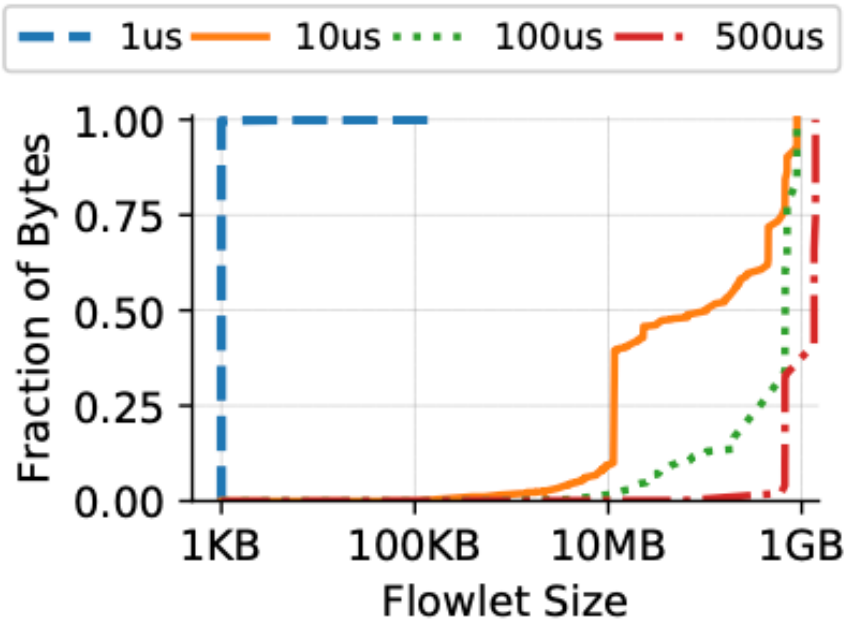
Introduction

Per-flowlet switching

- Conga, Letflow, PLB
- Rerouting avoids out-of-order arrivals via timegaps



(a) TCP



(b) RDMA

Hard to find time gaps in RDMA

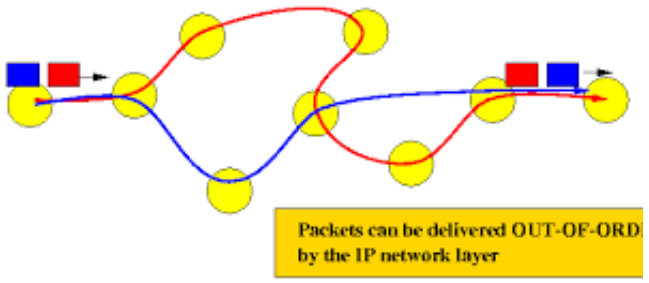
Introduction

Motivation of Conweave

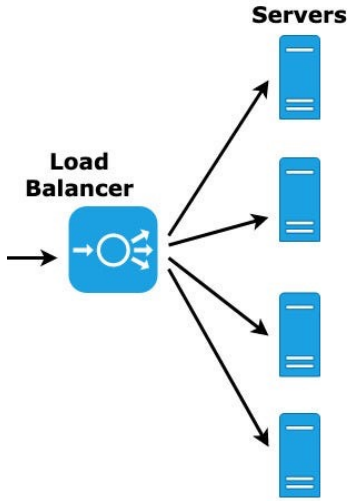
- RDMA, which is sensitive to large flowlets and packet order

  → Does not match ECMP, the existing load balancing protocol!

Intel Tofino2

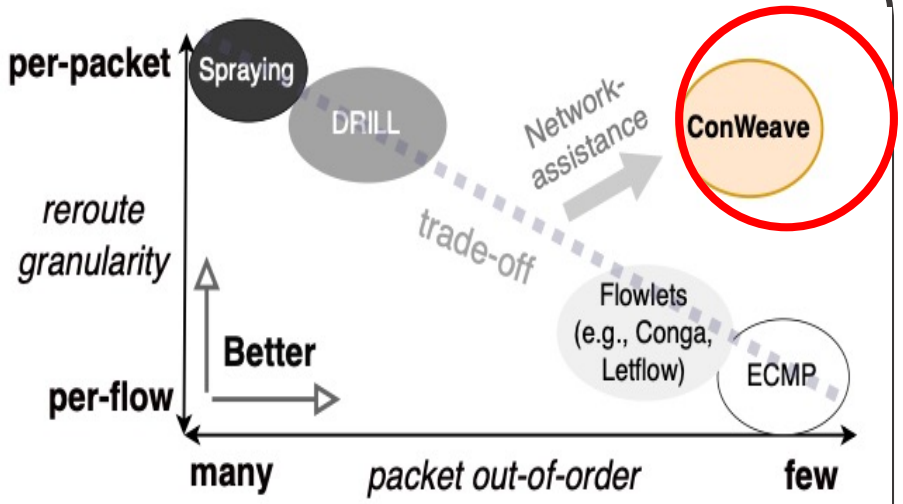


Not out-of-order packet



Effective Load Balancing

=



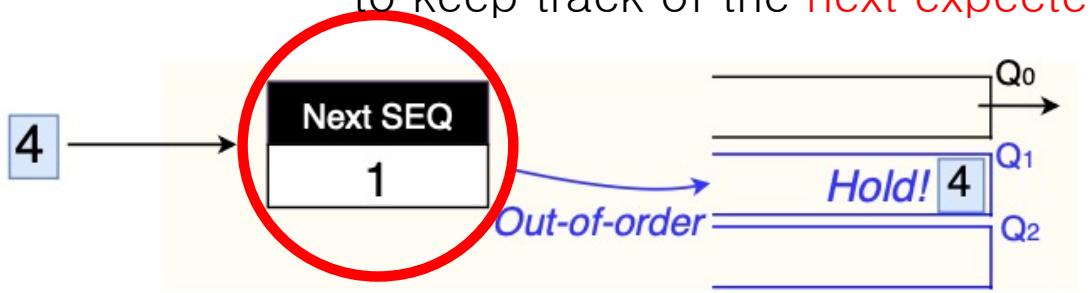
02.
Reordering out-of-order packets

Reordering out-of-order packets

How do we correct out-of-order packet in network?

- Buffering and releasing the received packets

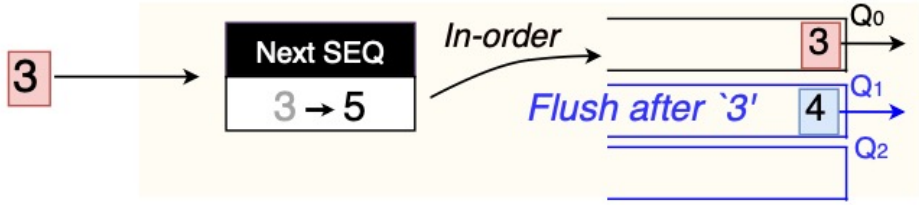
to keep track of the next expected packet sequence number



(a) Arrivals of packet 4, hold in Q_1.

(b) Arrival of packet 2, hold in Q_2.

(c) Arrival of packet 1, release 2.



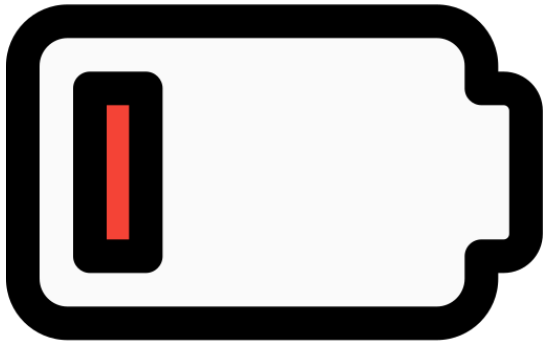
(d) Arrival of packet 3, release 4.

Figure 6: An illustration of the packet reordering mechanism

Reordering out-of-order packets

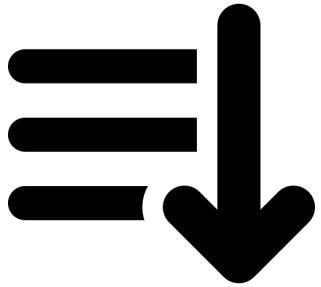
Practical Considerations

- While the above sorting mechanism is logically simple, there are a couple of practical issues.



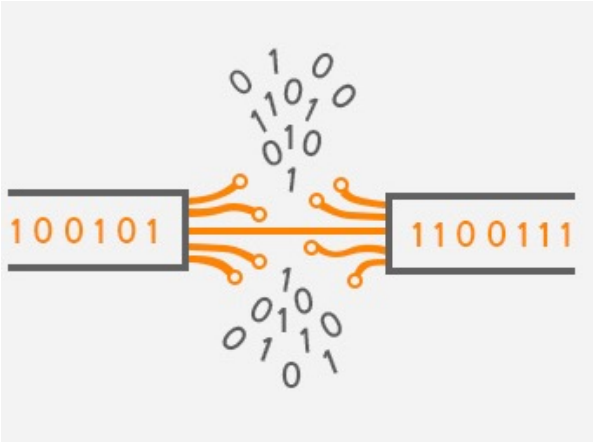
Limited Queue Resource

- Bounded by the number of queue available!
- Holding N out-of-order packet = require N queue in worst



Lack of sorting primitive

- Do not use packet recirculation, sorted queue(PIFO)
- A restricted mechanism but yet fulfills the packet reordering requirement is needed.



Dealing with packet loss

- Performing this task in the data plane with limited resources makes the task even more challenging.

03.
ConWeave

ConWeave

a new load-balancing framework

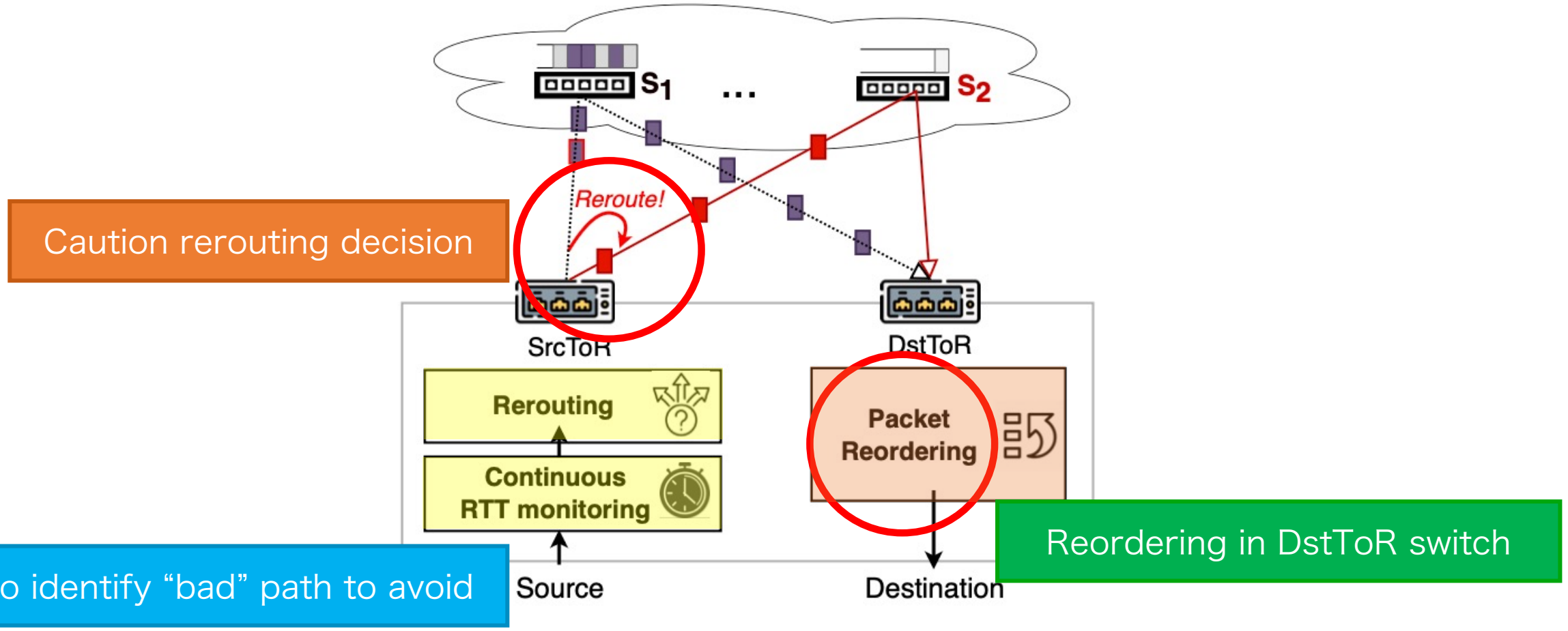
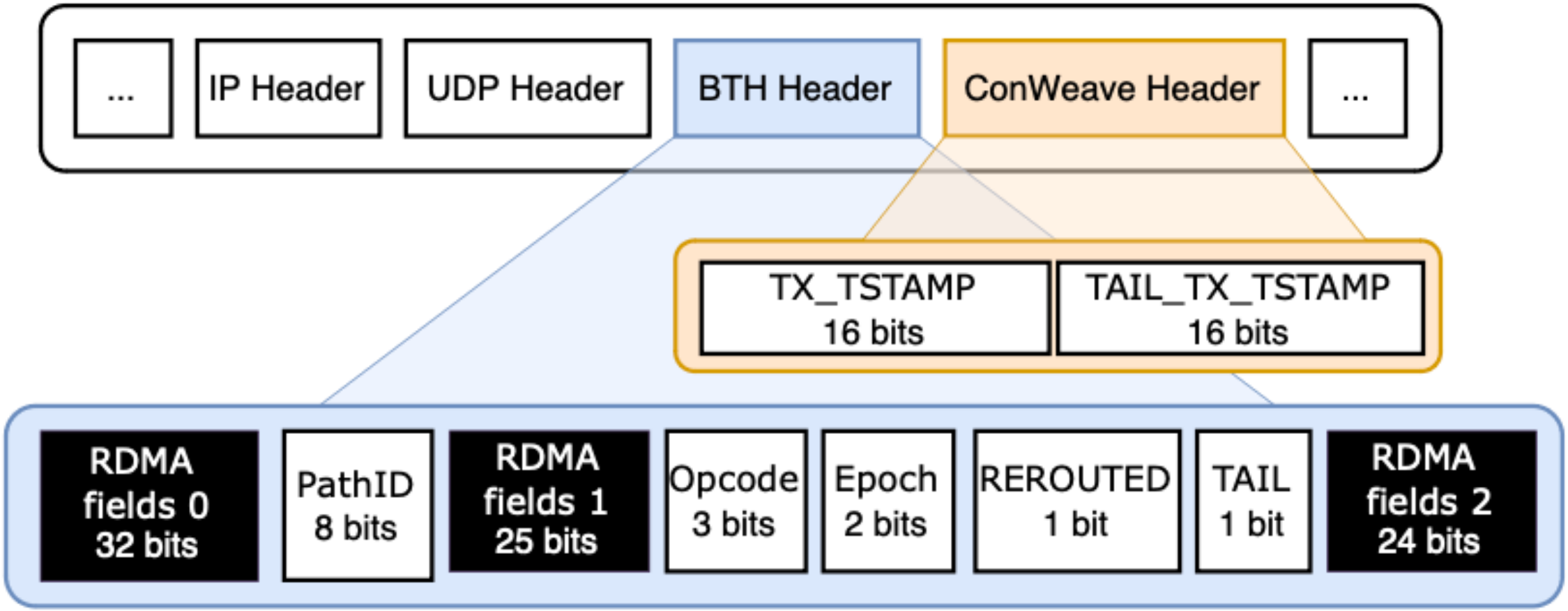


Figure 7: Overview of ConWeave. Only the ToR switches are required to be programmable.

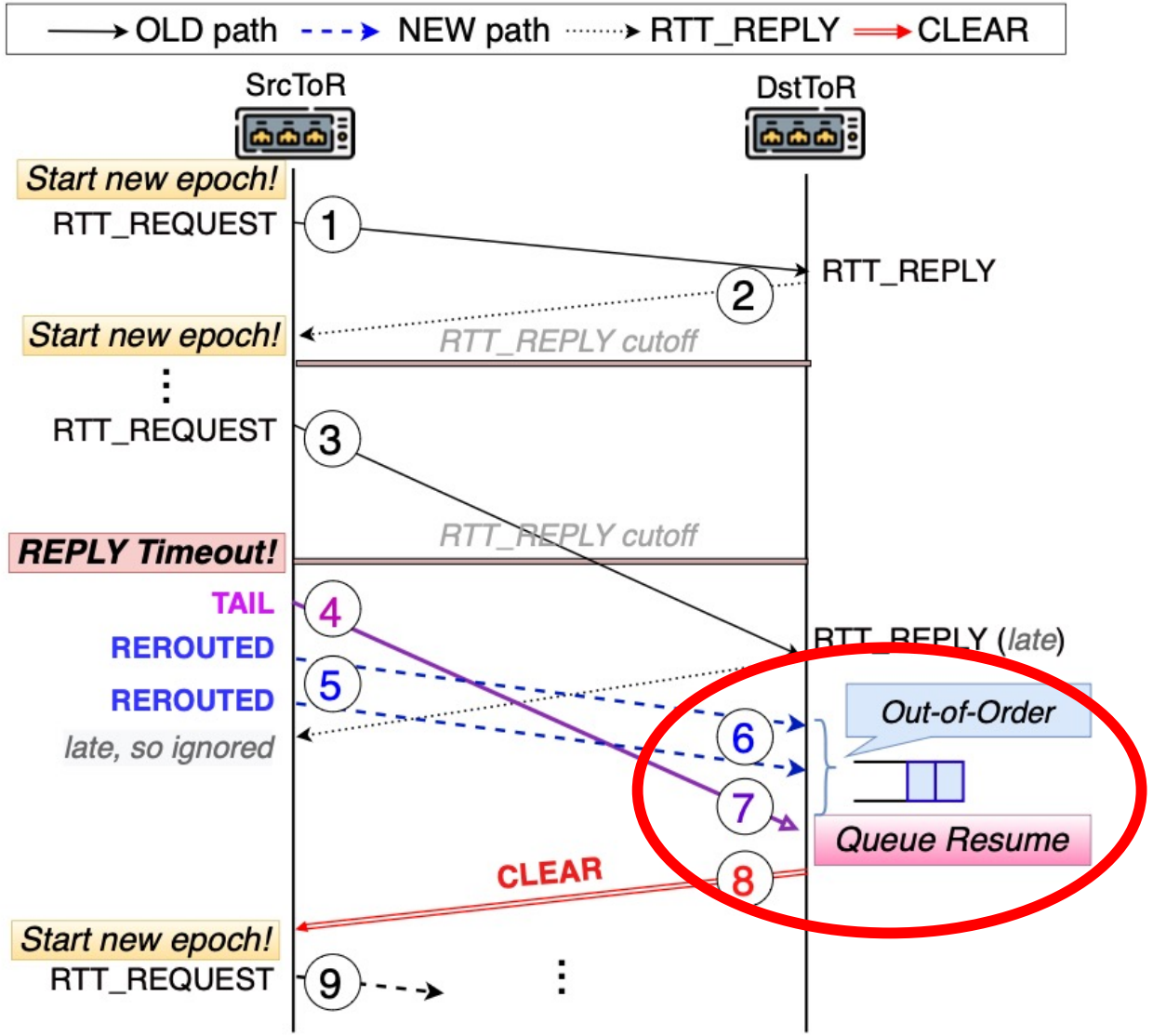
ConWeave

Implementation



ConWeave

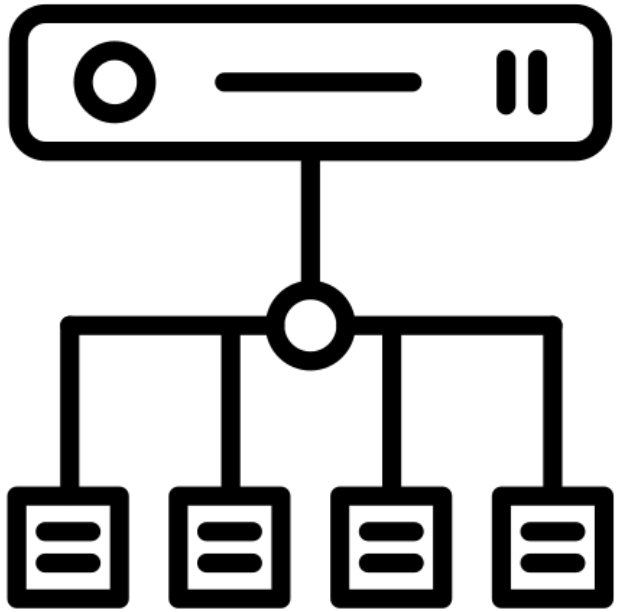
the example of a flow arrival and its rerouting



ConWeave

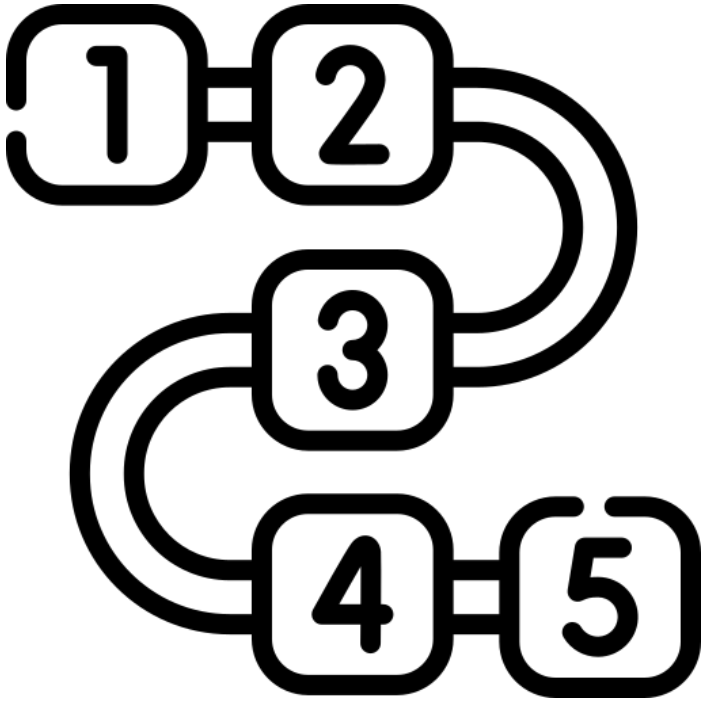
"Cautious" Rerouting Decisions

Goal : Producing predictable packet arrival patterns



Fine-grained traffic routing

=



out-of-order in unpredictable patterns

ConWeave

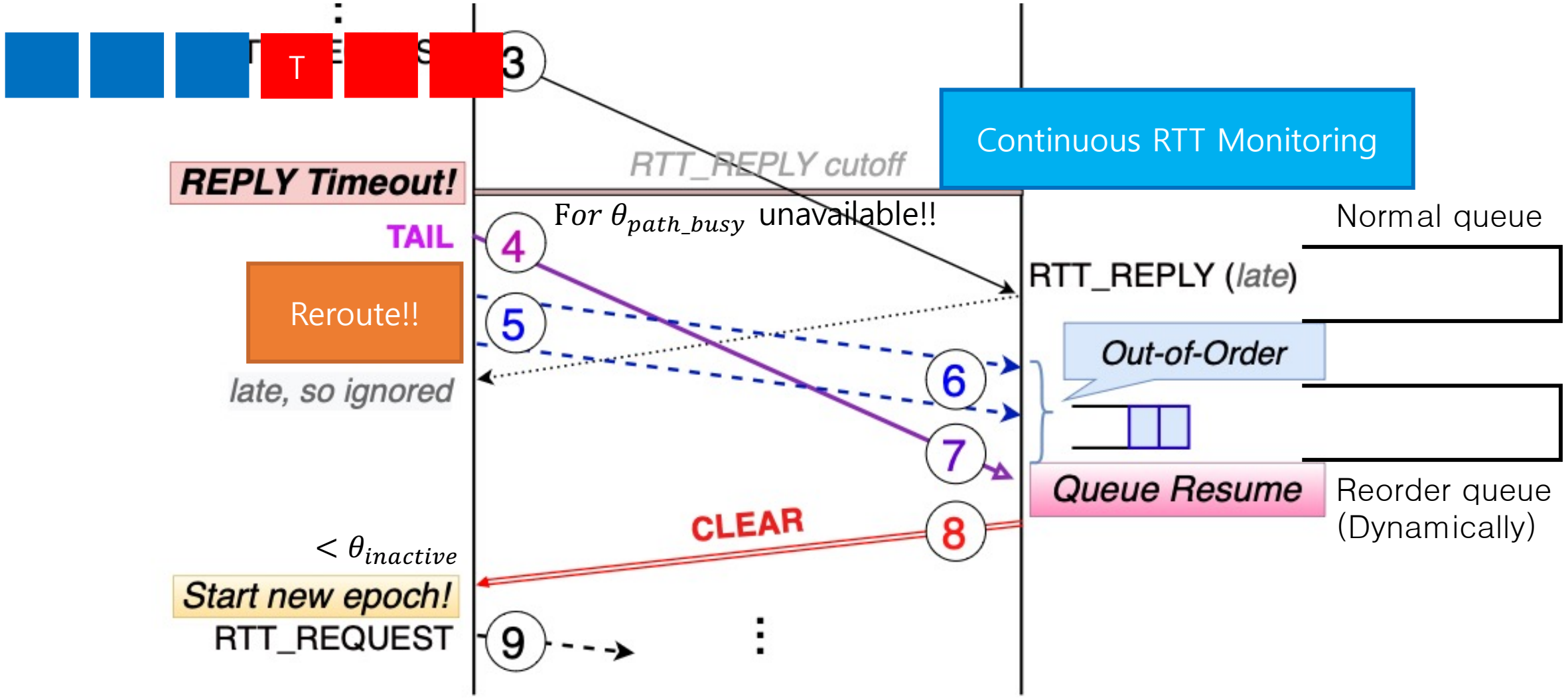
"Cautious" Rerouting Decisions

- Rerouting is needs and a good alternative path is available
  - The existing path is congested
  - There exists a viable path that is not congested

- To produce predictable arrival patterns that any flow can only have in-flight packets in at most two paths at any instance of time.
  - Any out-of-packets caused by previous reroutes have been received at the destination ToR

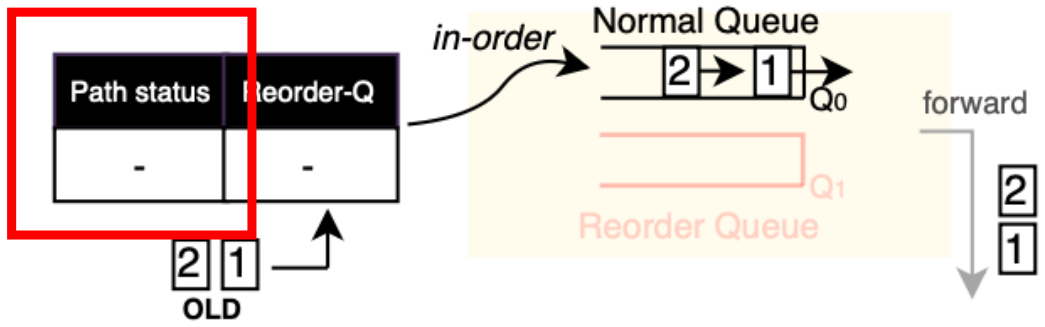
ConWeave

"Cautious" Rerouting Decisions

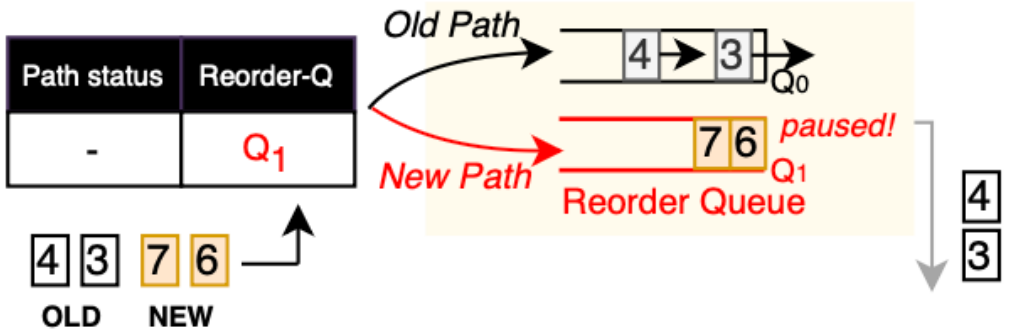


ConWeave

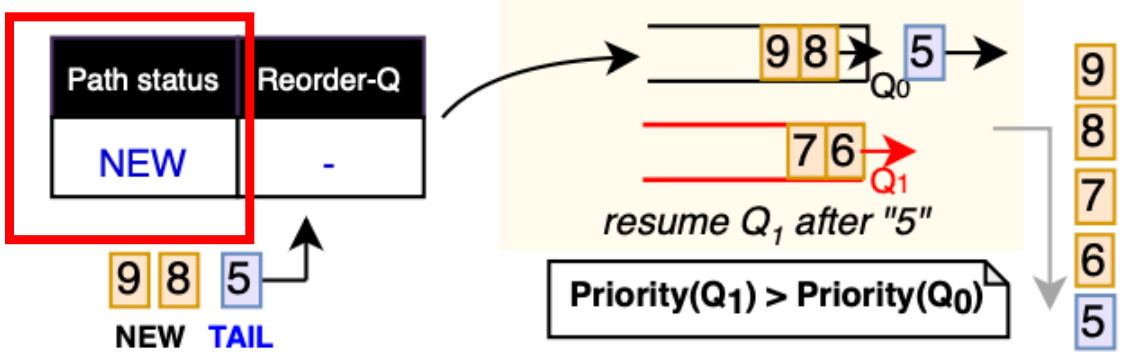
Masking packet reordering



(a) Forwarding in-order packets to a normal queue.



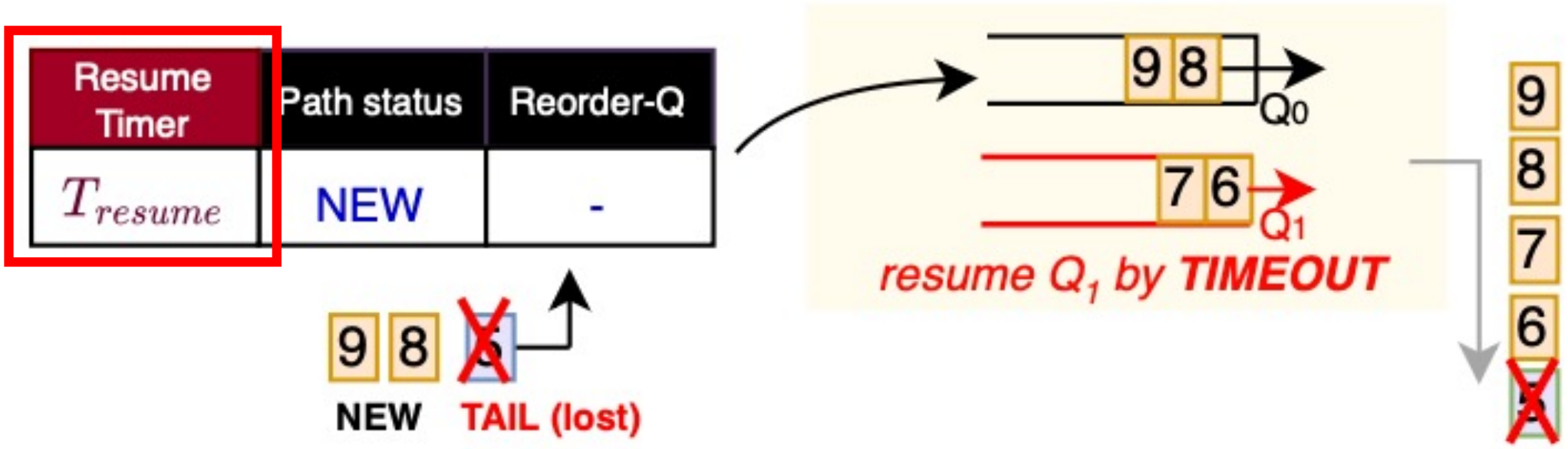
(b) Start enqueuing the first out-of-order packet.



(c) Resume reorder queue with priority queues.

ConWeave

Masking packet reordering – loss TAIL packet

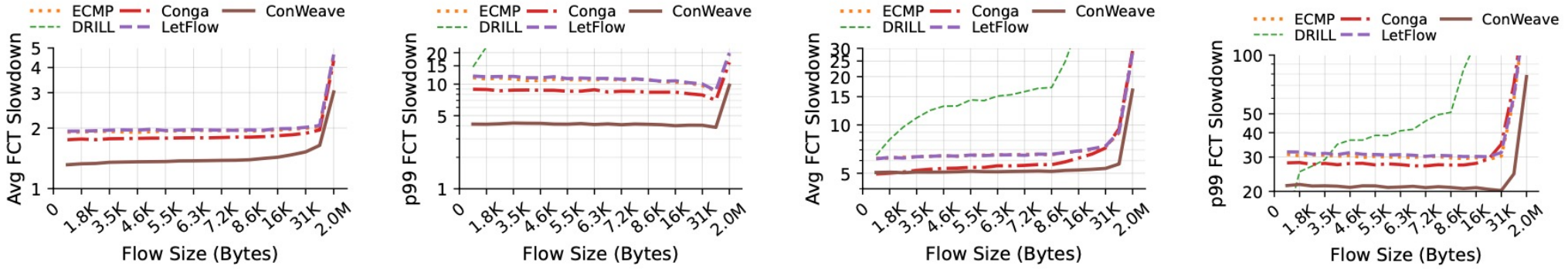


(d) In case of TAIL loss, timer triggers resume.

04.
Evaluation

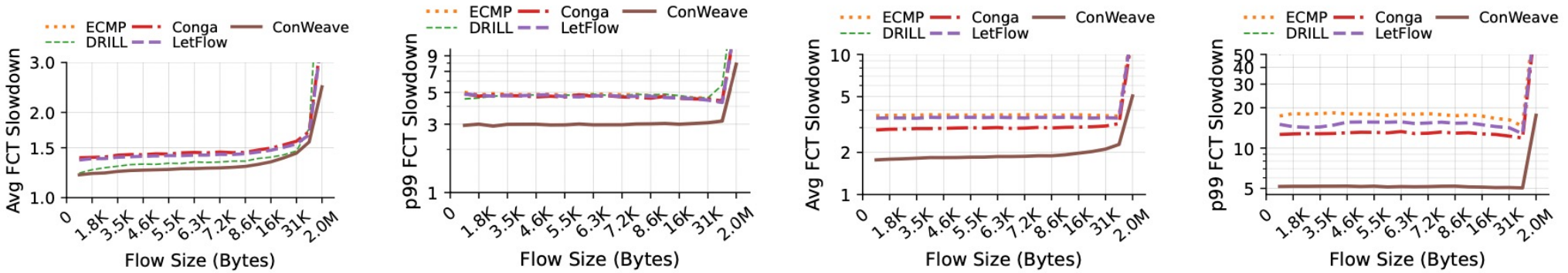
Evaluation

Reduction of FCT(Flow Completion Time)



(a) 50% Avg.Load (avg) (b) 50% Avg.Load (99%-ile) (c) 80% Avg.Load (avg) (d) 80% Avg.Load (99%-ile)

Figure 12: Avg. and tail FCT slowdown for *AliStorage* in *Lossless RDMA* (50% and 80% Avg.Load).



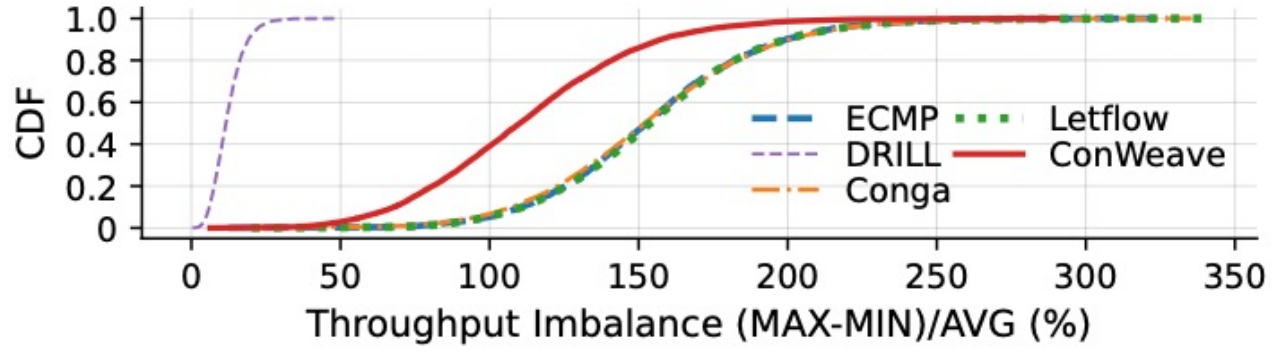
(a) 50% Avg.Load (avg) (b) 50% Avg.Load (99%-ile) (c) 80% Avg.Load (avg) (d) 80% Avg.Load (99%-ile)

Figure 13: Avg. and tail FCT slowdown for *AliStorage* in *IRN RDMA* (50% and 80% Avg.Load).

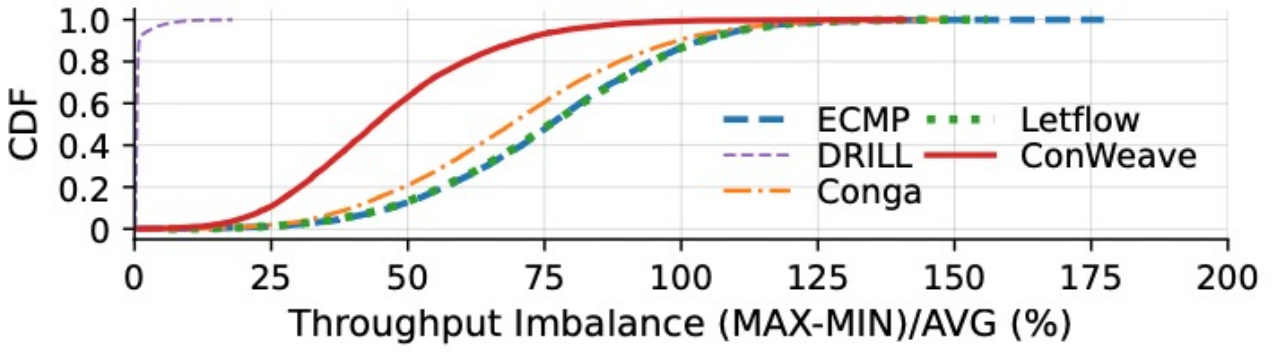
Evaluation

Load Balancing efficiency

- $the\ throughput\ imbalance = (the\ maxium\ throughput - the\ minimum\ throughput)/average$



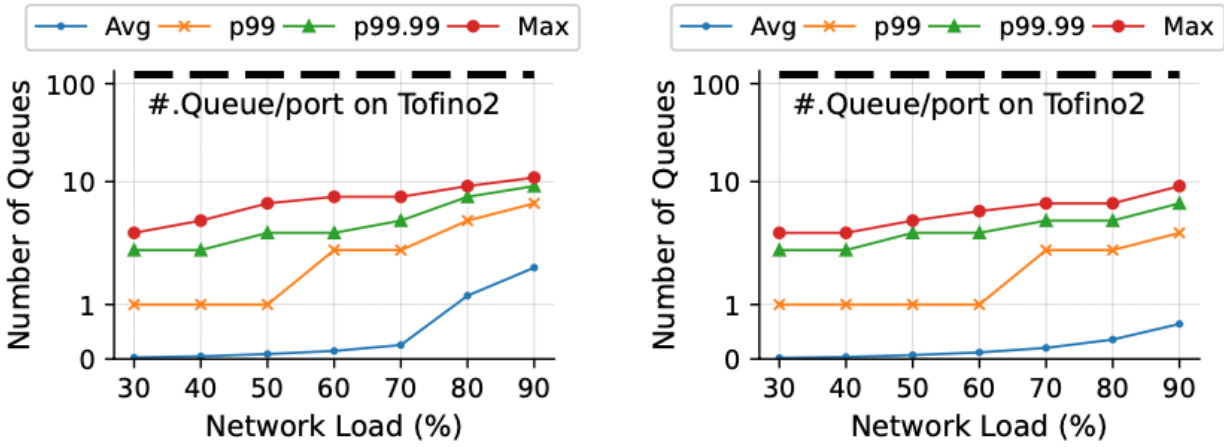
(a) 50% Avg.Load



(b) 80% Avg.Load

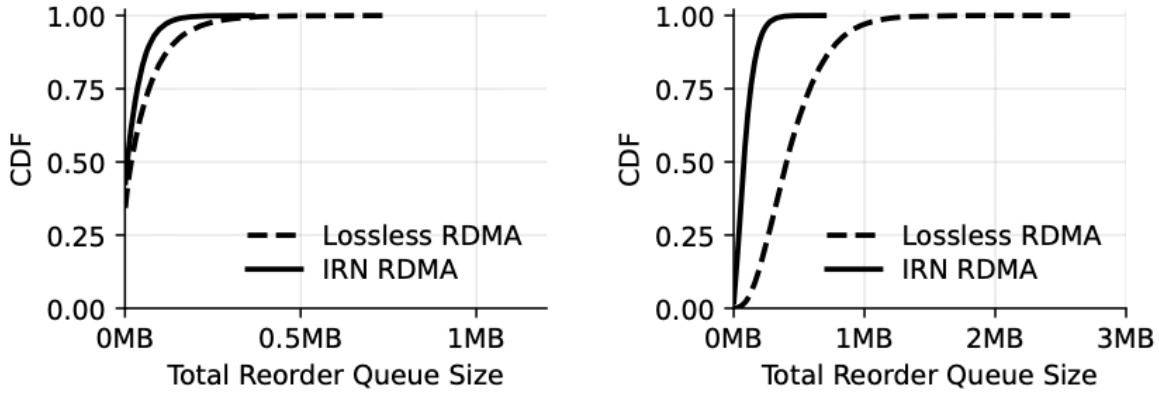
Evaluation

Hardware resource consumption



(a) *Lossless RDMA* (b) *IRN RDMA*

Figure 15: Number of queues usage per egress port. Note that the y-axis is in log scale while the dashed line on the top refers to the number of queues available per port.



(a) 50% Avg.Load (b) 80% Avg.Load

Figure 16: Total queue memory overhead per switch.

Evaluation

Three-tier topology

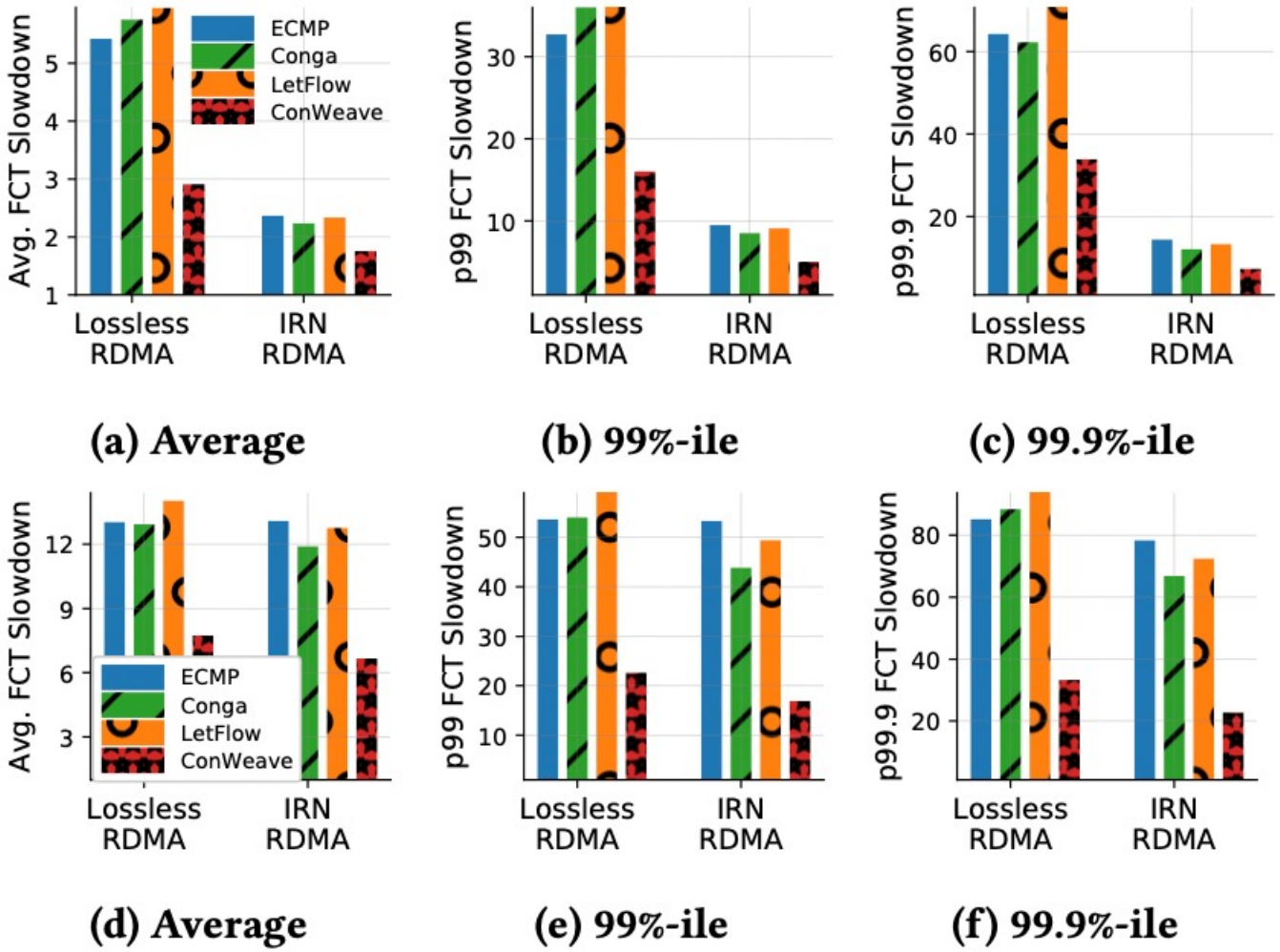


Figure 17: FCT slowdowns for short (a-c) and long (d-f) messages for fat-tree topology.

Thank you