RDMA over Commodity Ethernet at Scale

Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni,
Jianxi Ye, Jitendra Padhye, Marina Lipshteyn

Microsoft

SIGCOMM 2016 Experience Track

Chuanxiong Guo is the **first researcher** from China who published the 1st SIGCOMM paper in 2001!

Outline

- **Introduction to RDMA (Necessary Background)**
- RDMA at Microsoft (Explain the Paper)
- Takeaway (Some Thoughts)

What is RDMA?

- Before answering this question, let's take a look at what indeed is the latency within a DCN.
- We can use a Linux tool called *ping* to measure the RTT (round trip time)
- We connect two servers using the cable (showed to you in the 1st lecture, ~20m) via a 400Gbps switch

What's the latency?

- Propagation delay:
  - distance/speed = 2(hop) * 2 (round trip)* 20m/2.0*10^8 = 200ns
- Transmission delay:
  - packet size/BW = 2* 1500B * 8 / 400Gbps = 60ns
- Queueing delay: 0
  - 2 servers lead to no congestion
- Processing delay: Should be small, let's leave it as 0 right now…
- RTT should be < 1 microsecond

Hey, it's Not Right…

```
zhaoyu@node1:~$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.061 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.044 ms
64 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=0.038 ms
64 bytes from 10.0.1.2: icmp_seq=5 ttl=64 time=0.040 ms
64 bytes from 10.0.1.2: icmp_seq=6 ttl=64 time=0.042 ms
64 bytes from 10.0.1.2: icmp_seq=7 ttl=64 time=0.042 ms
64 bytes from 10.0.1.2: icmp_seq=8 ttl=64 time=0.042 ms
64 bytes from 10.0.1.2: icmp_seq=9 ttl=64 time=0.044 ms
64 bytes from 10.0.1.2: icmp_seq=10 ttl=64 time=0.039 ms
64 bytes from 10.0.1.2: icmp_seq=11 ttl=64 time=0.051 ms
^C
--- 10.0.1.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10278ms
rtt min/avg/max/mdev = 0.038/0.047/0.079/0.011 ms
```

Behind the *ping*

Application

↓

Socket API

↓

TCP/IP Stack

↓

Kernel

↓

NIC Driver

↓

NIC

↓

Network

- Multiple kernel crossings
- Interrupts
- TCP processing
- Memory copies

Latency Breakdown

Component	Cost
System call	~100 ns
Kernel networking stack	10-20 µs
Interrupt handling	~10 µs
Context switches	~10 µs

Then What About Throughput?

iperf: Send large message using TCP/IP

```
zhaoyu@node2:~$ iperf -c 10.0.1.3
------------------------------------------------------------
Client connecting to 10.0.1.3, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  1] local 10.0.1.2 port 43932 connected with 10.0.1.3 port 5001 (icwnd/mss/irtt=89/9164/204)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0077 sec  53.8 GBytes  46.2 Gbits/sec
```

Hey, our cable and switch are all 400Gbps, but why only 42Gbps…

CPU Bottleneck

```
----total-usage----
usr sys idl wai stl
  0   0 100   0   0
  0   0 100   0   0
  0   1  98   0   0
  0   1  98   0   0
  0   1  98   0   0
  0   1  98   0   0
  0   1  98   0   0
  0   1  98   0   0
  0   1  99   0   0
  0   1  98   0   0
  0   1  99   0   0
  0   1  99   0   0
  0   0 100   0   0
  0   0 100   0   0
  0   0 100   0   0
  0   0 100   0   0
  0   0 100   0   0^C
```

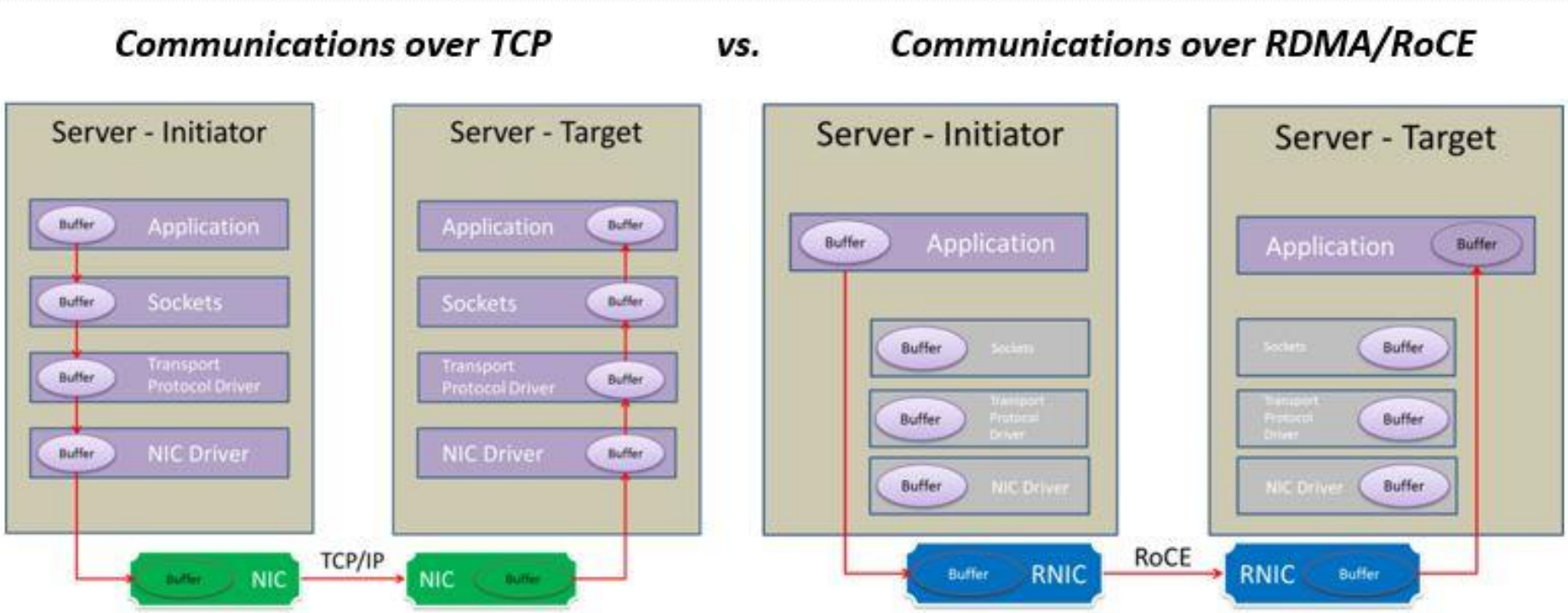
Problem with TCP/IP

- TCP/IP stack adds a **negligible latency** to DCN
  - **~50-100 microseconds**
  - Internet does not suffer from this problem
    - Propagation dominates the latency
    - Constrained by the speed of light
    - Beijing to Los Angeles: 100 milliseconds
- **CPU becomes a bottleneck** in DCN where NIC available **bandwidth** reaches 100-400Gbps

RDMA

- Remote Direct Memory Access
- **The idea is bypass the Kernel**
  - **Low latency**
  - **CPU is not a bottleneck -> High bandwidth**
- **Require advanced devices**
  - Network Interface Card (NIC)
  - Switch (depends on implementation)

RDMA


Communications over TCP
vs.
Communications over RDMA/RoCE
Server - Initiator
Server - Target
Application
Sockets
Transport Protocol Driver
NIC Driver
Buffer
NIC
TCP/IP
Server - Initiator
Server - Target
Application
Buffer
RNIC
RoCE

Low Latency/High Throughput

```
zhaoyu@node2:~$ ib_write_lat 10.0.1.3
 TX depth        : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Max inline data : 220[B]
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x0146 PSN 0x992a03 RKey 0x1ffbb9 VAddr 0x0055817a635000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x01f9 PSN 0xc8ac66 RKey 0x1ff8b6 VAddr 0x005979da804000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:03
---------------------------------------------------------------------------------------
 #bytes #iterations    t_min[usec]    t_max[usec]  t_typical[usec]    t_avg[usec]    t_stdev[usec]   99% percentile[usec]   99.9% p
ercentile[usec]
Conflicting CPU frequency values detected: 800.000000 != 3500.035000. CPU Frequency is not max.
Conflicting CPU frequency values detected: 800.000000 != 3499.964000. CPU Frequency is not max.
 2       1000          2.33           9.50         2.36                2.37             0.13            2.47                 9.50

---------------------------------------------------------------------------------------
```

Low Latency/High Throughput

```
zhaoyu@node1:~$ ib_write_bw  -s 4000000 --report_gbits -t 60 -q 32
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x013e PSN 0x821c93 RKey 0x203dbc VAddr 0x00727bd06b7800
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x013f PSN 0x557062 RKey 0x203dbc VAddr 0x00727bd0a88100
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0140 PSN 0x10dddb RKey 0x203dbc VAddr 0x00727bd0e58a00
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0141 PSN 0x730bc1 RKey 0x203dbc VAddr 0x00727bd1229300
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0142 PSN 0xe2e14c RKey 0x203dbc VAddr 0x00727bd15f9c00
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0143 PSN 0xa43c5e RKey 0x203dbc VAddr 0x00727bd19ca500
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0144 PSN 0xe9361c RKey 0x203dbc VAddr 0x00727bd1d9ae00
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
 remote address: LID 0000 QPN 0x0145 PSN 0x458db RKey 0x203dbc VAddr 0x00727bd216b700
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:00:01:02
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[Gb/sec]    BW average[Gb/sec]   MsgRate[Mpps]
 4000000    160000           391.78             391.78               0.012243
---------------------------------------------------------------------------------------
```

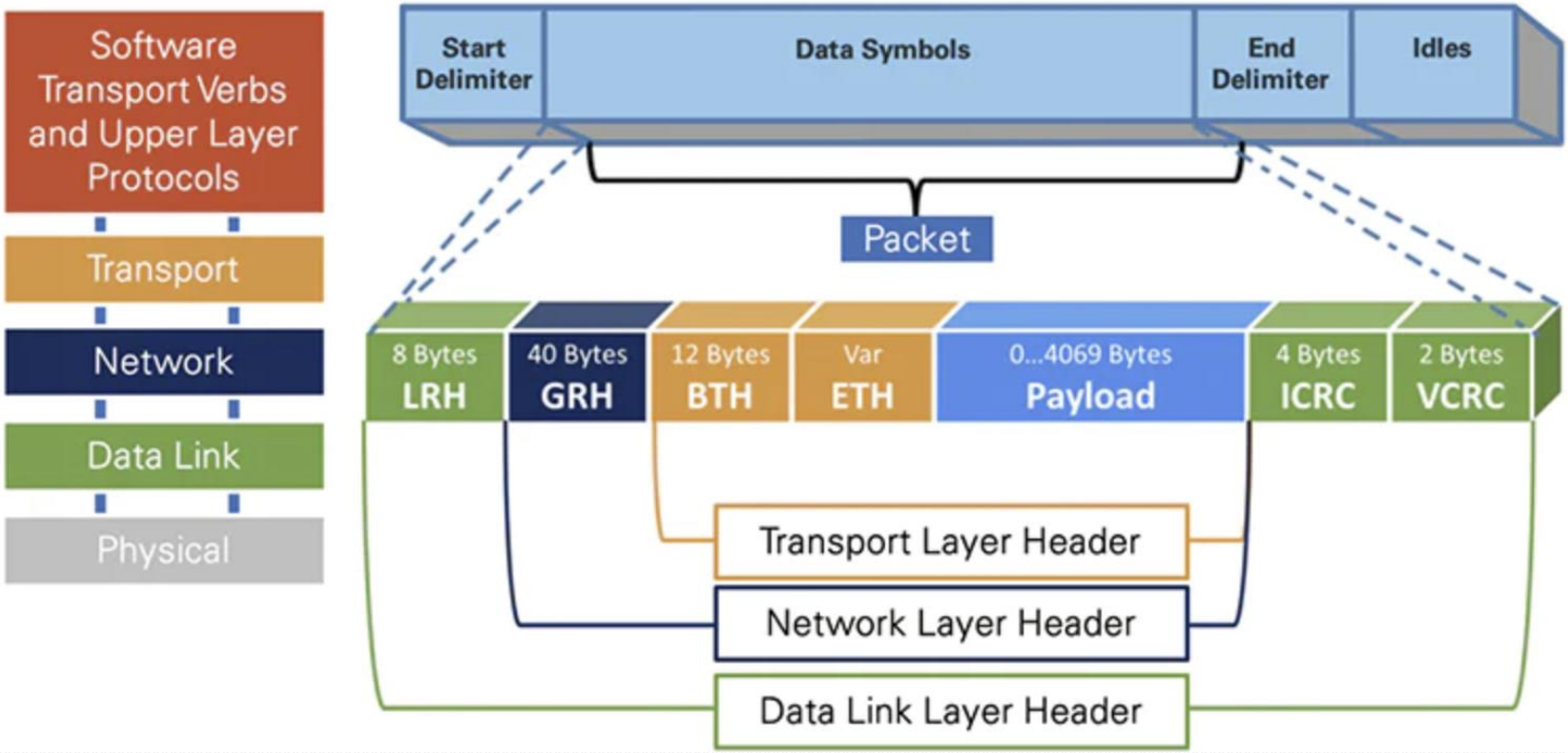
RDMA

- Several Implementations
  - InfiniBand
  - RoCE/RoCEv2
  - iWARP
- InfiniBand and RoCE/RoCEv2 are widely adopted nowadays. No one uses iWARP now.

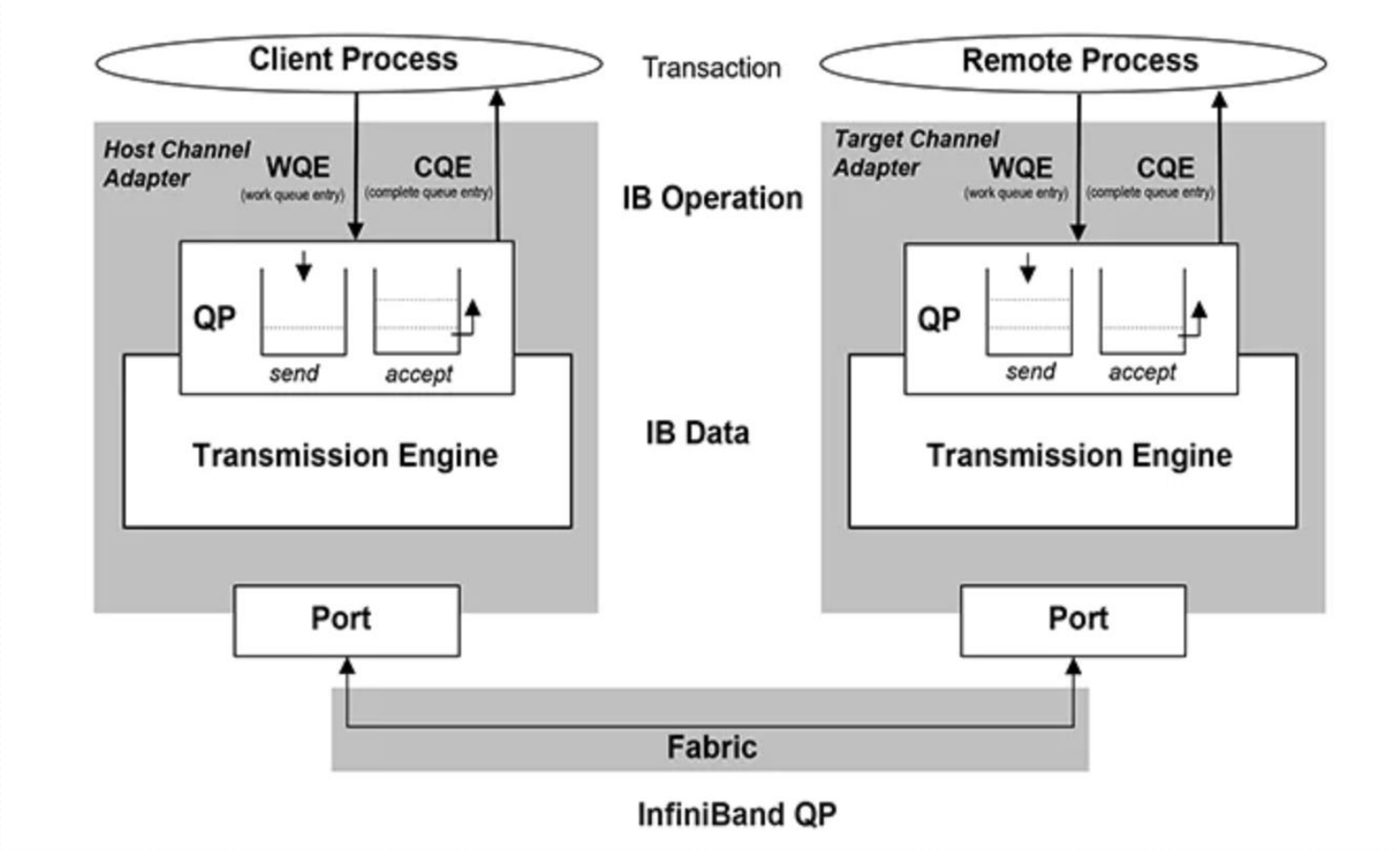
RDMA Implementation - InfiniBand

- InfiniBand is a network designed *from scratch* for RDMA
- Key component:
  - Host Channel Adapter (HCA or RDMA NIC)
  - Switch
  - Fabric
- Advantage:
  - **Easy to configure**
  - Best performance (latency/throughput)

RDMA Implementation - InfiniBand



InfiniBand Programming Model



Supported Connection Type

- RC: Reliable Connection
- UC: Unreliable Connection
- UD: Unreliable Datagram

Supported Operations

- SEND/RECEIVE
- WRITE (remote CPU is not involved)
- READ (remote CPU is not involved)

Programming RDMA

- You can use the following links to learn more about how to do RDMA programming/libverbs
  - https://docs.nvidia.com/doca/archive/doca-v2.2.0/rdma-programming-guide/index.html
  - https://www.rdmamojo.com/

Drawback of InfiniBand

- Require **special** InfiniBand switches and fabric
  - Not compatible with existing DCN infrastructure: **Ethernet**
  - High deployment cost
- **Single Vendor**
  - **Mellanox/NVIDIA**

Microsoft does not use InfiniBand at least in its Azure DCN

RDMA Implementation - RoCE

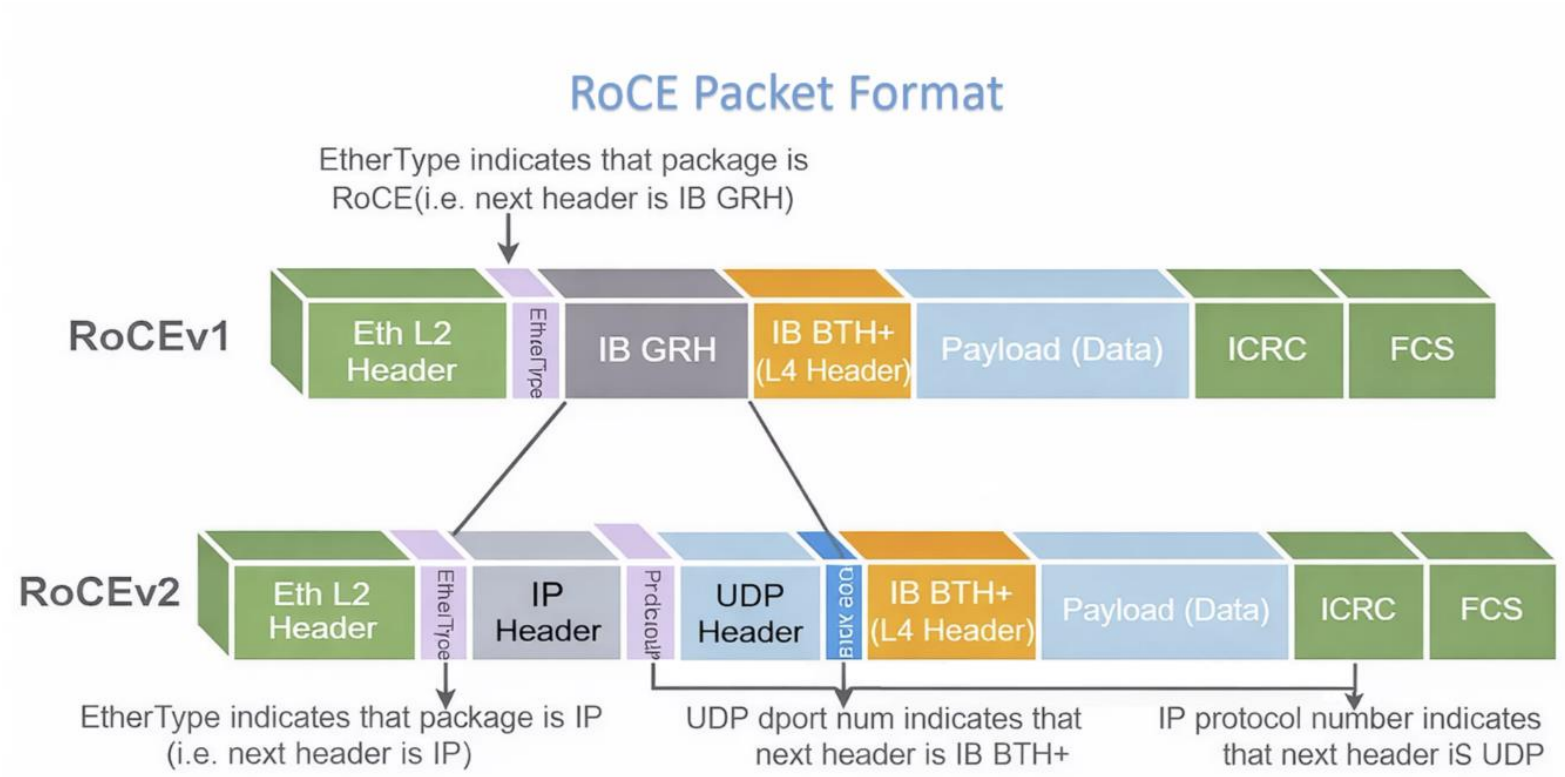
- **RDMA over Converged Ethernet**
- Why converged Ethernet?
  - The Ethernet has to be **lossless** (Why?)
- The core idea is to run InfiniBand RDMA semantics over Ethernet

RDMA Implementation - RoCE

- Advantages:
  - InfiniBand switch/fabrics -> **Ethernet switch/fabrics**
- Disadvantages:
  - Scalability issues as **pure L2** cannot support routing across subnets

RDMA Implementation – RoCEv2

- **Further use IP/UDP header**



RDMA Implementation – RoCEv2

- Advantages of RoCEv2:
  - L3 support
  - Cross subnet or even cross DC
- Why UDP header?
  - Commodity switches can only observe L2/L3/Transport header
    - Flow classification
    - **ECMP**

Why RDMA needs lossless?

- Ethernet is **lossy** by design
- TCP handles it via:
  - Retransmission
    - Complicated logic implemented in the kernel networking stack
    - Timeout may occur
- There exists lossy RDMA nowadays, but let's focus on lossless RDMA

Why RDMA needs lossless?

- Can RDMA NIC use the same algorithms in TCP?
- No.
  - RDMA NIC **has limited resources**, which is challenging to support advanced retransmission algorithms: such as SACK.
  - Go back 0/N cause the entire window to stall
  - Even small stall or timeout will cause **latency spike** (RDMA achieves very low latency)

Priority Flow Control

- Avoid packet drops inside the network
- Can congestion control, such as DCTCP, solve the problem?
  - No. congestion control is end-to-end, too slow for some cases, such as incast.
- We advocate a hop-to-hop solution.
  - **Pause the sender of one hop before buffers overflow**

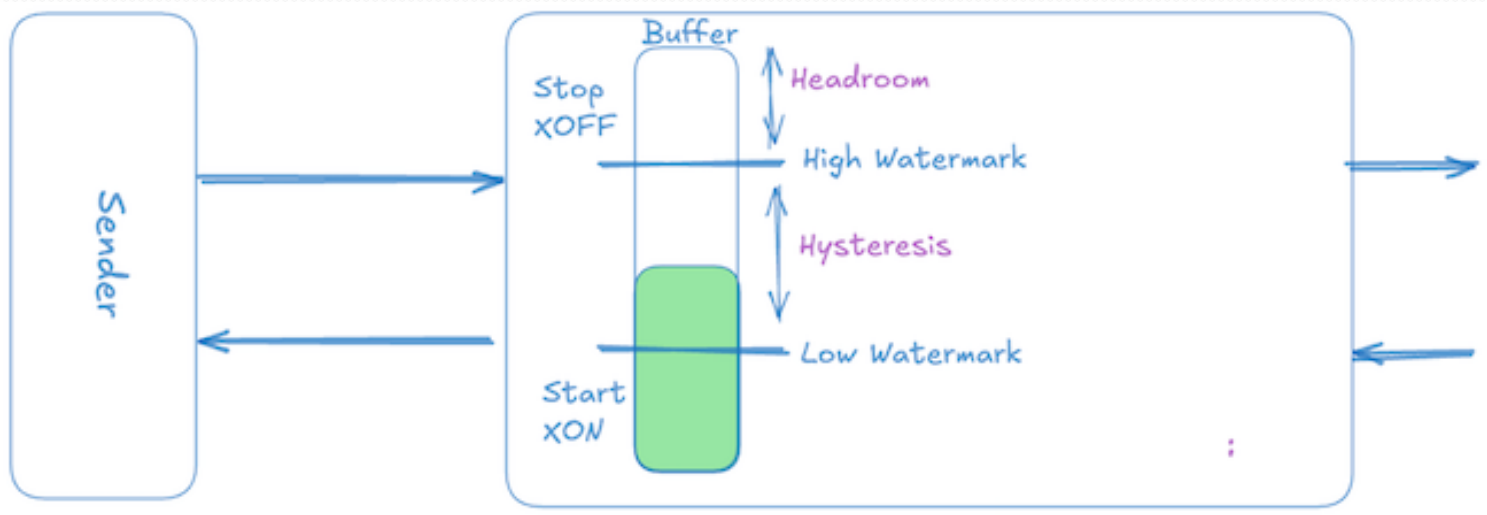
Priority Flow Control

- **Workflow of PFC**

  Receiver buffer almost full
  ↓
  Send PFC pause frame: XOFF
  ↓
  Sender stops transmitting

- **Key Properties:**
  - **L2 protocol:** Leverage VLAN PCP for priority
  - **Per-priority/queue** (not global pause)
  - **Hop-by-hop**
  - **Prevents packet drops due to congestion**
  - Very **complicated** to configure correctly

Priority Flow Control



XOFF: Send pause frame

XON: Send resume frame

Needs headroom buffer: pause frame requires time to arrive at the upper sender

PFC prevents loss but propagates congestion

Outline

- Introduction to RDMA (Necessary Background)
- **RDMA at Microsoft (Explain the Paper)**
- Takeaway (Some Thoughts)

Why Microsoft use RoCEv2?

- The advantages of RoCEv2
  - High performance
  - Low CPU overhead (Good for cloud providers)
  - Compatible with existing DCN infrastructure (Low deployment cost)
  - No risks of single vendor lock (Good for big companies)

The Contribution of the Paper

- New mechanism: DSCP-based PFC
- Safety Challenges
  - RDMA transport livelock
  - PFC Deadlock
  - NIC PFC pause frame storm
  - The Slow-receiver symptom

The Contribution of the Paper

- RDMA In Production
  - Configuration management and monitoring
  - PFC pause frame and traffic monitoring
  - RDMA Pingmesh
  - RDMA Performance

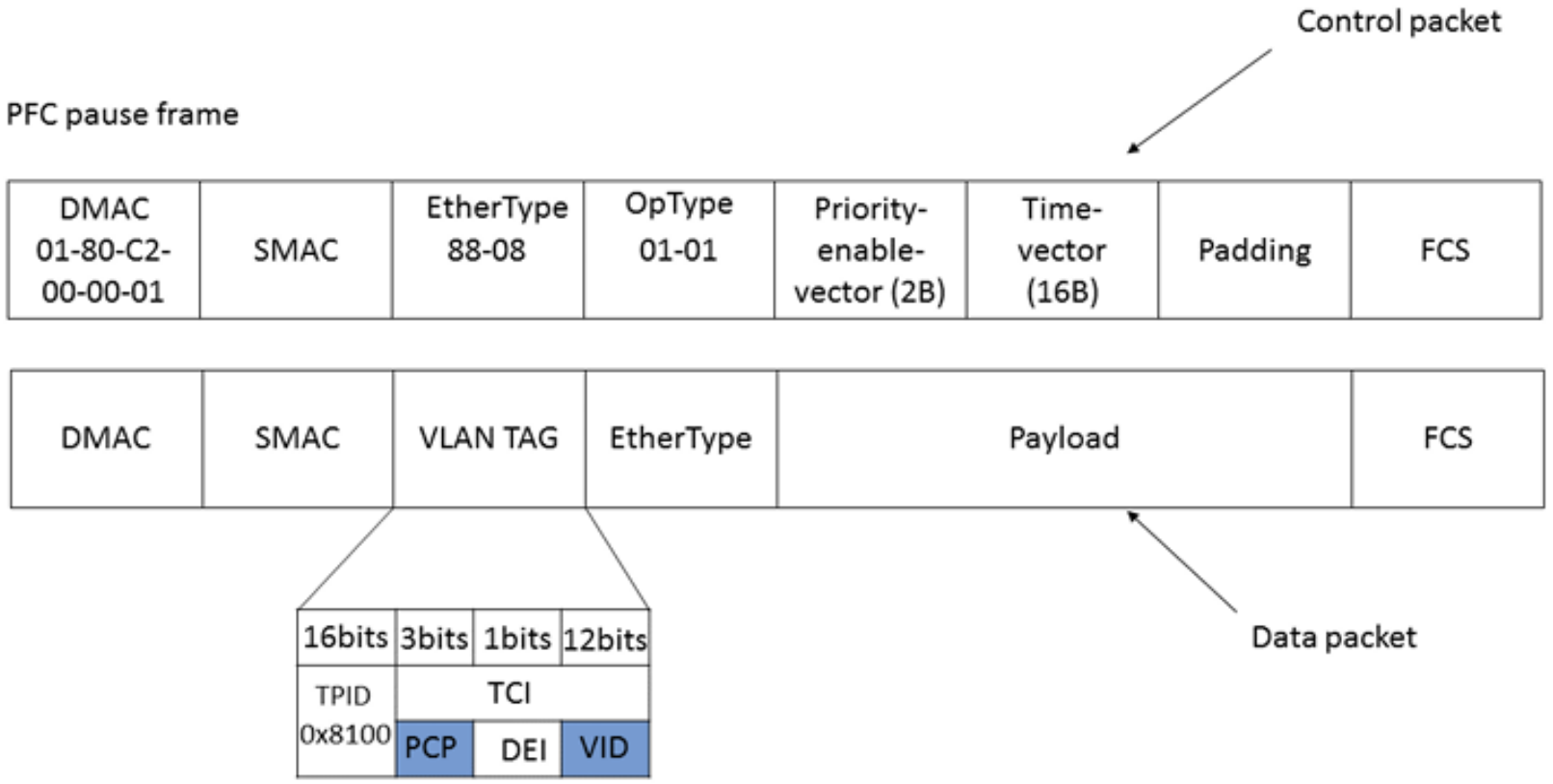
The Contribution of the Paper

- Deployment Experience
  - RDMA Deployment
  - Incidents
  - Lessons Learned and Discussion

New Mechanism: DSCP-based PFC

- VLAN-based PFC carries packet priority in the VLAN tag, which also contains VLAN ID.
- Coupling of packet priority and VLAN ID created two serious problems
  - The switch trunk mode has an undesirable inter action with our OS provisioning service
  - All switches in Microsoft use **L3** mode nowadays

VLAN-based PFC

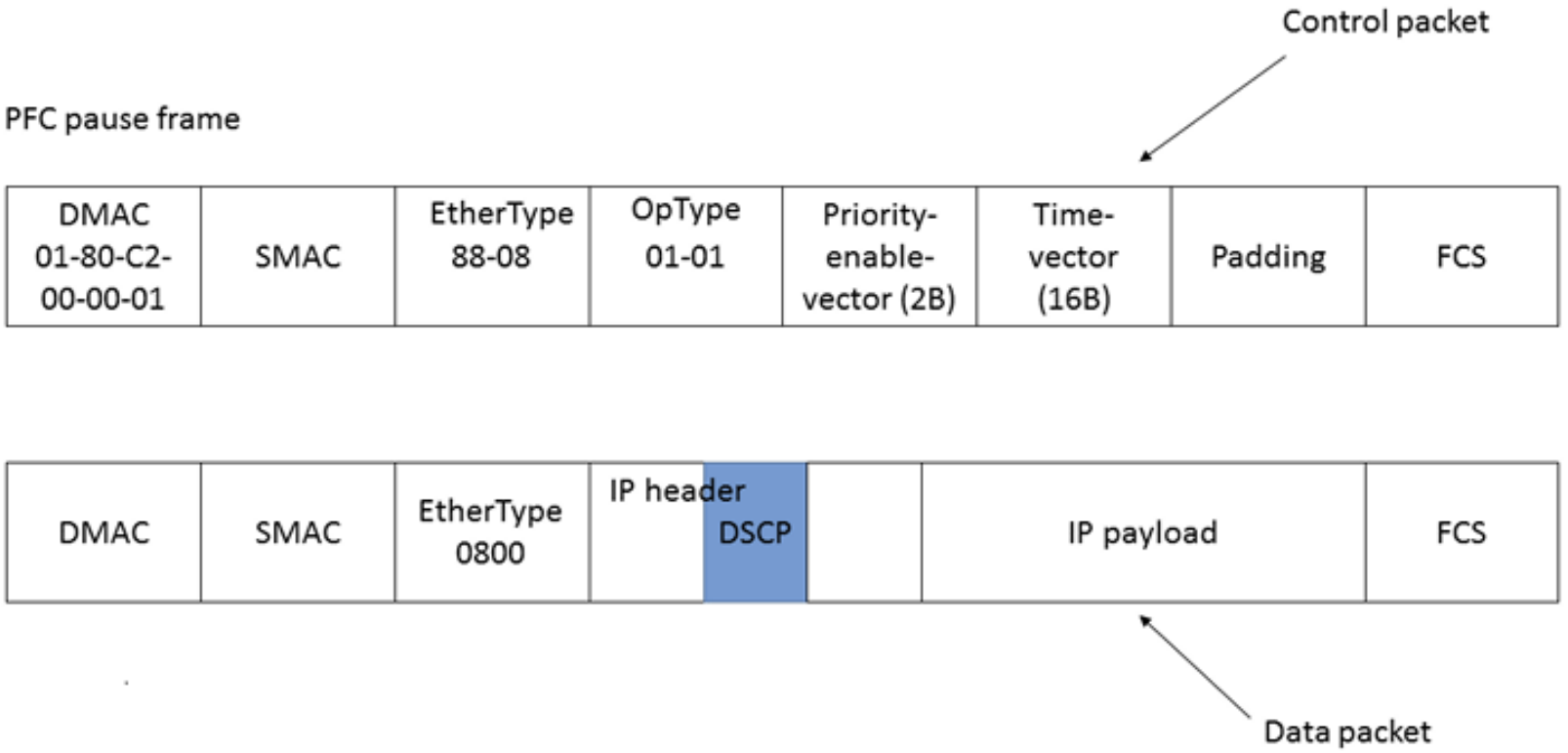


(a) VLAN-based PFC.

DSCP-based PFC

- Move the packet priority from the VLAN tag into **DSCP**

DSCP-based PFC



(b) DSCP-based PFC.

DSCP-based PFC

- Move the packet priority from the VLAN tag into **DSCP**
- DSCP-based PFC requires both **NICs** and **switches** to classify and queue packets based on the DSCP value instead of the VLAN tag
  - The switch and NIC ASICs are flexible enough to implement this (**only big company can do that…**)
  - The mapping between DSCP values and PFC priorities can be flexible and can even be many-to-one

DSCP-based PFC

- The reason behind some design choices of an experience paper is due to **some internal reasons** of a company (may not be widely recognized at that time)
- 2016 -> 2026, DSCP-based PFC has been the **de-facto standard** of current RoCEv2 deployment

Safety Challenges

- RDMA transport livelock
- PFC Deadlock
- NIC PFC pause frame storm
- The Slow-receiver symptom

RDMA Transport Livelock

- PFC ensures no packet loss due to congestion
- None congestion loss still exists (**very rare**)
  - FCS errors
  - Bugs in switches/software
- Theoretically, throughput should degrade only slightly
- However, the **goodput** drops dramatically while throughput is high (livelock)

RDMA Transport Livelock

- Micro-experiment to re-produce the problem:
  - Server A/B conncet to a switch, A uses RDMA SEND/WRITE/READ 4MB message to/from B
  - The switch is configured to drop packet at 0.4%
- Observation:
  - Throughput: full at line-rate
  - Goodput: ~0

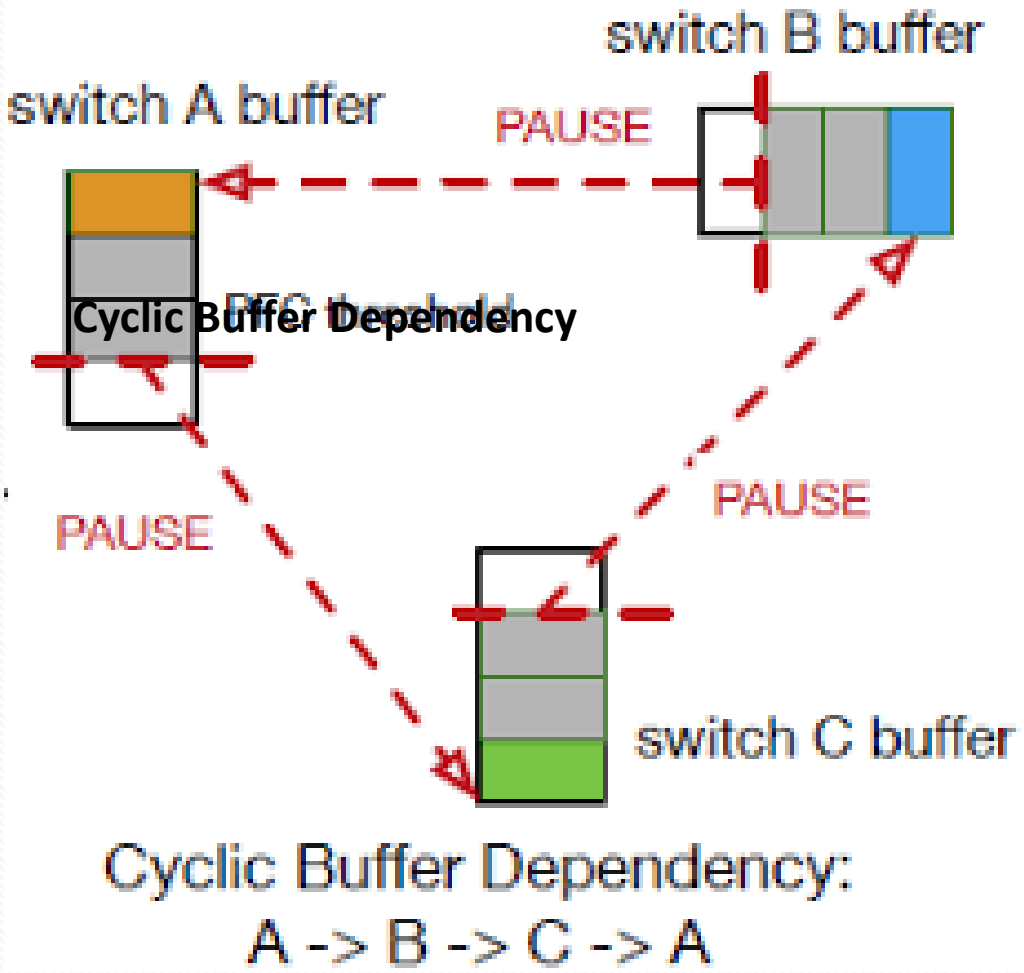
RDMA Transport Livelock

- **Root Cause:**
  - RDMA NIC (2016) used go-back-0
  - If a packet is dropped, sender retransmit the packet from 0
- **Take Away:**
  - **Packet losses can still occur even with PFC. A more sophisticated retransmission scheme is needed**
- **Solution**
  - **Replace go-back-0 with go-back-N**

PFC Deadlock

- A common belief is that CLOS topology does not cause deadlock
  - No cyclic buffer dependency

Cyclic Buffer Dependency



PFC Deadlock

- A common belief is that CLOS topology does not cause deadlock
  - No cyclic buffer dependency
- **However, Microsoft still suffers from PFC deadlock!!**
- **Reason:** PFC and Ethernet packet flooding broke the up-down routing
  - Host → ToR → Spine → ToR → Host

Ethernet Flooding

- Packets with **unknown destination MAC**, broadcast, or multicast are **flooded to all ports** (except ingress)
  - ARP table timeouts
- Flooding creates packet replication across multiple paths in the network
  - break the up-down routing
  - Host → ToR → Spine → ToR → Spine

NIC PFC Pause Frame Storm

- But PFC can cause collateral damage to innocent flows due to the head-of-line blocking
- Worst case:
  - The malfunctioning NIC of server 0 continually sends pause frames to its ToR switch
  - The ToR switch in turn pauses all the rest ports including all the upstream ports to the Leaf switches
  - The Leaf switches pause the Spine switches
  - The Spine switches pause the rest of the Leaf switches
  - The rest of the Leaf switches pause their ToR switches
  - The ToR switches pause the servers that connect to them

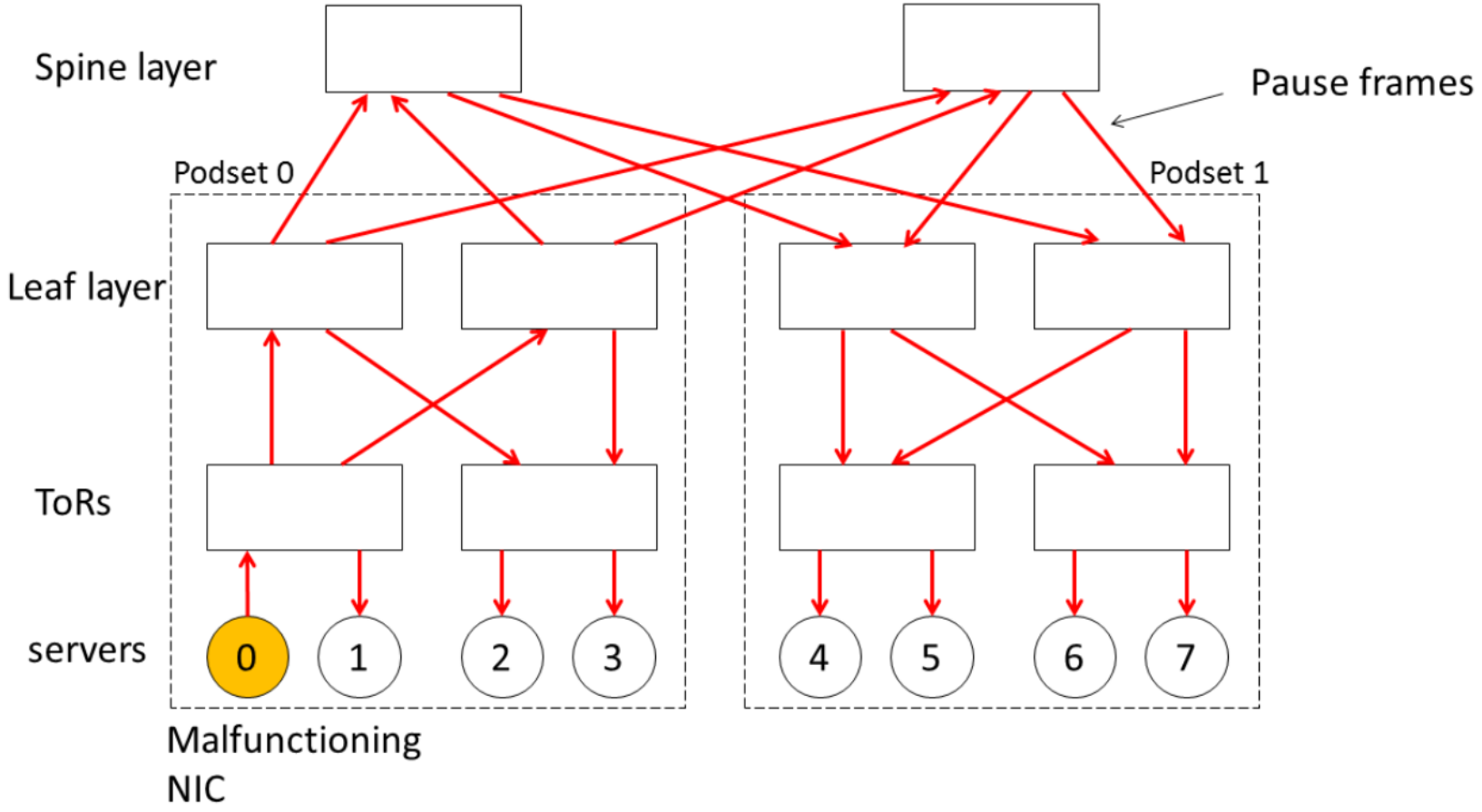


Figure 5: The PFC pause frame storm caused by the malfunctioning NIC of one single server (server 0).

NIC PFC Pause Frame Storm

- **Solution**: Broadcast and multicast packets should not be put into lossless classes

NIC PFC Pause Frame Storm

- **NIC PFC pause frame storm**: a single malfunctioning NIC may block the entire network from transmitting
- **Microsoft observed PFC storms in their networks multiple times and PFC storms caused several incidents**
- **Root Cause:** a bug in the NIC's receiving pipeline
- **Solution**: Microsoft works with NIC provider to fix this NIC bug

The Slow-receiver Symptom

- In Microsoft's DCN
  - A server NIC is connected to a ToR switch using a point-to-point cable
  - The NIC is connected to the CPU and memory systems via PCIe.
    - For a 40 GbE NIC, it uses PCIe Gen3x8 which provides 64Gb/s raw bidirectional bandwidth which is more than the 40Gb/s throughput of the NIC.

The Slow-receiver Symptom

- There seems to be no bottleneck between the switch and the server CPU and memory
- **Server NIC** should **not** be able to generate PFC pause frames to the switch
- In Microsoft's DCN
  - **Many servers** may generate up to thousands of PFC pause frames per second
- **Root Cause**: NIC itself becomes a bottleneck

The Slow-receiver Symptom

- The NIC has limited memory resources; hence it puts most of the data structures including QPC (Queue Pair Context) and WQE (Work Queue Element) in the main memory of the server
- If the virtual address in a WQE is not mapped in the MTT, it results in a cache miss, and the NIC has to replace some old entries for the new virtual address
  - Access the main memory of the server to get the entry for the new virtual address.
  - Take time and **slow down the packet processing pipeline**

The Slow-receiver Symptom

- **Solution:**
  - On the NIC side, Microsoft used **a large page size** of 2MB instead of 4KB to reduce the frequency of cache miss
  - On the switch side, Microsoft **enabled dynamic buffer sharing** among different switch ports to avoid pause frame propagation if NIC still generates pause frame from time to time

RDMA In Production

- Configuration management and monitoring
- PFC pause frame and traffic monitoring
- RDMA Pingmesh
- RDMA Performance

Configuration Management and Monitoring

- Configure PFC at the switch side
- Configure DCQCN/PFC at the server side
- Microsoft leverages a **configuration monitoring** service to check if the running configurations of the switches and the servers are the same as their desired configurations

PFC Pause Frame and Traffic Monitoring

- Microsoft also uses **a monitoring service** for the **PFC pause frames**
  - Number of pause frames sent and received by the switches and servers
  - Pause intervals at the server side
- Two monitoring services for RDMA
  - Packets and bytes sent and received per port per priority
  - Packet drops at the ingress ports, and packet drops at the egress queues

RDMA Pingmesh

- Microsoft has a TCP pingmesh system
  - **Chuanxiong Guo** et al. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. SIGCOMM 2015 experience paper
- A similar system for RDMA

RDMA Performance

- Latency reduction
- Throughput

RDMA Performance: Latency Reduction

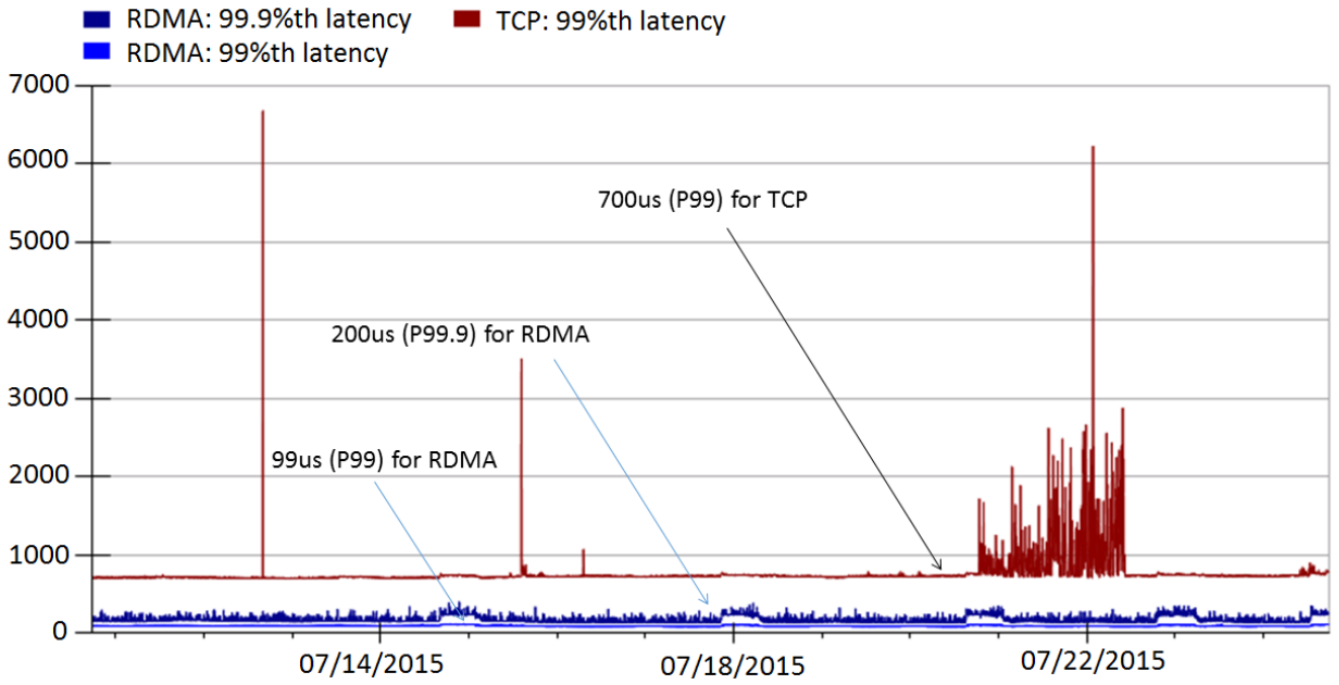
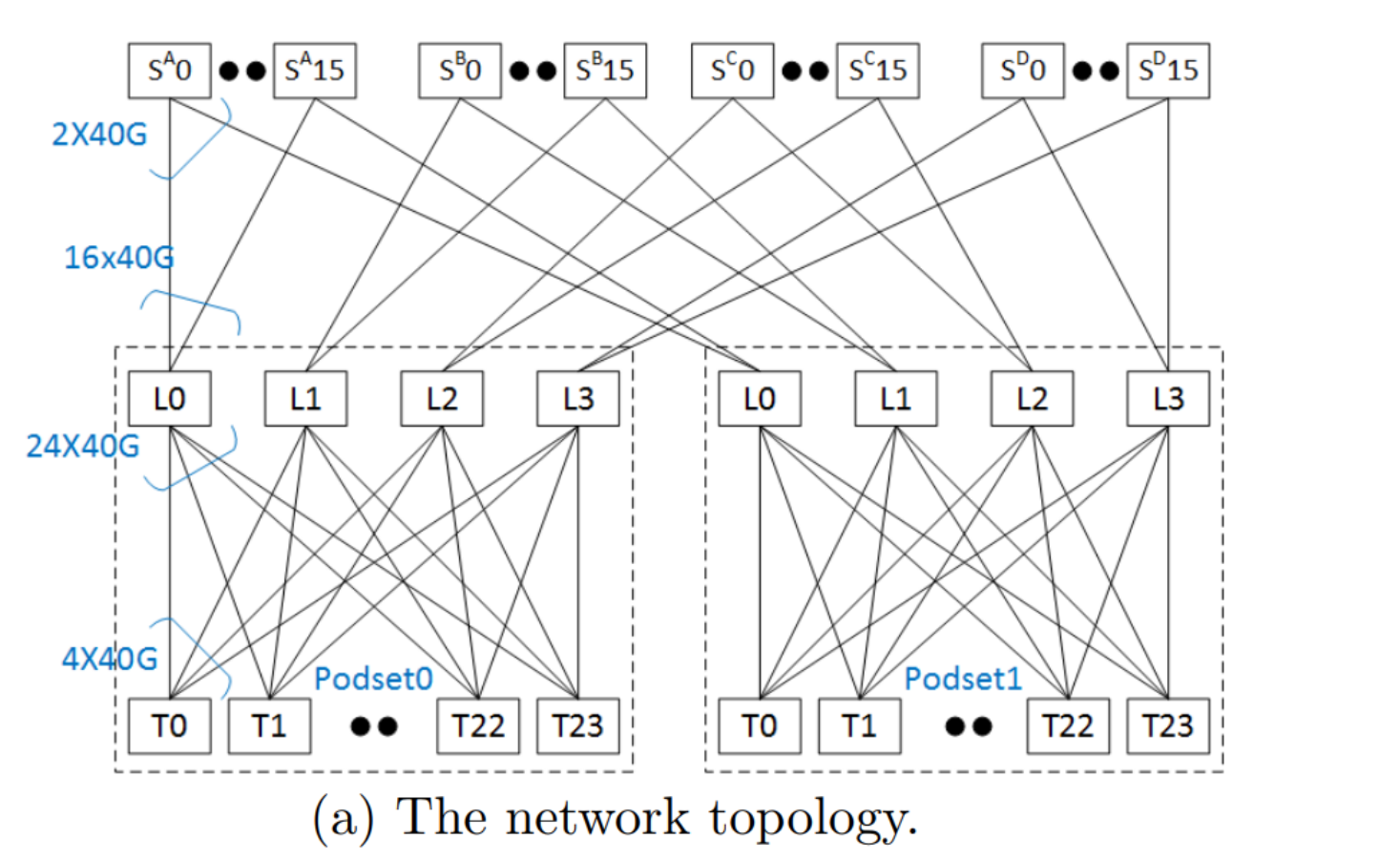


Figure 6: The comparison of the measured TCP and RDMA latencies for a latency-sensitive service.

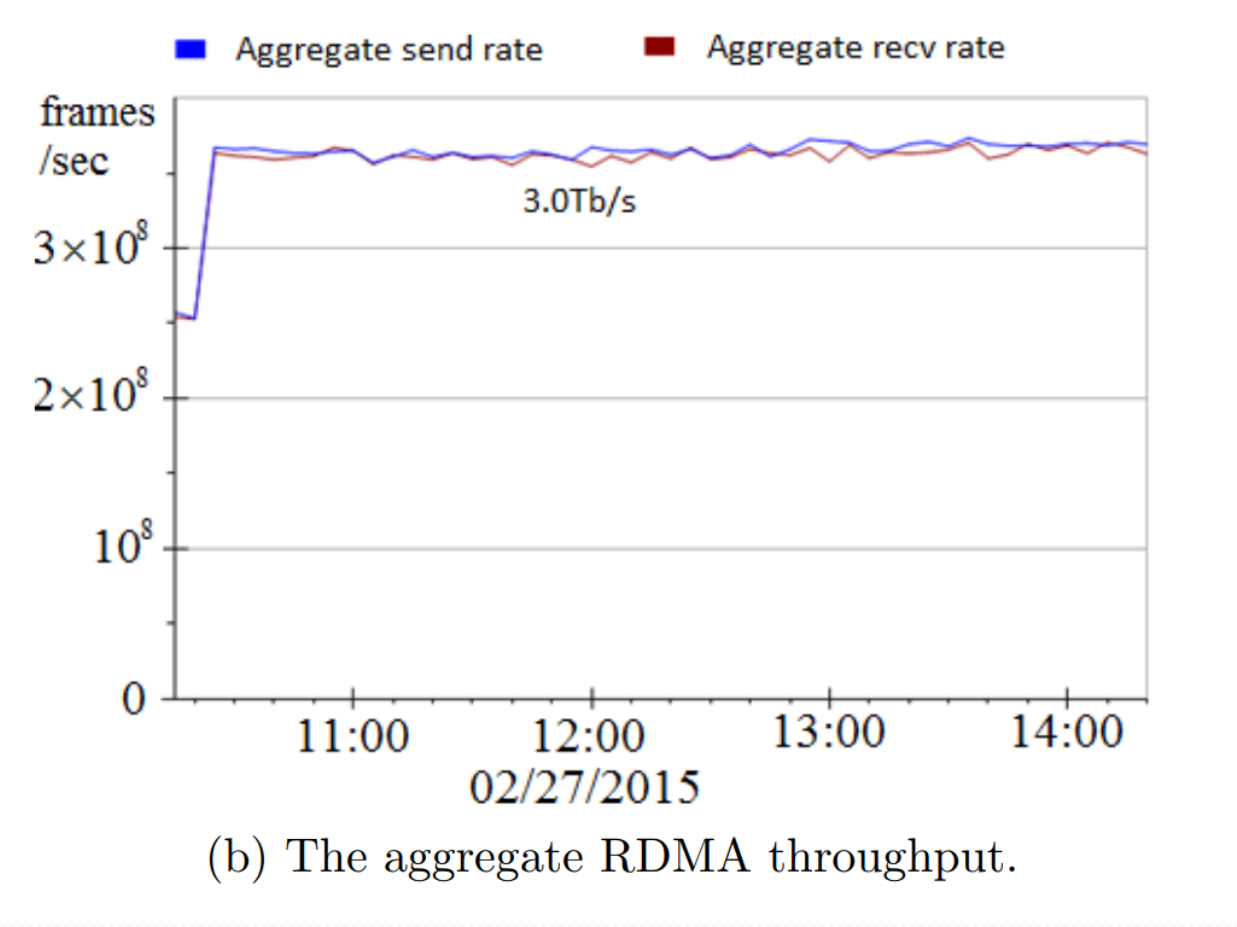
RDMA Performance: Throughput



(a) The network topology.

The aggregate bandwidth between a podset and the Spine switch layer is 64x40Gb/s = 2.56Tb/s. 5.12Tb/s total Capacity.

RDMA Performance: Throughput



(b) The aggregate RDMA throughput.

Microsoft uses ECMP for multi-path routing in our network, 60% utilization is what they can achieve for this experiment

Experience

- RDMA Deployment
- Incidents

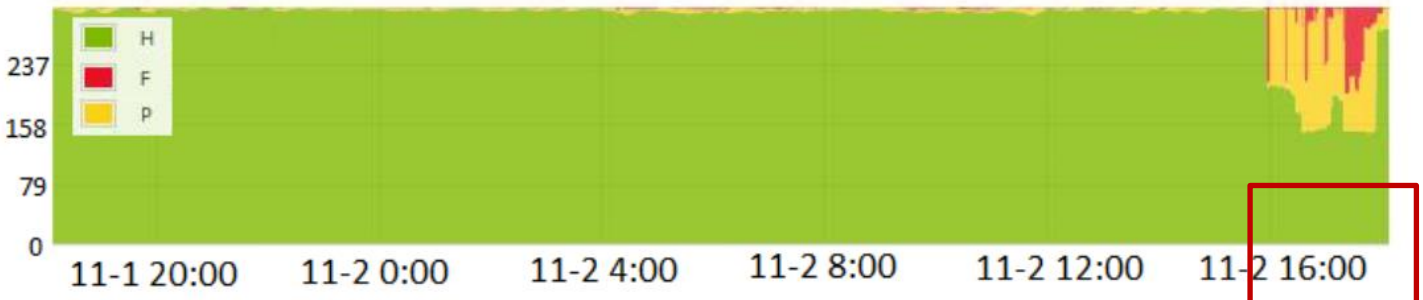
RDMA Deployment

- For the 1st step, we built **a small lab network** with tens of servers
- In the 2nd step, we used test clusters to **improve the maturity** of RoCEv2
- In the 3rd step, we enabled RDMA in production networks **at ToR level only**
- In the 4th step, we enabled PFC **at the Podset level**
- In the last step, we enabled PFC **up to the Spine switches**

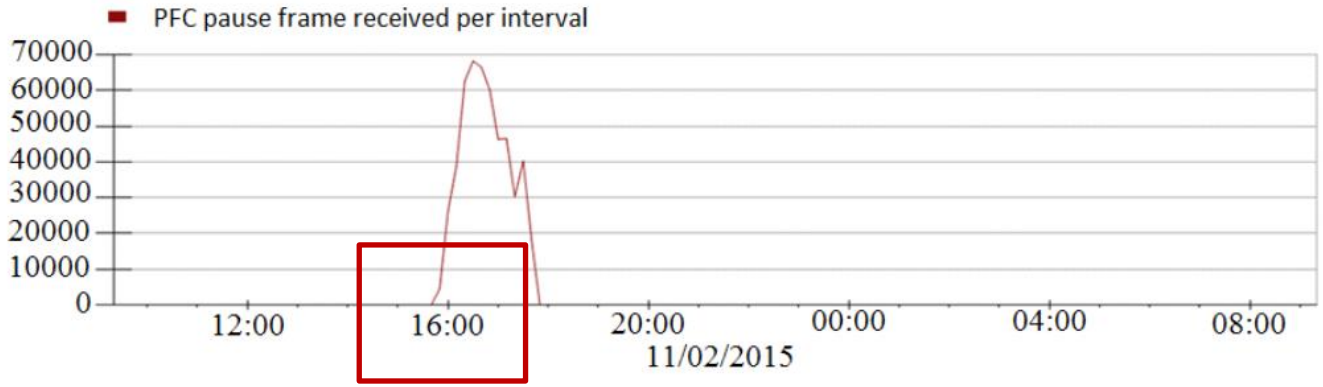
Incidents

- NIC PFC storm
- Switch buffer misconfiguration

NIC PFC Storm



(a) Server availability reduction. H (healthy), F (failing), and P (probation) are server states.



(b) The PFC pause frames received by the servers.

Figure 9: An incident caused by the NIC PFC storm problem of a single server.

NIC PFC Storm

- Many of Microsoft's servers became unavailable as shown in Figure 9(a).
- At the same time, we observed that many of the servers were continuously receiving large number of PFC pause frames as shown by our monitoring system in Figure 9(b).

NIC PFC Storm

- NIC PFC storms happened **very infrequently**
  - With hundreds of thousands of servers in production, the number of the NIC PFC storm events we have experienced is still single digits
  - Nonetheless, once a NIC PFC storm happens, the damage is huge due to the PFC pause frame propagation

Switch Buffer Misconfiguration

- Small switch buffers (9–12MB) require dynamic sharing
- Buffer allocation controlled by parameter **α**
- Misconfigured α leads to inefficient buffer utilization

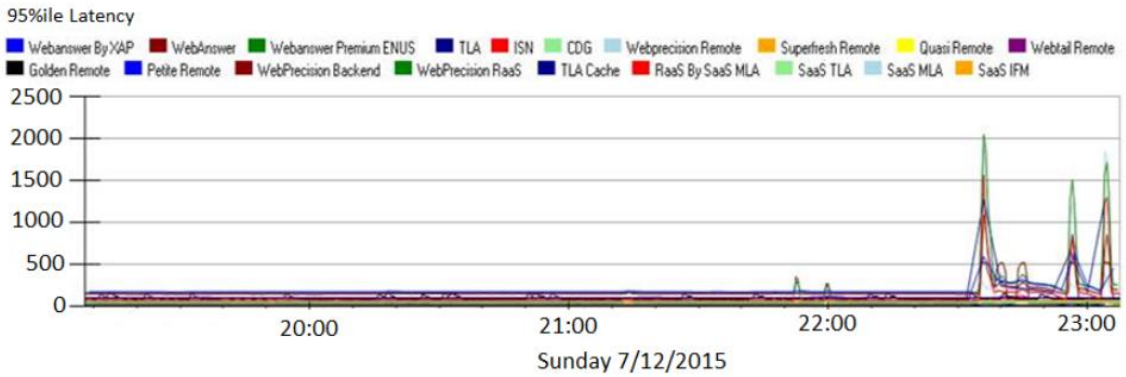
Switch Buffer Misconfiguration

- **Incident Symptoms**
- Dramatic latency increase in latency-sensitive services
- Excessive PFC pause frames (up to 60K in 5 minutes)
- Impact propagated to thousands of servers

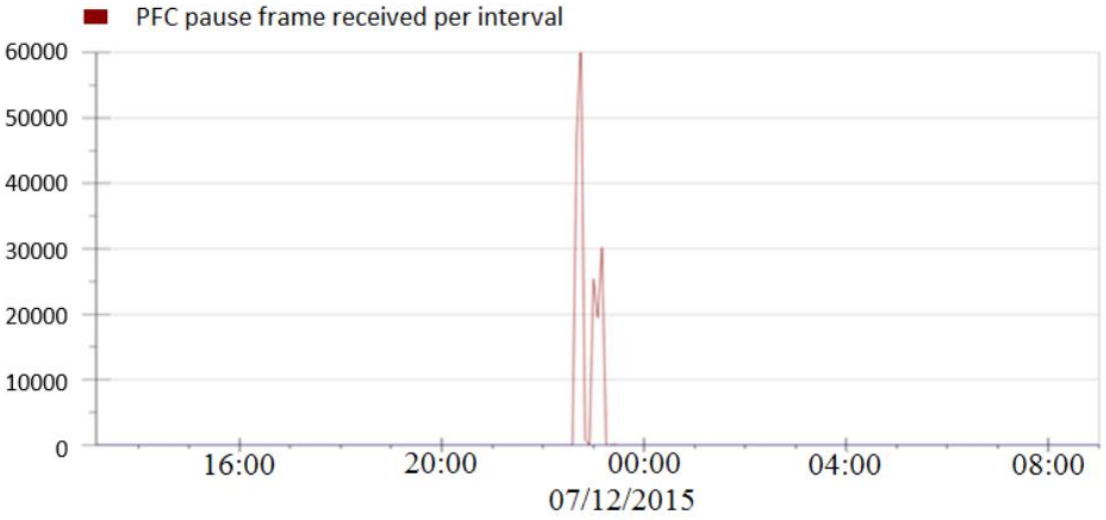
Switch Buffer Misconfiguration

- **Root Causes**
- **Incast traffic pattern**
  - Many-to-one responses create sudden congestion
- **Incorrect α setting**
  - Old ToR: α = 1/16
  - New ToR: α = 1/64 (smaller buffer allocation)
- Smaller α → less buffer per port → easier PFC triggering

Switch Buffer Misconfiguration



(a) Services latency increase caused by the PFC pause frame propagation. Every color here represents an impacted service.



(b) The PFC pause frames received by the servers.

Outline

- Introduction to RDMA (Necessary Background)
- RDMA at Microsoft (Explain the Paper)
- **Takeaway (Some Thoughts)**

Lessons Learned and Discussion

- Deadlock, livelock, and PFC pause frames propagation did happen
- NICs are the key to make RDMA/RoCEv2 work
  - Most of the RDMA/RoCEv2 bugs we ran into were caused by the NICs instead of the switches
- Be prepared for the unexpected
- Is lossless needed?
  - **Lossy RDMA has been deployed nowadays**

Topic Review

Data Center Networking

What We Have Learned?

- How are servers in DCN connected?
  - **Topology** Problem: FatTree
- How to decide the speed of one single flow?
  - **Transport** Problem: DCTCP
- How to decide priorities of flows?
  - **Flow Scheduling** Problem: PIAS
- How to decide the path of flows?
  - **Load Balance Problem**: CONGA
- What's the latest DCN infrastructure beyond TCP?
  - **Deployment Experience**: RDMA

Some Key Points

- FatTree
  - What is CLOS topology? How many shortest path between any pair of servers in a given topology?
  - For FatTree, given a K
    - How to draw the topology of a FatTree
    - How to decide the IP for servers/switches

Some Key Points

- DCTCP
  - Why is DCTCP better than TCP? What are the improvements?
    - Deterministic ECN marking
    - Instantaneous queue length
  - How to update the DCTCP congestion window?

Some Key Points

- PIAS
  - How to implement MLFQ with commodity switches?
  - What's the result of improper threshold? How to solve the problem?
    - Too large
    - Too small

Some Key Points

- CONGA
  - What's the advantages/disadvantages of flow-based/packet-based solutions?
  - Why is flowlet better than them?
  - Why do local congestion-aware solution fail to deliver optimal performance? Examples
  - How to compute DRE?

Some Key Points

- RDMA Deployment in Microsoft
  - What's the advantages of RDMA over TCP and how does RDMA achieve them?
  - What is priority flow control? How does PFC work?
  - What is PFC deadlock? Why does it happen?

Reading Materials
(For Group Presentation. Not in the Exam)

- **Topology: Improvements over Fat-Tree**
  - VL2: a scalable and flexible data center network, SIGCOMM 2009
- **Transport: Designed for RDMA**
  - Congestion Control for Large-Scale RDMA Deployments, SIGCOMM 2015
- **Flow Scheduling: From flow to coflow**
  - Coflow: A Networking Abstraction for Cluster Applications, SIGCOMM 2012
- **Load Balance: Simple is better**
  - Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching, NSDI 17