CompSci 514: Computer Networks
Lecture 17: Network Support for Remote Direct Memory Access

Xiaowei Yang
Some slides adapted from http://www.cs.unh.edu/~rdr/rdma-intro-module.ppt
Overview

• What is RDMA?

• Congestion control for RDMA over Ethernet
  – Congestion Control for Large-Scale RDMA Deployments

• Deploying RDMA at datacenter networks
  – Experience of deploying RDMA at a large scale datacenter network
What is RDMA?

• A (relatively) new method for high-speed inter-machine communication
  – new standards
  – new protocols
  – new hardware interface cards and switches
  – new software
Motivation

• Datacenter networks need
  – Ultra-low latency
  – High throughput
  – Low CPU overhead

• TCP/IP does not meet the requirements
Remote Direction Memory Access

- Read, write, send, receive etc. do not go through CPU
Throughput (Gbps)

Message size

(a) Mean Throughput

RDMA

TCP

(b) Mean CPU Utilization

(c) Mean Latency

Figure 1: Throughput, CPU consumption and latency of TCP and RDMA

Figure 2: Testbed topology. All links are 40Gbps. All switches are Arista 7050QX32. There are four ToRs (T1-T4), four leaves (L1-L4) and two spines (S1-S2). Each ToR represents a different IP subnet. Routing and ECMP is done via BGP. Servers have multiple cores, large RAMs, and 40Gbps NICs.

Figure 3: PFC Unfairness

(a) Topology

(b) Throughput of individual senders

Figure 4: Victim flow problem

Victim flow:

Because PAUSE frames can have a cascading effect, a flow can be hurt by congestion that is not even on its path. Consider Figure 4(a). Four senders (H11-H14) send data to R. In addition, we have a “victim flow” – VS sending to VR. Figure 4(b) shows the median throughput (250 transfers of 250MB each) of the victim flow. When there are no senders under T3, in the median case (two of H11-H14 map to T1-L1, others to T1-L2. Each of H11-H14 gets 10Gbps throughput. VS maps to one of T1’s uplinks), one might expect VS to get 20Gbps throughput. However, we see that it only gets 10Gbps. This is due to cascading PAUSEs. As T4 is the bottleneck of H11-H14 incast, it ends up PAUSEing its incoming links. This in turn leads to L3 and L4 to pause their incoming links, and so forth. Eventually, L1 and L2 end up pausing T1’s uplinks, and T1 is forced to PAUSE the senders. The flows on T1 that use these uplinks are equally affected by these PAUSEs, regardless of their destinations – this is also known as the head-of-the-line blocking problem.

The problem gets worse as we start senders H31 and H32 that also send to R. We see that the median throughput further falls from 10Gbps to 4.5Gbps, even though no path from H31 and H32 to R has any links in common with the path between VS and VR. This happens because H31 and H32 compete with H11-H14 on L3 and L4, make them PAUSE S1 and S2 longer, and eventually make T1 PAUSE senders longer.

Two machines (Intel Xeon E5-2660 2.2GHz, 16 core, 128GB RAM, 40Gbps NICs, Windows Server 2012R2) connected via a 40Gbps switch.
Figure 1: Throughput, CPU consumption and latency of TCP and RDMA

Figure 2: Testbed topology. All links are 40Gbps. All switches are Arista 7050QX32. There are four ToRs (T1-T4), four leaves (L1-L4) and two spines (S1-S2). Each ToR represents a different IP subnet. Routing and ECMP is done via BGP. Servers have multiple cores, large RAMs, and 40Gbps NICs.

(a) Topology

(b) Mean Throughput

(c) Mean Latency

Figure 3: PFC Unfairness

(a) Topology

(b) Throughput of individual senders

Figure 4: Victim flow problem

(a) Topology

(b) Median throughput of victim flow
Figure 1: Throughput, CPU consumption and latency of TCP and RDMA

Figure 2: Testbed topology. All links are 40Gbps. All switches are Arista 7050QX32. There are four ToRs (T1-T4), four leaves (L1-L4) and two spines (S1-S2). Each ToR represents a different IP subnet. Routing and ECMP is done via BGP. Servers have multiple cores, large RAMs, and 40Gbps NICs.

Figure 3: (a) Topology

(b) Throughput of individual senders

(c) Mean Latency

Figure 4: PFC Unfairness

(a) Topology

(b) Median throughput of victim flow

The problems using a 3-tier testbed (Figure 2) representative of modern datacenter networks.

Unfairness:

Consider Figure 3(a). Four senders (H1-H4) send data to the single receiver (R) using RDMA WRITE operation. All senders use the same priority class. Ideally, the four senders should equally share bottleneck link (T4 to R). However, with PFC, there is unfairness. When queue starts building up on T4, it pauses incoming links (ports P2-P4). However, P2 carries just one flow (from H4), while P3 and P4 may carry multiple flows since H1, H2 and H3 must share these two ports, depending on how ECMP maps the flows. Thus, H4 receives higher throughput than H1-H3. This is known as the parking lot problem [14].

This is shown in Figure 3(b), which shows the min, median and max throughput achieved by H1-H4, measured over 1000 4MB data transfers. H4 gets as much as 20Gbps throughput, e.g. when ECMP maps all of H1-H3 to either P3 or P4. H4's minimum throughput is higher than the maximum throughput of H1-H3.

Victim flow:

Because PAUSE frames can have a cascading effect, a flow can be hurt by congestion that is not even on its path. Consider Figure 4(a). Four senders (H11-H14), send data to R. In addition, we have a “victim flow” – VS sending to VR. Figure 4(b) shows the median throughput (250 transfers of 250MB each) of the victim flow.

When there are no senders under T3, in the median case (two of H11-H14 map to T1-L1, others to T1-L2. Each of H11-H14 gets 10Gbps throughput. VS maps to one of T1's uplinks), one might expect VS to get 20Gbps throughput. However, we see that it only gets 10Gbps. This is due to cascading PAUSEs. As T4 is the bottleneck of H11-H14 incast, it ends up PAUSEing its incoming links. This in turn leads to L3 and L4 to pause their incoming links, and so forth. Eventually, L1 and L2 end up pausing T1's uplinks, and T1 is forced to PAUSE the senders. The flows on T1 that use these uplinks are equally affected by these PAUSEs, regardless of their destinations – this is also known as the head-of-the-line blocking problem.

The problem gets worse as we start senders H31 and H32 that also send to R. We see that the median throughput further falls from 10Gbps to 4.5Gbps, even though no path from H31 and H32 to R has any links in common with the path between VS and VR. This happens because H31 and H32 compete with H11-H14 on L3 and L4, make them PAUSE S1 and S2 longer, and eventually make T1 PAUSE senders longer.
Remote Direct Memory Access

- **Remote**
  - data transfers between nodes in a network
- **Direct**
  - no Operating System Kernel involvement in transfers
  - everything about a transfer offloaded onto Interface Card
- **Memory**
  - transfers between user space application virtual memory
  - no extra copying or buffering
- **Access**
  - send, receive, read, write, atomic operations
RDMA Technologies

- InfiniBand
  - Lossless link-layer network
  - Developed for super computing

- iWarp
  - NIC implements TCP/IP

- RoCE – RDMA over Converged Ethernet
  - RDMA for datacenter networks
RDMA architecture layers
InfiniBand (IB)

• Link layer uses hop-by-hop flow control to prevent packet drops

• Transport layer is simple and efficient

• Much of IB is implemented on the NIC

• Single-sided read/write
  – A server registers a memory buffer with its NIC
  – Clients read (write) from (to) it
Similarities between TCP and RDMA

- Both utilize the client-server model
- Both require a connection for reliable transport
- Both provide a reliable transport mode
  - TCP provides a reliable in-order sequence of **bytes**
  - RDMA provides a reliable in-order sequence of **messages**
How RDMA differs from TCP/IP

- "zero copy" – data transferred directly from virtual memory on one node to virtual memory on another node

- "kernel bypass" – no operating system involvement during data transfers

- asynchronous operation – threads not blocked during I/O transfers
TCP/IP setup

Client

User App
Kernel Stack
CA
Wire

Setup

Connect

Server

User App
Kernel Stack
CA
Wire

Setup

Bind
Listen
Accept

blue lines: control information
red lines: user data
green lines: control and data
RDMA setup

client

<table>
<thead>
<tr>
<th>User App</th>
<th>rdma_connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Stack</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>Wire</td>
<td></td>
</tr>
</tbody>
</table>

server

<table>
<thead>
<tr>
<th>rdma_bind</th>
<th>rdma_listen</th>
<th>rdma_accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>User App</td>
<td>Kernel Stack</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>Wire</td>
<td></td>
</tr>
</tbody>
</table>

blue lines: control information
red lines: user data
green lines: control and data
TCP/IP setup

Client

User App
Kernel Stack
CA
Wire

Connect

Server

User App
Kernel Stack
CA
Wire

Setup
Bind
Listen
Accept

blue lines: control information
red lines: user data
green lines: control and data
TCP/IP transfer

**client**

<table>
<thead>
<tr>
<th>User App</th>
<th>setup</th>
<th>transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>connect</td>
<td>send copy</td>
</tr>
<tr>
<td>Kernel Stack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wire</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**server**

<table>
<thead>
<tr>
<th>User App</th>
<th>setup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bind listen accept</td>
</tr>
<tr>
<td>Kernel Stack</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>Wire</td>
<td></td>
</tr>
</tbody>
</table>

- **blue lines**: control information
- **red lines**: user data
- **green lines**: control and data
RDMA transfer

client

User App
- rdma_connect
Kernel Stack
CA
Wire

transfer
- rdma_post_send
- data

server

User App
- rdma_bind
Kernel Stack
CA
Wire

transfer
- rdma_post_recv
- data

setup
- rdma_accept
- rdma_listen

blue lines: control information
red lines: user data
green lines: control and data
“Normal” TCP/IP socket access model

- Byte streams – requires application to delimit / recover message boundaries
- Synchronous – blocks until data is sent/received
  - `O_NONBLOCK, MSG_DONTWAIT` are **not** asynchronous, are “try” and “try again”
- `send()` and `recv()` are paired
  - both sides must participate in the transfer
- Requires data copy into system buffers
  - order and timing of `send()` and `recv()` are **irrelevant**
  - user memory accessible immediately before and immediately after each `send()` and `recv()` call
TCP RECV()
RDMA RECV(

user
allocate
access
recv()
recv queue
metadata
status

channel adapter
recv queue
completion queue

virtual memory
parallel activity
poll_cq()

wire
control
data packets
ACK

channel adapter
register

wire

RDMA access model

- Messages – preserves user's message boundaries
- Asynchronous – no blocking during a transfer, which
  - starts when metadata added to work queue
  - finishes when status available in completion queue
- 1-sided (unpaired) and 2-sided (paired) transfers
- No data copying into system buffers
  - order and timing of send() and recv() are relevant
    - recv() must be waiting before issuing send()
  - memory involved in transfer is untouchable between start and completion of transfer
Congestion Control for Large-Scale RDMA Deployments

By Yibo Zhu et al.
Problem

• RDMA requires a lossless data link layer
• Ethernet is not lossless
• Solution → RDMA over Converged Ethernet RoCE
RoCE details

• Priority-based Flow Control (PFC)
  – A push back protocol
  – When busy, send Pause
  – When not busy, send Resume
Our goal is to support RDMA for intra data center (intra-DC) communications. Overflow is prevented by pausing the upstream sending entity when buffer occupancy exceeds a specified threshold. While some problems with PFC such as head-of-the-line blocking and potential for deadlock are well known [22, 33], we see several issues such as the RDMA transport livelock, the NIC PFC pause frame storm and the slow receiver symptom in our deployment that have not been reported in the literature. Even the root cause of the deadlock problem we have encountered is quite different from the toy examples often discussed in the research literature [22, 33].

We also note that VLAN [32] tags are typically used to identify PFC-enabled traffic in mixed RDMA/TCP deployments. As we shall discuss, this solution does not scale for our environment. Thus, we introduce a notion of DSCP (Differentiated Services Code Point) based PFC to scale RDMA from layer-2 VLAN to layer-3 IP.

Our RDMA deployment has now been running smoothly for over one and half years, and it supports some of Microsoft’s highly-reliable and latency-sensitive online services. Our experience shows that, by improving the design of RoCEv2, by addressing the various safety issues, and by building the needed management and monitoring capabilities, we can deploy RDMA safely in large-scale data centers using commodity Ethernet.

2. BACKGROUND

Our data center network is an Ethernet-based multi-layer Clos network [1, 3, 19, 31] as shown in Figure 1. Twenty to forty servers connect to a top-of-rack (ToR) switch. Tens of ToRs connect to a layer of Leaf switches. The Leaf switches in turn connect to a layer of tens to hundreds of Spine switches. Most links are 40Gb/s, and we plan to upgrade to 50GbE and 100GbE in near future [11, 25]. All switches use IP routing. The servers typically use copper cables of around 2 meters to connect to the ToR switches. The ToR switches and Leaf switches are within the distance of 10 - 20 meters, and the Leaf and Spine switches are within the distance of 200 - 300 meters. With three layers of switches, tens to hundreds of thousands of servers can be connected in a single data center. In this paper, we focus on supporting RDMA among servers under the same Spine switch layer.

RoCEv2:

We deployed RDMA over Converged Ethernet v2 (RoCEv2) [5] for both technical and economical reasons. RoCEv2 encapsulates an RDMA transport [5] packet within an Ethernet/IPv4/UDP packet. This makes RoCEv2 compatible with our existing networking infrastructure. The UDP header is needed for ECMP-based [34] multi-path routing. The destination UDP port is always set to 4791, while the source UDP port is randomly chosen for each queue pair (QP) [5]. The intermediate switches use standard five-tuple hashing. Thus, traffic belonging to the same QP follows the same path, while traffic on different QPs (even between the same pair of communicating endpoints) can follow different paths.

PFC and buffer reservation:

RoCEv2 uses PFC [14] to prevent buffer overflow. The PFC standard specifies 8 priority classes to reduce the head-of-line blocking problem. However, in our network, we are able to use only two of these eight priorities for RDMA. The reason is as follows.

PFC is a hop-by-hop protocol between two Ethernet nodes. As shown in Figure 2, the sender’s egress port sends data packets to the receiver’s ingress port. At the sending egress port, packets are queued in up to eight queues. Each queue maps to a priority. At the receiving ingress port, packets are received in corresponding ingress queues. In the shared-buffer switches used in our network, an ingress queue is implemented simply as a counter – all packets share a common buffer pool.

Once the ingress queue length reaches a certain threshold (XOFF), the switch sends out a PFC pause frame to the corresponding upstream egress queue. After the egress queue receives the pause frame, it stops sending packets. A pause frame carries which priorities need to be paused and the pause duration. Once the ingress queue length falls below another threshold (XON), the switch sends a pause with zero duration to resume transmission. XOFF must be set conservatively to ensure that there is no buffer overflow, while XON needs be set to ensure that there is no buffer underflow. It takes some time for the pause frame to arrive at the upstream egress port, and for the switch to react to the pause frame.
Problems with PFC

- Per-port, not per flow
- Unfairness: port-fair, not flow-fair
- Collateral damage: head-of-line blocking for some flows
Figure 2: Testbed topology. All links are 40Gbps. All switches are Arista 7050QX32. There are four ToRs (T1-T4), four leaves (L1-L4) and two spines (S1-S2). Each ToR represents a different IP subnet. Routing and ECMP is done via BGP. Servers have multiple cores, large RAMs, and 40Gbps NICs.
Unfairness

- H1-H4 write to R
- H4 has no contention at port P2
- H1, H2, and H3 has contention on P3, and P4

Figure 3: PFC Unfairness
Head of line blocking

- VS → VR
- H11,H14, H31-H32 → R
- T4 congested, sends PAUSE messages
- T1 Pauses all its incoming links regardless of their destinations

Figure 1: Throughput, CPU consumption and latency of TCP and RDMA

Figure 2: Testbed topology. All links are 40Gbps. All switches are Arista 7050QX32. There are four ToRs (T1-T4), four leaves (L1-L4) and two spines (S1-S2). Each ToR represents a different IP subnet. Routing and ECMP is done via BGP. Servers have multiple cores, large RAMs, and 40Gbps NICs.

Figure 4: Victim flow problem
Solution

• Per-flow congestion control control

• Existing work:
  – QCN (Quantized Congestion Notification)
    • Using Ethernet SRC/DST and a flow ID to define a flow
    • Switch sends congestion notification to sender based on source MAC address
    • Only works at L2

• This work: DCQCN
  – Works for IP-routed networks
Why QCN does not work for IP networks?

- Same packet has different SRC/DST MAC addresses.
DCQCN

• DCQCN is a rate-based, end-to-end congestion protocol
  – No notion of window
  – No reliability

• Most of the DCQCN functionality is implemented in the NICs
High level Ideas

- ECN-mark packets at an egress queue
- Receiver sends Congestion Notification to sender
- Sender reduces sending rates
Challenges

• How to set buffer sizes at the egress queue

• How often to send congestion notifications

• How a sender should reduce its sending rate to ensure both convergence and fairness
Solutions provided by the paper

• ECN must be set before PFC is triggered
  – Use PFC queue sizes to set ECN buffer

• Use a fluid model to tune congestion parameters

• Using ECN marking as a degree of congestion
  – Same as DCTCP
RDMA over Commodity Ethernet at Scale

Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitendra Padhye, Marina Lipshteyn
Microsoft
What this paper is about

• Extending PFC to IP-routed network
• Safety issues of RDMA
  – Livelock
  – Deadlock
  – Pause frame storm
  – Slower receiver symptoms
• Performance observed in production networks
DSCP-based PFC

- Issues of VLAN-based PFC
  - It breaks PXE boot
  - No standard way for carrying VLAN tag in L3 networks

- DSCP-based PFC
  - DSCP field for carrying the priority value
  - No change needed for the PFC pause frame
  - Supported by major switch/NIC vendors
RDMA transport livelock

4MB message, 1K packets
Drop packets with IP ID’s last byte 0xff (1/256)
go-back-to-0 → go-back-to-N
PFC deadlock

• Our data centers use Clos network
• Packets first travel up then go down
• No cyclic buffer dependency for up-down routing -> no deadlock
• But we did experience deadlock!
PFC deadlock

- Preliminaries
  - ARP table: IP address to MAC address mapping
  - MAC table: MAC address to port mapping
  - If MAC entry is missing, packets are flooded to all ports
S3 is dead.
T1.p2 is congested
Pause is sent to T1.p3, La.p1, To.p2, S1.
S4 → S2, S2 is dead
Blue packet flooded to T0.p2
To.p2 is paused. Ingress T0.p3 pauses Lb.p0
Lb.p1 pauses T1.p4. T1.p1 pauses S4
PFC deadlock

Path: \{S1, T0, L_a, T1, S3\}
Path: \{S1, T0, L_a, T1, S5\}
Path: \{S4, T1, L_b, T0, S2\}

Ingress port
Egress port
Congested port
Packet drop
Dead server

Server
S1
S2

T0

T1
Summary

• What is RDMA

• DCQCN: congestion control for RDMA

• Deployment issues for RDMA