Announcements

• Midterm – next Wednesday 3/7
Overview

• DCTCP

• Pfabric

• RDMA
Data Center Packet Transport

- Large purpose-built DCs
  - Huge investment: R&D, business

- Transport **inside** the DC
  - TCP rules (99.9% of traffic)

- How’s TCP doing?
TCP in the Data Center

- We’ll see TCP does not meet demands of apps.
  - Suffers from bursty packet drops, Incast [SIGCOMM ‘09], ...
  - Builds up large queues:
    - Adds significant latency.
    - Wastes precious buffers, esp. bad with shallow-buffered switches.

- Operators work around TCP problems.
  - Ad-hoc, inefficient, often expensive solutions
  - No solid understanding of consequences, tradeoffs
Partition/Aggregate Application Structure

- Time is money
  - Strict deadlines (SLAs)
- Missed deadline
  - Lower quality result

**Deadline**
1. **250ms**
2. **50ms**
3. **10ms**

**Artists**
- Picasso: "Everything you can imagine is real."
- "Bad artists copy. Good artists steal."
- "It is your work in life that is the ultimate seduction."
- "The chief enemy of creativity is good sense."
- "Inspiration does exist, but it must find you working."
- "I'd like to live as a poor man with lots of money."
- "Art is a lie that makes us realize the truth."
- "Computers are useless. They can only give you answers."
Generality of Partition/Aggregate

- The foundation for many large-scale web applications.
  - Web search, Social network composition, Ad selection, etc.

- Example: Facebook

Partition/Aggregate ~ Multiget
- Aggregators: Web Servers
- Workers: Memcached Servers
Workloads

• Partition/Aggregate (Query)

• Short messages [50KB-1MB] (Coordination, Control state)

• Large flows [1MB-50MB] (Data update)
Incast: Cluster-based Storage Systems

Synchronized Read

Client sends next batch of requests

1 2 3 4

Server Request Unit (SRU)

Data Block

Client

Switch

Storage Servers
Incast

Worker 1

Worker 2

Worker 3

Worker 4

• Synchronized mice collide.
  ➢ Caused by Partition/Aggregate.

RTO_{\text{min}} = 300 \text{ ms}

TCP timeout

Aggregator
Requests are jittered over 10ms window.

Jittering switched off around 8:30 am.

MLA Query Completion Time (ms)

Jittering trades of median for high percentiles
Queue Buildup

- Big flows buildup queues.
  - Increased latency for short flows.

- Measurements in Bing cluster
  - For 90% packets: RTT < 1ms
  - For 10% packets: 1ms < RTT < 15ms
1. High Burst Tolerance
   - Incast due to Partition/Aggregate is common.

2. Low Latency
   - Short flows, queries

3. High Throughput
   - Continuous data updates, large file transfers

The challenge is to achieve these three together.
Tension Between Requirements

High Throughput
High Burst Tolerance

Deep Buffers:
- Queuing Delays
- Increased Queuing Delays

Reduced $RTO_{\text{min}}$ (SIGCOMM ‘09)
- Doesn’t Help Latency

Low Latency

Shallow Buffers:
- Bad for Bursts & Throughput

AQM – RED:
- Avg Queue Not Fast Enough for Incast

Objective:
Low Queue Occupancy & High Throughput
Review: The TCP/ECN Control Loop

Sender 1

ECN = Explicit Congestion Notification

ECN Mark (1 bit)

Sender 2

Receiver
Two Key Ideas

1. React in proportion to the **extent** of congestion, not its presence.
   - Reduces **variance** in sending rates, lowering queuing requirements.

<table>
<thead>
<tr>
<th>ECN Marks</th>
<th>TCP</th>
<th>DCTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 1 1 0 1 1 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 40%</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 5%</td>
</tr>
</tbody>
</table>

2. Mark based on **instantaneous** queue length.
   - Fast feedback to better deal with bursts.
**Data Center TCP Algorithm**

**Switch side:**
- Mark packets when **Queue Length > K**.

**Sender side:**
- Maintain running average of *fraction* of packets marked ($\alpha$).

In each RTT:

$$ F = \frac{\text{# of marked ACKs}}{\text{Total # of ACKs}} $$

$$ \alpha \leftarrow (1 - g)\alpha + gF $$

- **Adaptive window decreases:**
  $$ Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd $$
  - Note: decrease factor between 1 and 2.
DCTCP in Action

Setup: Win 7, Broadcom 1Gbps Switch
Scenario: 2 long-lived flows, K = 30KB

Queue Length (Kbytes)

Time (seconds)

DCTCP, K=20, 2 flows
TCP, 2 flows

Setup: Win 7, Broadcom 1Gbps Switch
Scenario: 2 long-lived flows, K = 30KB
Why it Works

1. High Burst Tolerance
   ✓ Large buffer headroom → bursts fit.
   ✓ Aggressive marking → sources react before packets are dropped.

2. Low Latency
   ✓ Small buffer occupancies → low queuing delay.

3. High Throughput
   ✓ ECN averaging → smooth rate adjustments, low variance.
Conclusions

• DCTCP satisfies all our requirements for Data Center packet transport.
  ✓ Handles bursts well
  ✓ Keeps queuing delays low
  ✓ Achieves high throughput

• Features:
  ✓ Very simple change to TCP and a single switch parameter.
  ✓ Based on mechanisms already available in Silicon.
Overview

• DCTCP

• Pfabric

• RDMA
RDMA/RoCEv2 background

• RDMA addresses TCP’s latency and CPU overhead problems
  • RDMA: Remote Direct Memory Access
    • RDMA offloads the transport layer to the NIC
    • RDMA needs a lossless network
  • RoCEv2: RDMA over commodity Ethernet
    • DCQCN for connection-level congestion control
    • PFC for hop-by-hop flow control
RDMA

- RDMA performance typically better than TCP
- Key assumption → lossless network
- Need Priority Flow Control (PFC) in Ethernet
Priority-based flow control (PFC)

- Hop-by-hop flow control, with eight priorities for HOL blocking mitigation
- The priority in data packets is carried in the VLAN tag
- PFC pause frame to inform the upstream to stop
PFC deadlock

Path: \{S1, T0, La, T1, S3\}

Path: \{S1, T0, La, T1, S5\}

Path: \{S4, T1, Lb, T0, S2\}

Ingress port

Egress port

Dead server

Packet drop

PFC pause frames

Congested port

Server

S1

S2

S3

S4

S5

Sigcomm 2016
“Congestion Spreading” in Lossless Networks (Priority Flow Control of PFC)
DCQCN

- Microsoft/Mellanox proposal to run congestion control for RoCEv2
- Receiving NICS reflect back ECN info once per N microseconds
- Sender adjusts rate in DCTCP-like fashion
- Experience:
  - Although tree-like topology deadlocks can occur due to broadcast
  - Livelock issues if losses occur due to poor loss recovery
  - Hard to use > 2 priorities due to buffer limitations → need enough buffer to accommodate delays in pause
  - Simple fixes – but suggest fragile design
Overview

• DCTCP

• Pfabric

• RDMA
DC Fabric: Just a Giant Switch
DC Fabric: Just a Giant Switch
DC Fabric: Just a Giant Switch
DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT

DC transport = Flow scheduling on giant switch

Objective?

Minimize avg FCT
“Ideal” Flow Scheduling

Problem is NP-hard 😞 [Bar-Noy et al.]

• Simple greedy algorithm: 2-approximation

![Diagram showing flow scheduling process]
Key Insight

Decouple flow scheduling from rate control

Switches implement flow scheduling via local mechanisms

Hosts implement simple rate control to avoid high packet loss
pFabric Switch

- **Priority Scheduling**: send highest priority packet first
- **Priority Dropping**: drop lowest priority packets first

$prio = \text{remaining flow size}$

Small “bag” of packets per-port
pFabric Switch Complexity

- Buffers are very small (~2 × BDP per-port)
  - e.g., C=10Gbps, RTT=15µs → Buffer ~ 30KB
  - Today’s switch buffers are 10-30x larger

Priority Scheduling/Dropping

- Worst-case: Minimum size packets (64B)
  - 51.2ns to find min/max of ~600 numbers
  - Binary comparator tree: 10 clock cycles
  - Current ASICs: clock ~ 1ns
pFabric Rate Control

- With priority scheduling/dropping, queue buildup doesn’t matter
  \[\text{Greatly simplifies rate control}\]

**Only task for RC:**
Prevent congestion collapse when elephants collide
pFabric Rate Control

Minimal version of TCP algorithm

1. Start at line-rate
   - Initial window larger than BDP

2. No retransmission timeout estimation
   - Fixed RTO at small multiple of round-trip time

3. Reduce window size upon packet drops
   - Window increase same as TCP (slow start, congestion avoidance, …)
Why does this work?

Key invariant for ideal scheduling:
At any instant, have the highest priority packet (according to ideal algorithm) available at the switch.

- Priority scheduling
  - High priority packets traverse fabric as quickly as possible

- What about dropped packets?
  - Lowest priority → not needed till all other packets depart
  - Buffer > BDP → enough time (> RTT) to retransmit