Congestion Control in Data Centers

Lecture 16, Computer Networks (198:552)
100Kbps–100Mbps links
~100ms latency

Transport inside the DC

10–40Gbps links
~10–100μs latency
Transport inside the DC

Interconnect for distributed compute workloads

Fabric

INTERNET

web
app
cache
database
map-reduce
HPC
monitoring
What’s different about DC transport?

• Network characteristics
  • Very high link speeds (Gb/s); very low latency (microseconds)
• Application characteristics
  • Large-scale distributed computation
• Challenging traffic patterns
  • Diverse mix of mice & elephants
  • Incast
• Cheap switches
  • Single-chip shared-memory devices; shallow buffers
Additional degrees of flexibility

• Flow priorities and deadlines

• Preemption and termination of flows

• Coordination with switches

• Packet header changes to propagate information
Data center workloads

• Mice and Elephants!

• Short messages
  (e.g., query, coordination)

• Large flows
  (e.g., data update, backup)

→ Low Latency

→ High Throughput
Incast

Worker 1

Worker 2

Worker 3

Worker 4

Aggregator

• Synchronized fan-in congestion

Vasudevan et al. (SIGCOMM’09)

TCP timeout

$RTO_{\text{min}} = 300 \text{ ms}$
Requests are jittered over 10ms window.

Jittering switched off around 8:30 am.

Jittering trades of median for high percentiles
DC transport requirements

1. Low Latency
   - Short messages, queries

2. High Throughput
   - Continuous data updates, backups

3. High Burst Tolerance
   - Incast

The challenge is to achieve these together
Data Center TCP

Mohammad Alizadeh et al., SIGCOMM’10
TCP widely used in the data center

• Apps use familiar interfaces
  • TCP is deeply ingrained in the apps
  • ... And developers’ minds

• However, TCP not really designed for data center environments
  • Complex to work around TCP problems
  • Ad-hoc, inefficient, often expensive solutions

Practical deployment is hard
→ keep it simple!
Review: TCP algorithm

**Additive Increase:**
\[ W \rightarrow W + 1 \text{ per round-trip time} \]

**Multiplicative Decrease:**
\[ W \rightarrow W/2 \text{ per drop or ECN mark} \]

**ECN Mark (1 bit)**

**ECN = Explicit Congestion Notification**
TCP buffer requirement

- Bandwidth-delay product rule of thumb:
  - A single flow needs $C \times RTT$ buffers for 100% Throughput.
Reducing buffer requirements

- Appenzeller et al. (SIGCOMM ‘04):
  - Large # of flows: \( C \times RTT/\sqrt{N} \) is enough.
Reducing buffer requirements

• Appenzeller et al. (SIGCOMM ‘04):
  • Large # of flows: \( C \times RTT/\sqrt{N} \) is enough.

• Can’t rely on stat-mux benefit in the DC
  • Measurements show typically only 1-2 large flows at each server

Key observation:
Low variance in sending rate ➞ Small buffers suffice
DCTCP: Main idea

• Extract multi-bit feedback from single-bit stream of ECN marks
  • Reduce window size based on fraction of marked packets
### DCTCP: Main idea

<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>

**Graphs**

- **TCP**
  - Window Size (Bytes) vs Time (sec)
  - Variations in window size over time

- **DCTCP**
  - Window Size (Bytes) vs Time (sec)
  - Reduced fluctuations compared to TCP
DCTCP algorithm

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

Sender side:
- Maintain running average of **fraction** of packets marked (**α**).
  
  \[
  \text{each RTT: } F = \frac{\text{# of marked ACKs}}{\text{Total # of ACKs}} \Rightarrow \alpha \leftarrow (1 - g)\alpha + gF
  \]

- **Adaptive window decreases:** 
  \[
  W \leftarrow (1 - \frac{\alpha}{2})W
  \]
  - Note: decrease factor between 1 and 2.
DCTCP vs TCP

**Experiment:** 2 flows (Win 7 stack), Broadcom 1Gbps Switch

Buffer is mostly empty

DCTCP mitigates Incast by creating a large buffer headroom
Why it works

1. Low Latency
   ✓ Small buffer occupancies $\rightarrow$ low queuing delay

2. High Throughput
   ✓ ECN averaging $\rightarrow$ smooth rate adjustments, low variance

3. High Burst Tolerance
   ✓ Large buffer headroom $\rightarrow$ bursts fit
   ✓ Aggressive marking $\rightarrow$ sources react before packets are dropped
Setting parameters: A bit of analysis

• How much buffering does DCTCP need for 100% throughput?

➢ Need to quantify queue size oscillations (Stability).

\[ \alpha = \frac{\text{# of pkts in last RTT of Period}}{\text{# of pkts in Period}} \]

\[ W^* \]

\[ W^* + 1 \]

\[ (W^* + 1)(1 - \alpha/2) \]
Setting parameters: A bit of analysis

• How small can queues be without loss of throughput?

➢ Need to quantify queue size oscillations (Stability).

\[ K > (\frac{1}{7}) \ C \times \text{RTT} \]

for TCP:
\[ K > C \times \text{RTT} \]
Convergence time

- DCTCP takes at most ~40% more **RTTs** than TCP
  - “Analysis of DCTCP”, SIGMETRICS 2011
- **Intuition**: DCTCP makes smaller adjustments than TCP, but makes them much more frequently
Bing benchmark (baseline)

**Background Flows**

- DCTCP
- TCP

**Query Flows**

- DCTCP
- TCP

Flow Completion Time (ms)

<table>
<thead>
<tr>
<th>Flow Size</th>
<th>DCTCP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-100KB</td>
<td>916</td>
<td>13</td>
</tr>
<tr>
<td>100KB-1MB</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>1-10MB</td>
<td>63</td>
<td>64</td>
</tr>
<tr>
<td>&gt;10MB</td>
<td>182</td>
<td>182</td>
</tr>
</tbody>
</table>

Query Completion Time (ms)

<table>
<thead>
<tr>
<th>Percentile</th>
<th>DCTCP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>95th</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>99th</td>
<td>19</td>
<td>28</td>
</tr>
<tr>
<td>99.9th</td>
<td>40</td>
<td>68</td>
</tr>
</tbody>
</table>
Bing benchmark (scaled 10x)

Completion Time (ms)

- **Query Traffic (Incast bursts)**
  - TCP/ShallowBuf
  - TCP/DeepBuf
  - DCTCP/ShallowBuf

- **Short messages (Delay-sensitive)**
  - DCTCP good for both incast & latency

Deep buffers fix incast, but increase latency.
Discussion

• Between throughput, delay, and convergence time, what metrics are you willing to give up? Why?

• Are there other factors that may determine choice of K and B besides loss of throughput and max queue size?

• How would you improve on DCTCP?

• How could you add on flow prioritization over DCTCP?
Acknowledgment

• Slides heavily adapted from material by Mohammad Alizadeh