Catch the Whole Lot in an Action:
Rapid Precise Packet Loss Notification in Data Centers

Peng Cheng, Fengyuan Ren, Ran Shu, Chuang Lin
Tsinghua University
TCP Out-of-order

Long Query Completion Time

TCP Unfairness

TCP Incast

TCP Outcast
One simple TCP-complementary mechanism to solve all problem.
TCP Incast

**Requirement of MapReduce:**
\[ N = 154 \pm 558\sigma \]  
[M45 Yahoo! cluster]

**Solution:**
- DCTCP
- ICTCP
- Reducing $RTO_{\text{min}}$?
Throughput Cliff

Each Data Size = 64KB, Buffer = 128KB

<table>
<thead>
<tr>
<th>Goodput (Mbps)</th>
<th>Number of Senders</th>
</tr>
</thead>
<tbody>
<tr>
<td>900</td>
<td>1</td>
</tr>
<tr>
<td>850</td>
<td>25</td>
</tr>
<tr>
<td>800</td>
<td>50</td>
</tr>
<tr>
<td>750</td>
<td>75</td>
</tr>
<tr>
<td>700</td>
<td>100</td>
</tr>
<tr>
<td>650</td>
<td>125</td>
</tr>
<tr>
<td>600</td>
<td>150</td>
</tr>
<tr>
<td>550</td>
<td>175</td>
</tr>
<tr>
<td>500</td>
<td>200</td>
</tr>
<tr>
<td>450</td>
<td>225</td>
</tr>
<tr>
<td>400</td>
<td>250</td>
</tr>
</tbody>
</table>

- **DCTCP (10ms)**
- **TCP (10ms)**

**Throughput Cliff**

Each Data Size = 64KB, Buffer = 128KB

2 Apr 2014

NSDI 2014
Throughput Cliff

Each Data Size = 64KB, Buffer = 128KB

Synchronous packet loss ➔ increasing Timeout

DCTCP(10ms)
TCP(10ms)

PTC = 170
PTC = 14
PTC = 42
PTC = 85

Goodput (Mbps)
Number of Senders

2 Apr 2014
NSDI 2014
Avoid timeout after a large number of packet loss is the key to meet the requirement of MapReduce.

Synchronous packet loss $\rightarrow$ increasing Timeout
TCP Unfairness

Random Drop Congestion Control, SIGCOMM 1990

Multiple-bottleneck Flows: Low Throughput in Internet
TCP Unfairness

Random Drop Congestion Control, SIGCOMM 1990

Multiple-bottleneck Flows
Sigle-bottleneck Flows

[Data Center Topology]
TCP Unfairness

Multiple-bottleneck Flows: in DCNs

[Random Drop Congestion Control, SIGCOMM 1990]
TCP Unfairness

![Graph showing the sum of MBF's goodput (Mbps) against the ratio of MBF to SBF for different SBF values (SBF=5, SBF=25, SBF=50) and comparing to theory. The x-axis represents the ratio of MBF to SBF, and the y-axis represents the sum of MBF's goodput (Mbps). The graph illustrates the unfairness in TCP connections with varying SBF settings.]
TCP Unfairness

High Packet Loss Ratio: Significant Low Throughput
TCP Unfairness

Low Packet Loss Ratio:
TCP Outcast →
Unfairly High Throughput
According to the paper, primary cause of TCP Outcast and low throughput is timeout.
TCP Out-of-order

• Multipath routing techniques
  • Flow-based Splitting Schemes: ECMP, Hedera, VLB
  • Ideal Packet-level scheme is better
TCP Out-of-order

• Multipath routing techniques
  • Flow-based Splitting Schemes: ECMP, Hedera, VLB
  • Ideal Packet-level scheme is better

• TCP Out-of-order Problem
  • Using a fixed threshold
    • Spurious retransmission
    • Sluggish congestion control
TCP Out-of-order

- Multipath routing techniques
  - Flow-based Splitting Schemes: ECMP, Hedera, VLB

Accurately distinguishing between lost or out-of-order packets can tackle TCP out-of-order problem

- Spurious retransmission
- Sluggish congestion control
• Each of 20 DCTCP flows transfers 50KB (Avg: 8ms)
• 10,000 random instances
• Finding: 1,231 experiments > 15ms

• Suffer many retransmission and waste time retransmitting
Long Query Completion Time

Each of 20 DCTCP flows transfers 50KB (Avg: 8ms)

10,000 random instances

Finding: 1,231 experiments > 15ms

2 Apr 2014

NSDI 2014

Retransmission delay imposes a significant influence on query completion time

• Suffer many retransmission and waste time retransmitting
Long Query Completion Time

Each of 20 DCTCP flows transfers 50KB (Avg: 8ms)

Finding: 1,231 experiments > 15ms

Retransmission delay imposes a significant influence on query completion time

Retransmission delay = Retransmission Time + Detection Time

• Suffer many retransmission and waste time

Flow Completion Time

Flow Number

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Flow Completion Time

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

15.371
Each of 20 DCTCP flows transfers 50KB (Avg: 8ms) for 10,000 random instances. Finding: 1,231 experiments > 15ms. Retransmission delay imposes a significant influence on query completion time. Reducing detection time is a better way to improve query completion time.
Our Goal

- TCP Incast
- TCP Unfairness
- TCP Out-of-order
- Long Completion Time
- Avoid Timeout
- Distinguish accurately
- Reduce Packet Loss Detection Time
Our Solution: CP

Switch: CP Drop Processing

Receiver: Packet-loss Feedback (PACK)

Sender: Reaction to PACK
Our Solution: CP

Switch: CP Drop Processing

Receiver: Packet-loss Feedback (PACK)

Sender: Reaction to PACK
Our Solution: CP

Switch: CP Drop Processing

Receiver: Packet-loss Feedback (PACK)

Sender: Reaction to PACK
Our Solution: CP

Switch: CP Drop Processing

Receiver: Packet-loss Feedback (PACK)

Sender: Reaction to PACK

IF input < 13.1 * output,
Extra Buffer: < 1 packet size

[Average length: 864B, Header: 66B]
• When Receiving the Payload-cut Packet:
  • **Step1:** Parse the payload-cut packet
  • **Step2:** Generate the PACK option
    • Swap the left and right edge of block
  • **Compatible with SACK**

• **Step3:** Send the PACK immediately
Our Solution: CP

When Receiving the PACK:

- **Step1:** Which packet is lost
  - Parse the PACK option
- **Step2:** Whether to be retransmitted
  - Check the packet status
- **Step3:** Trigger FRFR and retransmit the lost packet
Benefits

- Payload-cut Packet + PACK
  (Maintain Self-clocking)

- PACK
  (Rapid Precise Packet Loss Notification)

- Avoid Timeout

- Distinguish accurately

- Reduce Packet Loss Detection Time
Implementation

NetFPGA

Input Arbiter
Output Port Lookup
CP Marker
FIFO
TCP Checksum Calculation (6 cycle delay)
New Packet Length Calculation (7 cycle delay)
CP Controller (no delay)
CP Handler
Packet Reformer
Drop Payload
Output Queue
Buffer Size Registers

Data Path
Control Path

2 Apr 2014 NSDI 2014
Resource Usage: < 2% increase
Resource Usage: < 2% increase

Delay: 7 cycle $\times$ 8 = 56ns
Evaluation: TCP Incast

![Graph showing Goodput vs Number of Senders for different protocols: TCP (10ms), DCTCP (200ms), CP, CP & DCTCP, DCTCP (10ms), TCP (200ms). The graph demonstrates the performance of these protocols under varying senders.]
Achieve high throughput when N is large.
Evaluation: TCP Unfairness

Low Packet Loss Ratio:
Avoid TCP Outcast

High Packet Loss Ratio:
Ease Low Throughput
Evaluation: TCP Out-of-order Problem

- TCP (DupACK=3)
- DCTCP
- CP (DupACK=∞)
- Theory

Solve the TCP Out-of-order Problem

<table>
<thead>
<tr>
<th>Probability of Packets Through Non-congested Path</th>
<th>Goodput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>200</td>
</tr>
<tr>
<td>1/4</td>
<td>300</td>
</tr>
<tr>
<td>3/8</td>
<td>400</td>
</tr>
<tr>
<td>1/2</td>
<td>500</td>
</tr>
<tr>
<td>5/8</td>
<td>600</td>
</tr>
<tr>
<td>3/4</td>
<td>700</td>
</tr>
<tr>
<td>7/8</td>
<td>800</td>
</tr>
</tbody>
</table>
## Evaluation: Query Completion Time

<table>
<thead>
<tr>
<th></th>
<th>DCTCP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Realistic Traffic (384KB)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99th</td>
<td>10.2ms $\rightarrow$ 5.7ms (44.34%)</td>
<td>11.3ms $\rightarrow$ 7.0ms (38.09%)</td>
</tr>
<tr>
<td>99.9th</td>
<td>36.5ms $\rightarrow$ 34.1ms (6.58%)</td>
<td>47.7ms $\rightarrow$ 42.3ms (11.40%)</td>
</tr>
<tr>
<td><strong>10X Traffic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99th</td>
<td>31ms $\rightarrow$ 25ms (18.53%)</td>
<td>117ms $\rightarrow$ 75ms (35.71%)</td>
</tr>
<tr>
<td>99.9th</td>
<td>52ms $\rightarrow$ 39ms (26.21%)</td>
<td>482ms $\rightarrow$ 113ms (76.55%)</td>
</tr>
</tbody>
</table>
Conclusion

• **CP**
  • is a TCP-complementary method
  • can solve many TCP performance issue
  • is compatible with other TCP protocol in DCNs

• **Source Code Release**
  • Linux Patch + NetFPGA implementation
Thank you!