ACM SIGCOMM 2017

Credit–Scheduled Delay–Bounded Congestion Control for Datacenters

Inho Cho, Keon Jang*, Dongsu Han

KAIST

Google*
Datacenter Network

**Small Latency**

$< 100 \, \mu s$

**Shallow Buffer**

$< 30 \, \text{MB for ToR}$

**High Bandwidth**

$10/40 \sim 100 \, \text{Gbps}$

**Large Scale**

$> 10,000 \, \text{machines}$
Datacenter Network

**Small Latency**
\[ < 100 \, \mu s \]

**High Bandwidth**
\[ 10/40 \sim 100 \text{ Gbps} \]

**Shallow Buffer**
\[ < 30 \text{ MB for ToR} \]

**Large Scale**
\[ > 10,000 \text{ machines} \]

Congestion control is more challenging in datacenter.
Challenge with small BDP

BDP*(100\mu s, 40Gbps) \approx 300 \text{ MTUs}

* BDP: Bandwidth-delay Product
Rate-based CC + incast traffic

Queueing up to N-1!
Rate-based CC + incast traffic

Total buffer of Trident+

Data Queue

pkts

max

75%-ile

median

25%-ile

min

Number of Flows

32  64  128  256  512  1024  2048

Total buffer of Trident+

1024  512  256  128  64  32

10 k

1 k

100

10

1
Rate-based CC vs. credit-based CC

Data Queue

Number of Flows

DCTCP

Credit-based Approach

Number of Flows

Total buffer of Trident+
Prior Work with Bounded Queue

Credit-based Flow Control
- InfiniBand
- ATM Network
- PCI Express
  + Bounded queue

PFC
- RoCE/DCQCN
  + Bounded queue

Centralized
- FastPass
  + Bounded queue
Prior Work with Bounded Queue

Credit-based Flow Control
- InfiniBand
- ATM Network
- PCI Express

+ Bounded queue
- Does not scale to datacenter
- Requires switch support

PFC
- RoCE/DCQCN

+ Bounded queue
- Head of line blocking
- Possible deadlock

Centralized
- FastPass

+ Bounded queue
- Hard to scale
- Global time sync
- Single point of failure
Prior Work with Bounded Queue

Credit-based Flow Control
- InfiniBand

- Does not scale to datacenter
- Requires switch support

- Head of line blocking
- Possible deadlock

- Hard to scale
- Global time sync
- Single point of failure

Centralized
- RoCE/DCQCN

- PFC

How can we get the benefits of credit-based flow control on Ethernet?
Goal & Our Approach

Goal

To achieve \textit{bounded queue} even with heavy incast using \textit{Ethernet switches}.

ExpressPass

Proactive end-to-end credit-based congestion control using unreliable credits.
ExpressPass
End host behavior

I need credit!

Sender  Credit Request

Data

Credit Stop

Receiver
ExpressPass
End host behavior

Credit
Credit Request
Data
Credit Stop

Sender

Receiver
ExpressPass
End host behavior
ExpressPass
End host behavior

Credit Request

Credit Stop

Data

Credit

Sender

Receiver
ExpressPass
End host behavior

No more !
ExpressPass
End host behavior

stop credit!
ExpressPass
End host behavior

stop credit!

Sender

Credit

Data

Credit Request

Credit Stop

Receiver
ExpressPass
Switch behavior

Switch throttles credits.
(Throttling rate ≈ 5 %)
ExpressPass
Switch behavior

Switch forwards the data symmetric to the credit.
Credit-scheduled data transmission

Receiver

Switch

Senders

Credit

Data

Queue

Number of flows
## Challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Techniques to address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signaling overhead</td>
<td>Piggybacking to handshake packets</td>
</tr>
<tr>
<td>Non-zero queueing</td>
<td>Bounded queue</td>
</tr>
<tr>
<td>Credit waste</td>
<td>Credit feedback control</td>
</tr>
<tr>
<td>Fair drop on switch</td>
<td>Jitter, variable-sized credits</td>
</tr>
<tr>
<td>Path symmetry</td>
<td>Deterministic ECMP, packet level load balancing</td>
</tr>
<tr>
<td>Multiple traffic classes</td>
<td>Prioritizing credits rather than data</td>
</tr>
</tbody>
</table>
Signaling Overhead

- Credit
- Credit Request
- Data
- Credit Stop

Sender → Receiver

Credit Request
Data
Credit
Credit Stop
Maximum Bound of Data Queue

- Credit
- Data

Receiver

Sender 1

Sender 2

Sender 3
Maximum Bound of Data Queue

- Credit
- Data

Receiver

Sender 1: delay = 1
Sender 2: delay = 2
Sender 3: delay = 3
Maximum Bound of Data Queue

- Credit
- Data

Receiver

Senders:
1. Sender 1
2. Sender 2
3. Sender 3
Maximum Bound of Data Queue

Credit  Data

Receiver  Sender 1  Sender 2  Sender 3

Queuing!
Maximum Bound of Data Queue

\[
\text{max(buffer)} = C \ast \{\text{max(delay)} - \text{min(delay)}\}
\]

* Trident+ (10G), Trident II (40G), Tomahawk (100G)
Credit Waste

No more data!

Sender

Receiver
Credit Feedback Control

Proactive Congestion Control

Prevents the congestion *before* actual congestion happens using credits.

Cheap credit drop

We can increase rate aggressively.
Bandwidth probing is cheap.
Convergence can be faster.
Credit Waste & Convergence Time

- **Credit Waste**
  - More Aggressive: 80, 40, 20, 10, 4, 2 credits
  - Less Aggressive: 0

- **Convergence Time**
  - More Aggressive: 2, 3, 4, 5, 6 RTTs
  - Less Aggressive: 14 RTTs
Credit Waste & Convergence Time

Level of Aggressiveness

Rate vs Time

More Aggressive

Less Aggressive

Credit Waste

Convergence Time

RTTs

Rate vs Time

More Aggressive

Less Aggressive
Credit Waste & Convergence Time

Credit Waste

Level of Aggressiveness

More Aggressive

Less Aggressive

Credits

More Aggressive

Less Aggressive

Convergence Time

RTTs

More Aggressive

Less Aggressive

Level of Aggressiveness

80

40

20

10

4

2

2

3

4

5

6

14
Evaluation Setup

Testbed setup
- Dumbbell topology
- Implementation on SoftNIC
- 12 hosts (Xeon E3/E5) connected to single ToR (Quanta T3048)
- Each host has 10Gbps x 1port

NS-2 Simulation Setup
- Fat-tree topology
- 192 hosts / 32 ToR / 16 aggr. / 8 core switches
- Each host has 10Gbps x 1port
Evaluation

(1) Does ExpressPass provides low & bounded queueing with realistic workloads?
(2) Is the convergence fast and stable?
(3) How low & bounded queuing and fast & stable convergence translate into the flow completion time?
# Realistic Workloads

<table>
<thead>
<tr>
<th></th>
<th>Data Mining</th>
<th>Web Search</th>
<th>Cache Follower</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 10KB (S)</td>
<td>78%</td>
<td>49%</td>
<td>50%</td>
<td>63%</td>
</tr>
<tr>
<td>10 – 100KB (M)</td>
<td>5%</td>
<td>3%</td>
<td>3%</td>
<td>18%</td>
</tr>
<tr>
<td>100KB–1MB (L)</td>
<td>8%</td>
<td>18%</td>
<td>18%</td>
<td>19%</td>
</tr>
<tr>
<td>1MB- (XL)</td>
<td>9%</td>
<td>20%</td>
<td>29%</td>
<td>-</td>
</tr>
<tr>
<td>Average flow size</td>
<td>7.41MB</td>
<td>1.6MB</td>
<td>701KB</td>
<td>64KB</td>
</tr>
</tbody>
</table>
## Realistic Workloads

<table>
<thead>
<tr>
<th></th>
<th>Data Mining</th>
<th>Web Search</th>
<th>Cache Follower</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 10KB (S)</td>
<td>78%</td>
<td>49%</td>
<td>50%</td>
<td>63%</td>
</tr>
<tr>
<td>10 – 100KB (M)</td>
<td>5%</td>
<td>3%</td>
<td>3%</td>
<td>18%</td>
</tr>
<tr>
<td>100KB–1MB (L)</td>
<td>8%</td>
<td>18%</td>
<td>18%</td>
<td>19%</td>
</tr>
<tr>
<td>1MB- (XL)</td>
<td>9%</td>
<td>20%</td>
<td>29%</td>
<td>-</td>
</tr>
<tr>
<td>Average flow size</td>
<td>7.41MB</td>
<td>1.6MB</td>
<td>701KB</td>
<td>64KB</td>
</tr>
</tbody>
</table>
Bounded Queue

cache follower workload / load 0.2 – 0.4 / 0KB ~ (All Size)

ExpressPass

Max Queue vs Load

<table>
<thead>
<tr>
<th>Load</th>
<th>Max Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>34</td>
</tr>
<tr>
<td>0.4</td>
<td>41.78</td>
</tr>
<tr>
<td>0.6</td>
<td>32.3</td>
</tr>
</tbody>
</table>

DCTCP

Max Queue vs Load

<table>
<thead>
<tr>
<th>Load</th>
<th>Max Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>153.8</td>
</tr>
<tr>
<td>0.4</td>
<td>178.3</td>
</tr>
<tr>
<td>0.6</td>
<td>214.7</td>
</tr>
</tbody>
</table>
Low Average Queue
cache follower workload / load 0.6 / 0KB –
Low Average Queue

cache follower workload / load 0.6 / 0KB –
Fast & Stable Convergence

ExpressPass

DCTCP

Throughput

Throughput

Gbps

Gbps

Time (us)

Time (ms)

100 us

70 ms

10

10
Fast & Stable Convergence

ExpressPass

Throughput vs. Time (us)

DCTCP

Throughput vs. Time (ms)

ExpressPass reaches 100 Gbps in 100 us, while DCTCP takes 70 ms to achieve x700 of throughput with respect to ExpressPass.
Flow Completion Time

cache follower workload / load 0.6 / 0 – 10KB

<table>
<thead>
<tr>
<th></th>
<th>Average FCT</th>
<th>99%-ile FCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-pass</td>
<td>0.11</td>
<td>0.14</td>
</tr>
<tr>
<td>RCP</td>
<td>0.94</td>
<td>3.54</td>
</tr>
<tr>
<td>DCTCP</td>
<td>0.28</td>
<td>0.47</td>
</tr>
<tr>
<td>DX</td>
<td>0.13</td>
<td>0.17</td>
</tr>
<tr>
<td>HULL</td>
<td>0.13</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>Average FCT</td>
<td>99% - ile FCT</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>X-pass</td>
<td>0.11</td>
<td>0.14</td>
</tr>
<tr>
<td>RCP</td>
<td>0.28</td>
<td>0.47</td>
</tr>
<tr>
<td>DCTCP</td>
<td>0.13</td>
<td>0.17</td>
</tr>
<tr>
<td>DX</td>
<td>0.13</td>
<td>0.18</td>
</tr>
<tr>
<td>HULL</td>
<td>0.94</td>
<td>3.54</td>
</tr>
</tbody>
</table>

Flow Completion Time
(cache follower workload / load 0.6 / 0 – 10KB)
Flow Completion Time

cache follower workload / load 0.6 / 1MB –

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Average FCT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Pass</td>
<td>12.81</td>
</tr>
<tr>
<td>RCP</td>
<td>13.03</td>
</tr>
<tr>
<td>DCTCP</td>
<td>10.81</td>
</tr>
<tr>
<td>DX</td>
<td>32.89</td>
</tr>
<tr>
<td>HULL</td>
<td>55.66</td>
</tr>
</tbody>
</table>
Flow Completion Time

cache follower workload / load 0.6 / 1MB –
Conclusion

• ExpressPass is **end-to-end, credit-scheduled**, and **delay-bounded** congestion control for datacenter.

• ExpressPass propose a new **proactive** datacenter congestion control.

• Our evaluation on testbed and ns-2 simulation show that ExpressPass achieves
  (1) Low & bounded queueing
  (2) Fast & stable convergence
  (3) Short flow completion time especially for small flows
Thanks

Happy to answer your questions