High Speed Networks Need Proactive Congestion Control

Lavanya Jose, Lisa Yan, Nick McKeown, Sachin Katti
Stanford University

Mohammad Alizadeh
MIT

George Varghese
Microsoft Research
The Congestion Control Problem
Ask an oracle.

<table>
<thead>
<tr>
<th>Link 0</th>
<th>Link</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow A</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>Flow B</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>Flow C</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>Flow D</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow A</td>
<td>35</td>
</tr>
<tr>
<td>Flow B</td>
<td>25</td>
</tr>
<tr>
<td>Flow C</td>
<td>5</td>
</tr>
<tr>
<td>Flow D</td>
<td>5</td>
</tr>
</tbody>
</table>

Flow A = 35G  
Flow B = 25G  
Flow C = 5G  
Flow D = 5G
Traditional Congestion Control

• No explicit information about traffic matrix
• Measure congestion signals, then react by adjusting rate after measurement delay
• Gradual, can’t jump to right rates, know direction
• “Reactive Algorithms”
Flow $A = 35G$
Flow $B = 25G$
Flow $C = 5G$
Flow $D = 5G$
Flow $A = 35G$

Flow $B = 25G$

Flow $C = 5G$

Flow $D = 5G$
Flow $A = 35G$

Flow $B = 25G$

Flow $C = 5G$

Flow $D = 5G$
Flow A = 35G
Flow B = 25G
Flow C = 5G
Flow D = 5G

Link 0 100 G
Link 1 60 G
Link 2 30 G
Link 3 10 G
Link 4 100 G

RCP (dashed)
Ideal (dotted)
Link 0 100 G → Link 1 60 G → Link 2 30 G → Link 3 10 G → Link 4 100 G

Flow A = 35G
Flow B = 25G
Flow C = 5G
Flow D = 5G
Flow A = 35G
Flow B = 25G
Flow C = 5G
Flow D = 5G
Flow A = 35G
Flow B = 25G
Flow C = 5G
Flow D = 5G

Link 0 100 G
Link 1 60 G
Link 2 30 G
Link 3 10 G
Link 4 100 G

30 RTTs to Converge
RCP (dashed)
Ideal (dotted)

Transmission Rate (Gbps)
Time (# of RTTs, 1 RTT=24us)
Convergence Times Are Long

- If flows only last a few RTTs, then we can’t wait 30 RTTs to converge.
- At 100G, a typical flow in a search workload is < 7 RTTs long.

Fraction of Total Flows in Bing Workload

- Small (1-10KB): 30%
- Medium (10KB-1MB): 14%
- Large (1MB-100MB): 56%

1MB / 100 Gb/s = 80 µs
Why “Reactive” Schemes Take Long

1. No explicit information
2. Therefore measure congestion signals, react
3. Can’t leap to correct values but know direction
4. Reaction is fed back into network
5. Take cautious steps
Reactive algorithms trade off *explicit flow information* for *long convergence times*

Can we use explicit flow information and get shorter convergence times?
Back to the oracle, how did she use traffic matrix to compute rates?

Flow A = 35G

Flow B = 25G

Flow C = 5G

Flow D = 5G
Waterfilling Algorithm

Link 0 (0/ 100 G)
Link 1 (0/ 60 G)
Link 2 (0/ 30 G)
Link 3 (0/ 10 G)
Link 4 (0/ 100 G)

Flow A (0 G)
Flow B (0 G)
Flow C (0 G)
Flow D (0 G)
Waterfilling - 10 G link is fully used

Link 0 (5/100 G)

Link 1 (10/60 G)

Link 2 (10/30 G)

Link 3 (10/10 G)

Link 4 (5/100 G)

Flow A (5 G)

Flow B (5 G)

Flow C (5 G)

Flow D (5 G)
Waterfilling- 30 G link is fully used

- Link 0 (25/100 G)
- Link 1 (50/60 G)
- Link 2 (30/30 G)
- Link 3 (10/10 G)
- Link 4 (5/100 G)

Flow A (25 G)
Flow B (25 G)
Flow C (5 G)
Flow D (5 G)
Waterfilling- 60 G link is fully used

Link 0 (35/100 G)

Flow A (35 G)
Flow B (25 G)

Link 1 (60/60 G)

Flow C (5 G)
Flow D (5 G)

Link 2 (30/30 G)
Link 3 (10/10 G)
Link 4 (5/100 G)
Fair Share of Bottlenecked Links

Link 0 (35/100 G)
Link 1 (60 G)
Link 2 (30 G)
Link 3 (10 G)
Link 4 (5/100 G)

Flow A (35 G)
Flow B (25 G)
Flow C (5 G)
Flow D (5 G)
A centralized water-filling scheme may not scale.

Can we let the network figure out rates in a distributed fashion?
Fair Share for a Single Link

Capacity at Link 1: 30G
So Fair Share Rate: 30G/2 = 15G

<table>
<thead>
<tr>
<th>flow</th>
<th>demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
</tr>
</tbody>
</table>
A second link introduces a *dependency*

<table>
<thead>
<tr>
<th>flow</th>
<th>demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>100 G</td>
</tr>
</tbody>
</table>
Dependency Graph

Flow A

Flow B

Link 1
30 G

Link 2
10 G
Dependency Graph

Flow A

Flow B

10

Link 1

30 G

Link 2

10 G
Proactive Explicit Rate Control (PERC) Overview

- Flows and links alternately exchange messages.
- A flow sends a “demand”
  - $\infty$ when no other fair share
  - min. fair share of other links
- A link sends a “fair share”
  - $C/N$ when demands are $\infty$
  - otherwise use water-filling
Proactive Explicit Rate Control (PERC) Overview

- Flows and links alternately exchange messages.
- A flow sends a “demand”
  - $\infty$ when no other fair share
  - min. fair share of other links
- A link sends a “fair share”
  - $C/N$ when demands are $\infty$
  - otherwise use water-filling
- Messages are approximate, jump to right values quickly with more rounds

Round 2 (Flows $\rightarrow$ Links)

- Flow A
  - $\infty$ to Link 1, $30 \, G$
- Flow B
  - 15 to Link 2, $10 \, G$
Proactive Explicit Rate Control (PERC) Overview

- Flows and links alternately exchange messages.
- A flow sends a “demand”
  - $\infty$ when no other fair share
  - min. fair share of other links
- A link sends a “fair share”
  - $C/N$ when demands are $\infty$
  - otherwise use water-filling
- Messages are approximate, jump to right values quickly with more rounds
Message Passing Algorithms

Decoding error correcting codes (LDPC- Gallager, 1963)

Flow counts using shared counters (Counter Braids- Lu et al, 2008)

$x_1$ 0
$x_2$ 1
$x_3$ 1

Parity Check 1: $x_1 + x_3 = 0$
Parity Check 2: $x_2 + x_3 = 0$

Flow A
Counter 1
36

Flow B
Counter 2
32
Making PERC concrete
PERC Implementation

Control Packet For Flow B

| d | ∞ | ∞ |
| f | ? | ? |

Link 1 30 G → Link 2 10 G

Flow A

Flow B
PERC Implementation

Control Packet For Flow B

<table>
<thead>
<tr>
<th>d</th>
<th>∞</th>
<th>∞</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>15</td>
<td>?</td>
</tr>
</tbody>
</table>

Link 1 30 G → Link 2 10 G

Flow A

Flow B
PERC Implementation

Control Packet For Flow B

<table>
<thead>
<tr>
<th>d</th>
<th>∞</th>
<th>∞</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>

Link 1
30 G

Link 2
10 G

Flow A

Flow B
PERC Implementation

Control Packet For Flow B

<table>
<thead>
<tr>
<th>d</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Link 1: 30 G
Link 2: 10 G

send at 15G!

Flow A
Flow B
PERC converges fast

![Graph showing PERC convergence time and transmission rates](image)

- **Flow A**: 35G
- **Flow B**: 25G
- **Flow C**: 5G
- **Flow D**: 5G

- **Link 0**: 100 Gbps
- **Link 1**: 60 Gbps
- **Link 2**: 30 Gbps
- **Link 3**: 10 Gbps
- **Link 4**: 100 Gbps

**Time (# of RTTs, 1 RTT=24us)**

- **RCP took 30 RTTs to Converge**
- **PERC (solid)**
- **Ideal (dotted)**

**PERC converges fast**
PERC Converges Fast

CDF

4 vs 14 at Median
10 vs 71 at Tail (99th)
Some unanswered questions

• How to calculate fair shares in PERC switches?
• How to bound convergences times in theory?
• What about other policies?
Takeways

• Reactive schemes are slow for short flows (majority) at 100G
• Proactive schemes like PERC are fundamentally different and can converge quickly because they calculate explicit rates based on out of band information about set of active flows.
• Message passing promising proactive approach—could be practical, need further analysis to understand good convergence times in practice.
Thanks!
Shorter FCTs For Flows That Last A Few RTTs (“Medium”)

Tail FCT (norm. by IDEAL) vs. Flow Size

- Small Flows
- Medium Flows
- Large Flows

100G, 12us

Graph showing Tail FCT and Mean FCT for Small, Medium, and Large Flows, normalized by IDEAL, with comparison among RCP, DCTCP, and PERC.
As XCP is window-based, the EC computes a desired increase or decrease in the number of bytes that the aggregate traffic transmits in a control interval (i.e., an average RTT). This aggregate feedback \( \phi \) is computed each control interval:

\[
\phi = \alpha \cdot d \cdot S - \beta \cdot Q, \tag{1}
\]

\( \alpha \) and \( \beta \) are constant parameters, whose values are set based on our stability analysis (§ 4) to 0.4 and 0.226, respectively. The term \( d \) is the average RTT, and \( S \) is the spare bandwidth defined as the difference between the input traffic rate and link capacity. (Note that \( S \) can be negative.) Finally, \( Q \) is the persistent queue size (i.e., the
$R_l$ based on congestion signals, such as the input traffic rate $y_l(t)$ and queue length $q_l(t)$ at the link at time $t$. Every RTT, $d$, RCP updates $R_l$ at each link as follows:

$$R_l(t) = R_l(t - d) \left( 1 + \frac{\alpha(C_l - y_l(t)) - \beta \frac{q_l(t)}{d}}{C_l} \right),$$
ATM/ Charny etc.

CDF

RTTs to converge

- PERC
- CHARNY
- RCP
Discussion

• Fundamentally any limit on how fast we can get max-min rates? Explicit or implicit whatever