Copa: Practical Delay-Based Congestion Control

Venkat Arun and Hari Balakrishnan

MIT, CSAIL
web.mit.edu/copa
The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)
The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn
The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

- Higher bandwidth-delay product
- Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss
- Greater flow-churn

Large flows (e.g. video streaming) co-exist with short-flows
The Internet is more challenging than ever
(Why are we still talking about congestion control in 2018?)

Higher bandwidth-delay product

Greater bandwidth $\Rightarrow$ Lower tolerance for non-congestive loss

Greater flow-churn

Large flows (e.g. video streaming) co-exist with short-flows

Wireless links with variable bandwidths are commonplace
The Internet is more challenging than ever

(Why are we still talking about congestion control in 2018?)

Greater bandwidth ⇒ Large flows (e.g. video streaming) co-exist with short-flows

Greater bandwidth ⇒ Lower tolerance for non-congestive loss

Greater flow-churn

Simultaneously, users are more sensitive to performance!

Higher bandwidth-delay product

Wireless links with variable bandwidths are commonplace
Loss-based schemes have long-standing problems

- Buffer-filling
- Vulnerable to non-congestive loss
- Loss is a coarse signal

Worsens with increasing bandwidth
Delay-based congestion control?

Benefits

Challenges
Delay-based congestion control?

**Benefits**
- Maintain low delay

**Challenges**
- Not competitive with buffer-filling schemes
Delay-based congestion control?

**Benefits**
- Maintain low delay
- Robust to misleading loss

**Challenges**
- Not competitive with buffer-filling schemes
- Delay can mislead too!
Delay-based congestion control?

**Benefits**
- Maintain low delay
- Robust to misleading loss
- Rich signal

**Challenges**
- Not competitive with buffer-filling schemes
- Delay can mislead too!
### Delay-based congestion control?

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain low delay</td>
<td>Not competitive with buffer-filling schemes</td>
</tr>
<tr>
<td>Robust to misleading loss</td>
<td>Delay can mislead too!</td>
</tr>
<tr>
<td>Rich signal</td>
<td>Finding true minimum RTT is hard</td>
</tr>
</tbody>
</table>
Delay-based congestion control?

**Challenges**

- Not competitive with buffer-filling schemes
- Delay is noisy too!
- Finding true minimum RTT is hard

**Our Solution**
Delay-based congestion control?

**Challenges**

- Not competitive with buffer-filling schemes
- Delay is noisy too!
- Finding true minimum RTT is hard

**Our Solution**

- If buffer-fillers are present, give up on low delay
Delay-based congestion control?

**Challenges**

- Not competitive with buffer-filling schemes
- Delay is noisy too!
- Finding true minimum RTT is hard

**Our Solution**

- If buffer-fillers are present, give up on low delay
- MIN: a more robust statistic for queuing delay
Delay-based congestion control?

**Challenges**

Not competitive with buffer-filling schemes

Delay is noisy too!

Finding true minimum RTT is hard

**Our Solution**

If buffer-fillers are present, give up on low delay

MIN: a more robust statistic for queuing delay

Empty queues periodically
Basic Goals

Avoid *congestion collapse*

+ Efficient and Fair allocation of bandwidth

+ Low delay
Target rate $= r_t = \frac{1}{\delta \cdot d_q}$

Adjustable Parameter default = 0.5

Queuing delay
Target Rate $\equiv$ Nash Equilibrium

Selfishly optimize for:

$$Utility_i = \log(tput) - \delta_i \log(d_q)$$

Assuming Poisson arrivals (more details in paper)
Target Rate $\equiv$ Nash Equilibrium

Selfishly optimize for:

$$Utility_i = \log(tput) - \delta_i \log(d_q)$$

Assuming Poisson arrivals (more details in paper)
Computing the Target Rate
Estimating queuing delay from RTT
Estimating queuing delay from RTT

\[ \text{Queuing delay} = \text{RTT} - \text{RTT}_{\text{min}} \]
Estimating queuing delay from RTT

New flow starts here

RTT

True minimum RTT

Time

Wrong!
Estimating queuing delay from RTT

True minimum RTT = $\text{RTT}_{\text{min}}$ for new flows!
A “noisy” cellular link: Stanford to AWS
Decoupling queuing delay from other delay variation

- Wireless links
- Cross traffic
- ACK compression
- Bursty transmission
- ...

Queue length vs. Time

Measured RTT
Decoupling queuing delay from other delay variation

Queue length

Positive Additive Noise!

Take Min over last 0.5 RTT of samples

Time

- Wireless links
- Cross traffic
- ACK compression
- Bursty transmission
- ...
A “noisy” cellular link: Stanford to AWS

Using the MIN delay estimator improves throughput from 0.5 Mbits/s to 3.9 Mbits/s
Attaining the Target
The Copa Algorithm

Calculate target rate $r_t = \frac{1}{\delta d_q}$

If current rate $< r_t$: additively increase by $\frac{v}{\delta}$ pkts/RTT
Else: additively decrease by $\frac{v}{\delta}$ pkts/RTT
The Copa Algorithm

Calculate target rate $r_t = \frac{1}{\delta d_q}$

If current rate $< r_t$: additively increase by $\frac{v}{\delta}$ pkts/RTT
Else: additively decrease by $\frac{v}{\delta}$ pkts/RTT
The Copa Algorithm

Calculate target rate: 
\[ r_t = \frac{1}{\delta d_q} \]

If current rate < \( r_t \): additively increase by \( \frac{v}{\delta} \) pkts/RTT
Else: additively decrease by \( \frac{v}{\delta} \) pkts/RTT
Steady-State Dynamics of Copa

Queue Length (in pkts) vs Time

- Period: 5 RTTs
- Amplitude: $2.5 \delta^{-1}$

Equilibrium queue length
Steady-State Dynamics of Copa

Queue Length (in pkts)

0 RTT 0.5 RTT 1.5 RTT 2.5 RTT 3 RTT 4 RTT

Time

Switch point

0.5 RTT window

Queue length corresponding to $\text{RTT}_{\text{standing}}$ at the switch point

Period: 5 RTTs
Amplitude: $2.5 \delta^{-1}$

Equilibrium queue length
Steady-State Dynamics of Copa

Queue Length (in pkts)

Switch point

Period: 5 RTTs
Amplitude: 2.5 $\delta^{-1}$
Steady-State Dynamics of Copa

Queue Length (in pkts)

Period: 5 RTTs
Amplitude: 2.5 $\delta^{-1}$
Steady-State Dynamics of Copa

Queue Length (in pkts)

Period: 5 RTTs
Amplitude: 2.5 $\delta^{-1}$

Switch point

Queue length corresponding to $\text{RTT}_{\text{standing}}$ at the switch point

Equilibrium queue length
Queue empties every 5 RTTs! ⇒

Estimate true minimum RTT

Detect buffer-filling TCP
TCP-Competitiveness
Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)
Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)
Mode switching for TCP competitiveness

Delay sensitive
($\delta = 0.5$)

TCP Competitive
(AIMD on $\delta^{-1}$)
Mode switching for TCP competitiveness

Delay sensitive
\( (\delta = 0.5) \)

TCP Competitive
(AIMD on \( \delta^{-1} \))

![Graph showing mode switching for TCP competitiveness. The graph illustrates different phases of low delay, low throughput, low delay, high delay, high throughput, and high delay for Copa and TCP Cubic. The X-axis represents time (s) and the Y-axis represents throughput (Mbits/s) and queuing delay (ms).]
Mode switching for TCP competitiveness

Best of both worlds!

When queue doesn’t empty once every 5 RTTs, switch to TCP Competitive mode!
Copa gets higher throughput without hurting TCP Cubic!
Copa gets higher throughput without hurting TCP Cubic!
Copa gets higher throughput without hurting TCP Cubic!
Limitations

• Cannot ignore low frequency noise

• Queues don’t empty periodically if:
  • Propagation delay is much smaller than queuing delay
  • Flows with very different propagation delays share a bottleneck queue

• Needs precise RTT measurements
Consistent Performance on Real Paths

Cellular Networks

Avg. Normalized Throughput vs. Avg. Queuing Delay (ms)

Wired Networks

Avg. Normalized Throughput vs. Avg. Queuing Delay (ms)
Satellite link: High BDP, high loss

Throughput (Mbit/s) vs. Median Queuing Delay (ms)

- PCC
- Copa
- Remy
- BBR
- Cubic Vegas

Better
Fairness during flow-churn

Throughput (Mbit/s)

Avg ± Std. Dev.

Time (s)
Fairness during flow-churn

![Graph showing throughput and Jain index over time for Copa, BBR, Cubic, and PCC protocols. The graph includes a zoomed-in view highlighting the Jain index and throughputs for each protocol.](image-url)
Summary

A *practical* delay-based congestion control algorithm

https://web.mit.edu/copacopa
A *practical* delay-based congestion control algorithm

https://web.mit.edu/copa
A *practical* delay-based congestion control algorithm

https://web.mit.edu/copa

---

**Summary**

**Move toward target**

(AIAD-variant)

---

**Estimate queuing delay**

and compute target

---

**Outer control loop**

Mode switching and Competitive mode

---

**Rate/cwnd**

---

**Network**