Adapting TCP for Reconfigurable Datacenter Networks

*Carnegie Mellon University, ○UC San Diego, †Nefeli Networks, ‡Microsoft Research

February 26, 2020
Reconfigurable Datacenter Network (RDCN)

all-to-all connectivity

higher bandwidth, between certain racks

RDCN is a black box:
Do not segregate flows between networks

[Liu, NSDI '14]
2010: RDCNs speed up DC workloads

Hybrid networks achieve higher performance on datacenter workloads

Figure 9: The completion of Hadoop Gridmix tasks

[Wang, SIGCOMM '10]
Advances in circuit switch technology have led to a 10x reduction in reconfiguration delay ⇒ today, circuits can reconfigure much more frequently.

**Better for datacenters:** More flexibility to support dynamic workloads

**Better for hosts:** Less data must be available to saturate higher bandwidth NW

---

[Porter, SIGCOMM '13]
Short-lived circuits pose a problem for TCP

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s

No TCP variant makes use of the high-bandwidth circuits
TCP cannot ramp up during short circuits

achieved bandwidth (BW) = slope

8x BW

1x BW

no circuit

180 μs

circuit

no circuit

What we expect vs. reality
What is the problem?

All TCP variants are designed to adapt to changing network conditions
• E.g., congestion, bottleneck links, RTT

But *bandwidth fluctuations* in modern RDCN are an order of magnitude
*more frequent* (10x shorter circuit duration) and *more substantial* (10x
higher bandwidth) than TCP is designed to handle
• RDCNs break the implicit assumption of relatively-stable network
  conditions

This requires an *order-of-magnitude shift in how fast TCP reacts*
This talk: Our 2-part solution

**In-network:** Use information about upcoming circuits to transparently “trick” TCP into ramping up more aggressively

- High utilization, at the cost of tail latency

**At endhosts:** New TCP variant, reTCP, that explicitly reacts to circuit state changes

- Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together
Naïve idea: Enlarge switch buffers

**Want we want:** TCP’s congestion window ($cwnd$) to parallel the BW fluctuations

**First attempt:** Make $cwnd$ large all the time  

**How?** Use large ToR buffers

---

**Diagram:**

- **Available bandwidth** vs **Time**
- **$cwnd$** vs **Time**
  - Purple dotted line: desired
  - Blue solid line: large, static buffers
Naïve idea: Enlarge switch buffers

Sender → ToR buffer → Circuit Switch → ToR buffer → Receiver

low BDP

high BDP
Naïve idea: Enlarge switch buffers

Larger ToR buffers increase utilization of the high-BDP circuit network
Naïve idea: Enlarge switch buffers
Large queues increase utilization...

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s
...but result in high latency

**Median latency**

**99th percentile latency**

How can we improve this latency?

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s
Use large buffers only when circuit is up

*Dynamic buffer resizing:* Before a circuit begins, transparently enlarge ToR buffers

Full circuit utilization with a latency degradation only during ramp-up period
Resize ToR buffers before circuit begins

Circuit coming!
Resize ToR buffers before circuit begins
Resize ToR buffers before circuit begins
Resize ToR buffers before circuit begins
Configuring dynamic buffer resizing

**How long in advance should ToR buffers resize ($\tau$)?**

- Long enough for TCP to grow $cwnd$ to the circuit BDP

**How large should ToR buffers grow to?**

- circuit BDP = $80 \text{ Gb/s} \times 40 \mu s = 45 \text{ 9000-byte packets}$

For our configuration, the ToR buffers must hold $\sim 40$ packets to achieve 90% utilization, which requires $1800 \mu s$ of prebuffering

We resize ToR buffers between sizes of 16 and 50 packets
How long in advance to resize, $\tau$?

**no circuit**

- achieved bandwidth (BW) = slope
- util./latency trade-off

**circuit**

- 180 $\mu$s

---

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

- optimal
- 2400 $\mu$s 98%
- 1800 $\mu$s 91%
- 600 $\mu$s 65%
- 0 $\mu$s 49%
- packet NW only
1800μs of prebuffering yields 91% util.

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets
Latency degradation during ramp-up

**Median latency**

- Static buffers (vary size)
- Dynamic buffers (vary $\tau$)

**99th percentile latency**

- Static buffers (vary size)
- Dynamic buffers (vary $\tau$)

**We cannot use large queues for so long. Can we get the same high utilization with shorter prebuffering?**

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets
This talk: Our 2-part solution

**In-network:** Use information about upcoming circuits to transparently “trick” TCP into ramping up more aggressively
  - High utilization, at the cost of tail latency

**At endhosts:** New TCP variant, *reTCP*, that explicitly reacts to circuit state changes
  - Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together
This talk: Our 2-part solution

**In-network:** Use information about upcoming circuits to transparently “trick” TCP into ramping up more aggressively
  - High utilization, at the cost of tail latency

**At endhosts:** New TCP variant, reTCP, that explicitly reacts to circuit state changes
  - Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together
reTCP: Rapidly grow $cwnd$ before a circuit

1) Communicate circuit state to sender TCP
2) Sender TCP reacts by multiplicatively increasing/decreasing $cwnd$

Diagram:
- Available bandwidth vs. time
- $cwnd$ vs. time
- Desired behavior
- Large, static buffers
- Dynamic buffers
- Dynamic buffers + reTCP

Resize!
reTCP: Explicit circuit state feedback

Circuit coming!

Packet Switch

Reuse existing ECN-Echo (ECE) bit

Sender

ToR buffer

Packet Switch

ToR buffer

Receiver

reTCP marks:
reTCP: Explicit circuit state feedback

Circuit coming!

Reusing existing ECN-Echo (ECE) bit

Sender

ToR buffer

Packet Switch

ToR buffer

Circuit Switch

Receiver

0 → 1 increase!

reTCP marks: 0
reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit
reTCP: Explicit circuit state feedback

Reusing existing ECN-Echo (ECE) bit

Sender -> ToR buffer -> Packet Switch -> ToR buffer -> Receiver

Circuit Switch

reTCP marks: 1 1

1 \rightarrow 0 decrease!
Single multiplicative increase/decrease

On $0 \rightarrow 1$ transitions:
\[
\text{cwnd} = \text{cwnd} \times \alpha
\]

On $1 \rightarrow 0$ transitions:
\[
\text{cwnd} = \frac{\text{cwnd}}{\alpha}
\]

\(\alpha\) depends on ratio of circuit BDP to ToR queue capacity:
- Circuit network BDP: 45 packets
- Small ToR queue capacity: 16 packets

We use \(\alpha = 2\)

More advanced forms of feedback are possible
Dynamic buffers + reTCP achieve high utilization

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets
Short prebuffer time means low latency

**Median latency**

**99th percentile latency**

With 150μs of prebuffering, dynamic buffers + reTCP achieve 93% circuit utilization with an only 1.20x increase in tail latency

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets
Limitations and future work

Dynamic buffer resizing and reTCP are designed to be minimally invasive
• Higher performance may be possible by involving the end-host further

Our evaluation used a simple traffic pattern to isolate TCP’s behavior
• Important to consider complex workloads as well

Is TCP the right protocol for hybrid networks?
Summary: Adapting TCP for RDCNs

Bandwidth fluctuations in reconfigurable datacenter networks break TCP’s implicit assumption of relative network stability

Two techniques to ramp up TCP during short-lived circuits
  - *Dynamic buffer resizing*: Adapt ToR queues to packet or circuit network
  - *reTCP*: Ramp up aggressively to fill new queue capacity

Etalon emulator open source at: [github.com/mukerjee/etalon](http://github.com/mukerjee/etalon)

*Christopher Canel ~ ccanel@cmu.edu*

Thank you!
One more thing: Etalon emulator

Click hybrid switch (physical host)

Emulated rack 1 (physical host)
  Container 1
  ...
  Container M

Emulated rack N (physical host)
  Container 1
  ...
  Container M
One more thing: Etalon emulator

Use *time dilation* to emulate high-bandwidth links
- “slows down” rest of the machine
- *libVT*: Catches common syscalls

*Flowgrind* to generate traffic

*Strobe schedule*: Each rack pair gets a circuit for an equal share.
Summary: Adapting TCP for RDCNs

Bandwidth fluctuations in reconfigurable datacenter networks break TCP’s implicit assumption of relative network stability.

Two techniques to ramp up TCP during short-lived circuits:

- **Dynamic buffer resizing**: Adapt ToR queues to packet or circuit network
- **reTCP**: Ramp up aggressively to fill new queue capacity

Etalon emulator open source at: github.com/mukerjee/etalon

Christopher Canel ~ ccanel@cmu.edu

Thank you!
Backup Slides
Circuit uptime impacts FCT

Simulation; packet network: 10 Gb/s; circuit network: 80 Gb/s
Buffer resizing benefits many TCP variants

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small queues: 16 packets; large queues: 50 packets
Higher latency percentiles perform similarly

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small queues: 16 packets; large queues: 50 packets