Datacenter Transport Control in the Era of Small Buffers

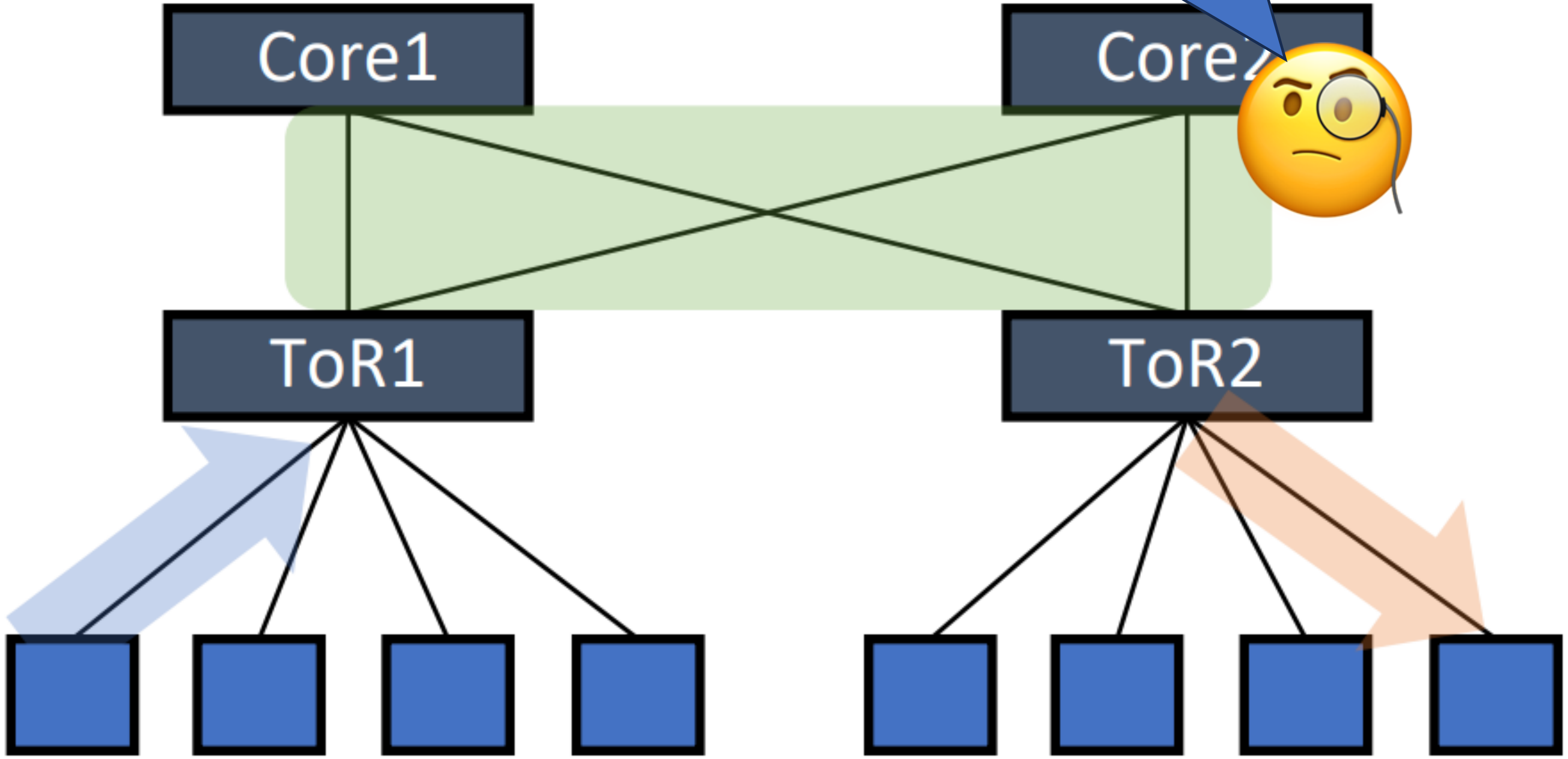
Marios Kogias

IMPERIAL

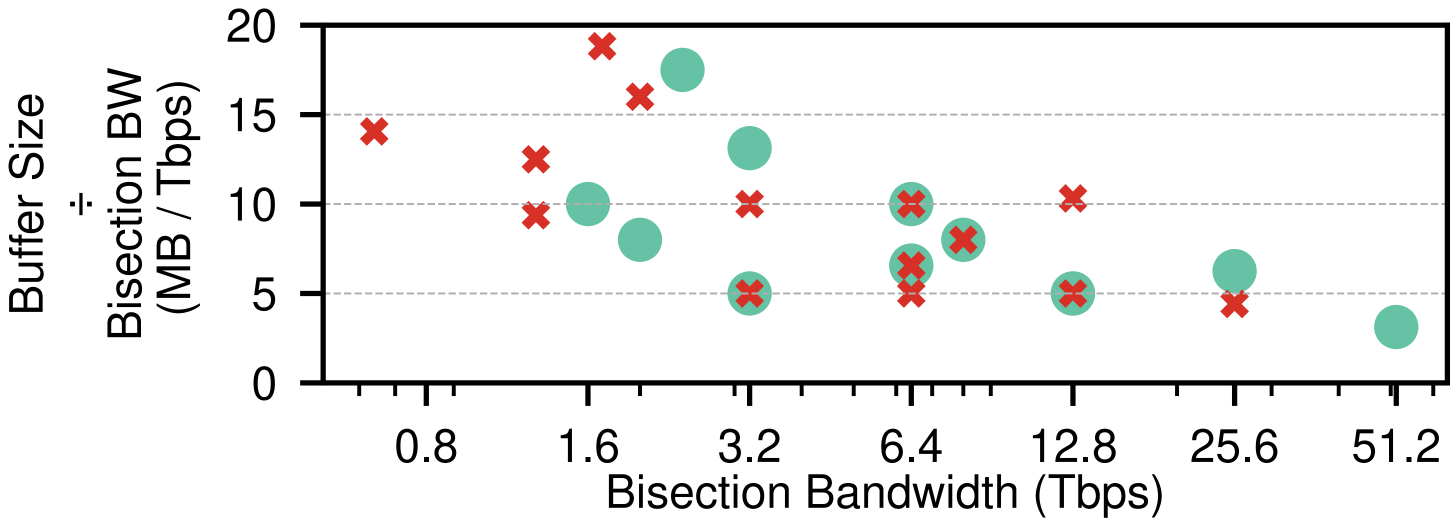
DC Networks & Queuein

Even that is an old problem.
Why is it more crucial to
solve it now?

- Well-known (fixed) topology
- Specific workload patterns
- High bandwidth
- µs-scale RTTs



Switch Buffer Trends



The available switch memory does not follow the bandwidth increase trends

Congestion Control Scheme Classification

Sender-Driven

- Reactive schemes based on:
  - Packet loss
  - ECN marking
  - Delay
- Senders adjust their sending rate after distributed coordination
- Handle all types of congestion
- Slow to react -> More queueing
- Examples: DCTCP, Swift

Receiver-Driven

- Proactive schemes
- Receivers explicitly ask senders to send through special packets (tokens, grants,…)
- Only handle ToR downlinks, but do that great!
- Assume a single link owner
- Hardware dependency and persistent queueing
- Examples: Homa, NDP

Problem Statement

Can we have a congestion control scheme that:

- Handles all types of congestion
- Deals with incast equally good as other RD schemes
- Minimizes buffer requirements
- Is deployable on existing network infrastructure

Yes! SIRD can achieve all the above

SIRD Insights

- Treat **single-owner** bottlenecks **proactively** and **shared** ones **reactively** through a unified mechanism.
  - ☞ RD scheme that reacts to congestion signals
  - ☞ Practically: Dynamically adjust the rate of issuing grants

- Aggressively reduce buffering by carefully restricting the amount of bytes in the network.
  - ☞ Informed overcommitment for high utilization and low queueing
  - ☞ Practically: Issue grants only to receivers that can use them

SIRD Design Summary

- Receiver driven scheme

- Cap the overall #credits per receiver with a global bucket of grants B
  - B is fixed and slightly above BDP to guarantee utilization
- Cap the #credits per sender in every receiver with a per sender bucket SB
  - SB is dynamic and managed by the receiver based on congestion signals

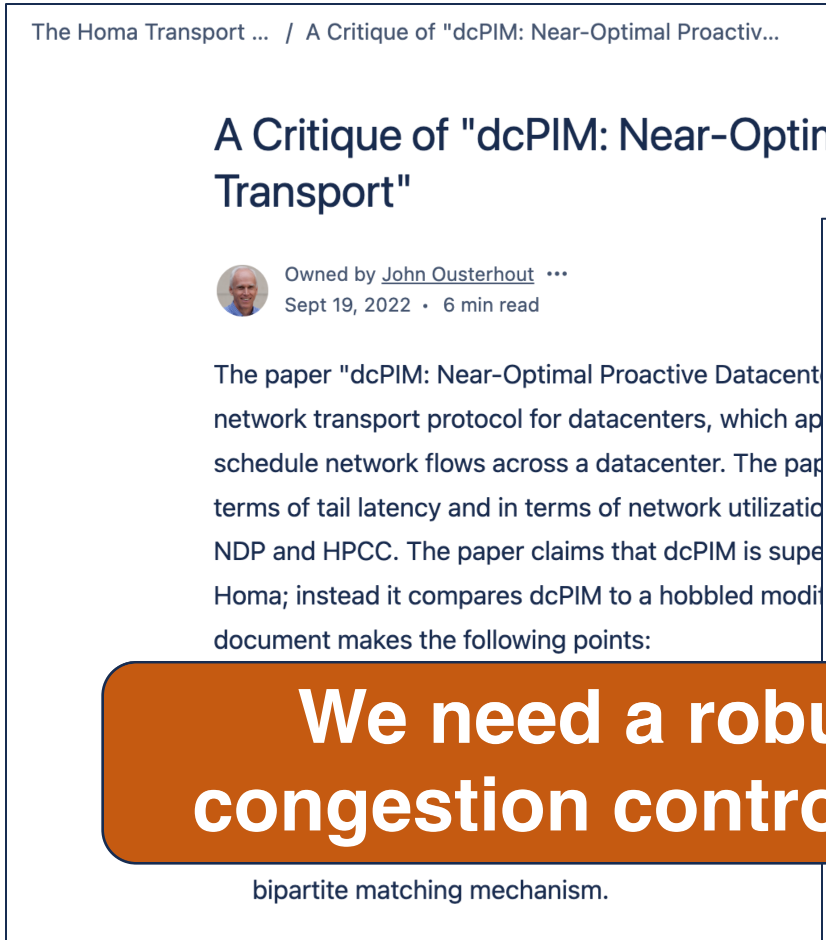
- ECN marking for core congestion
- **C**ongested **S**ender **N**otification (CSN) to deal with sender congestion and prevent credit accumulation

Design Details

- How to configure B?
- Does SIRD add an extra RTT to request credits?
- How to use network priorities if available?
- How does pacing help SIRD?
- How does SIRD support receiver and sender policies?

Evaluation - Simulation

Problem 1: I don’t trust/agree with your simulator configuration



The Homa Transport ... / A Critique of "dcPIM: Near-Optimal Proactiv...

Summarize

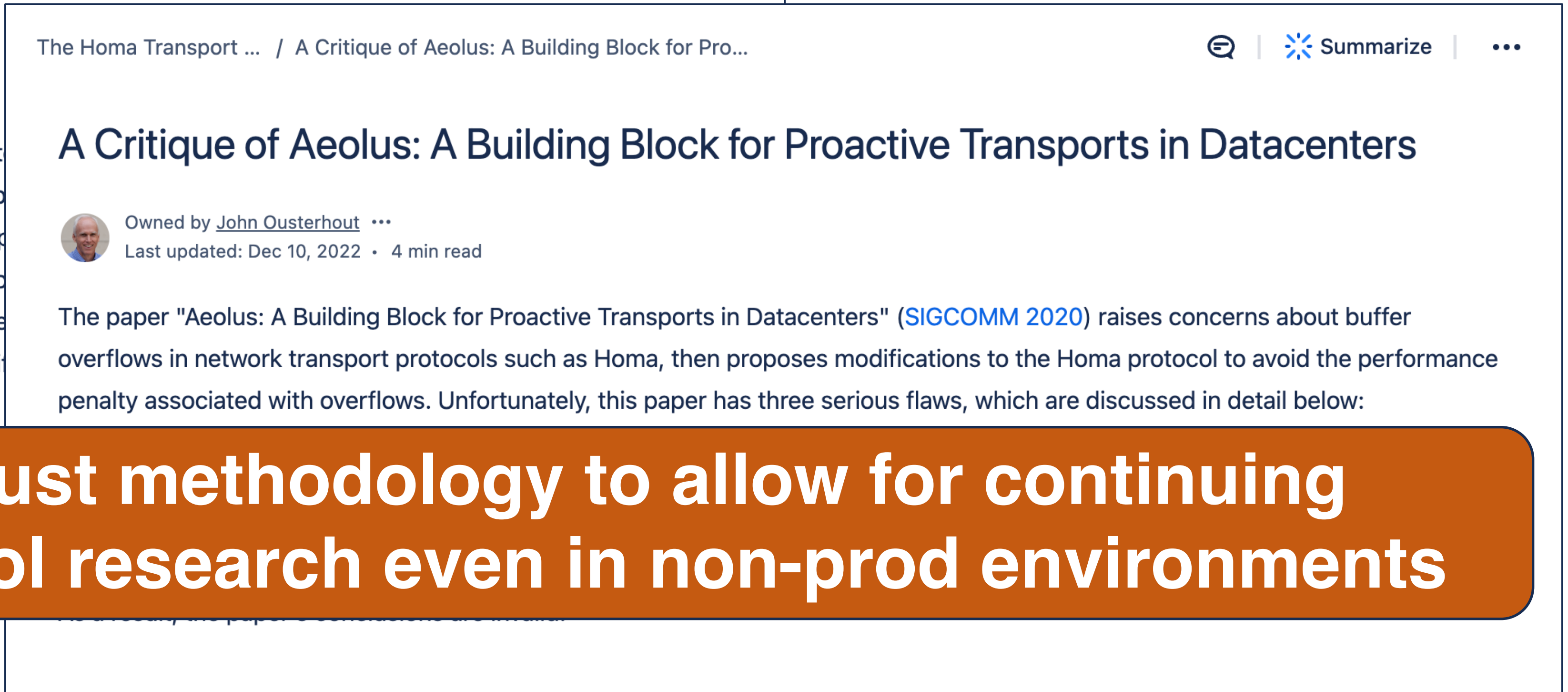
A Critique of "dcPIM: Near-Optimal Proactive Datacenter Transport"

Owned by John Ousterhout

Sept 19, 2022 • 6 min read

The paper "dcPIM: Near-Optimal Proactive Datacent
network transport protocol for datacenters, which ap
schedule network flows across a datacenter. The pap
terms of tail latency and in terms of network utilizatio
NDP and HPCC. The paper claims that dcPIM is supe
Homa; instead it compares dcPIM to a hobbled modi
document makes the following points:

bipartite matching mechanism.



The Homa Transport ... / A Critique of Aeolus: A Building Block for Pro...

Summarize

A Critique of Aeolus: A Building Block for Proactive Transports in Datacenters

Owned by John Ousterhout

Last updated: Dec 10, 2022 • 4 min read

The paper "Aeolus: A Building Block for Proactive Transports in Datacenters" (SIGCOMM 2020) raises concerns about buffer overflows in network transport protocols such as Homa, then proposes modifications to the Homa protocol to avoid the performance penalty associated with overflows. Unfortunately, this paper has three serious flaws, which are discussed in detail below:

We need a robust methodology to allow for continuing congestion control research even in non-prod environments

Evaluation - Simulation

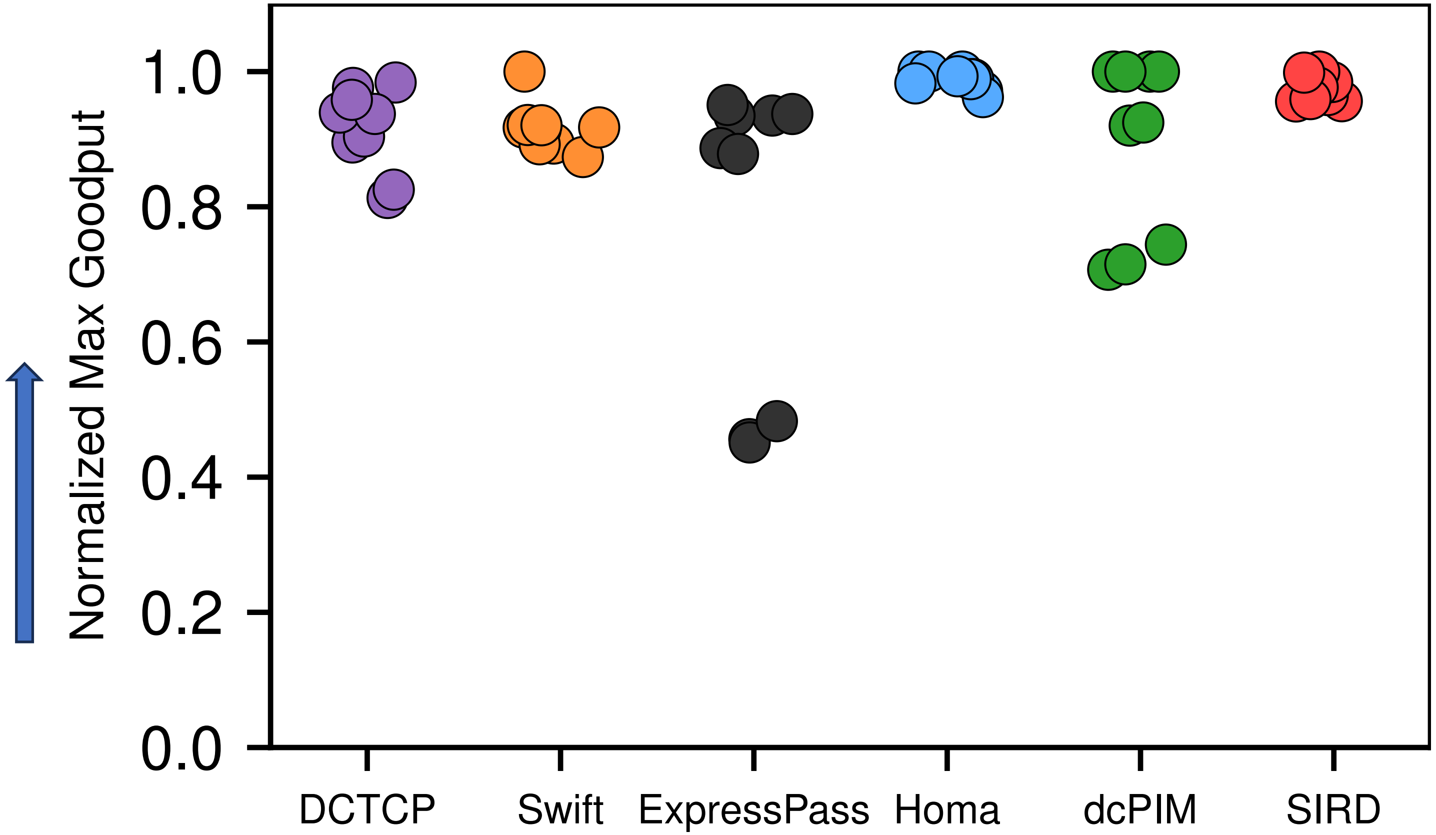
Problem 2: It is a multi-objective and workload-specific problem

- **Baselines**: DCTCP, Swift, Homa, dcPIM, ExpressPass
- **Traffic**: 1) All-to-all, 2) Oversubscribed, 3) Incast overlay
- **Workloads**: 1) Google all RPC 2) FB Hadoop 3) Websearch
- **Metrics:** Goodput, Tail latency, Queuing

SIRD is the only protocol that consistently achieves near-ideal scores across all metrics.

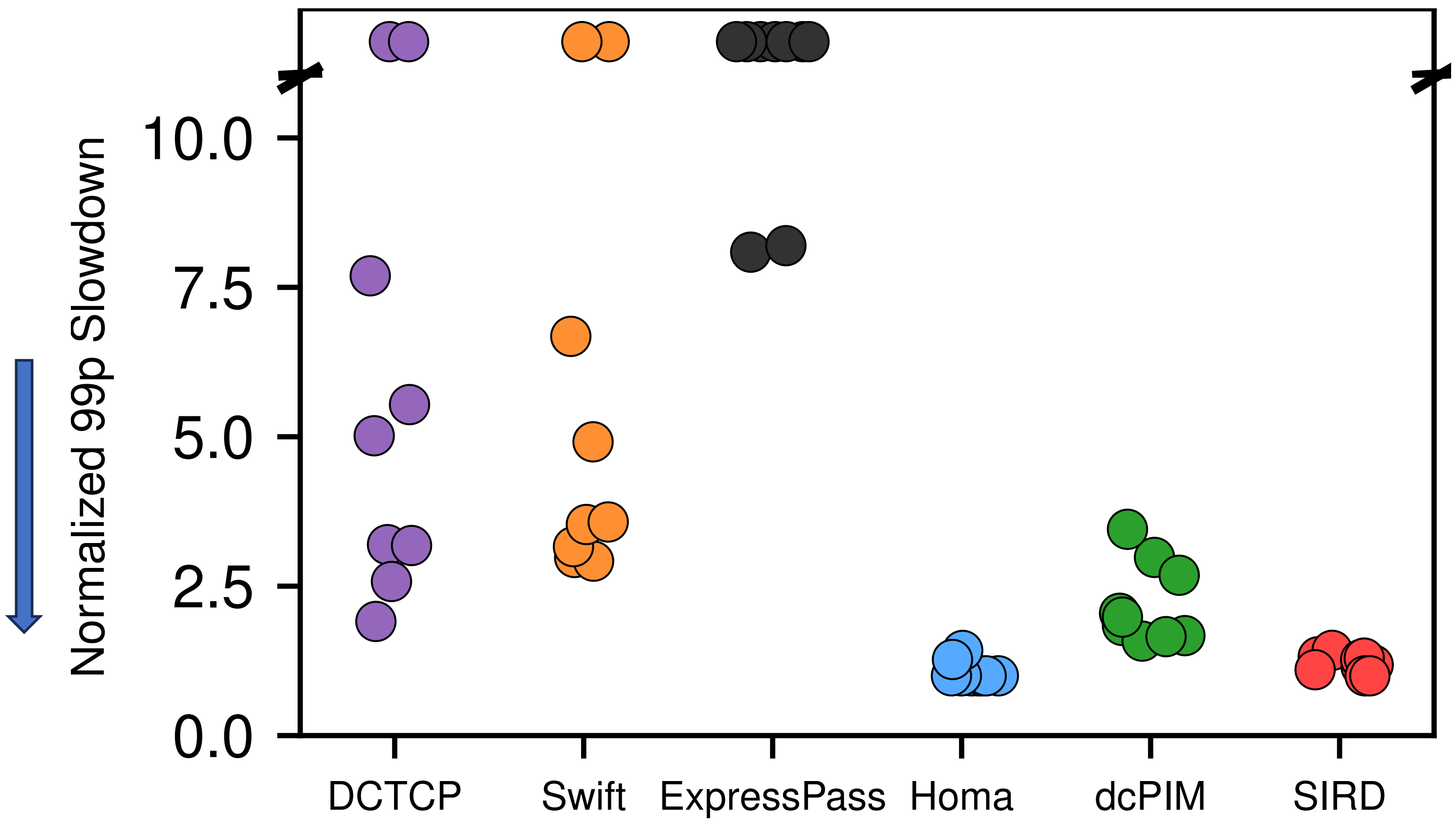
Maximum Goodput

Higher is better



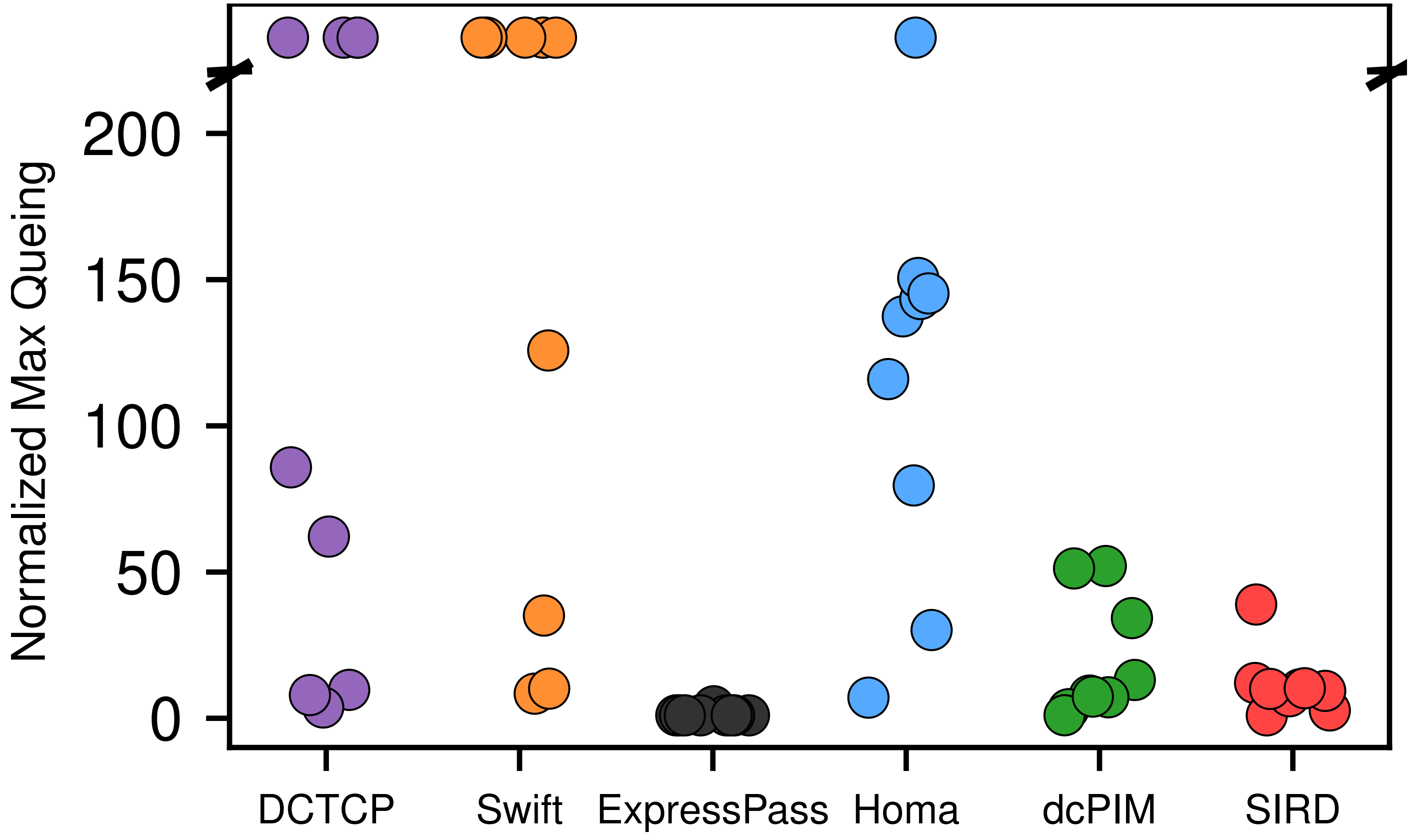
Slowdown @ 50% Load

Lower is better



Maximum Queuing

Lower is better



Thanks

Konstantinos Prasopoulos

Edouard Bugnion

https://arxiv.org/pdf/2312.15403

SIRD Summary

SIRD: A Sender-Informed Receiver-Driven CC scheme

1. <u>Generality</u>
   => explicitly handle all bottlenecks through network feedback
2. <u>Both high BW utilization and minimal queuing</u>
   => Distribute *limited* credit *efficiently*
3. <u>Minimal deployment assumptions</u>
   => commodity switches & no requirement for priorities

Takeaways

- People keep trying to solve datacenter congestion control…
  - There are still cool ideas to explore

- HW trends: Switch memory does not follow bandwidth increase.

- Methodology challenge: We need a robust approach to evaluate and compare CC schemes across a range of scenarios.

Thank you!