Burst-tolerant Datacenter Networks with VERTIGO

Sepehr Abdous*, Erfan Sharafzadeh*, Soudeh Ghorbani

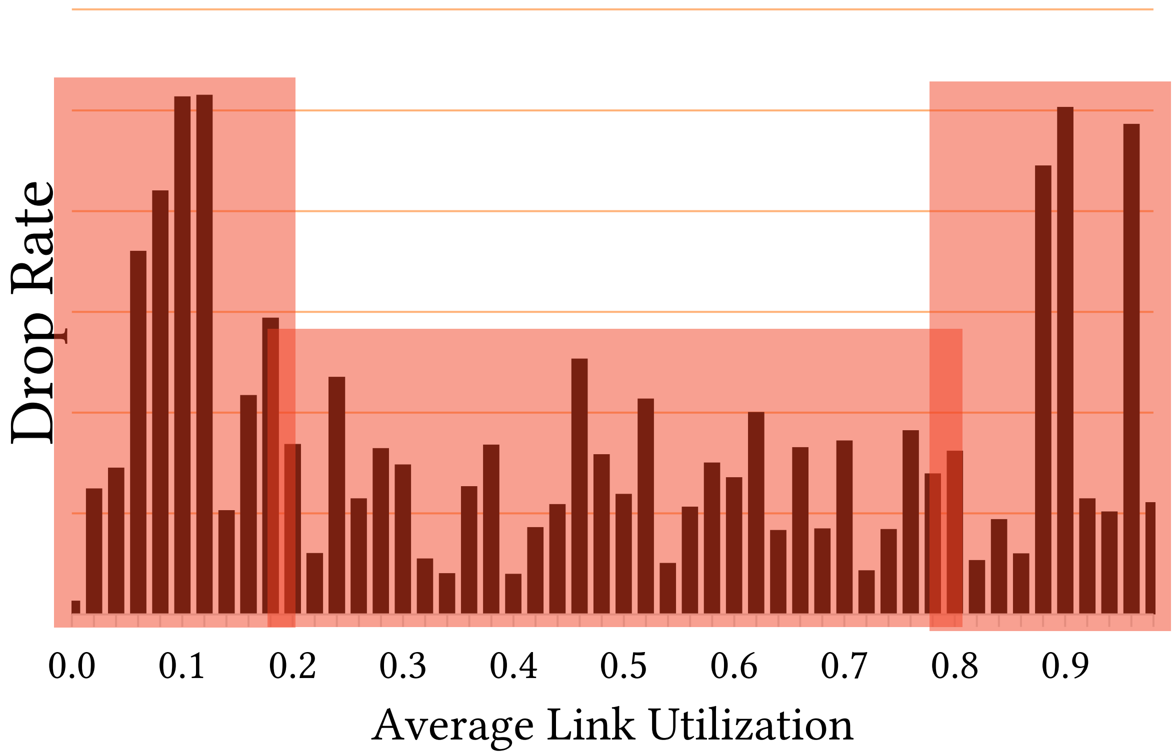
***Co-first Authors**

The 17th International Conference on emerging Networking EXperiments and Technologies, **CoNEXT '21**

JOHNS HOPKINS UNIVERSITY

Datacenter traffic is **bursty** in short timescales

Majority of drops are due to microbursts



[Zhang et al., “High-Resolution Measurement of Data Center Microbursts.”, IMC ’17]

!Microbursts!

High utilization periods in switch buffers that lasting 10s of µseconds

Edge-centric congestion control: slow for microbursts

Congestion control using queue occupancy data

- HPCC [SIGCOMM '19]

Congestion control using round-trip time variations

- Swift [SIGCOMM '20]

Deployed at the edge

Require at least **1 RTT** to identify and recover from packet loss

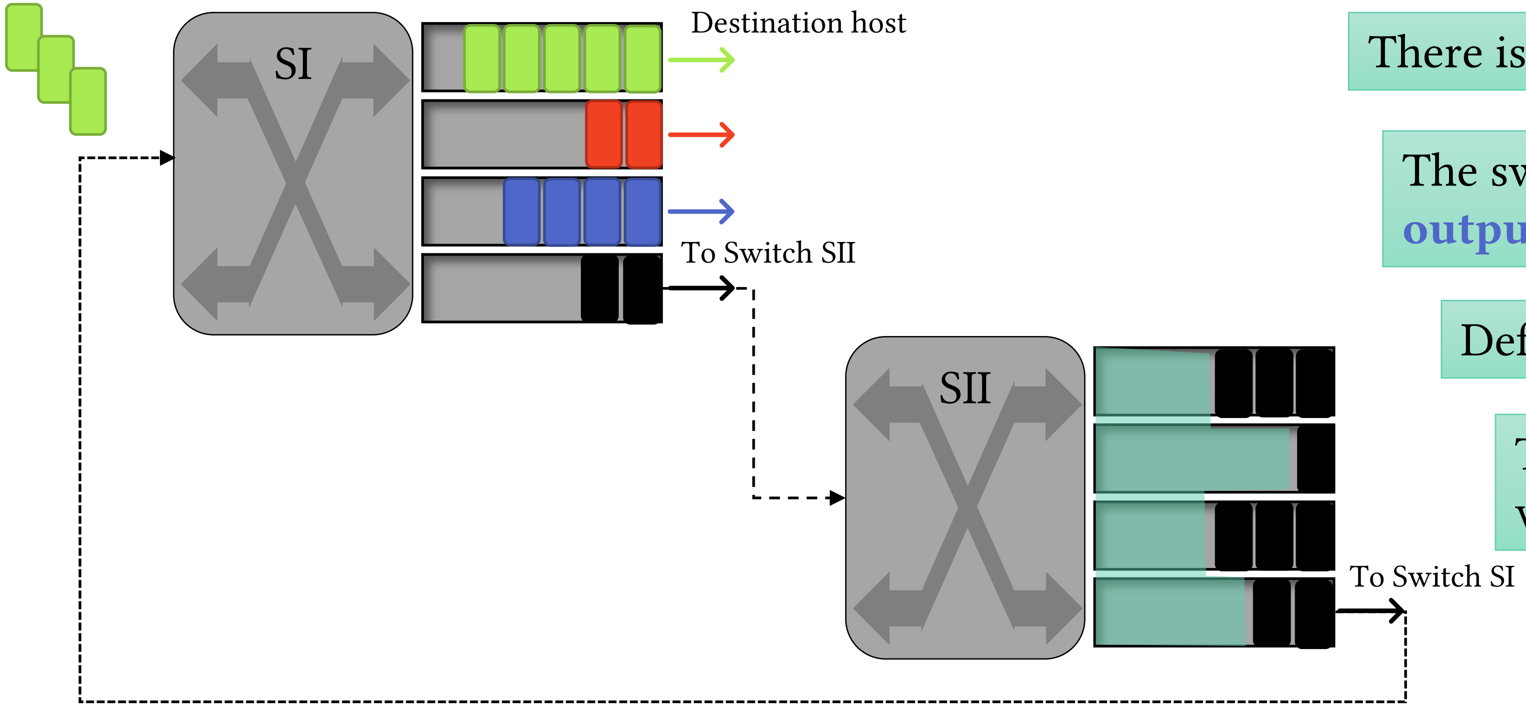
Edge is still slow for microbursts.

Why not react to them in the **network core**?

Goal: Managing microbursts in the network, in real-time

Deflection: a realization of in-network reaction

Randomly re-routing packets that arrive at a full buffer



There is plenty of free buffer in **neighbors**

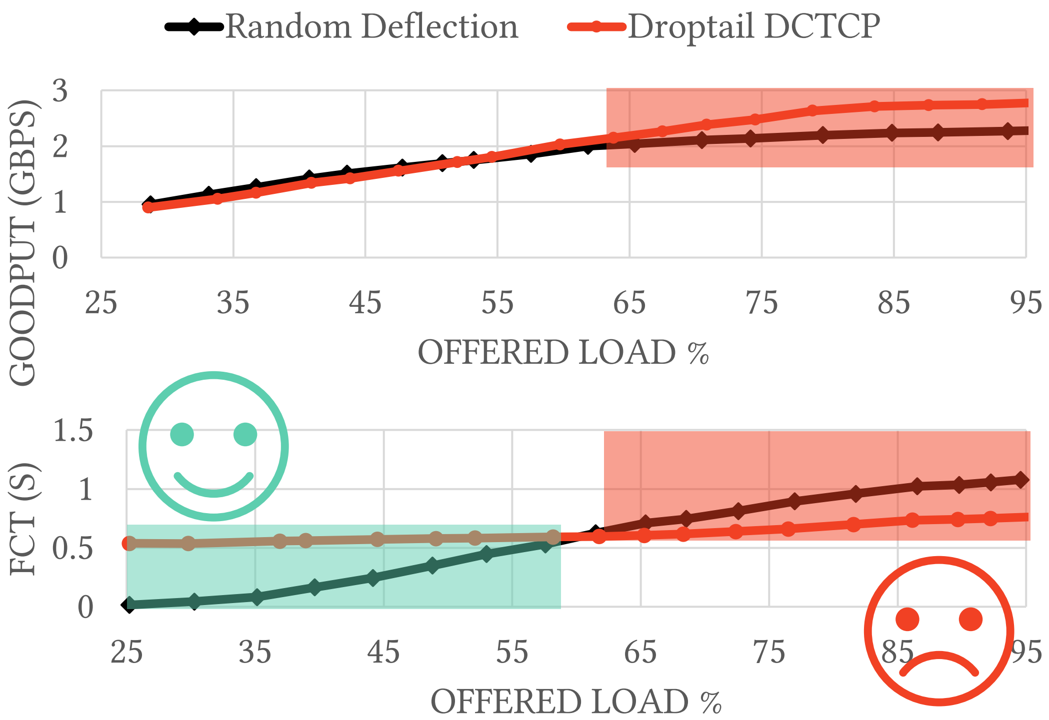
The switch pushes the packet into **another output buffer** instead of **dropping** it

Deflected packets shortly linger in the network

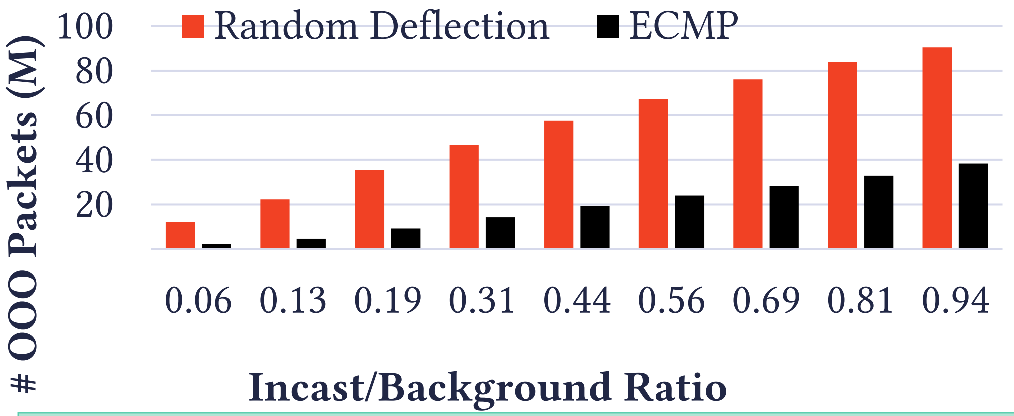
There is enough room in the destination buffer when they arrive back at ToR

Challenges of random deflection

Setup
- 4-8-40 Two-tiered leaf-spine
- 10GB server-to-ToR, 40GB aggregate links
- DCTCP transport
- Workload: FB cache, fixed background + variable Incast



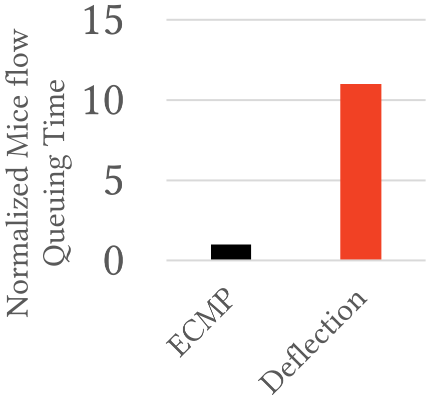
1. Deflection **collapses** under high loads.



2. Deflection causes heavy reordering

Up to **10x** more out-of-order packets

~17% Goodput reduction

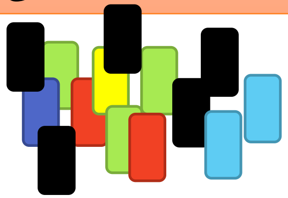


3. Deflection leads to head of line blocking & starvation

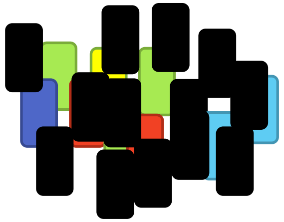
111% longer waits for **mice** flows (<100KB)

Random deflection causes head-of-the-line blocking

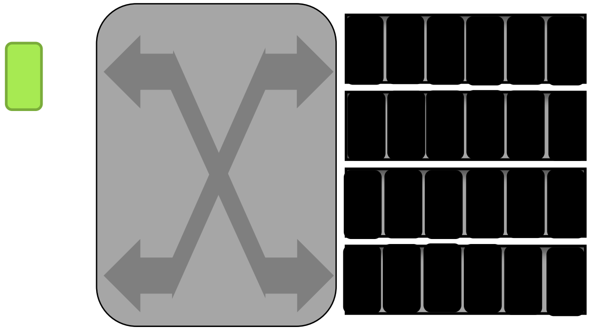
Random deflection saves flows regardless of their **size**



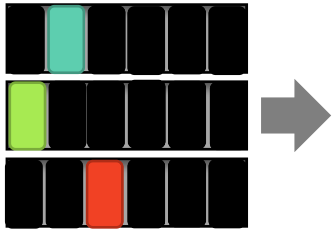
A Large flow **continues** to send traffic instead of **backing off**



Neighbor buffers **fill up**, innocent flows are victimized



Short flows are **stuck** in congested buffers



Random deflection breaks under load

Problem

Random deflection treats the flows contributing to **long lasting congestion** similar to **short-lived microbursts**

Solution

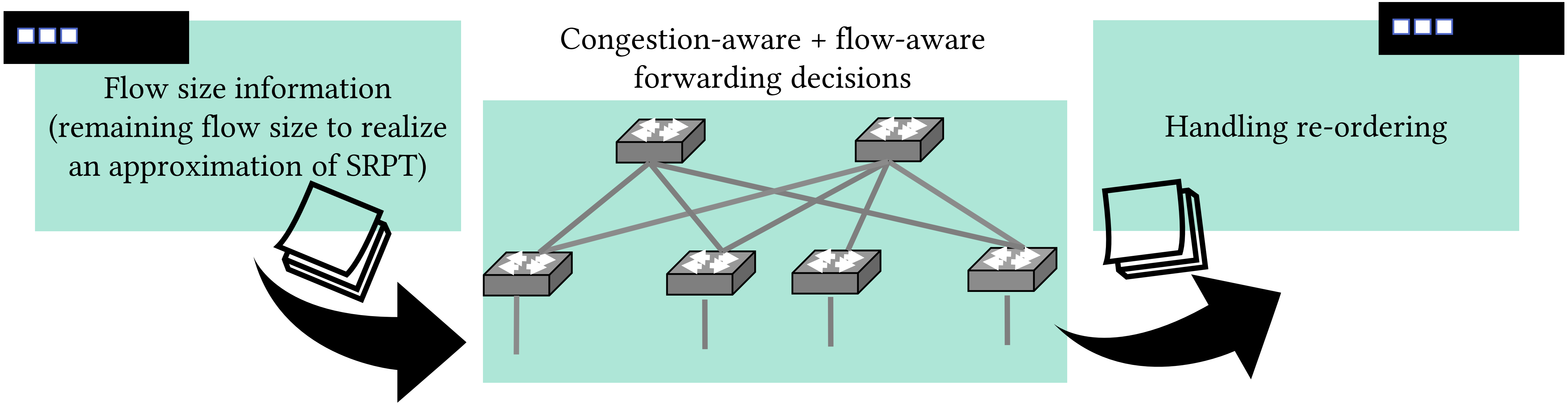
Detecting the flows that are more likely to contribute to **lasting congestion** and **prioritizing** their packets for:

(a) **deflection** under light load

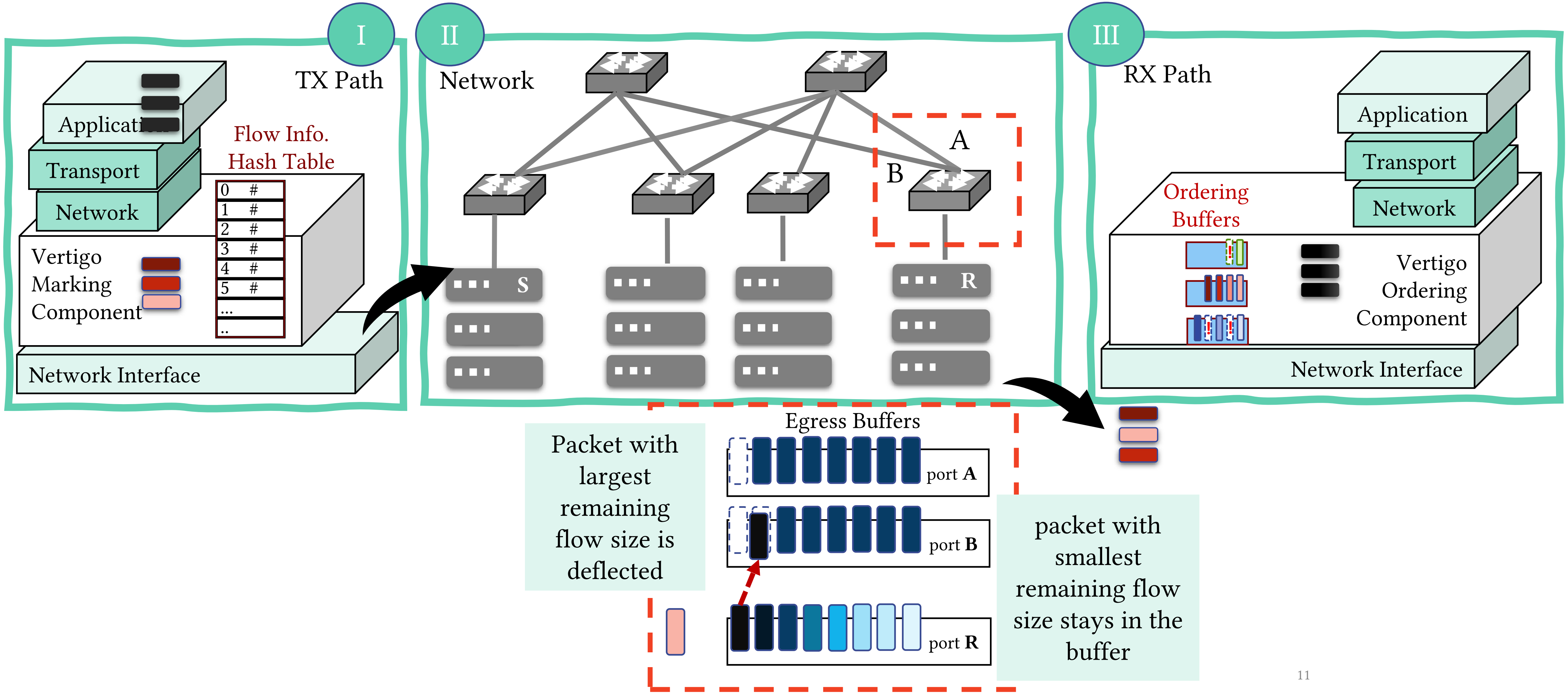
(b) **drop** under high load

Host-assisted deflection

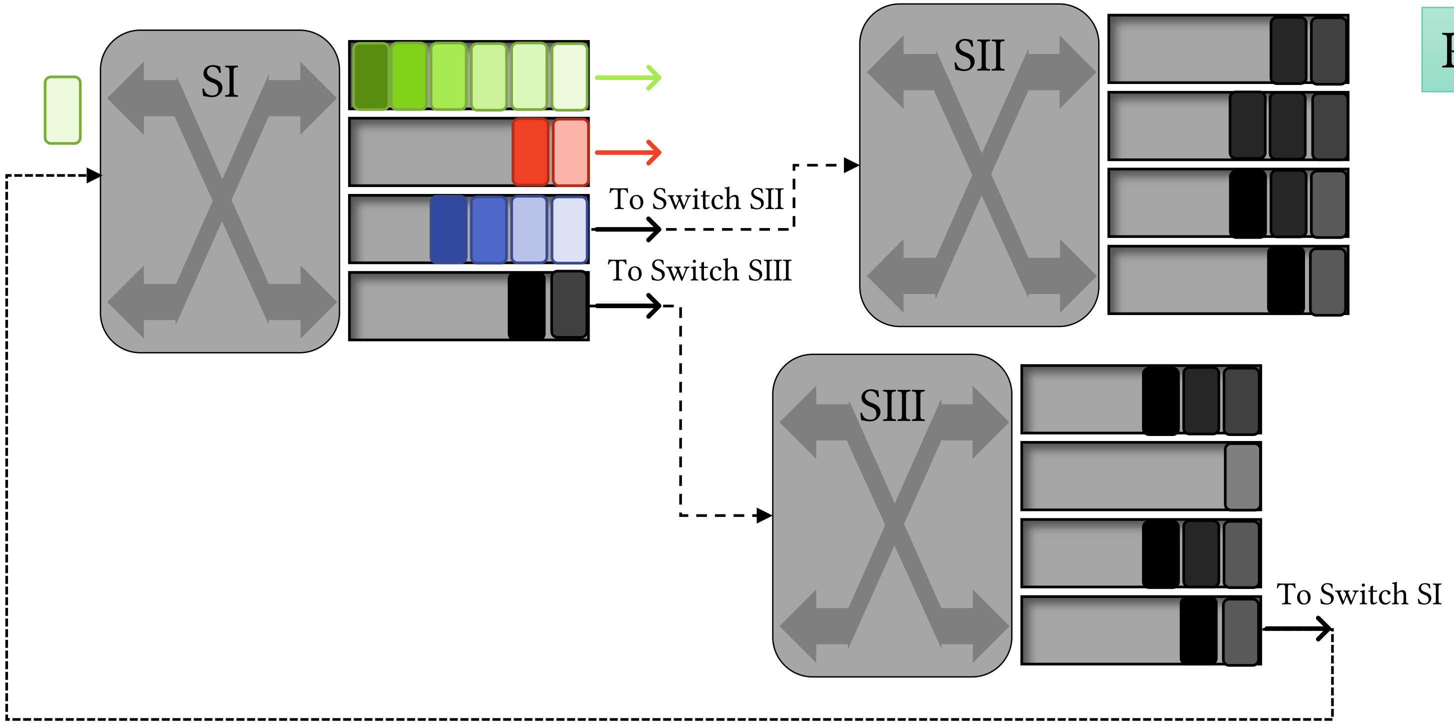
Remaining flow size, a good indicator for lasting congestion.



Vertigo: the big picture



Preventing collapse using flow length information



Vertigo Fabric

I. **Forwarding:** Least remaining flow size

II. **Congestion:** Deflect instead of Drop

III. **Deflection:** Highest remaining flow size

IV. **Load-balancing:** Power of 2 choices

Packet from a short flow arrives at a full buffer

Vertigo identifies the packet with highest remaining flow size from a full buffer

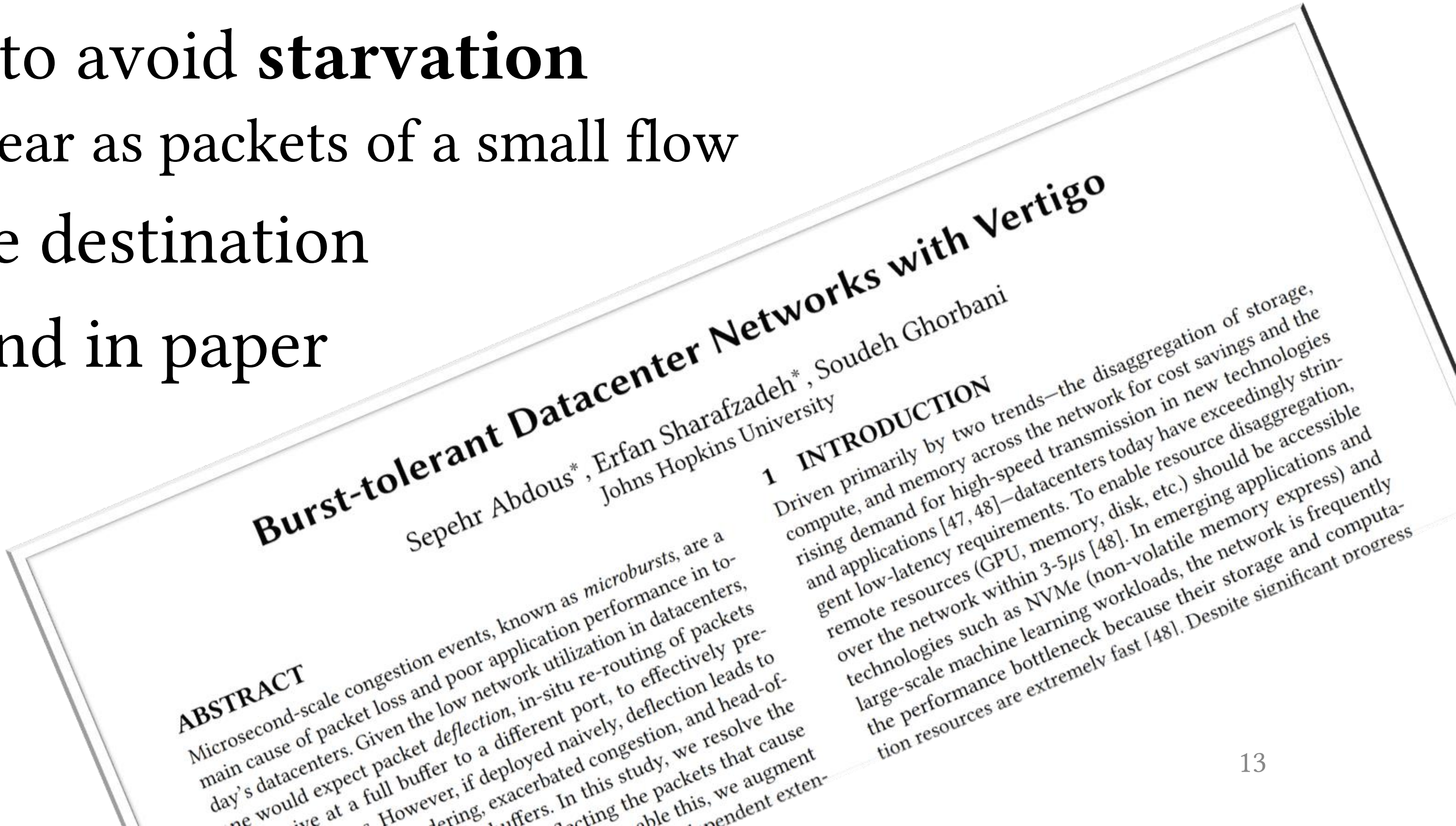
Randomly chooses two destination buffers, selects the one with least queue occupancy

Deflects the selected packet to chosen buffer

Inserts the arrived flow to its correct position w.r.t. its remaining flow size

Vertigo components at the host

- **Marking** the packets based on remaining flow size
- Detecting re-transmissions to ensure **consistency**
- Boosting re-transmissions to avoid **starvation**
    - Re-transmitted packets appear as packets of a small flow
- **Ordering** shim layer at the destination
- Detailed design can be found in paper



Burst-tolerant Datacenter Networks with Vertigo

Sepehr Abdous*, Erfan Sharafzadeh*, Soudeh Ghorbani
Johns Hopkins University

ABSTRACT

Microsecond-scale congestion events, known as *microbursts*, are a main cause of packet loss and poor application performance in today's datacenters. Given the low network utilization in datacenters, one would expect packet *deflection*, in-situ re-routing of packets ...ive at a full buffer to a different port, to effectively pre-... However, if deployed naively, deflection leads to ...dering, exacerbated congestion, and head-of-... ...uffers. In this study, we resolve the ...cting the packets that cause ...able this, we augment ...pendent exten-

1 INTRODUCTION

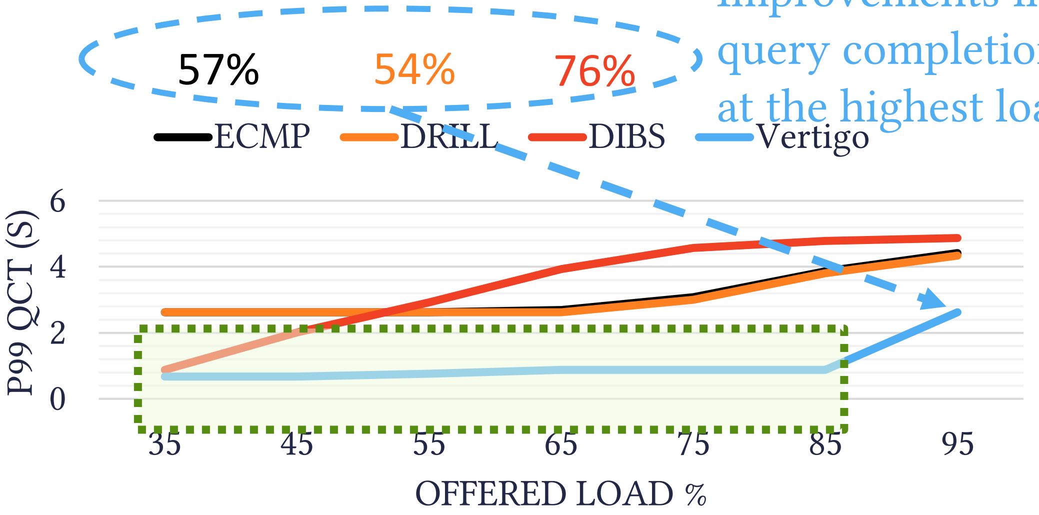
Driven primarily by two trends—the disaggregation of storage, compute, and memory across the network for cost savings and the rising demand for high-speed transmission in new technologies and applications [47, 48]—datacenters today have exceedingly stringent low-latency requirements. To enable resource disaggregation, remote resources (GPU, memory, disk, etc.) should be accessible over the network within 3-5μs [48]. In emerging applications and technologies such as NVMe (non-volatile memory express) and large-scale machine learning workloads, the network is frequently the performance bottleneck because their storage and computation resources are extremely fast [48]. Despite significant progress

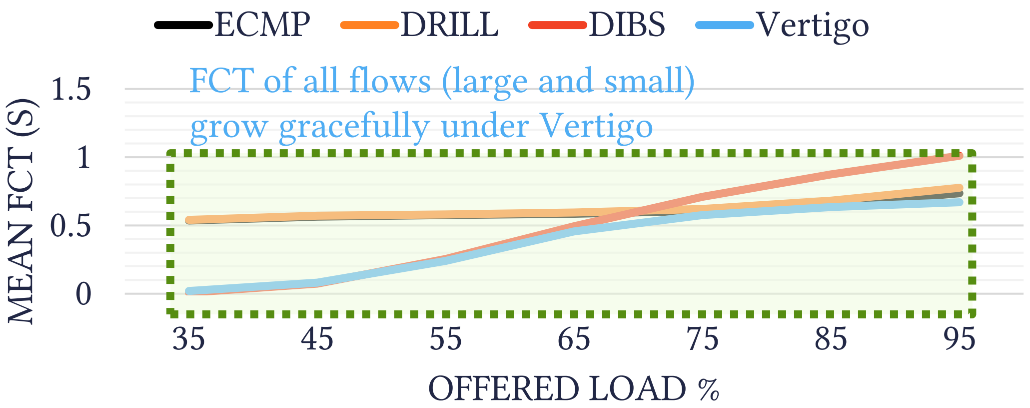
Simulation results: Vertigo's superior performance

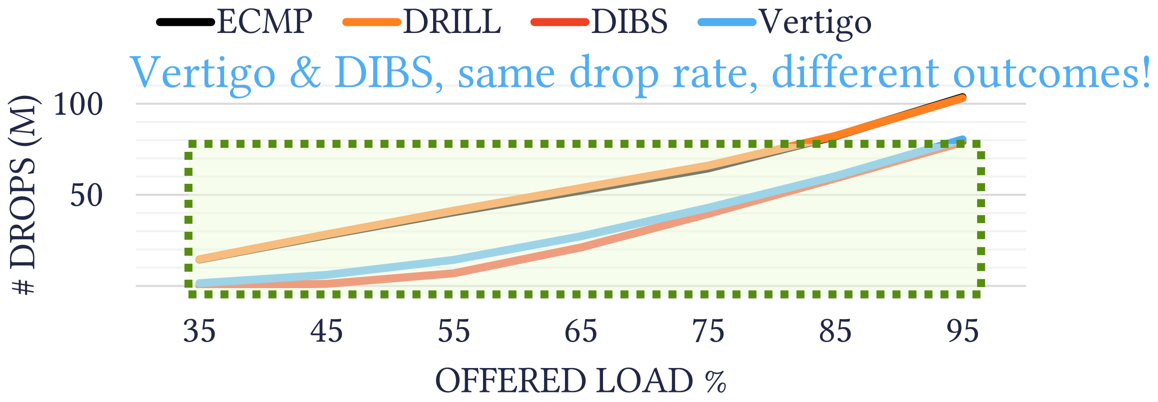
Setup
- 4-8-40 Two-tiered leaf-spine
- 10GB server-to-ToR, 40GB aggregate links
- DCTCP transport
- Workload: FB cache, fixed background + var. Incast

ECMP | DRILL | DIBS | Vertigo

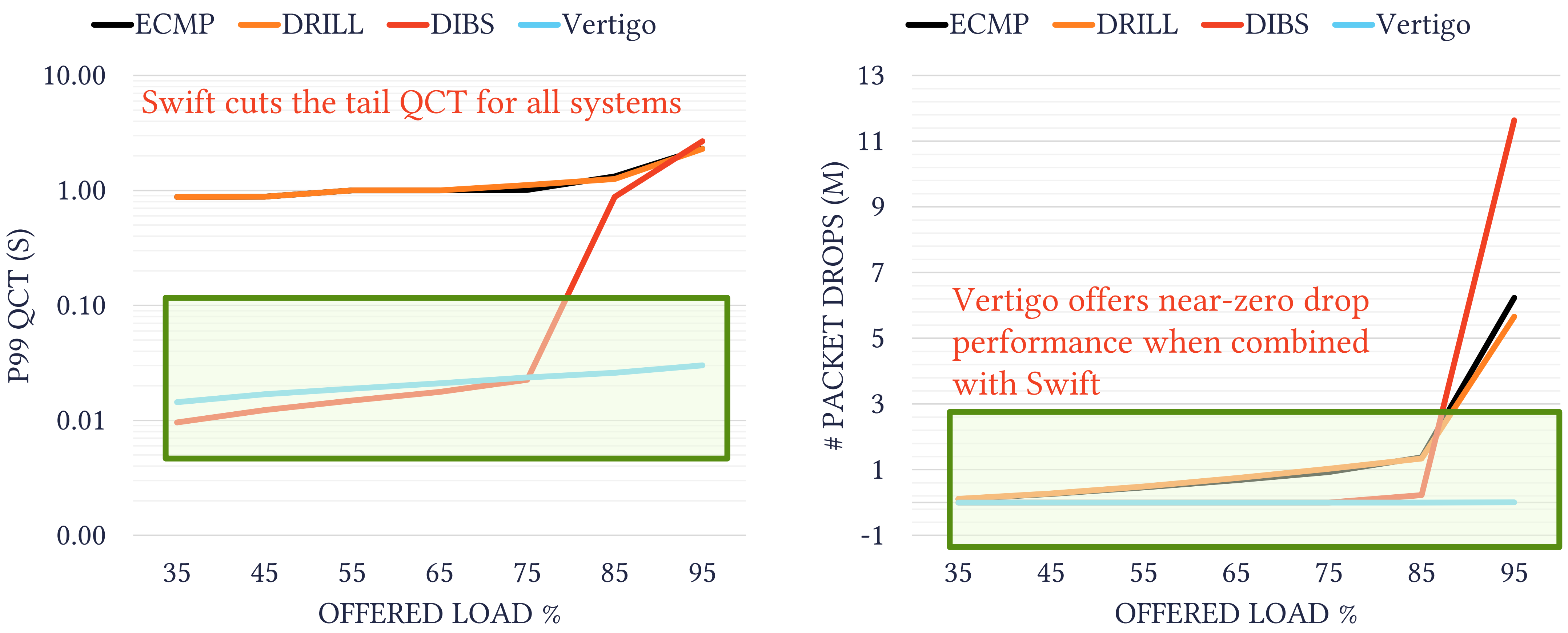
Improvements in tail query completion times at the highest load



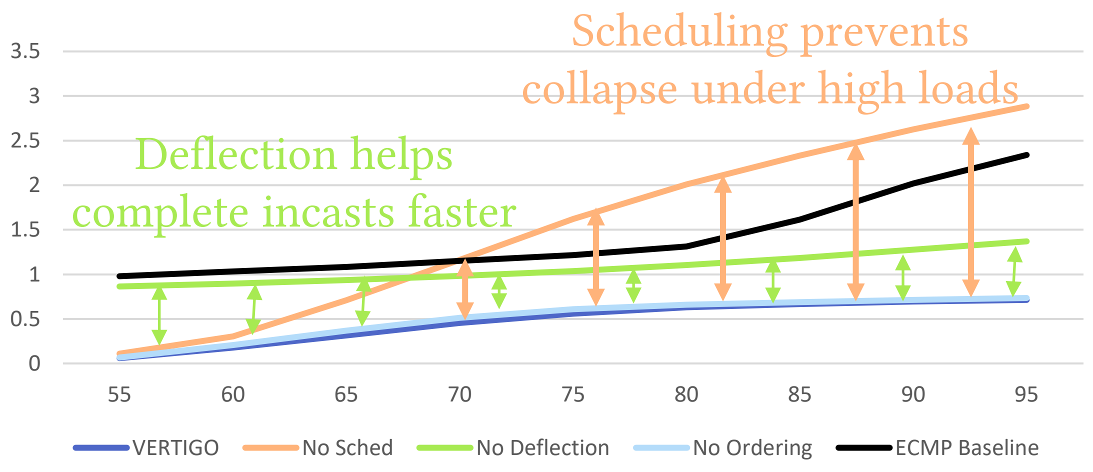


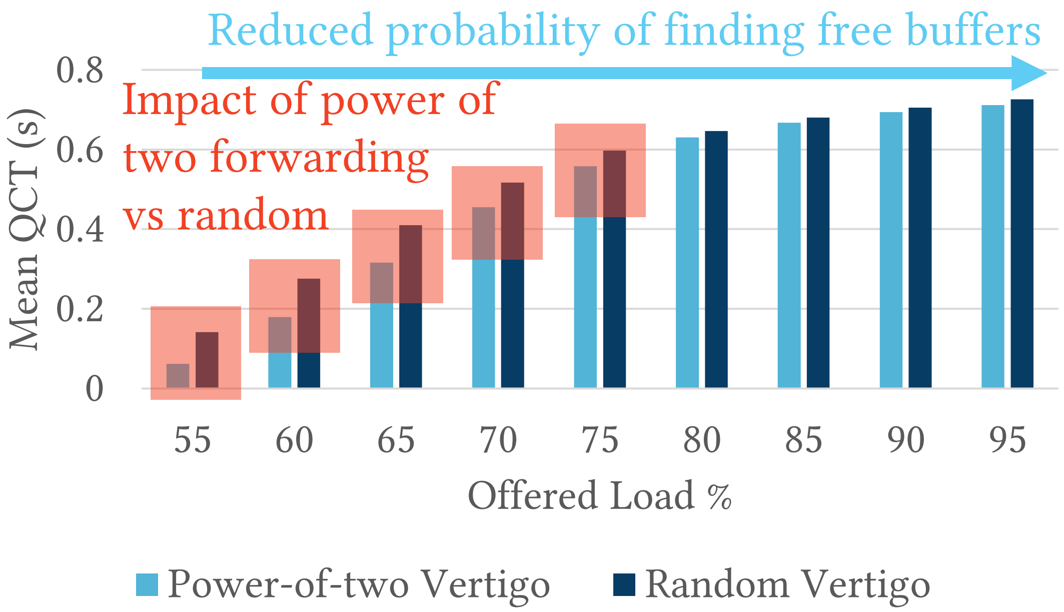


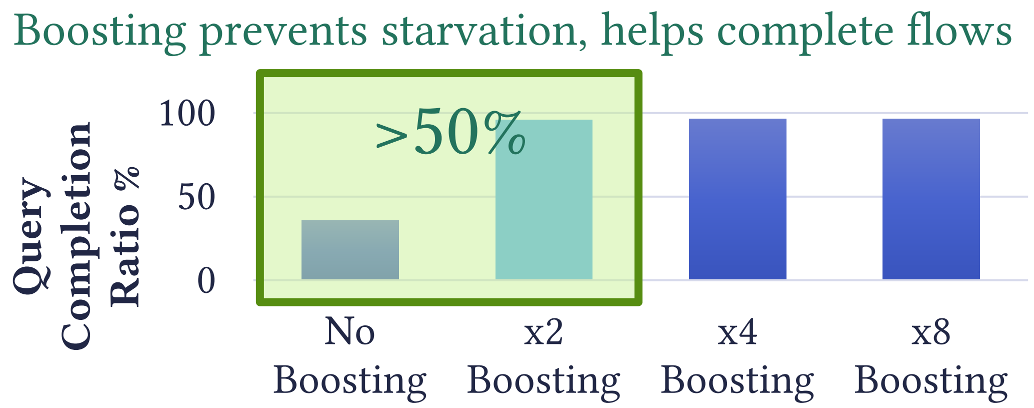
Vertigo achieves near-0 drops with Swift

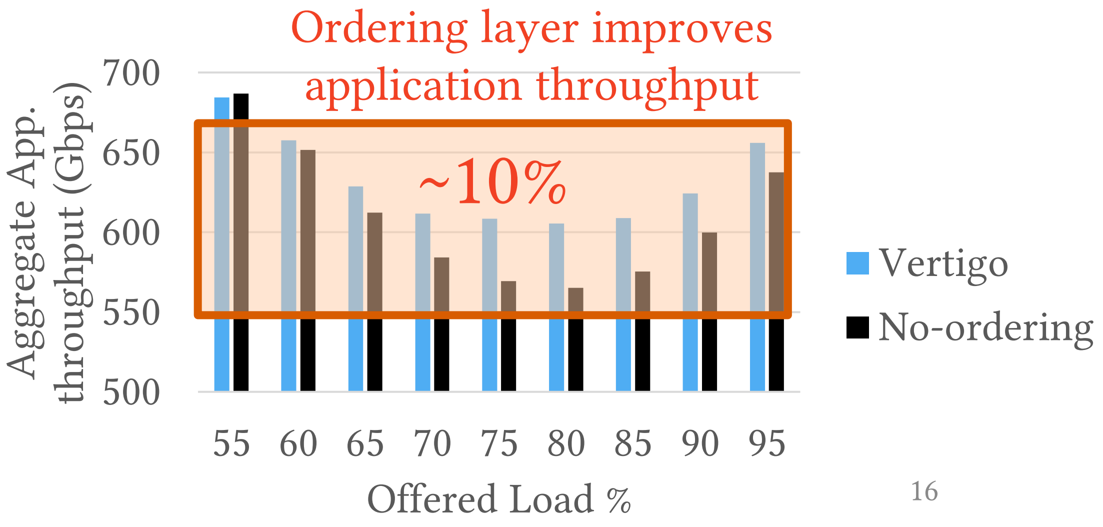


Vertigo component analysis









Deflection: Cuts the completion time tail

Scheduling: Prevents the collapse

Ordering: Preserves app throughput

Boosting: Prevents starvation

Vertigo Conclusions

Key Takeaway:

To properly react to microbursts, network-centric **real-time action** and end-host's **advance knowledge of flow sizes** are vital!

Vertigo:

A hybrid solution to tolerate micro-scale bursty traffic by changing the forwarding decisions upon facing imminent packet loss

Challenges:

- Both host and network must be changed
- Existing queue management abstractions are not enough

Check out Vertigo artifacts!
https://github.com/hopnets/vertigo-artifacts

Thank you!

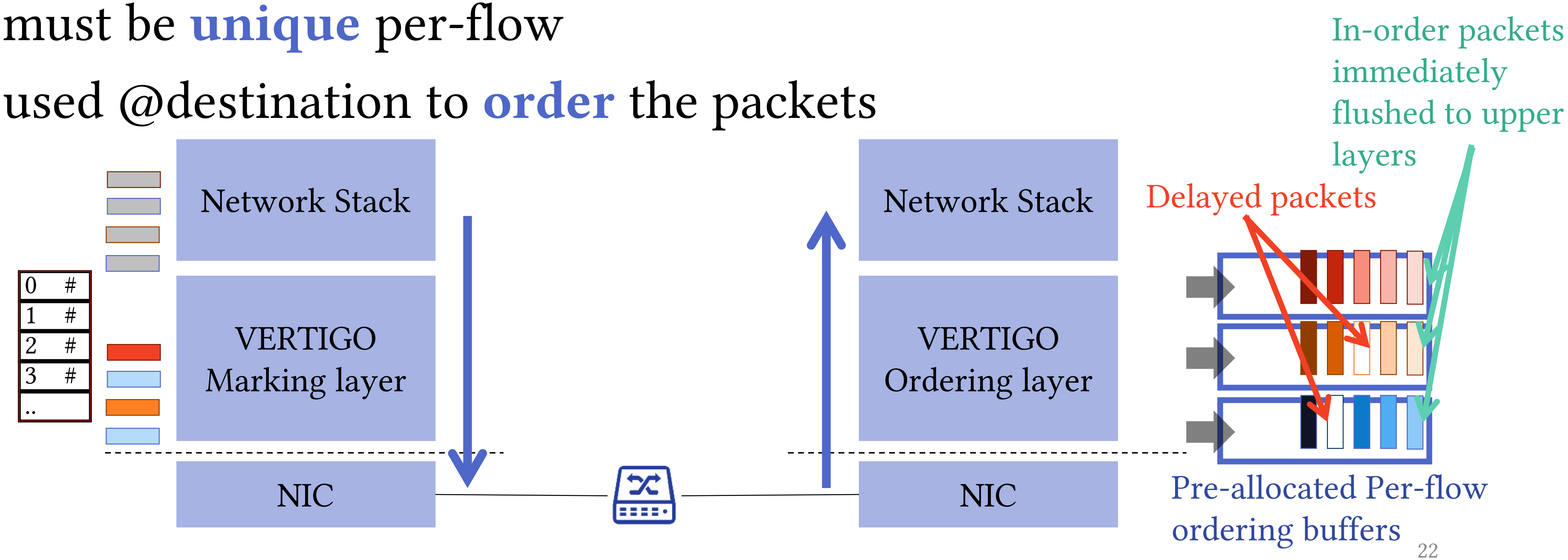
Contact us

- sabdous1@jhu.edu
- erfan@cs.jhu.edu

Backup slides

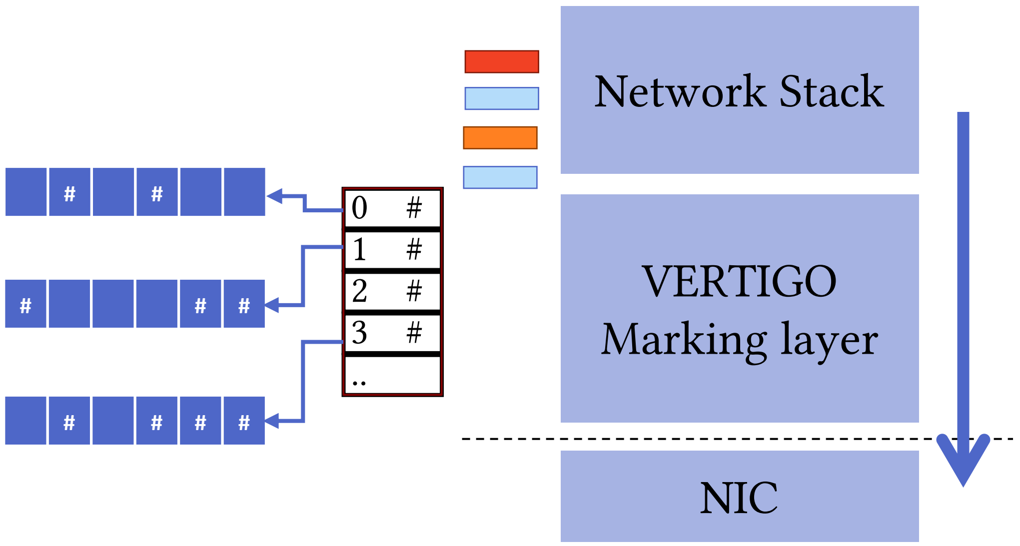
Handling packet reordering

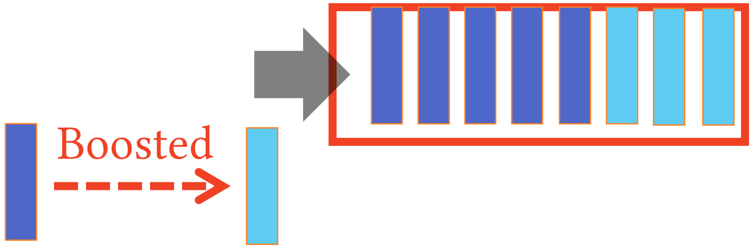
- Mark packets with **remaining flow size (RFS)** @sender
- Flow size tracking is **transport-independent**
- RFS must be **unique** per-flow
- RFS used @destination to **order** the packets



Saying no to starvation

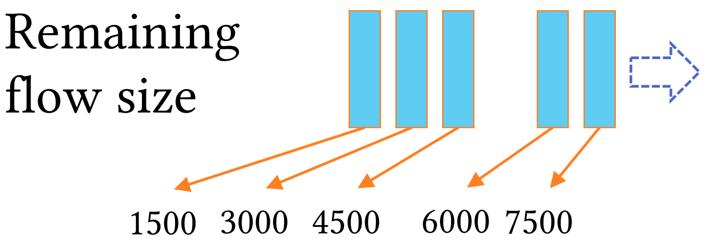
- Keeping track of **re-transmissions** to ensure RFS **consistency**
- **Boost** the re-transmitted packet by cutting its RFS

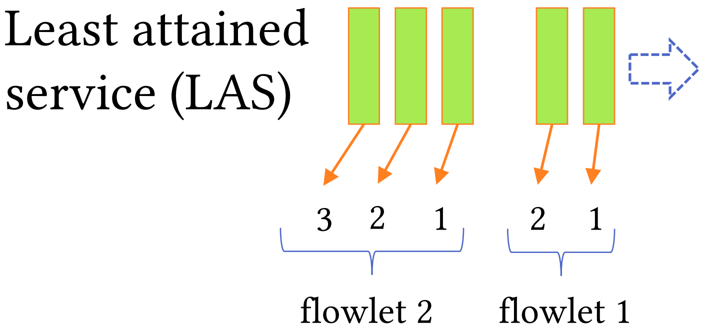


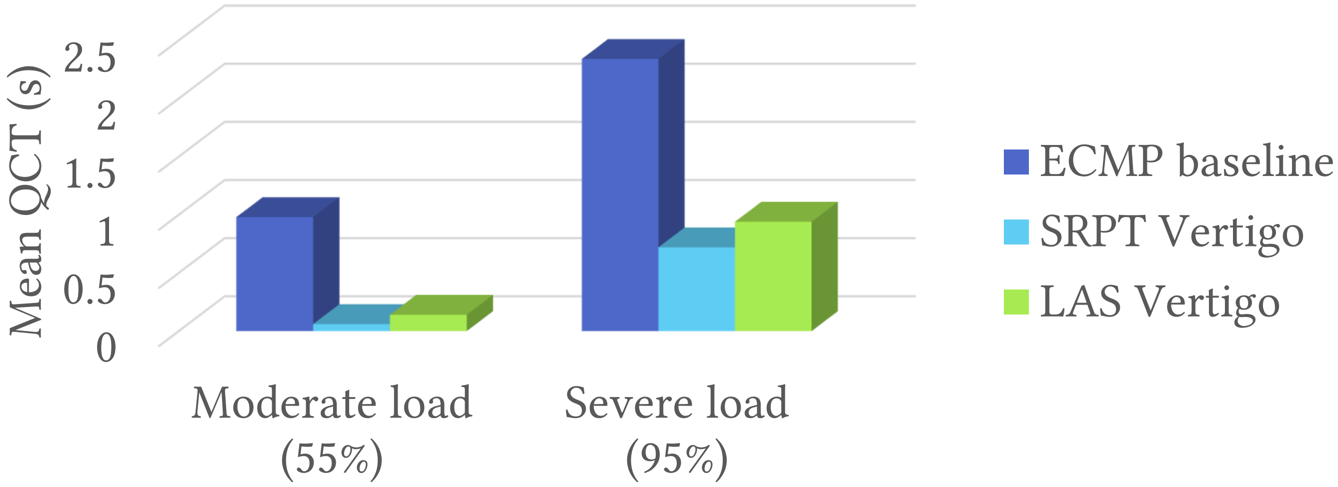


Simple marking by counting upwards

What if flow size information is not available?







- The granularity of load-balancing
- Choosing ordering timeouts
- Vertigo's performance under larger flows and larger-scale Incasts