CoNEXT '23

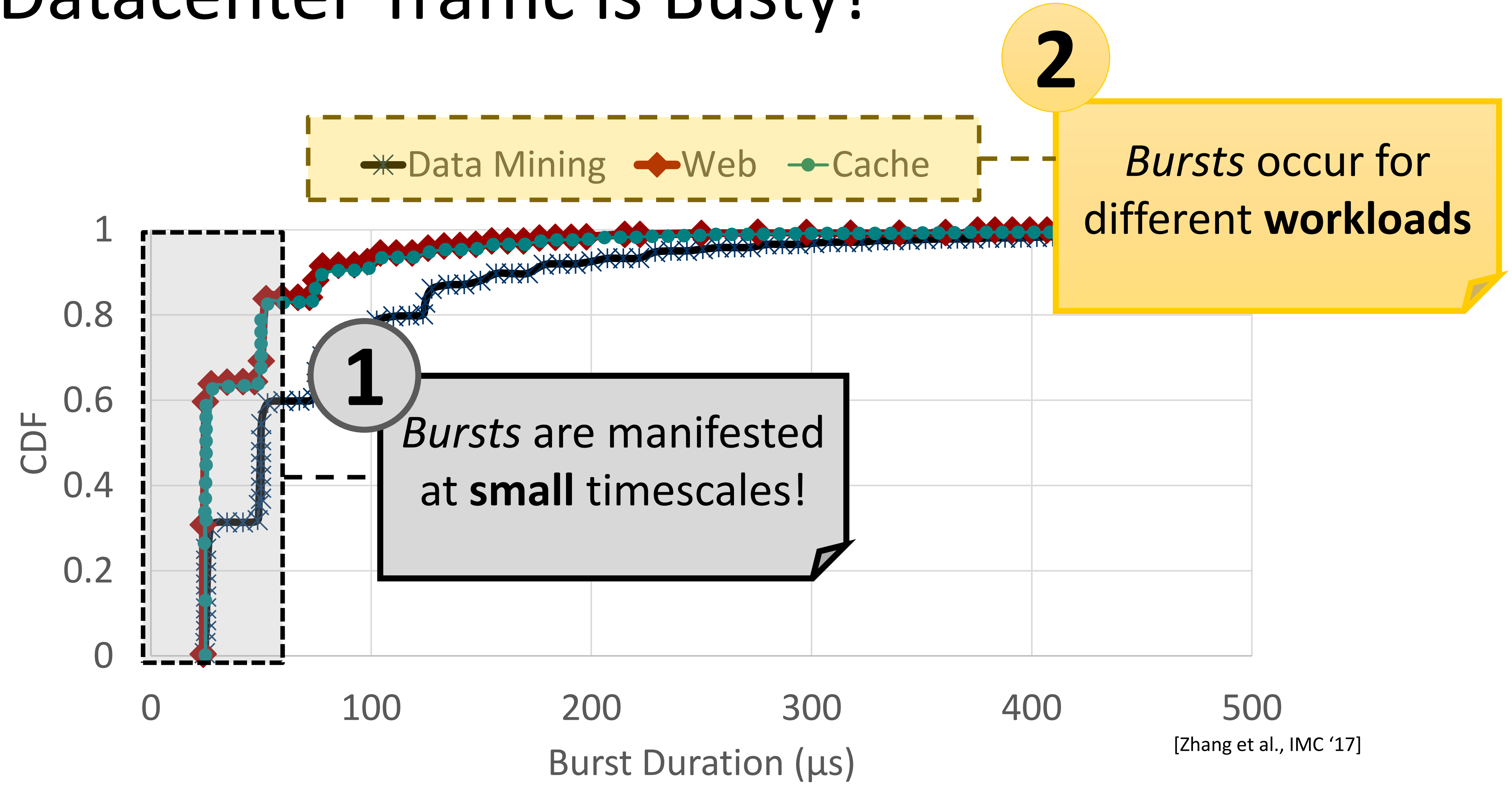
Practical Packet Deflection in Datacenters

Sepehr Abdous, Erfan Sharafzadeh, Soudeh Ghorbani

Johns Hopkins University

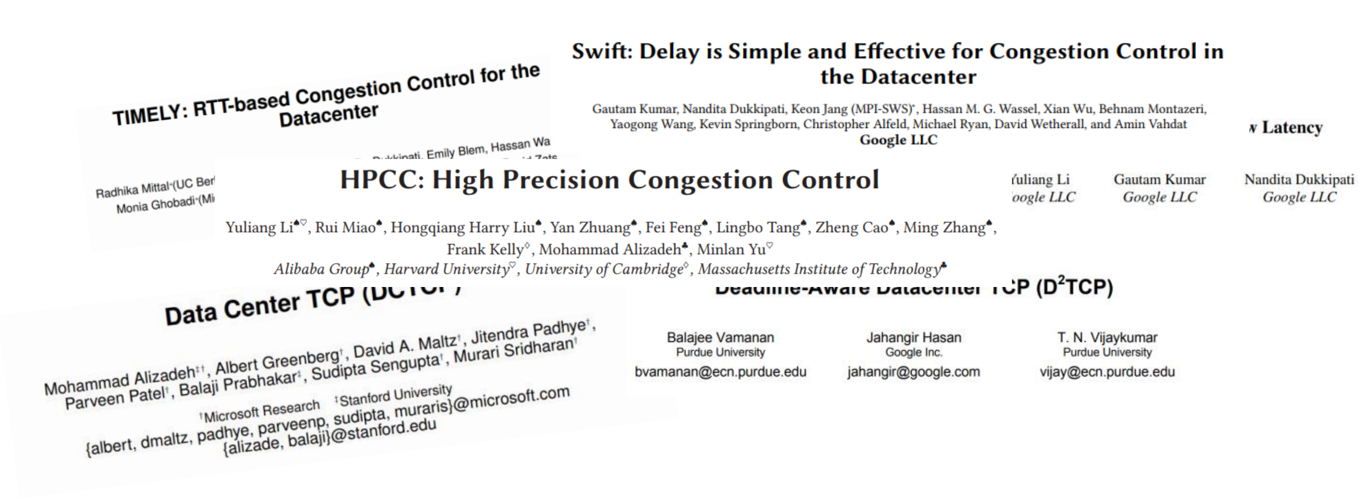
December 2023

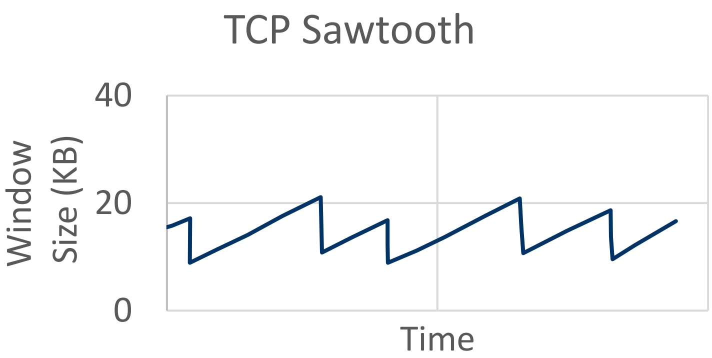
Datacenter Traffic is Busty!



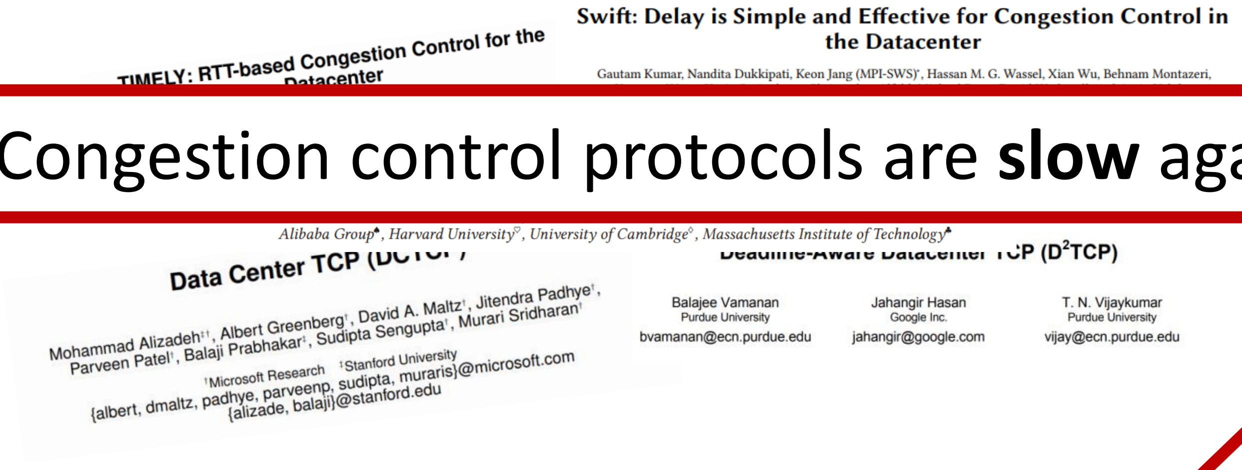
Existing Techniques Fall Short Against Bursts

Congestion control

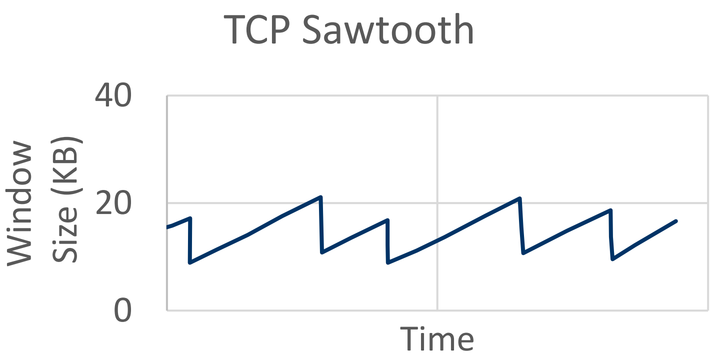




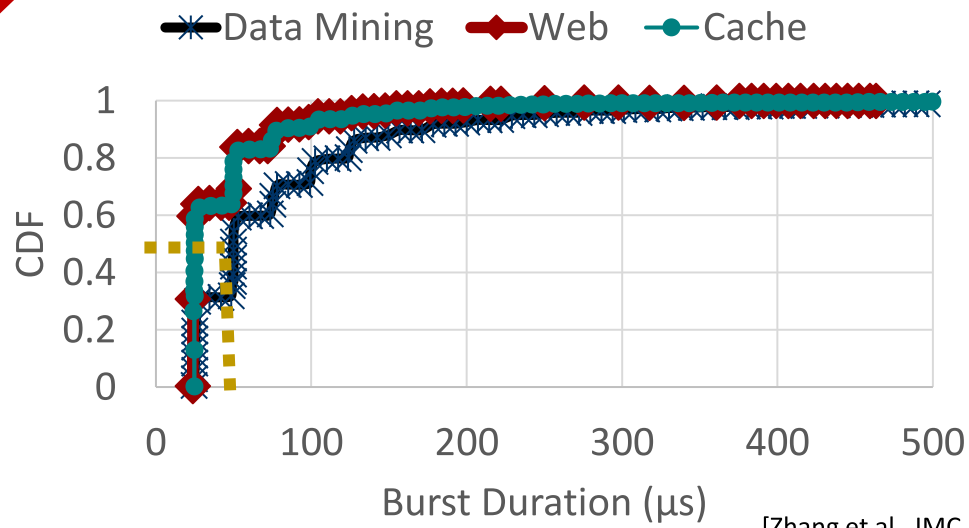
Existing Techniques Fall Short Against Bursts



Congestion control protocols are **slow** against bursts!



~ few milliseconds >> 50 µs



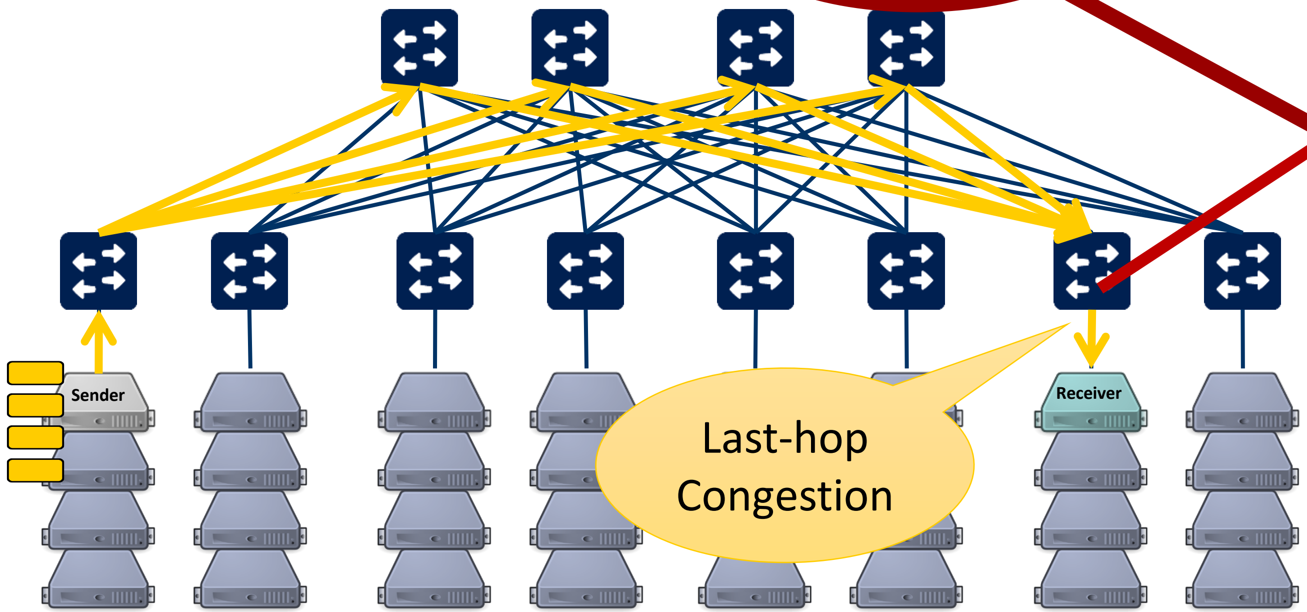
[Zhang et al., IMC '17]

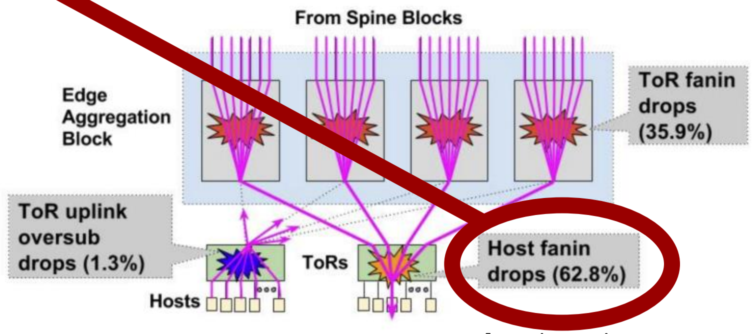
Existing Techniques Fall Short Against Bursts

Load balancing paradigms are ineffective against drops at the **last hop**.

~~Congestion control~~

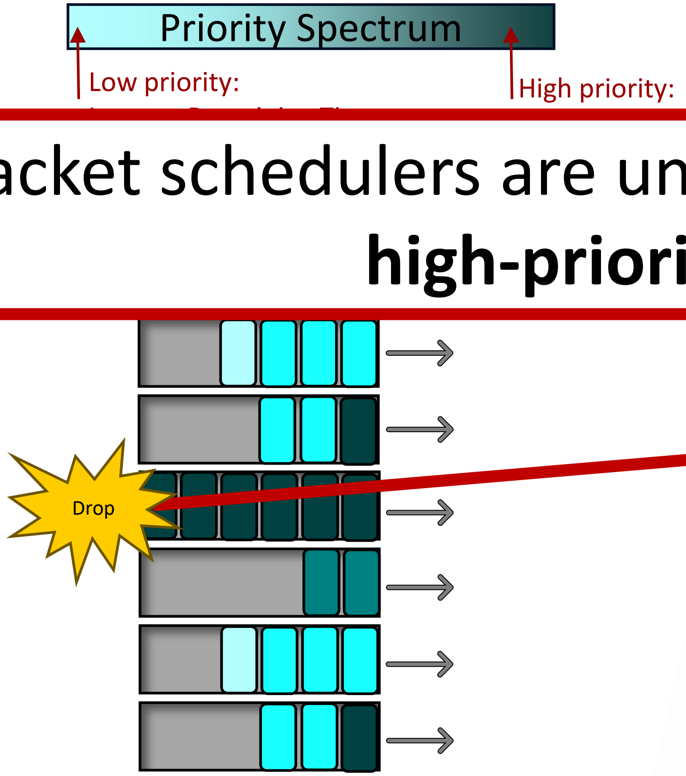
~~Load balancing~~





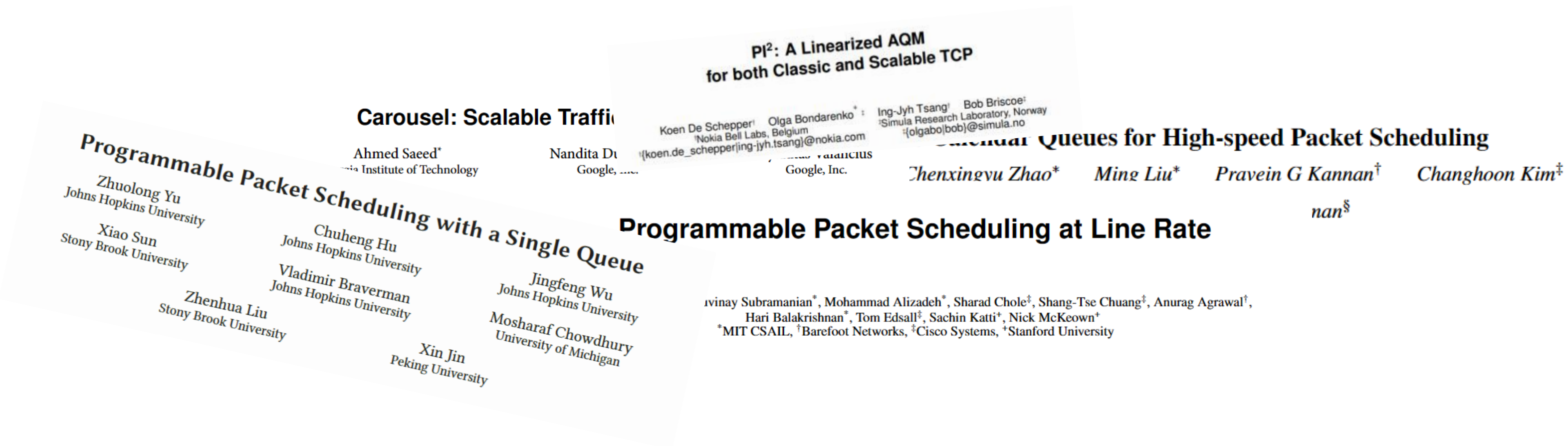
[Singh et al., SIGCOMM '12]

Existing Techniques Fall Short Against Bursts

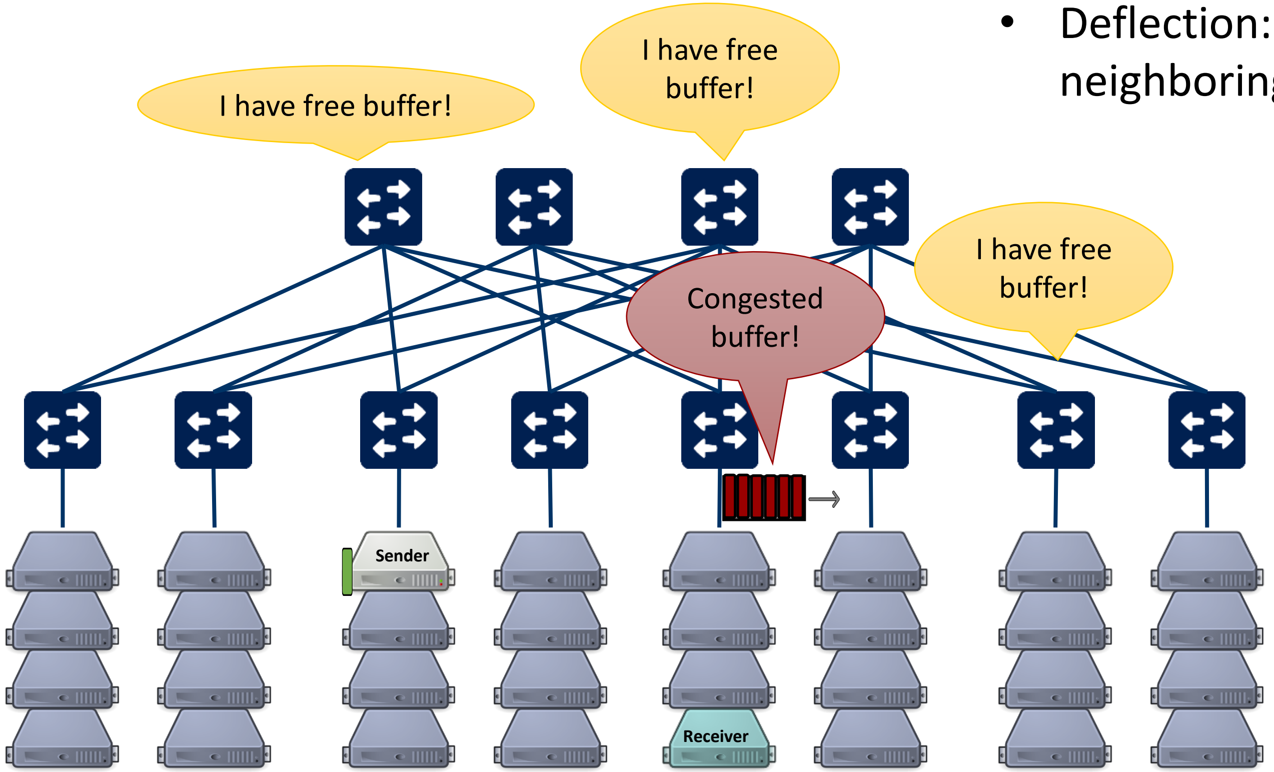


Packet schedulers are unable to address **bursts of high-priority packets**.

- ~~Congestion control~~
- ~~Load balancing~~
- ~~Packet scheduling~~



Packet Deflection Avoids Drops in the Hotspots!

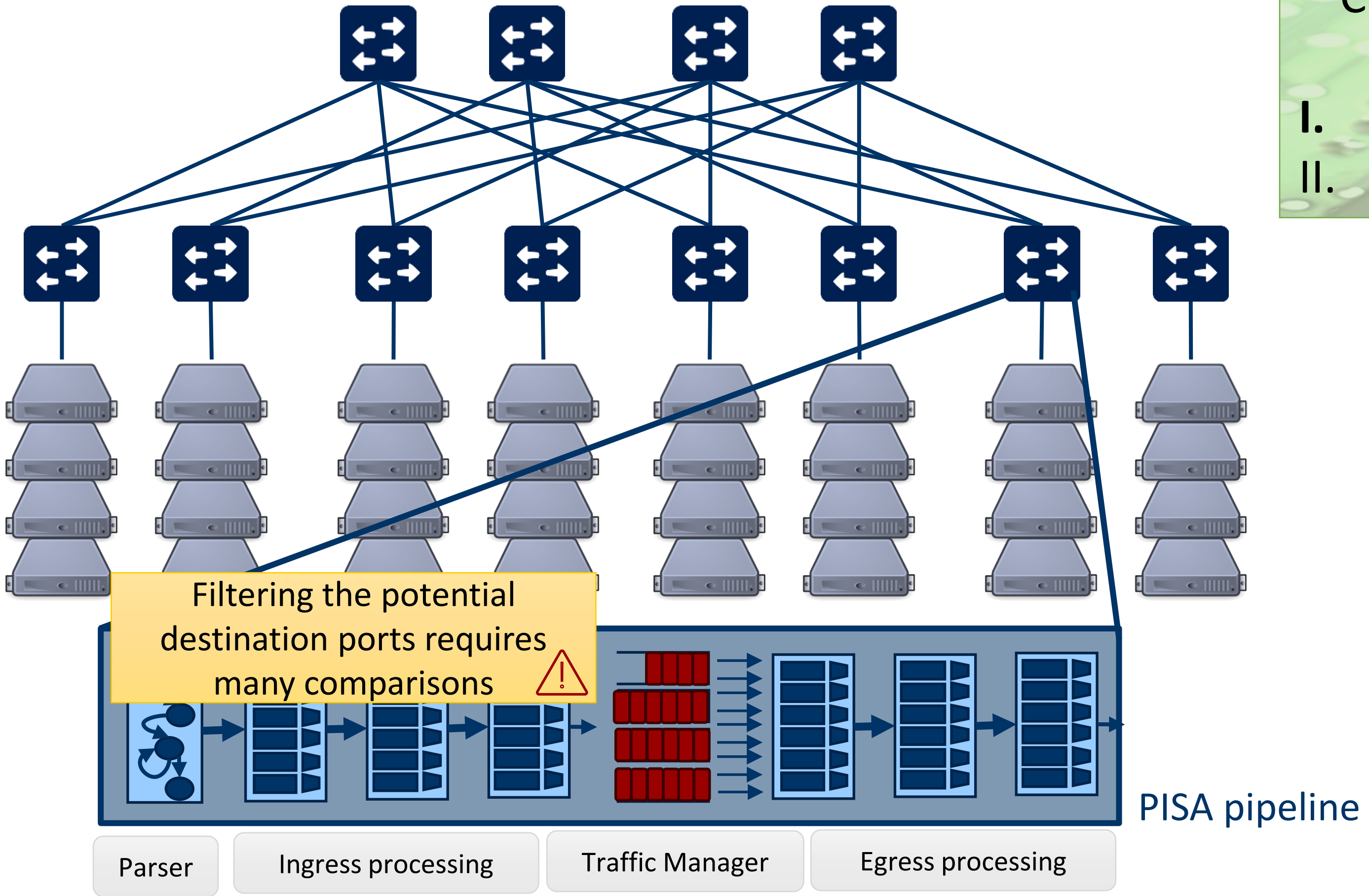


- Deflection: **Re-routing** packets that arrive at a full buffer to a neighboring switch.

Deflection improves the query performance by up to **43x**! [Vertigo, CoNEXT '21]

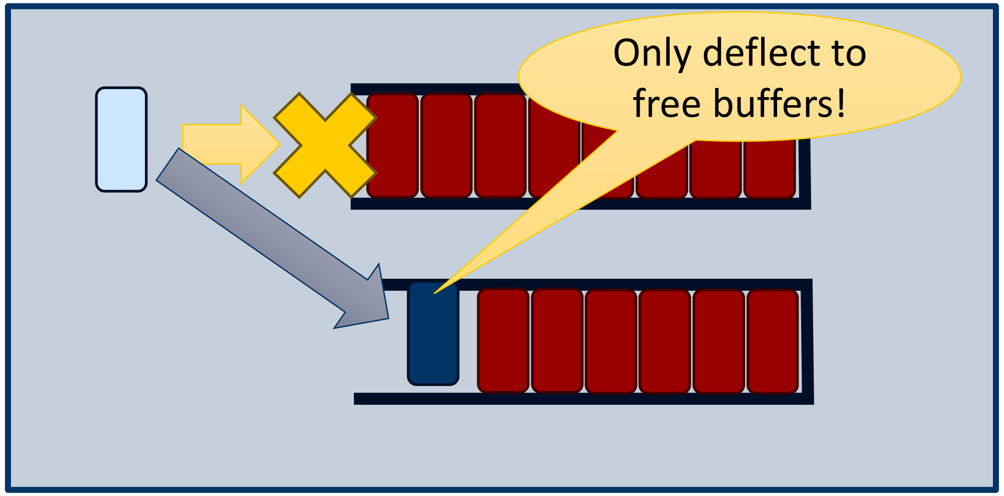
State-of-the-art **deflection** proposals
are **not implementable**
in existing programmable hardware!

State-of-the-art Deflection Depends on **Two** Primitives



Clean-slate deflection requirements in **real-time**:

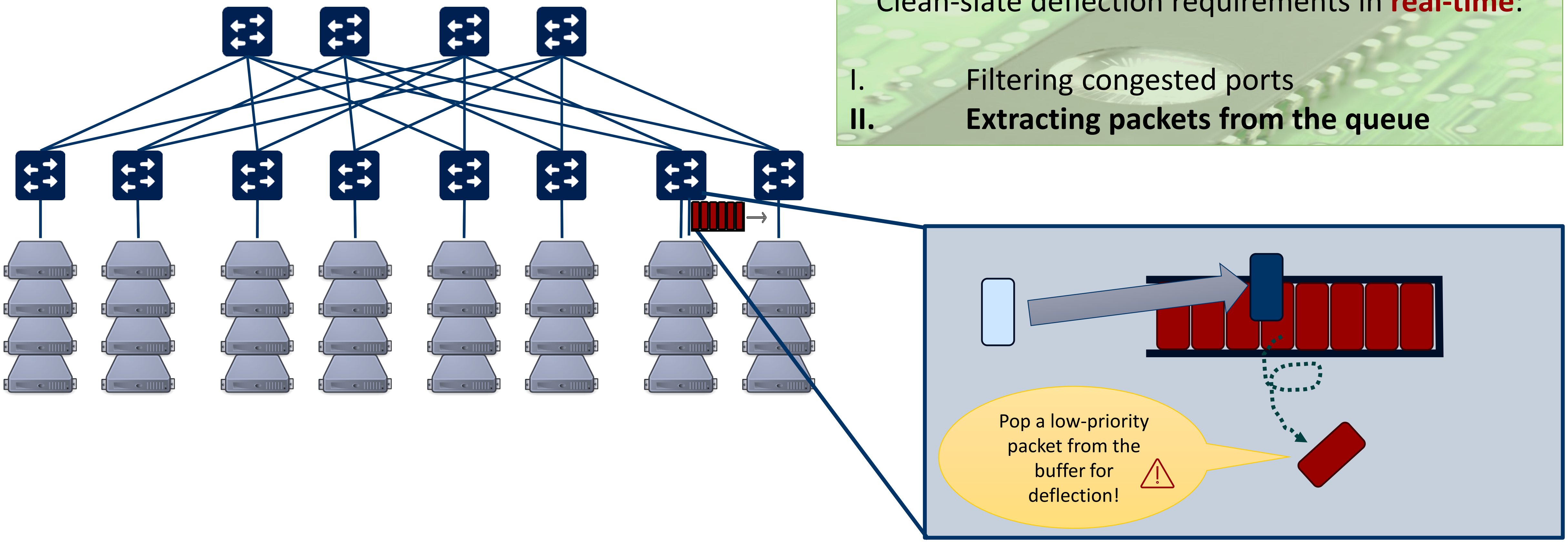
I. **Filtering congested ports**
II. Extracting packets from the queue



State-of-the-art Deflection Depends on **Two** Primitives

Clean-slate deflection requirements in **real-time**:

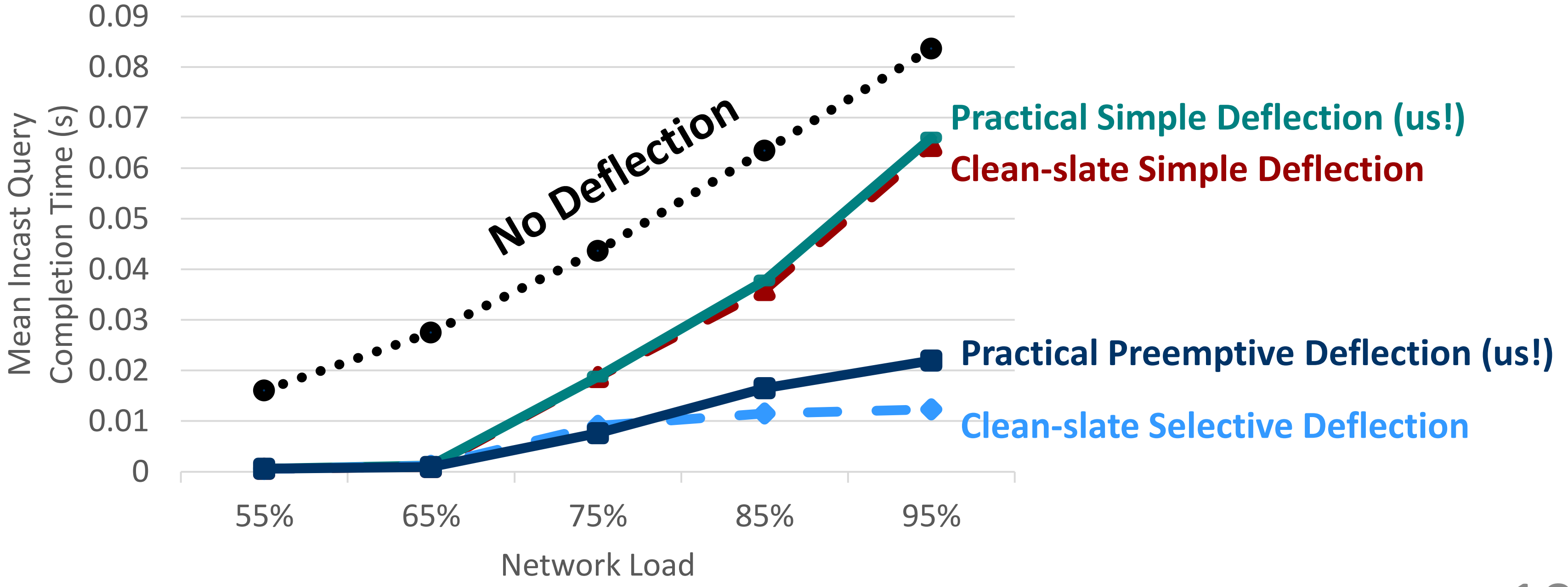
I. Filtering congested ports
II. Extracting packets from the queue



Our Contribution:
Implementing Deflection in Programmable Hardware

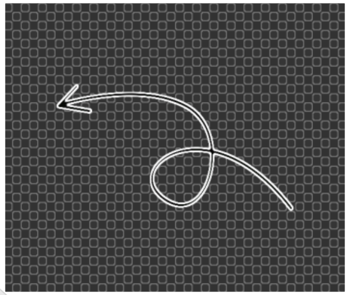
- Implementing two approaches to deflection:
  - **Simple Deflection**
  - Approximation of **Selective Deflection** called **Preemptive Deflection**
- Intuitions:
  - Using **packet recirculation** instead of expensive memory manipulation.
  - Using **admission control** instead of packet extraction from the queue.

- Using DCTCP Congestion control
- 100 Gbps 8-ary fat-tree cluster
- Incast size of 100 Requests per Query

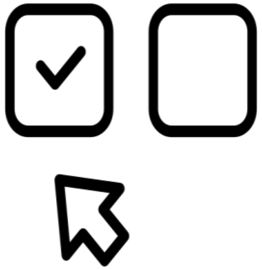


Approaches to Deflection Suited for Different Needs

Just deflect to a non-congested port, **randomly**!

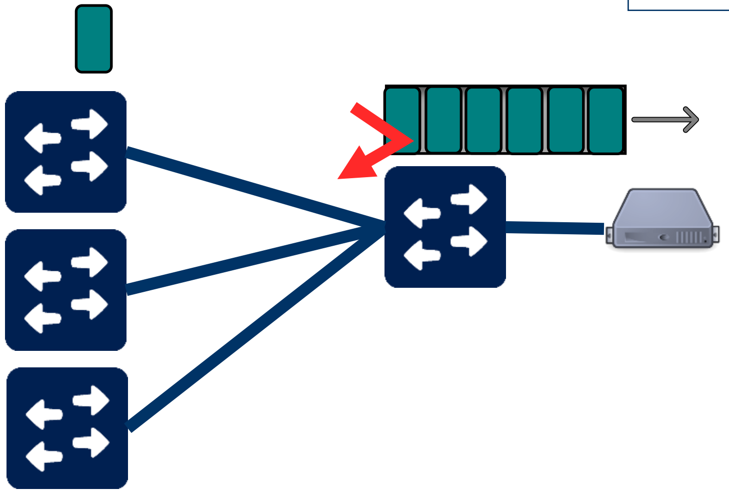


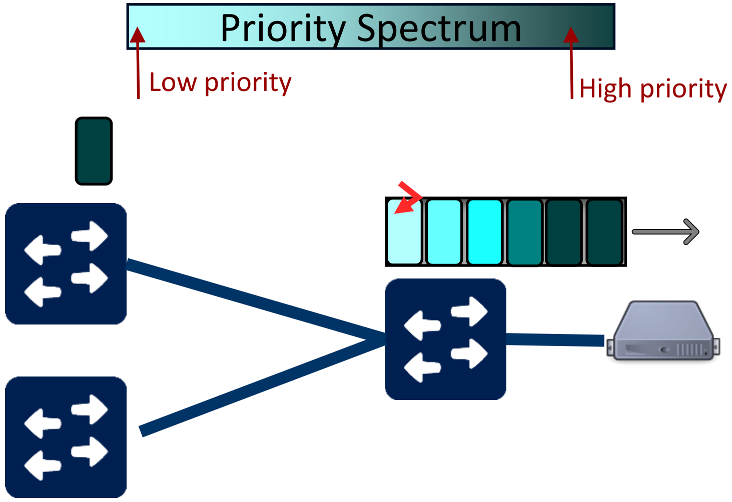
Simple



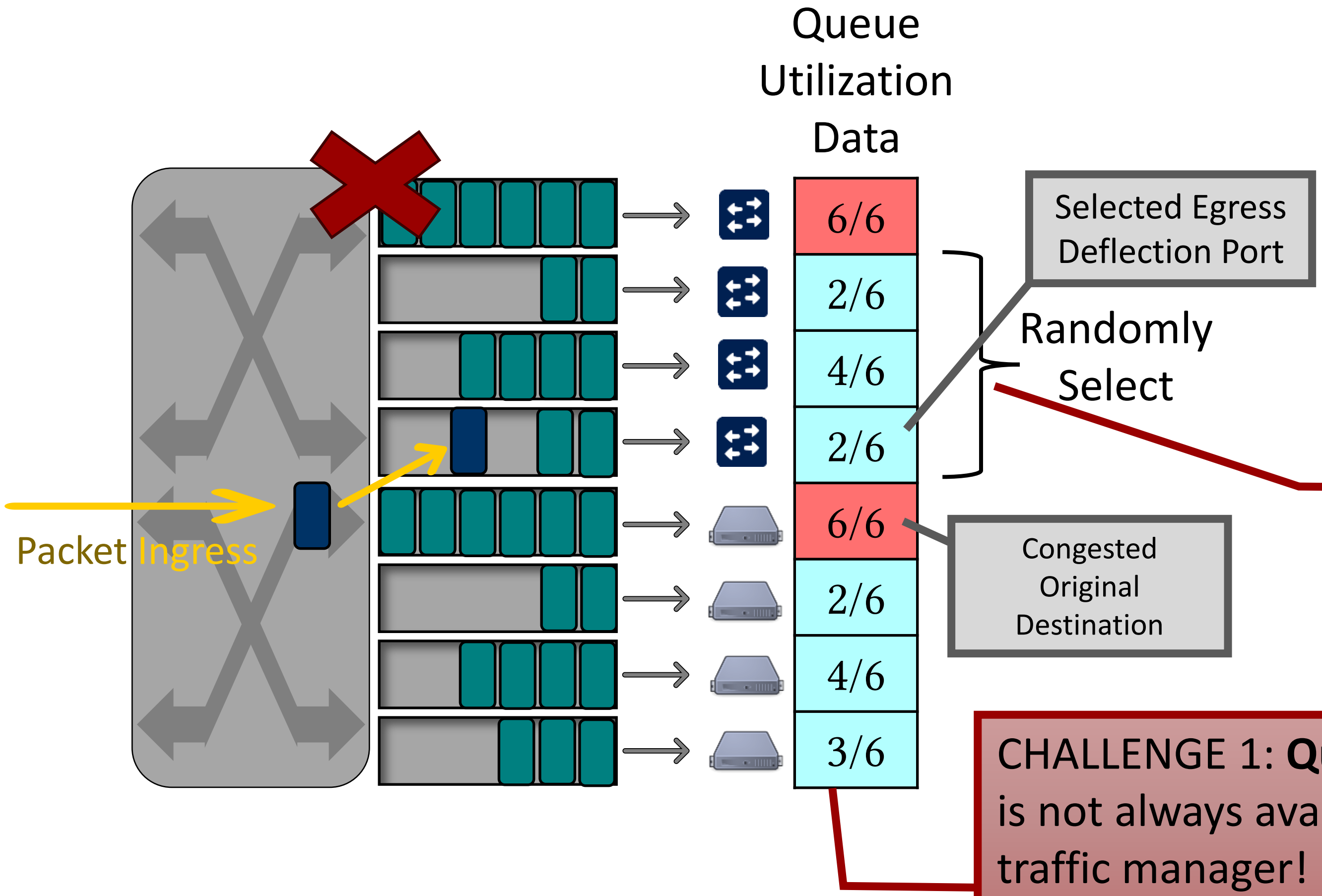
Selective

Sort packets in the queue and deflect lower-priority packets!





What Makes **Simple Deflection** Hard to Implement ?

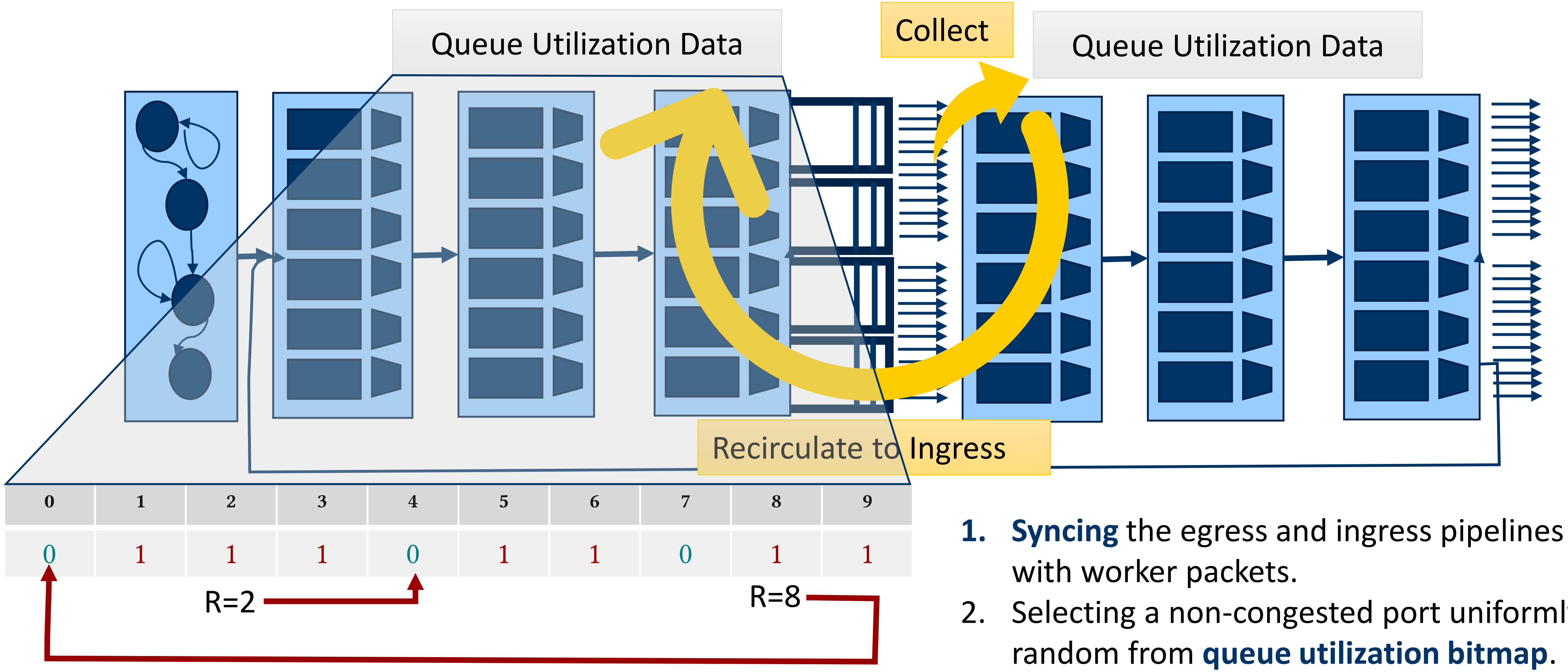


Simple Deflection is effective when congestion is infrequent
(71% lower QCT under moderate Incast)

CHALLENGE 2: **Filtering** non congested ports is non-trivial and computationally **expensive**!

CHALLENGE 1: **Queue utilization data** is not always available behind the traffic manager!

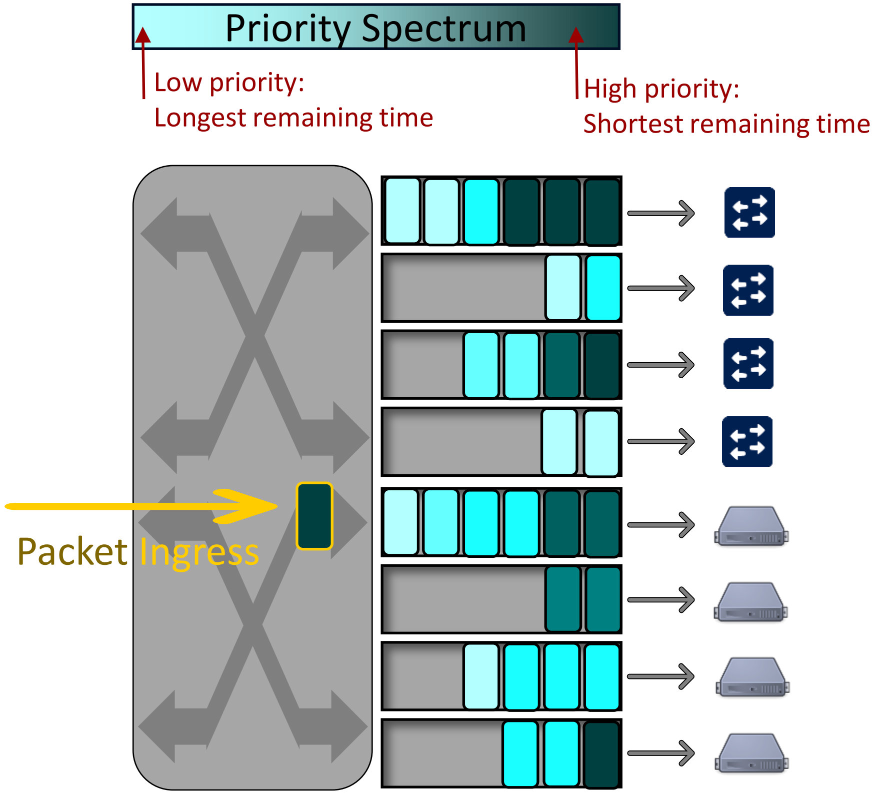
Implementable Simple Deflection in PISA



0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	1	0	1	1

1. **Syncing** the egress and ingress pipelines with worker packets.
2. Selecting a non-congested port uniformly at random from **queue utilization bitmap**.

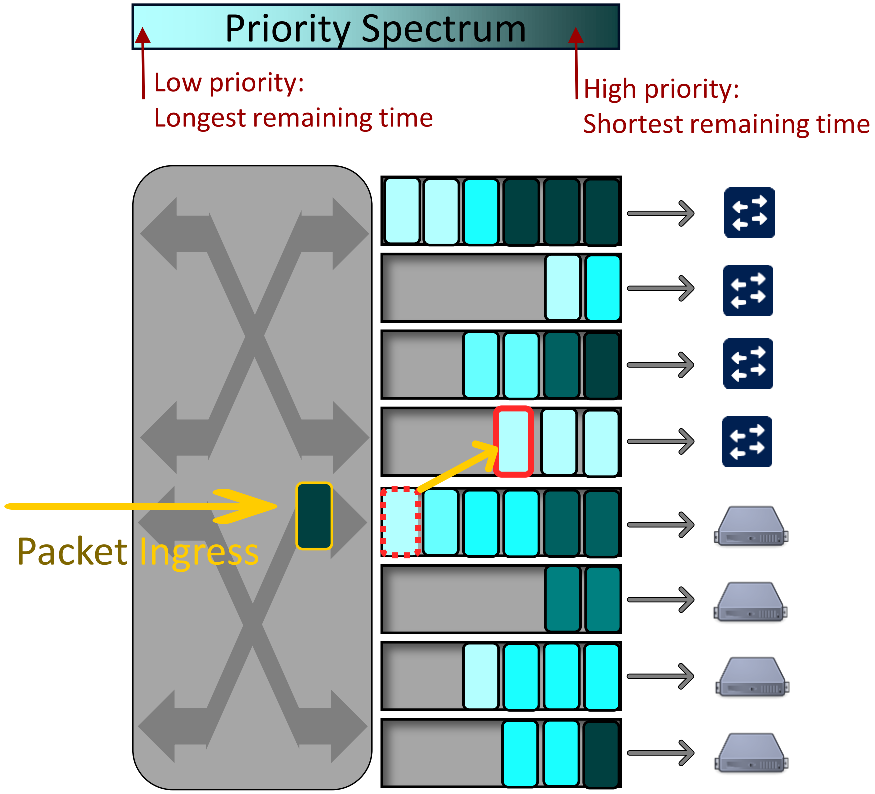
What Makes **Selective Deflection** Hard to Implement ?



Operation Steps

1. **High-priority** packet arrives
2. Extract and **deflect** a **low-priority** packet

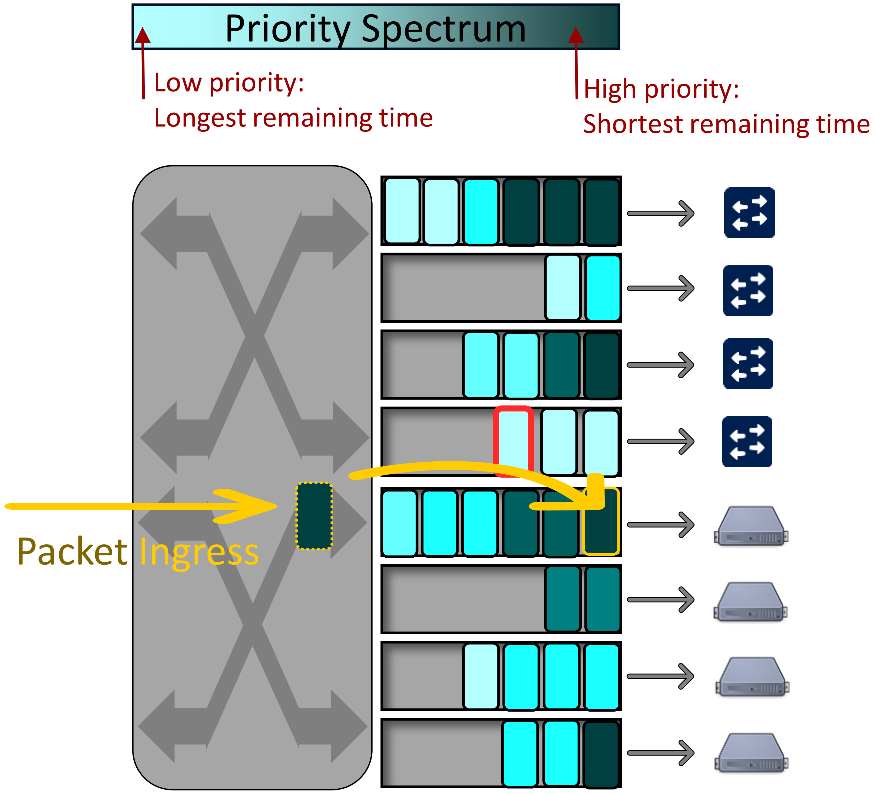
What Makes **Selective Deflection** Hard to Implement ?



Operation Steps

1. **High-priority** packet arrives
2. Extract and **deflect** a low-priority packet

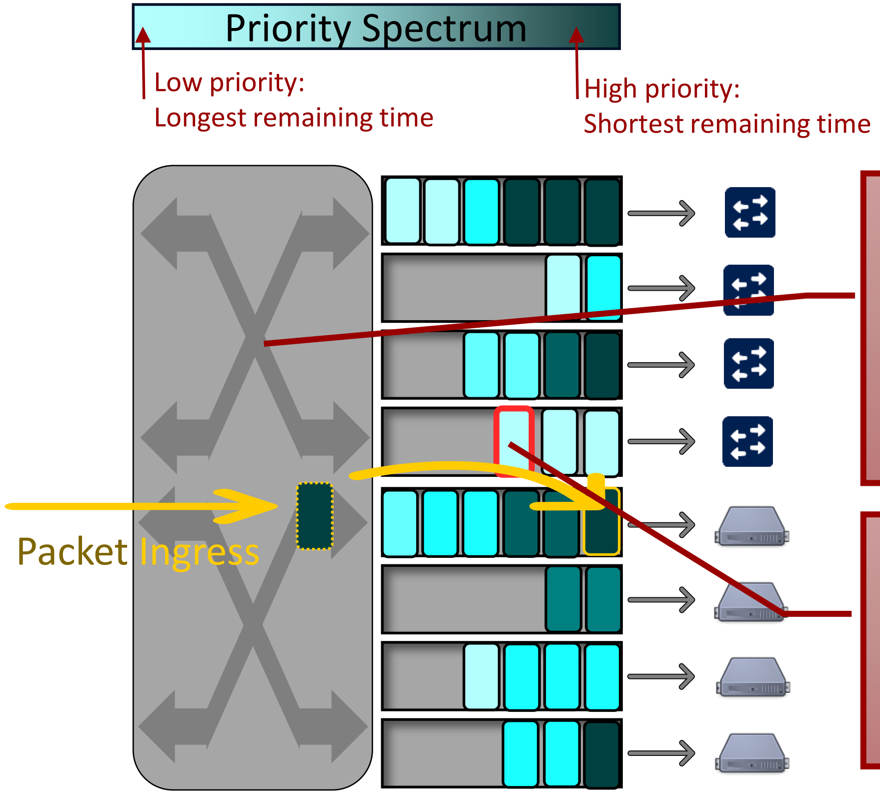
What Makes **Selective Deflection** Hard to Implement ?



Operation Steps

1. **High-priority** packet arrives
2. Extract and **deflect** a **low-priority** packet
3. **Insert** the **new packet** at the head of its queue
4. The deflected packet experiences **extra hop latency** instead of retransmission!

What Makes **Selective Deflection** Hard to Implement ?



CHALLENGE 1: Switch traffic manager is limited to **FIFO** queues

CHALLENGE 2: **Packet extraction** is not feasible after inserting the packet into a queue

Operation Steps

1. **High-priority** packet arrives
2. Extract and **deflect** a **low-priority** packet
3. **Insert** the **new packet** at the head of its queue
4. The deflected packet experiences **extra hop latency** instead of retransmission!

Selective Deflection is effective, even under **extreme load**!

(up to **43x** lower query completion times)

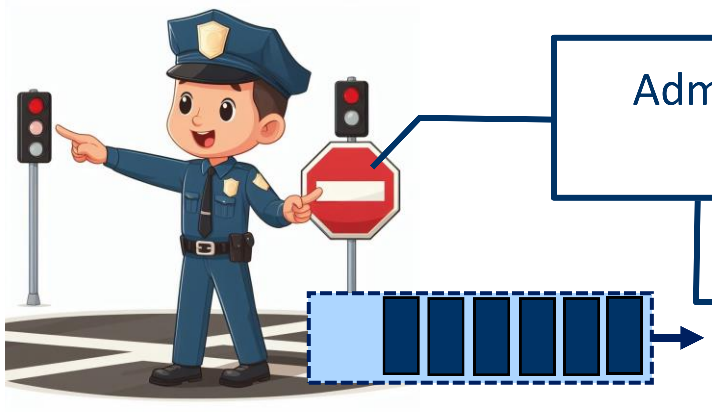
[Vertigo, Conext '21]

Can we use **existing traffic management capabilities** to approximate Selective Deflection?

Implementable Selective Deflection in PISA

Can we use existing traffic management capabilities to approximate **Selective Deflection**?

Yes, with **Preemptive Deflection!**



Admission policy: Should I **admit** the packet that just arrived?

Compare its priority to a **representation** of existing packets

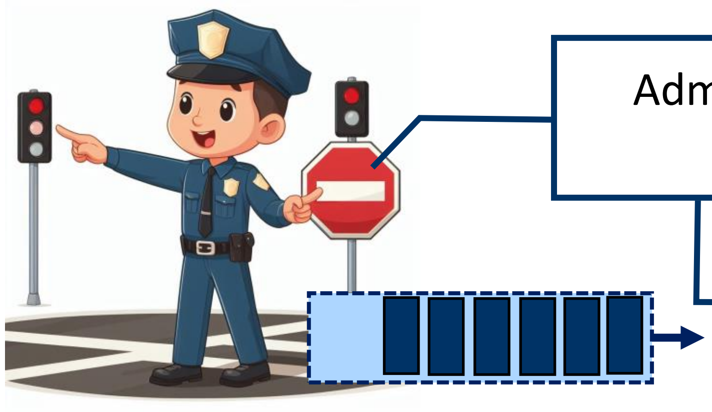
Queue Capacity | Relative Packet Priority | Sensitivity Knob

Deflection Threshold = × [1 − ×]
(τ)

Implementable Selective Deflection in PISA

Can we use existing traffic management capabilities to approximate **Selective Deflection**?

Yes, with **Preemptive Deflection!**



Admission policy: Should I **admit** the packet that just arrived?

Compare its priority to a **representation** of existing packets

Quantile-based

Queue Size

Sensitivity Parameter

Arriving packet: 1

> > > > > ≤

Queue: 4 5 2 3 2 1

$\tau = 8 \times [1 - 0.5 \times {}^{1}/_{6}] =$ **7.3** ✔ ADMIT

Arriving packet: 4

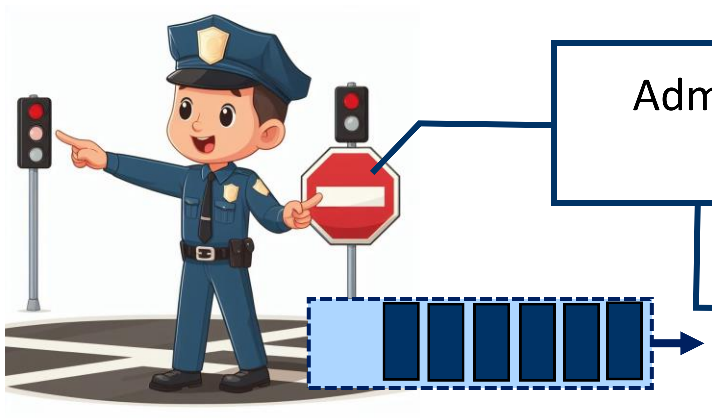
≤ > ≤ ≤ ≤ ≤

$\tau = 8 \times [1 - 0.5 \times {}^{5}/_{6}] =$ **4.6** DEFLECT

Implementable Selective Deflection in PISA

Can we use existing traffic management capabilities to approximate **Selective Deflection**?

Yes, with **Preemptive Deflection!**



Admission policy: Should I admit the packet that just arrived?

Compare its priority to a **representation** of existing packets

Quantile-based

- ✓ Implementable using a sample rolling window for arriving packets
- \- Requires consecutive comparisons in limited PISA stages.

Distribution-based

- ✓ Implementable using pre-filled tables
- ✓ Takes only 2 processing stages!
- \- Less accurate than quantile-based deflection

Putting it All Together

Practical Deflection in Datacenters

Simple Deflection

- Packet **recirculation** for syncing queue utilization data
- **Bitmaps** for randomly selecting non-congested ports

✓ Effective when congestion is infrequent

✓ Requires minimal resources, no external input

Quantile-based Preemptive Deflection

- **Admission vs deflection** policy on FIFO queues
- **Priority comparators** parallelized in PISA pipelines

✓ Can Handle Extreme degrees of congestion

✓ Accurately approximates selective deflection

Distribution-based Preemptive Deflection

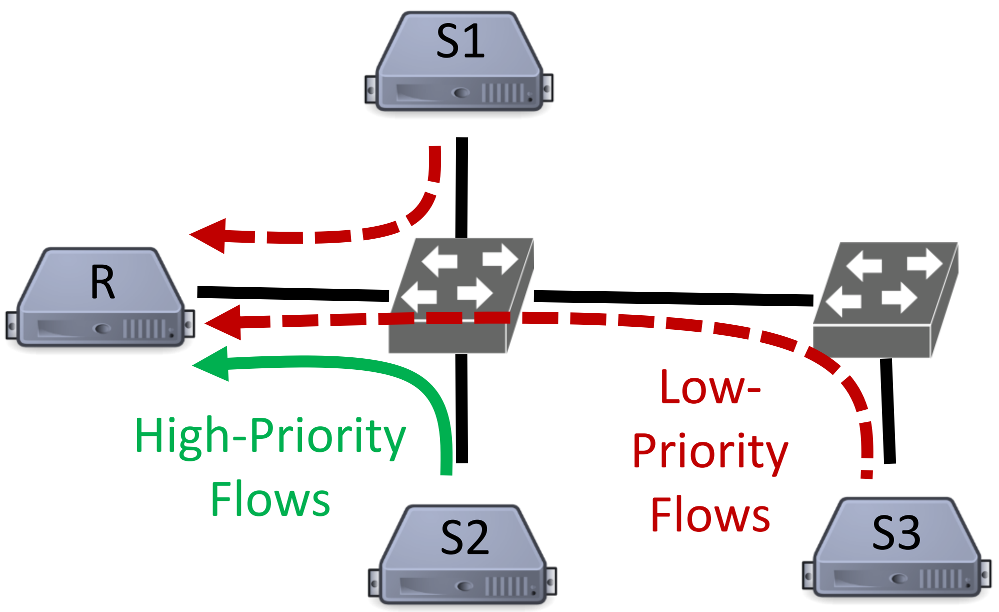
- Using tables prefilled with **statistical distribution** mean-values

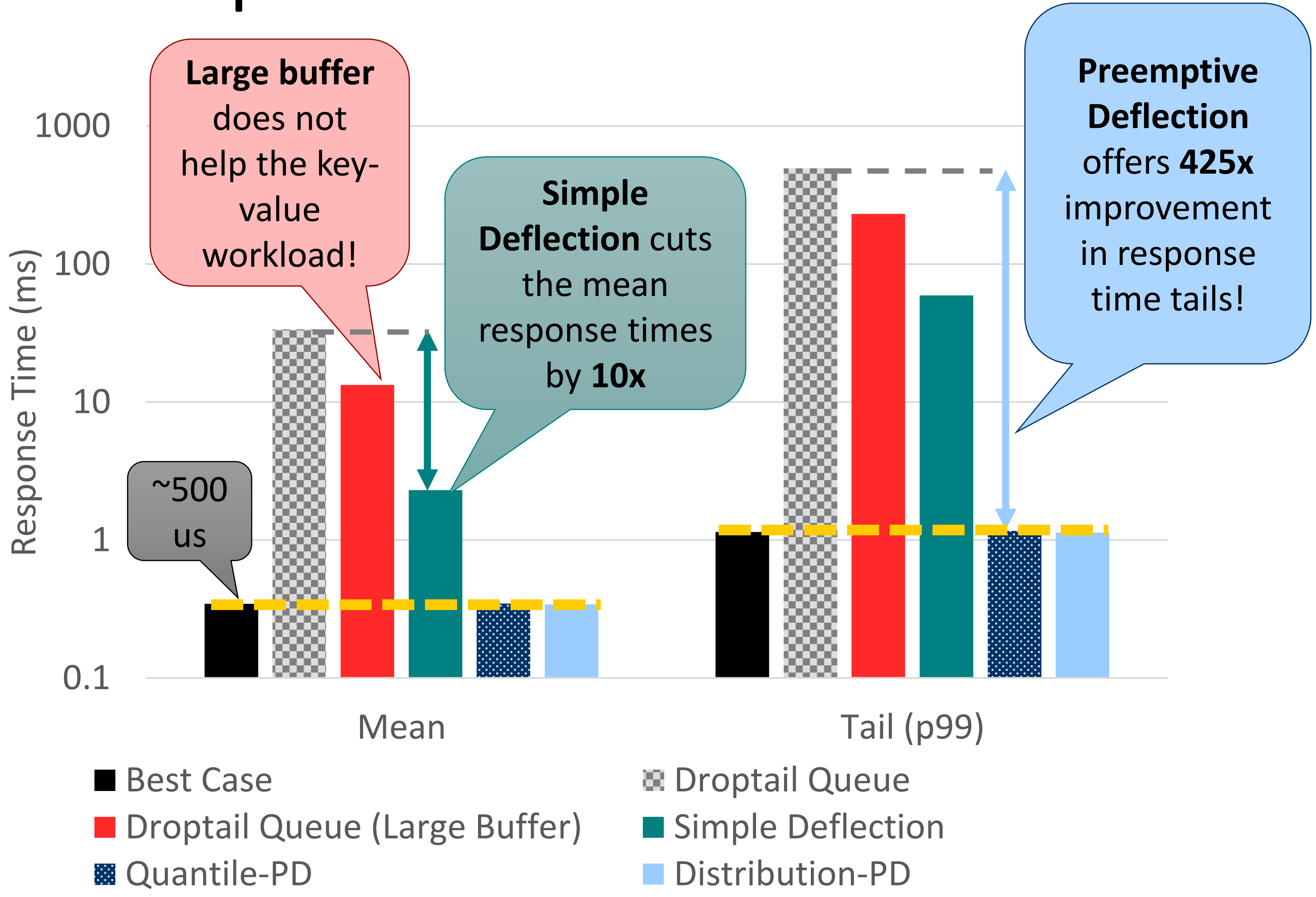
✓ Can handle large degrees of congestion

✓ Requires few processing stages in Tofino

Implementable Deflection Improves the Performance

Physical Testbed Setup

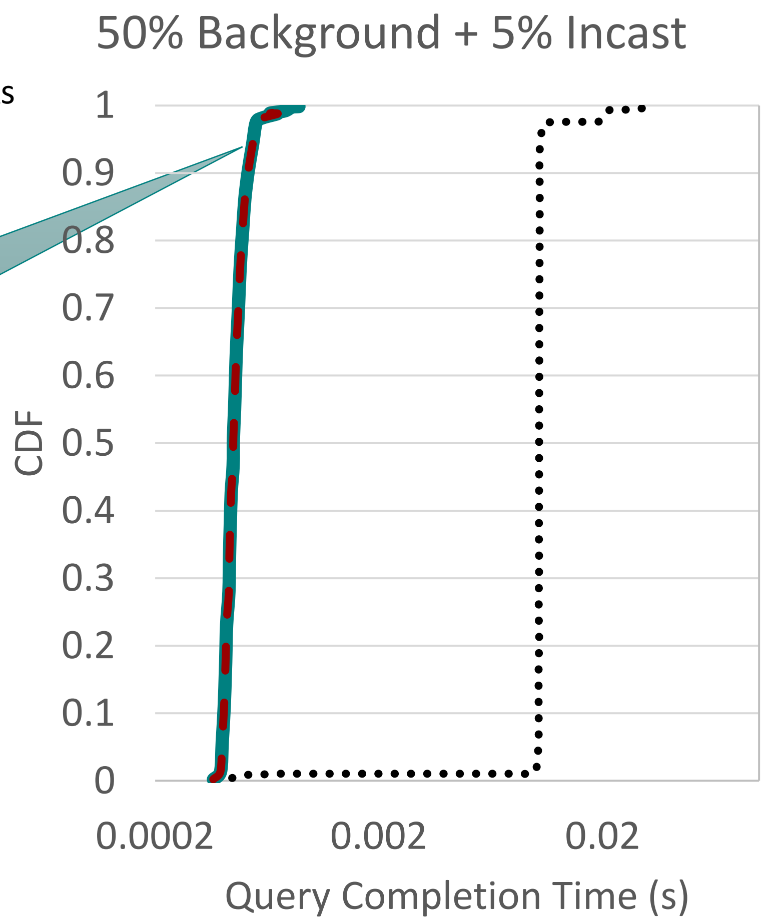


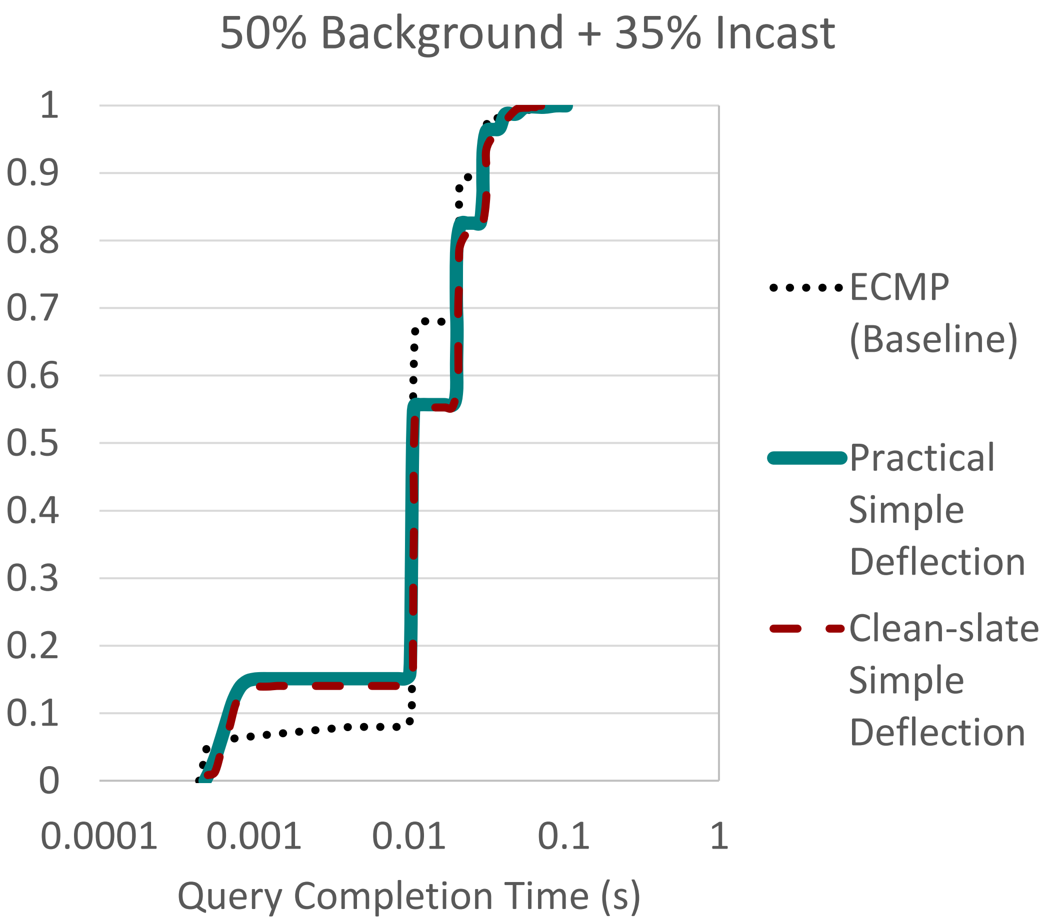


Implementable Deflection Under Large-scale Incast

- Large-scale OMNET simulations
- 2-tier leaf-spine with 100 Gbps links
- 40 machines with all-to-all traffic
- Swift Congestion control

Simple Deflection is effective under moderate congestion!



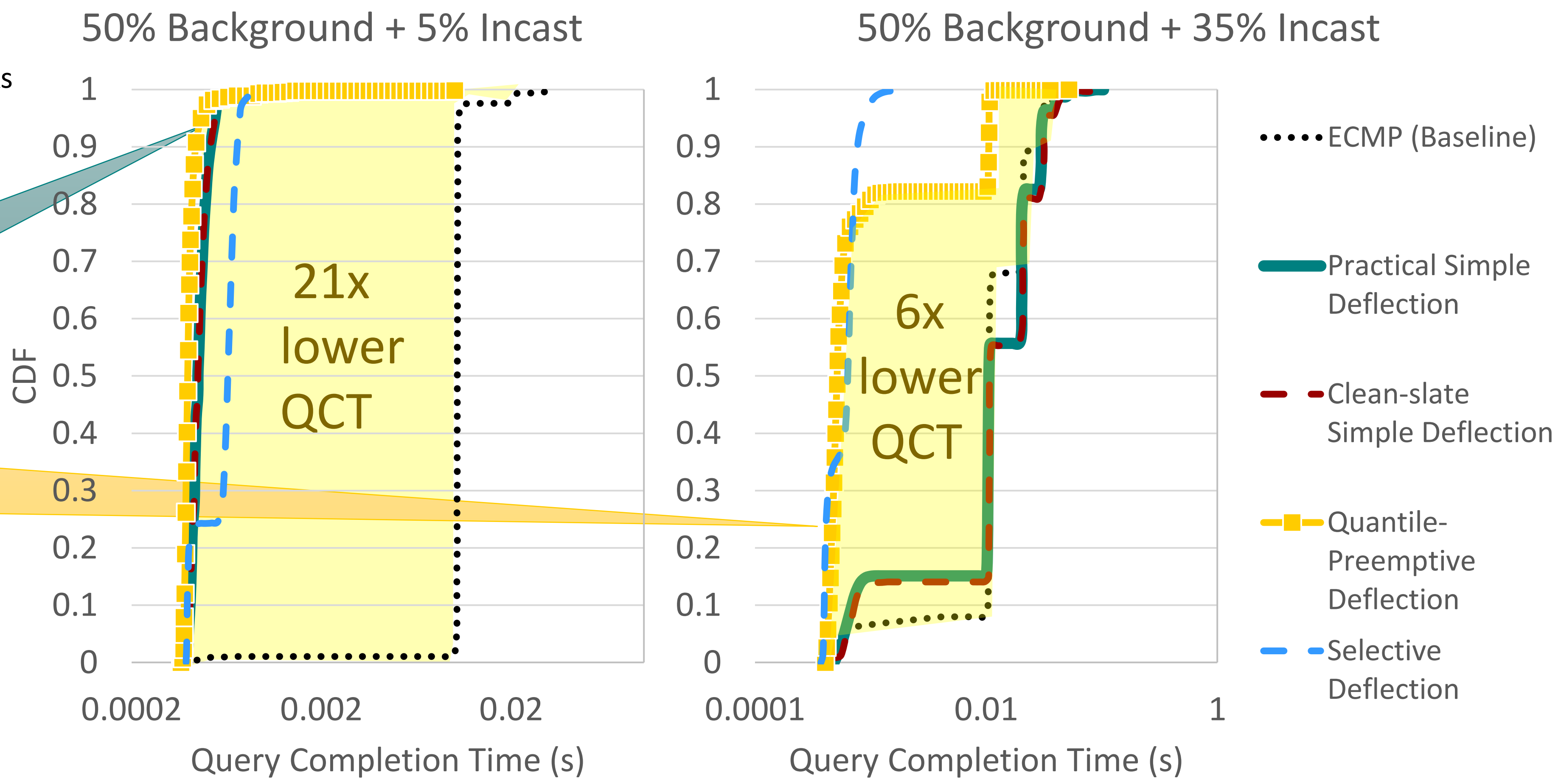


Implementable Deflection Under Large-scale Incast

- Large-scale OMNET simulations
- 2-tier leaf-spine with 100 Gbps links
- 40 machines with all-to-all traffic
- Swift Congestion control

Simple Deflection is effective under moderate congestion!

Preemptive Deflection offers **superior** performance under **large Incast**



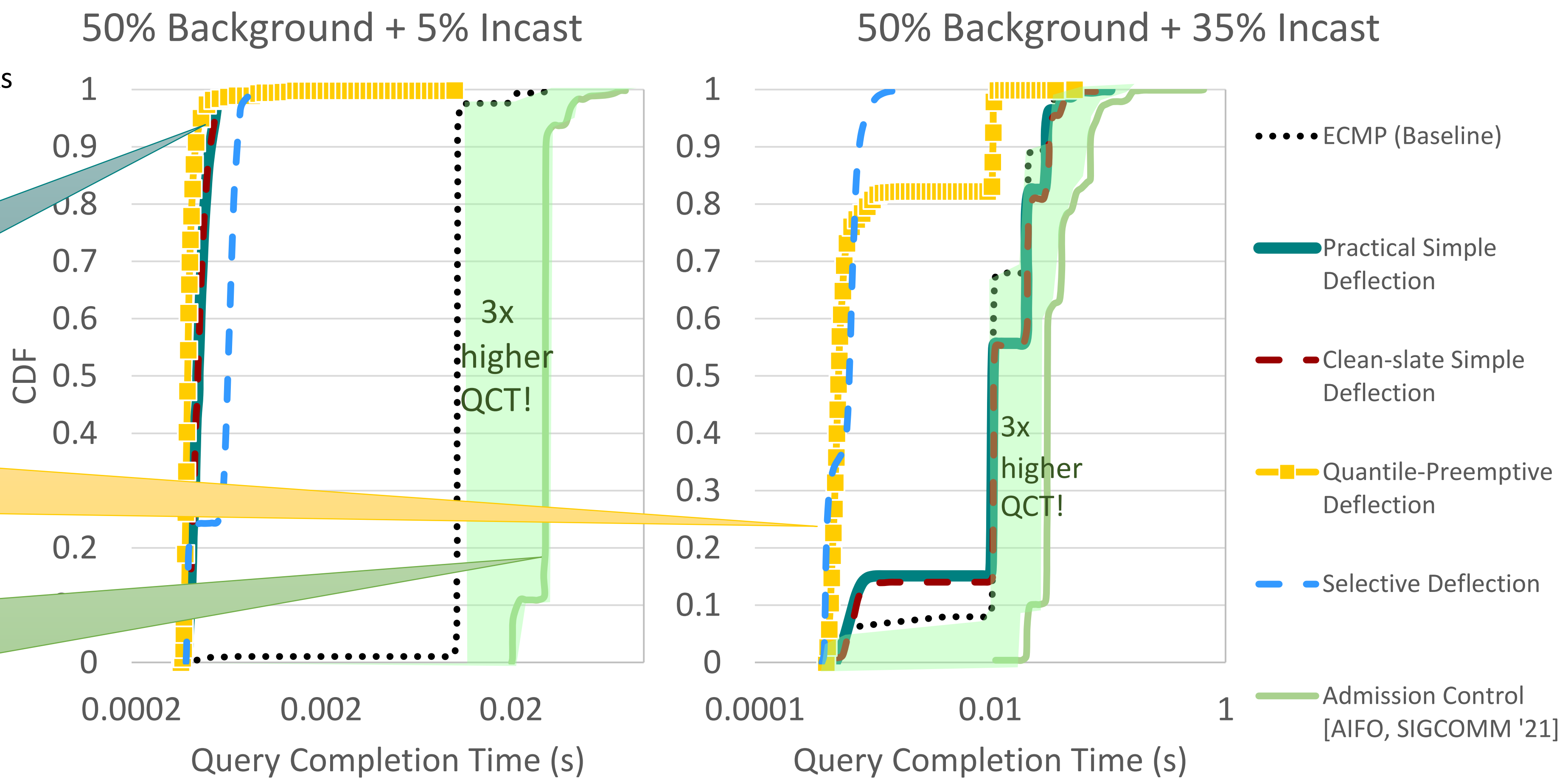
Implementable Deflection Under Large-scale Incast

- Large-scale OMNET simulations
- 2-tier leaf-spine with 100 Gbps links
- 40 machines with all-to-all traffic
- Swift Congestion control

Simple Deflection is effective under moderate congestion!

Preemptive Deflection offers **superior** performance under **large Incast**

Early drop and packet prioritization **alone cannot recover** from short-term local bursts



JOHNS HOPKINS UNIVERSITY

We Made Packet Deflection Practical

- We propose an accurate implementation of **Simple Deflection** on PISA architecture.
- We introduce **Preemptive Deflection**, an approximation of selective deflection on PISA.
- Choosing among deflection techniques depends on:
  - Network utilization & congestion intensity
  - Resource availability
  - Performance requirements
- Preemptive Deflection improves high-priority Flow Completion Times by **425x** in a physical testbed.
- Visit https://hopnets.github.io/practical_deflection for the codebase
- Contact Authors: sabdous1@jhu.edu, erfan@cs.jhu.edu

Illustrations generated using Bing AI

Backup slides