



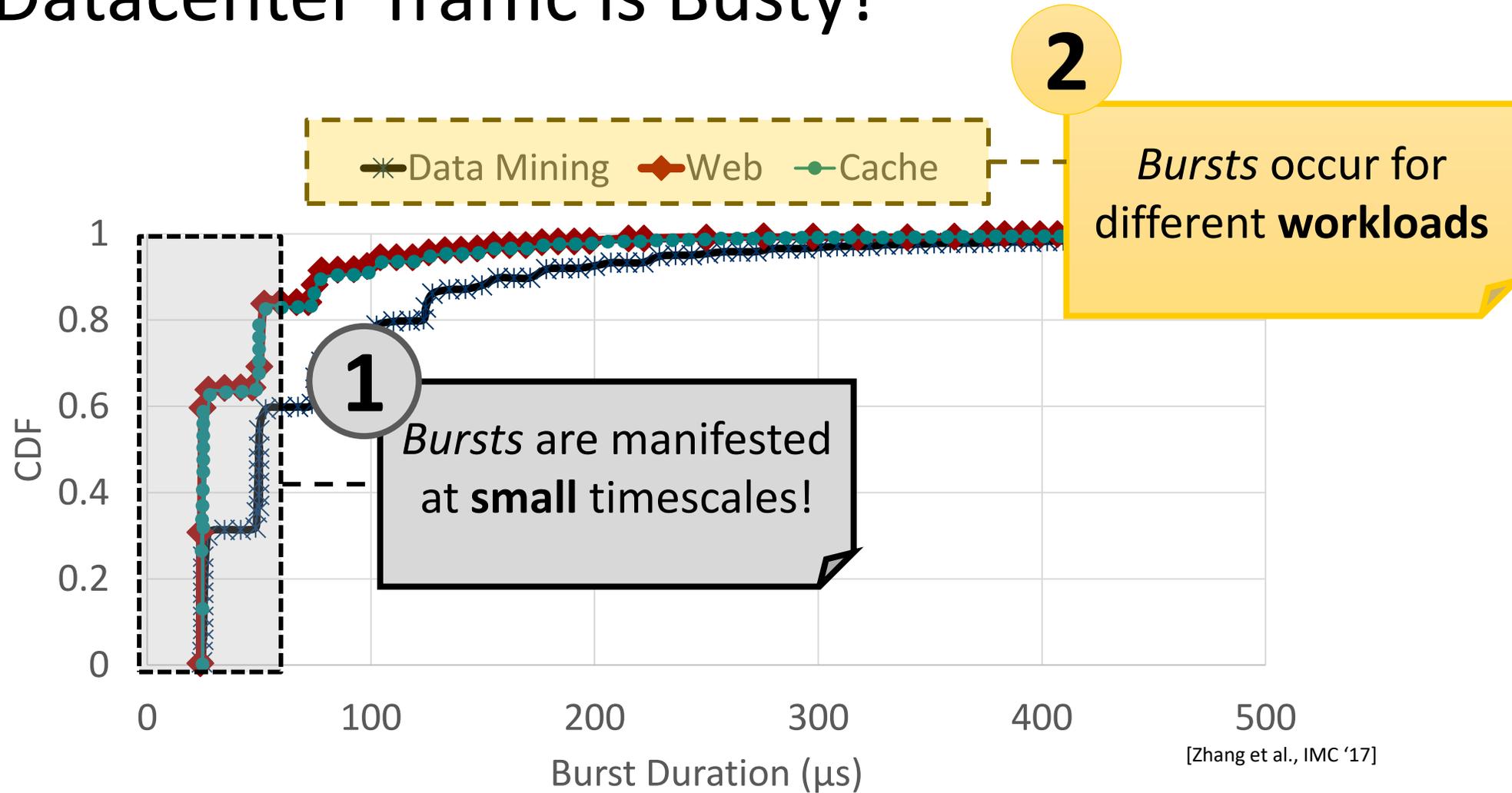
Practical Packet Deflection in Datacenters

Sepehr Abdous, Erfan Sharafzadeh, Soudeh Ghorbani



December 2023

Datacenter Traffic is Busty!



Existing Techniques Fall Short Against Bursts



Congestion control

TIMELY: RTT-based Congestion Control for the Datacenter

Radhika Mittal¹(UC Berkeley),
Monia Ghobadi¹(Microsoft), Emily Blem, Hassan War

Swift: Delay is Simple and Effective for Congestion Control in the Datacenter

Gautam Kumar, Nandita Dukkipati, Keon Jang (MPI-SWS), Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat
Google LLC

Low Latency

HPCC: High Precision Congestion Control

Yuliang Li^{1,2}, Rui Miao¹, Hongqiang Harry Liu¹, Yan Zhuang¹, Fei Feng¹, Lingbo Tang¹, Zheng Cao¹, Ming Zhang¹, Frank Kelly³, Mohammad Alizadeh¹, Minlan Yu²
Alibaba Group¹, Harvard University², University of Cambridge³, Massachusetts Institute of Technology⁴

Yuliang Li
Google LLC

Gautam Kumar
Google LLC

Nandita Dukkipati
Google LLC

Data Center TCP (DCTCP)

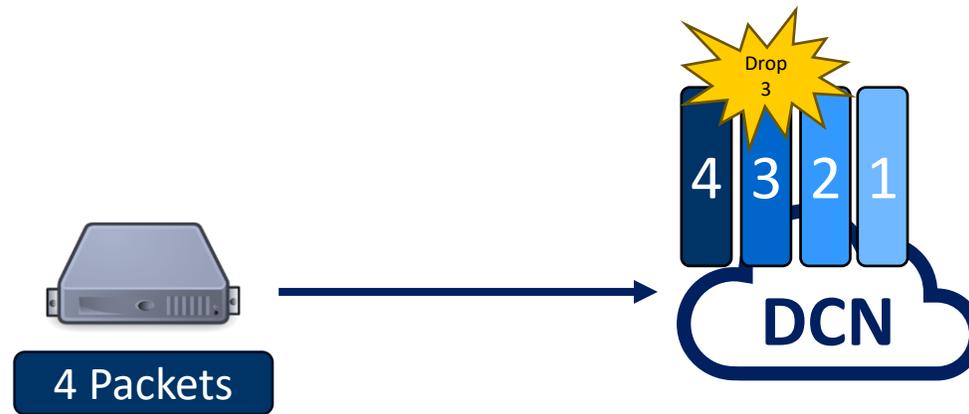
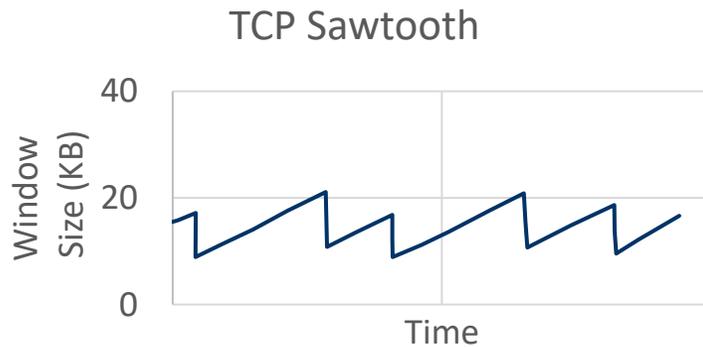
Mohammad Alizadeh^{1,2}, Albert Greenberg¹, David A. Maltz¹, Jitendra Padhye¹, Parveen Patel¹, Balaji Prabhakar¹, Sudipta Sengupta¹, Murari Sridharan¹
¹Microsoft Research, ²Stanford University
{albert, dmaltz, padhye, parveenp, sudipta, muraris}@microsoft.com
{alizade, balaji}@stanford.edu

Deadline-Aware Datacenter TCP (D²TCP)

Balajee Vamanan
Purdue University
bvamanan@ecn.purdue.edu

Jahangir Hasan
Google Inc.
jahangir@google.com

T. N. Vijaykumar
Purdue University
vijay@ecn.purdue.edu



Existing Techniques Fall Short Against Bursts



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

TIMELY: RTT-based Congestion Control for the Datacenter

Swift: Delay is Simple and Effective for Congestion Control in the Datacenter

Gautam Kumar, Nandita Dukkipati, Keon Jang (MPI-SWS), Hassan M. G. Wassef, Xian Wu, Behnam Montazeri.

Alibaba Group^{*}, Harvard University[†], University of Cambridge[‡], Massachusetts Institute of Technology[§]

Data Center TCP (DCTCP)

Deadline-Aware Datacenter TCP (D²TCP)

Mohammad Alizadeh^{1†}, Albert Greenberg¹, David A. Maltz¹, Jitendra Padhye¹, Parveen Patel¹, Balaji Prabhakar¹, Sudipta Sengupta¹, Murari Sridharan¹
¹Microsoft Research [†]Stanford University
{albert, dmaltz, padhye, parveenp, sudipta, muraris}@microsoft.com
{alizade, balaji}@stanford.edu

Balajee Vamanan
Purdue University
bvamanan@ecn.purdue.edu

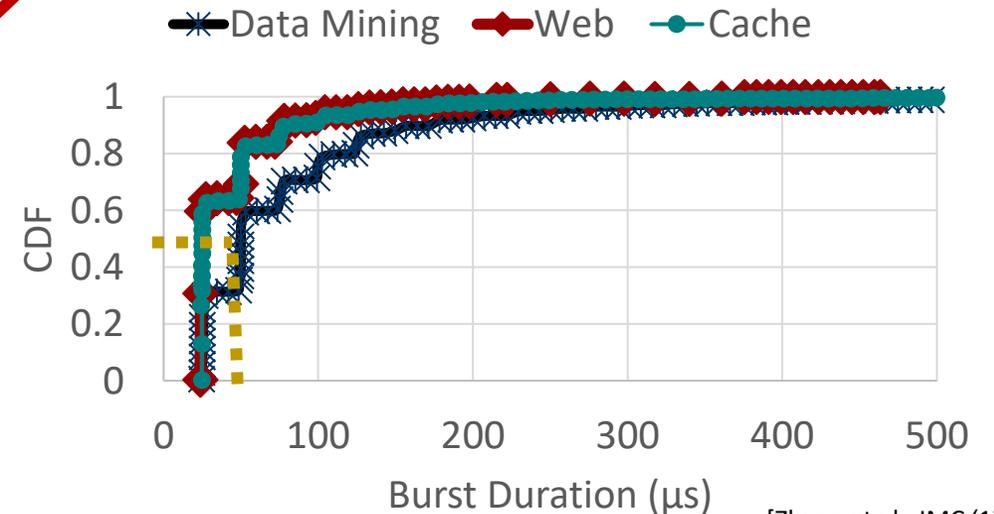
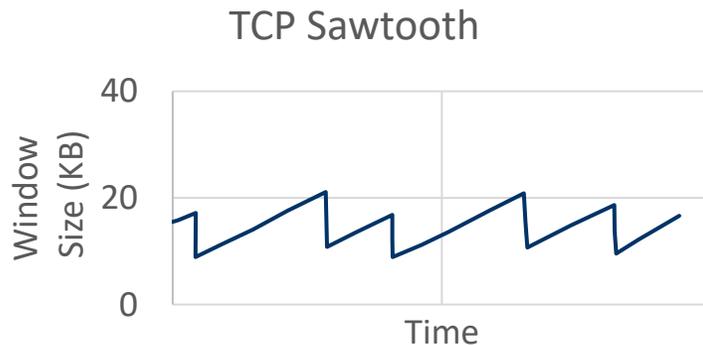
Jahangir Hasan
Google Inc.
jahangir@google.com

T. N. Vijaykumar
Purdue University
vijay@ecn.purdue.edu

~ few milliseconds >> 50 μ s



2 Packets



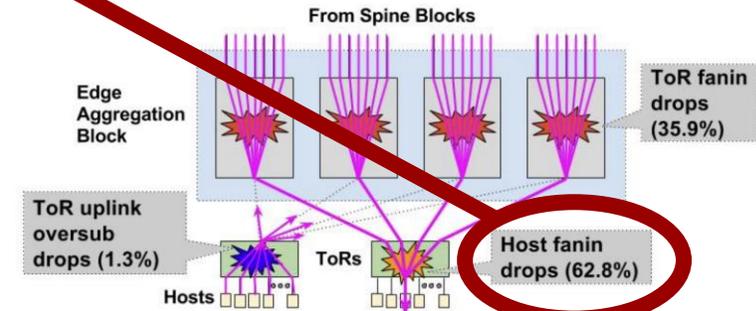
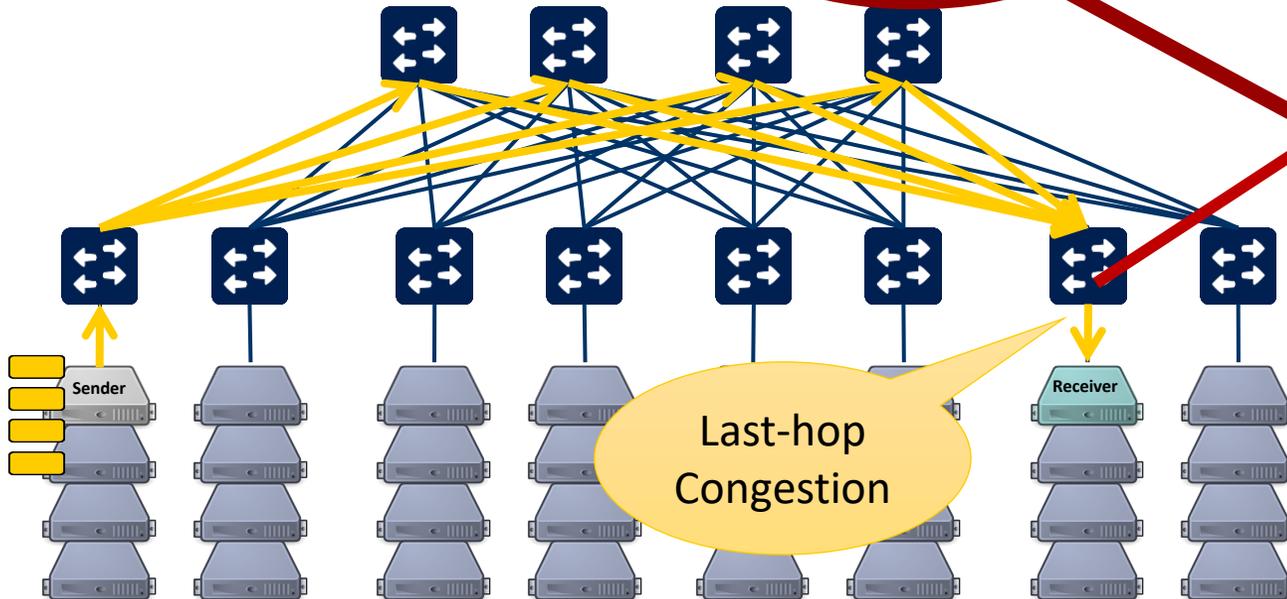
[Zhang et al., IMC '17]

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

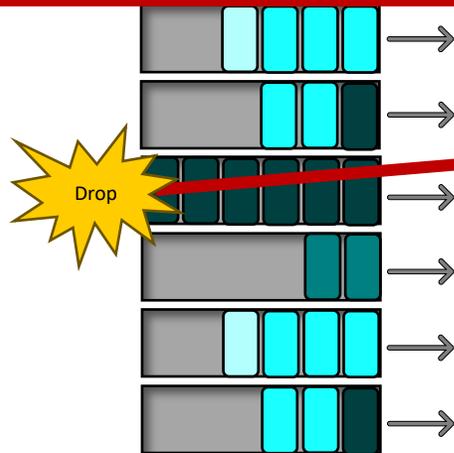
Priority Spectrum

Low priority:

High priority:

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

-  Congestion control
-  Load balancing
-  Packet scheduling



PI²: A Linearized AQM for both Classic and Scalable TCP
 Koen De Schepper¹, Olga Bondarenko², Ing-Jyh Tsang³, Bob Briscoe⁴
¹Nokia Bell Labs, Belgium, ²Simula Research Laboratory, Norway, ³{joigabo|bob}@simula.no

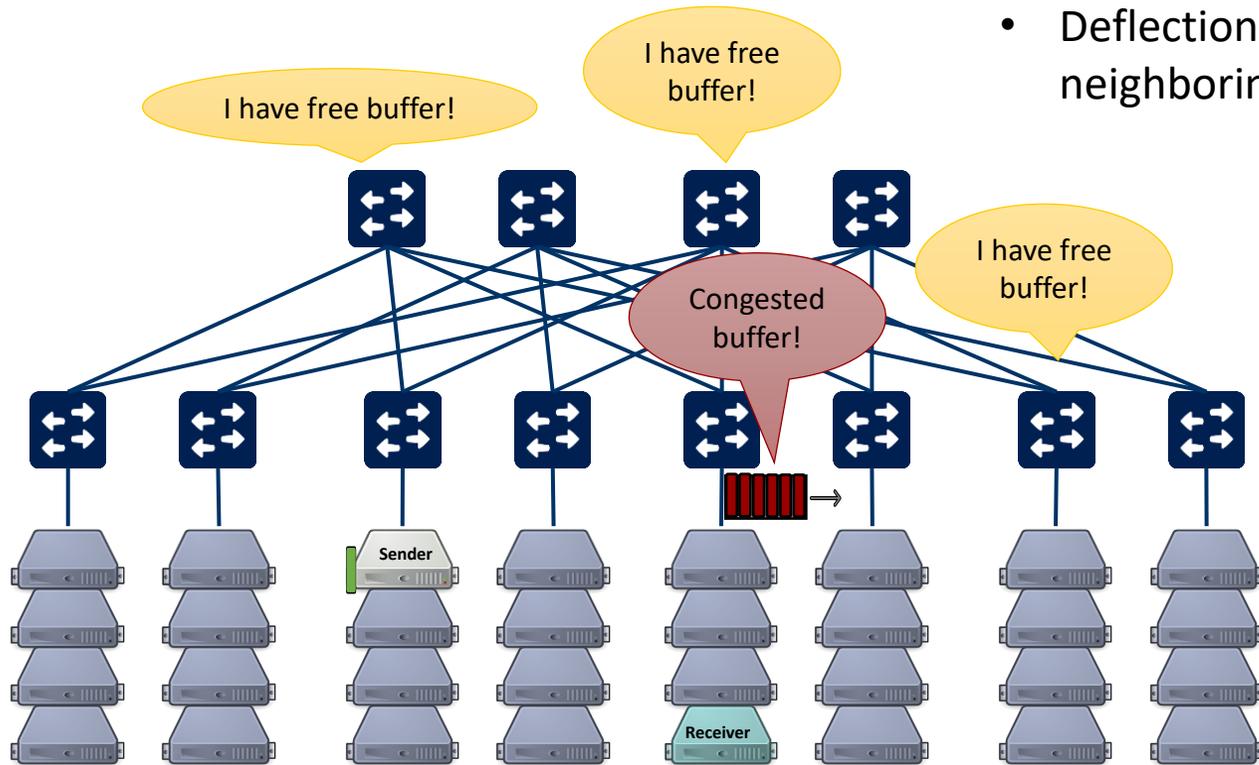
Carousel: Scalable Traffic
 Ahmed Saeed¹, Nandita D. Srikant²
¹Institute of Technology, ²Google, Inc.

Programmable Packet Scheduling with a Single Queue
 Zhuolong Yu¹, Chuheng Hu¹, Jingfeng Wu¹
 Johns Hopkins University, Johns Hopkins University, Johns Hopkins University
 Xiao Sun¹, Vladimir Braverman¹, Mosharaf Chowdhury¹
 Stony Brook University, Johns Hopkins University, University of Michigan
 Zhenhua Liu¹, Xin Jin¹
 Stony Brook University, Peking University

Programmable Queues for High-speed Packet Scheduling
 Chenxinevu Zhao^{*}, Ming Liu^{*}, Pravein G Kannan[†], Changhoon Kim[‡],
 nan[§]

Programmable Packet Scheduling at Line Rate
 rvinay Subramanian¹, Mohammad Alizadeh², Sharad Chole³, Shang-Tse Chuang¹, Anurag Agrawal¹,
 Hari Balakrishnan¹, Tom Edsall¹, Sachin Katti¹, Nick McKeown¹
^{*}MIT CSAIL, [†]Barefoot Networks, [‡]Cisco Systems, [§]Stanford University

Packet Deflection Avoids Drops in the Hotspots!



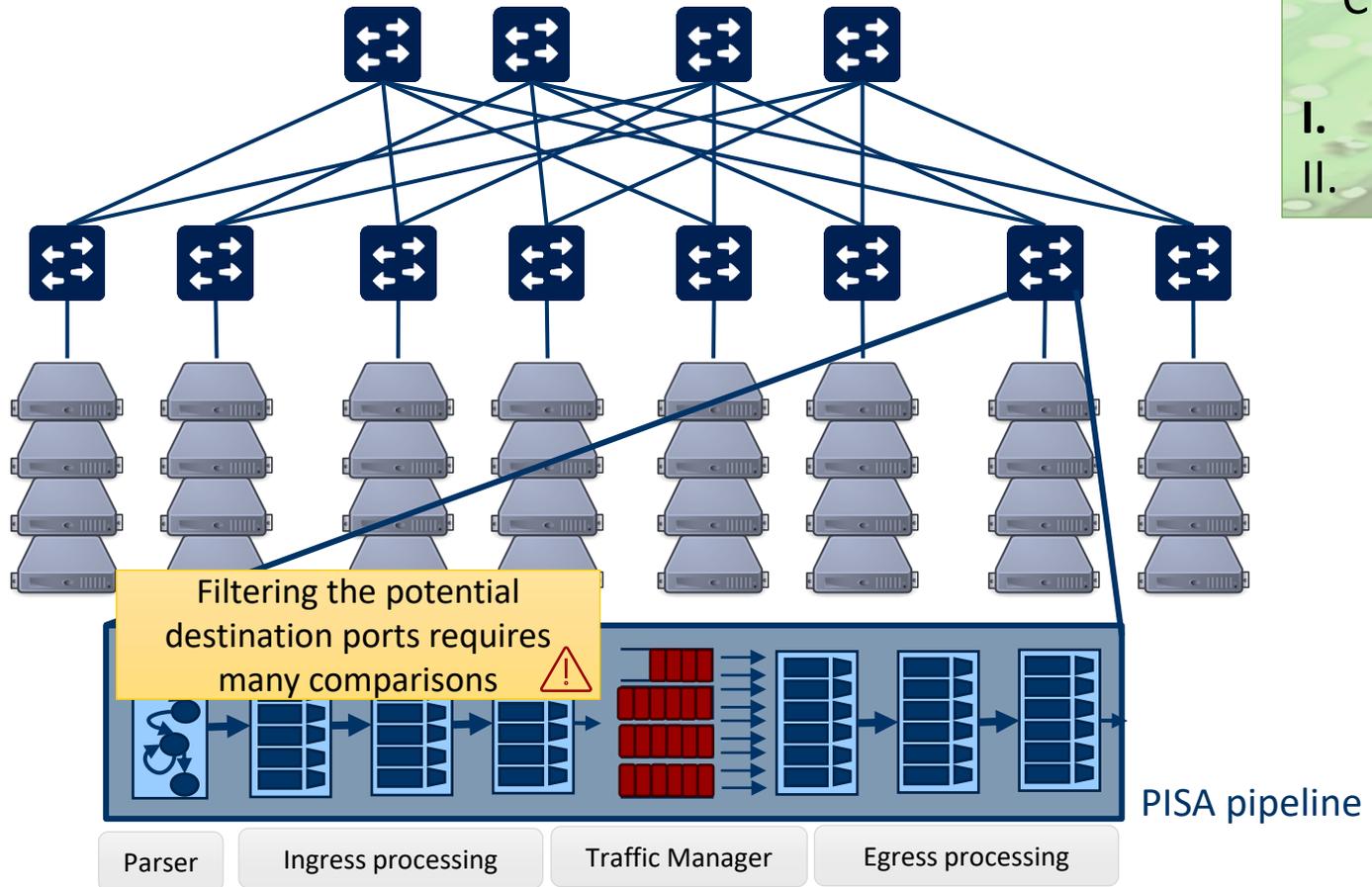
- Deflection: **Re-routing** packets that arrive at a full buffer to a neighboring switch.

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



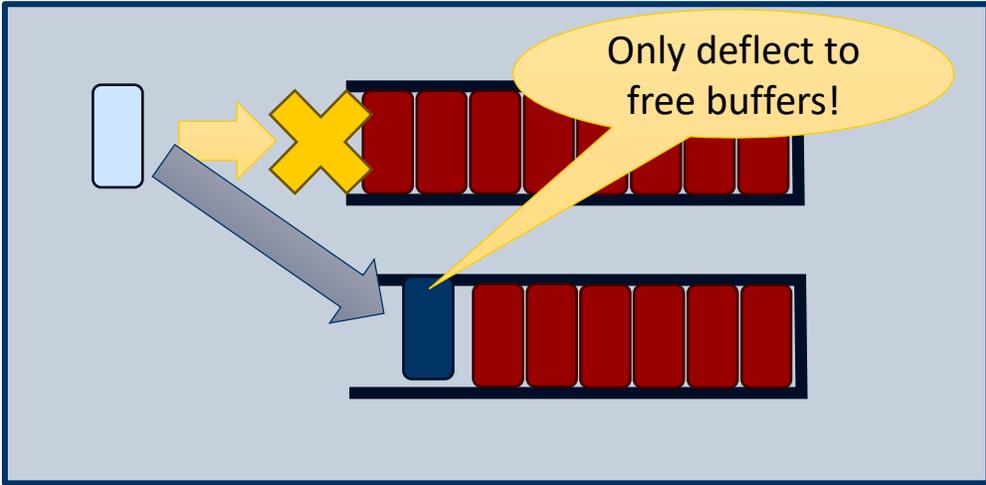
State-of-the-art **deflection** proposals
are **not implementable**
in existing programmable hardware!

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

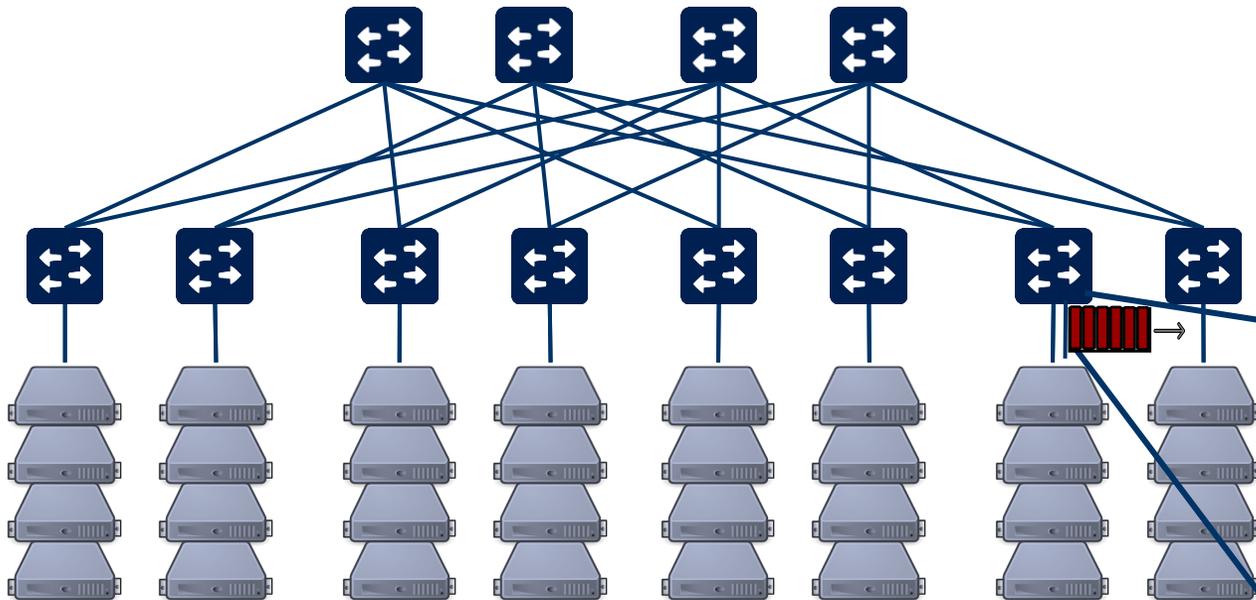


Clean-slate deflection requirements in **real-time**:

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

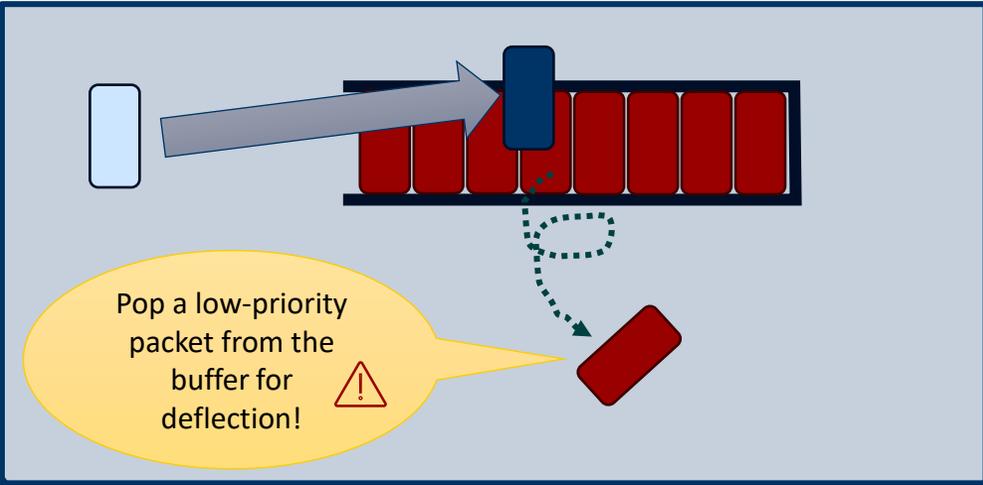


State-of-the-art Deflection Depends on **Two** Primitives



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

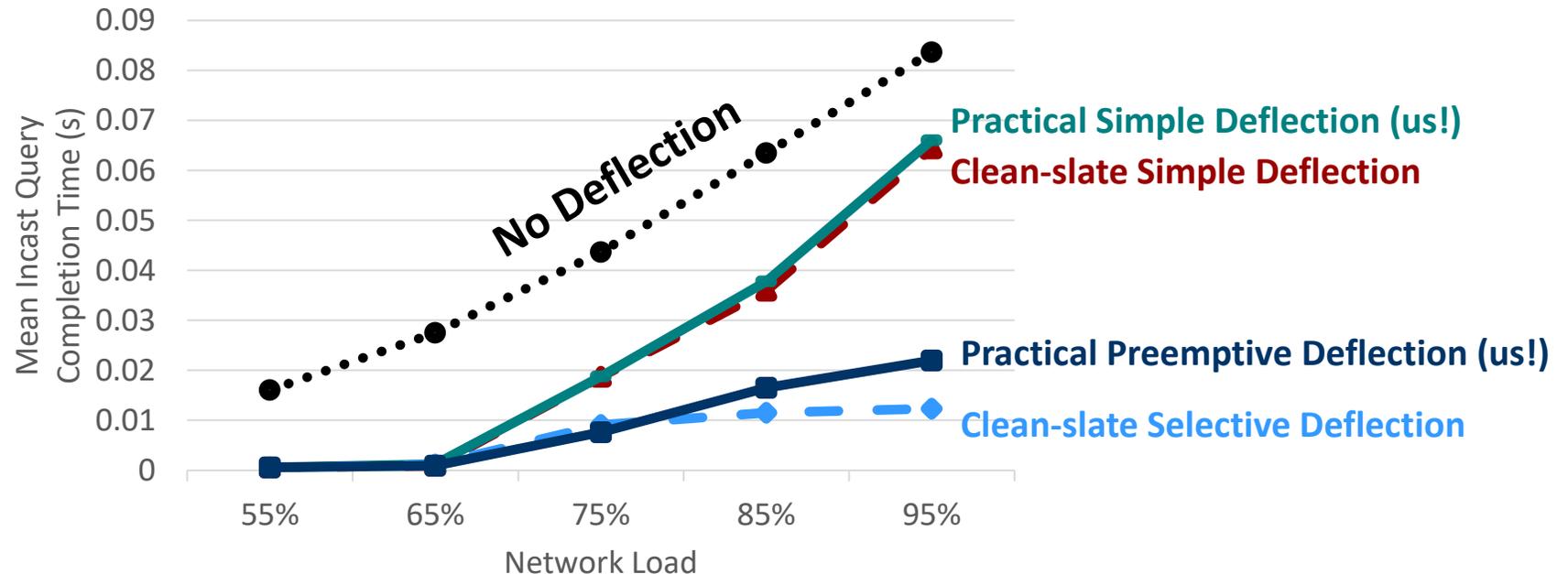
- I. Filtering congested ports
- II. **Extracting packets from the queue**



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.

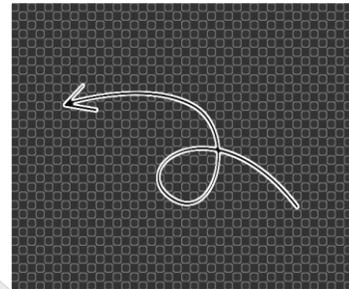
- Using DCTCP Congestion control
- 100 Gbps 8-ary fat-tree cluster
- Incast size of 100 Requests per Query



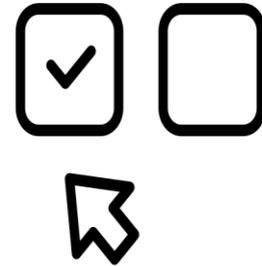
Practical Packet Deflection in Datacenters, CoNEXT 2023

Approaches to Deflection Suited for Different Needs

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

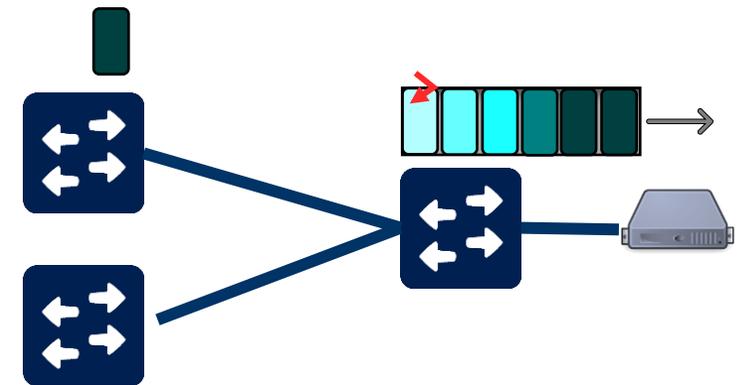
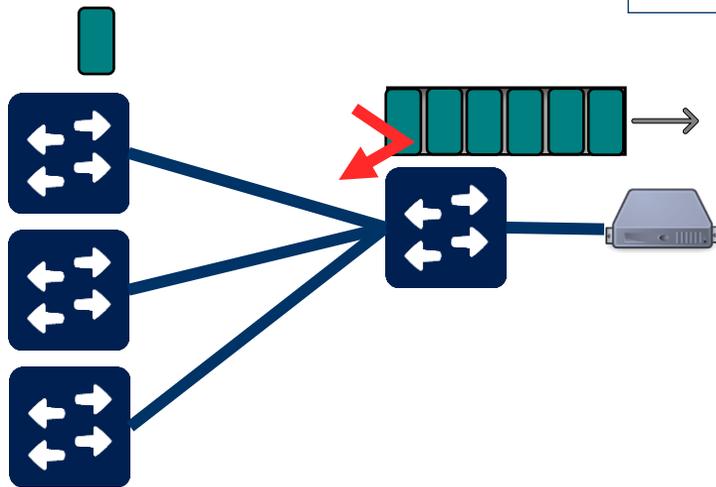
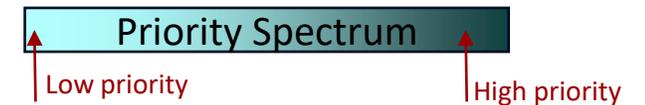


Simple

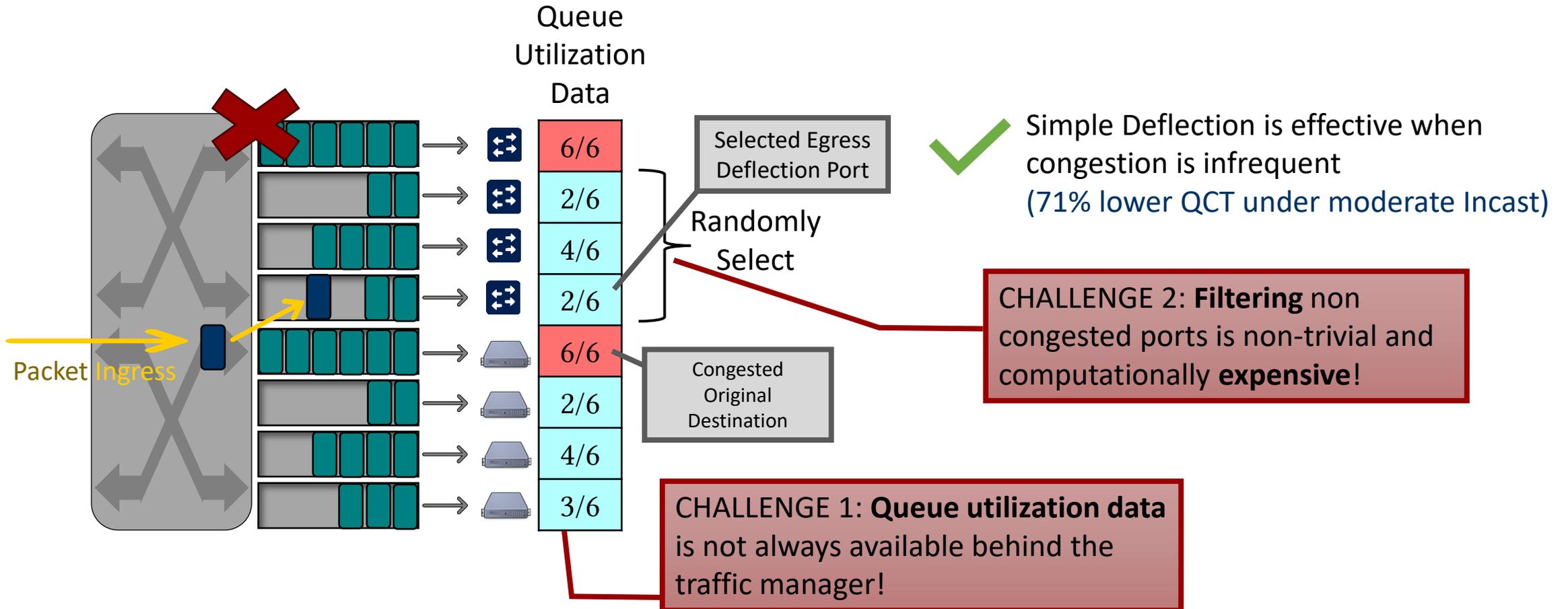


Selective

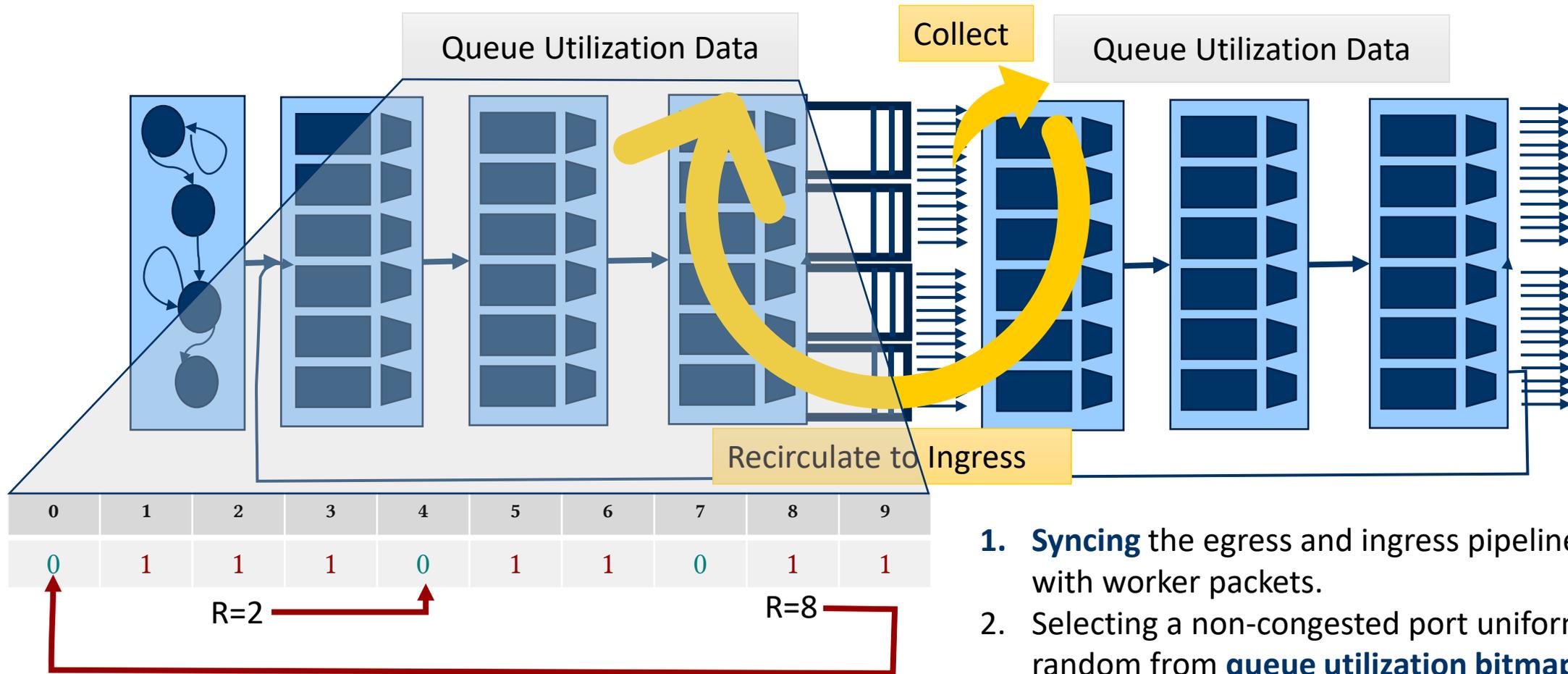
Sort packets in the queue and deflect lower-priority packets!



What Makes Simple Deflection Hard to Implement



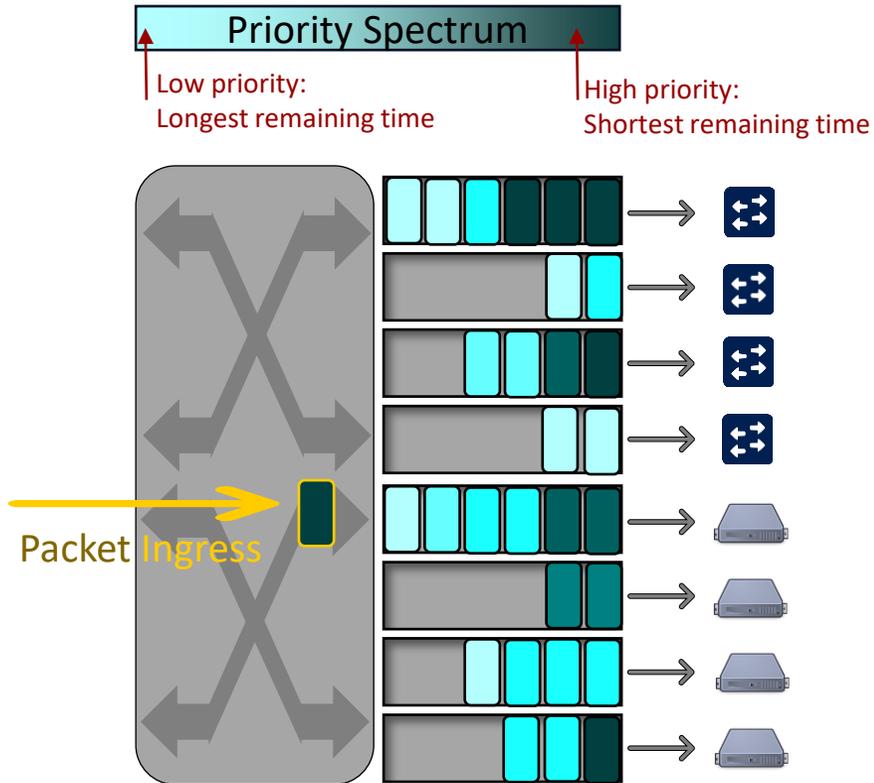
Implementable Simple Deflection in PISA



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



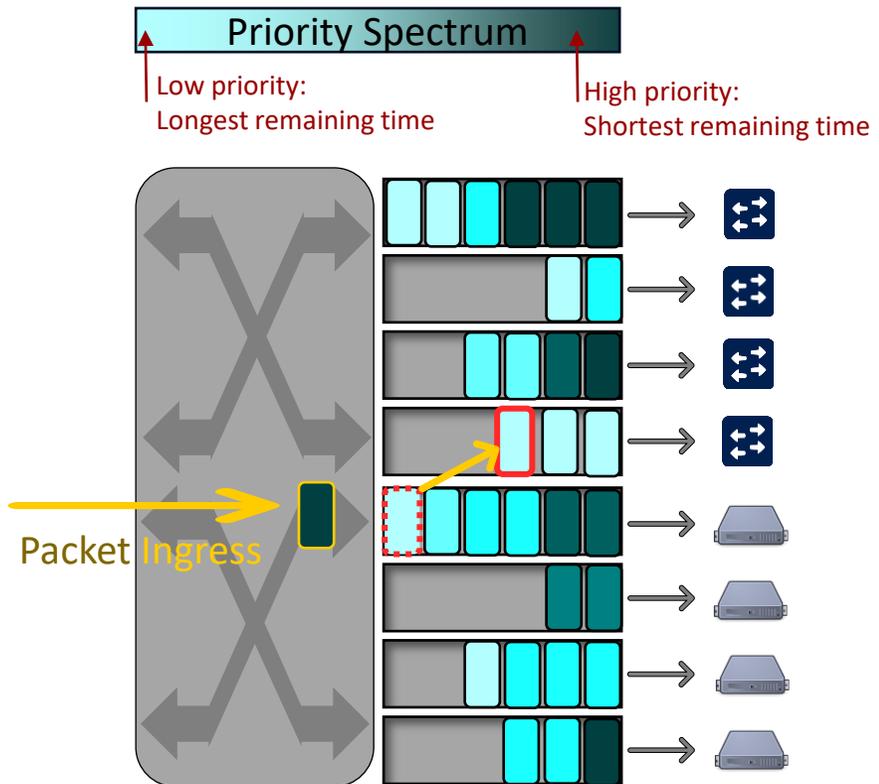
What Makes **Selective Deflection** Hard to Implement



Operation Steps

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

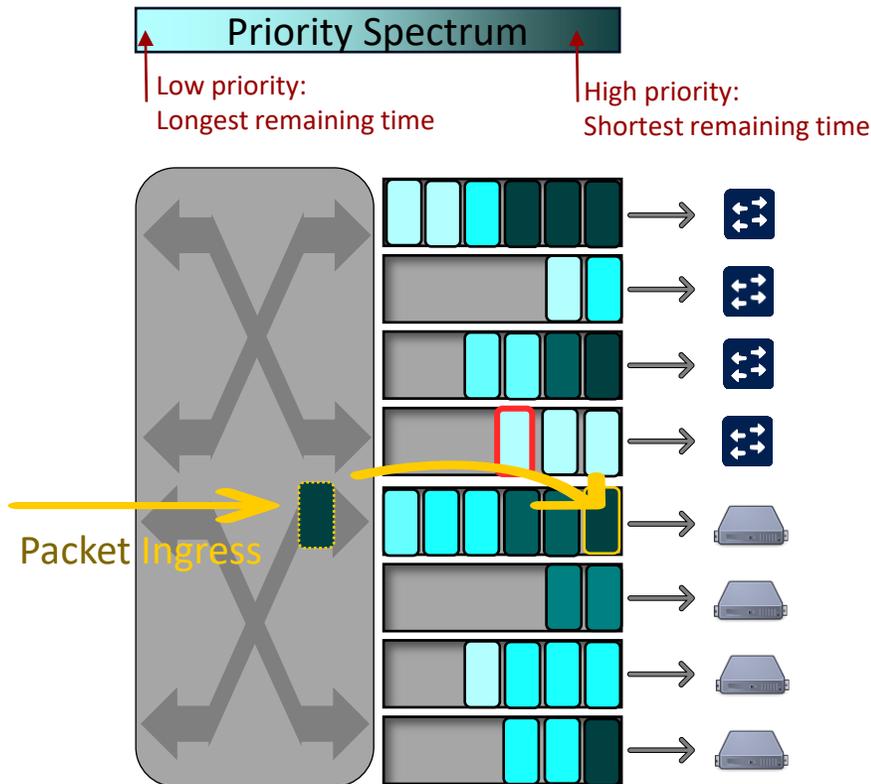
What Makes Selective Deflection Hard to Implement



Operation Steps

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

What Makes Selective Deflection Hard to Implement

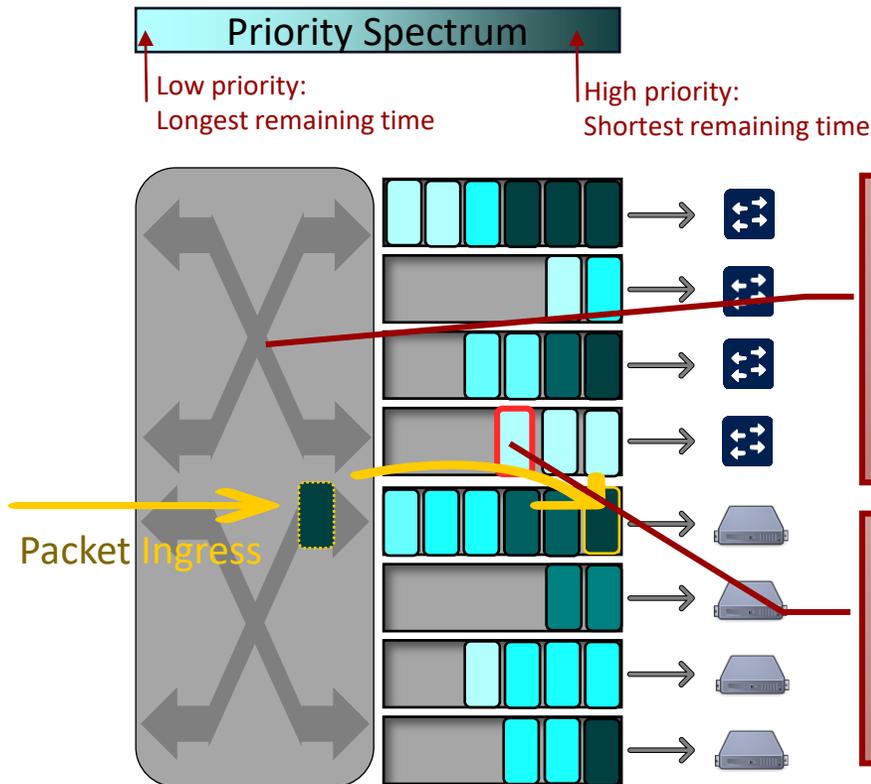


Operation Steps

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!

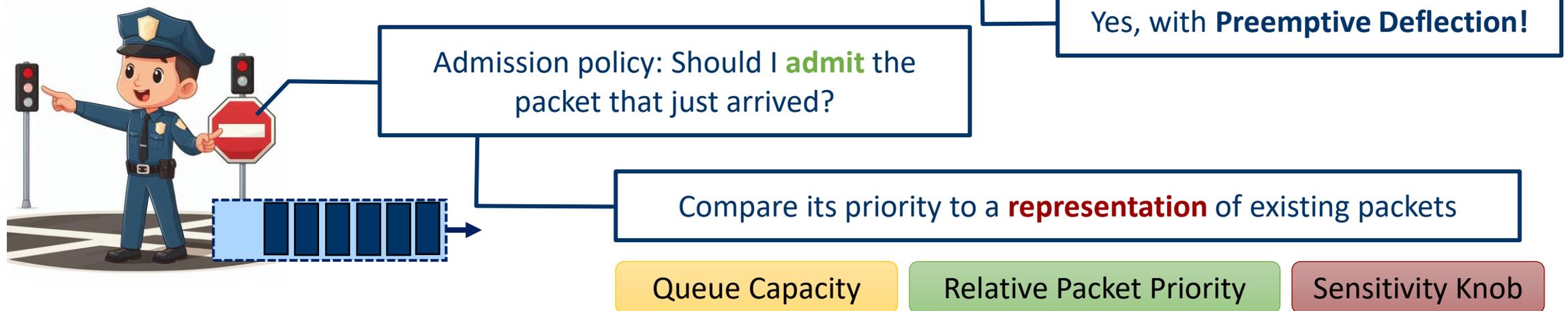
Selective Deflection is effective, even under **extreme load!**

(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**?



$$\text{Deflection Threshold} = (\tau) \times [1 - \times \times]$$

Implementable Selective Deflection in PISA

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

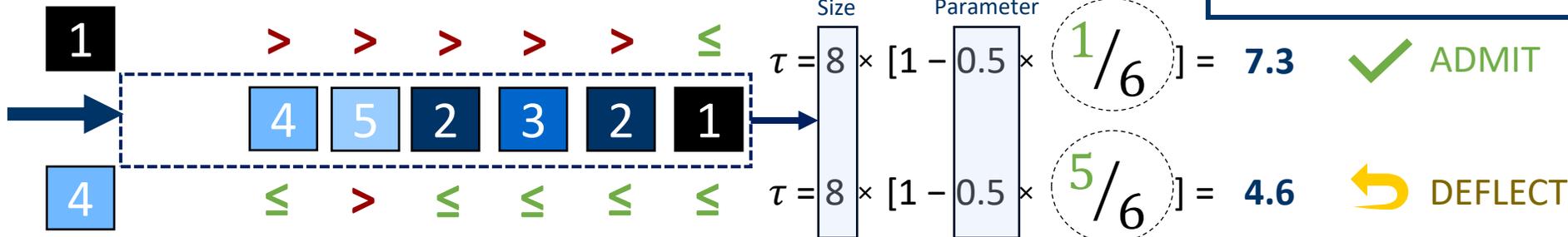
Yes, with **Preemptive Deflection!**



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

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

Quantile-based



Implementable 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

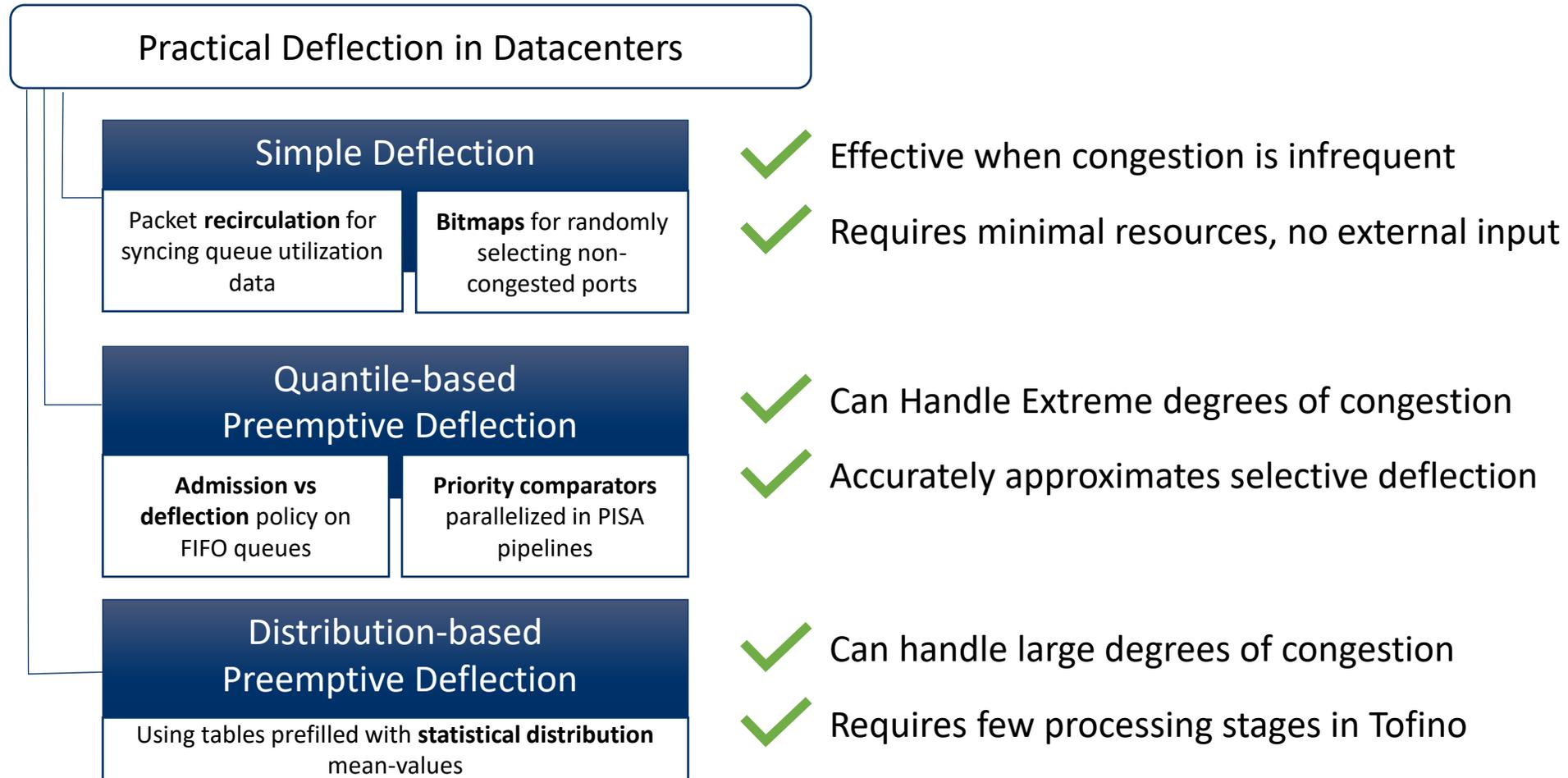
Quantile-based

Distribution-based

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

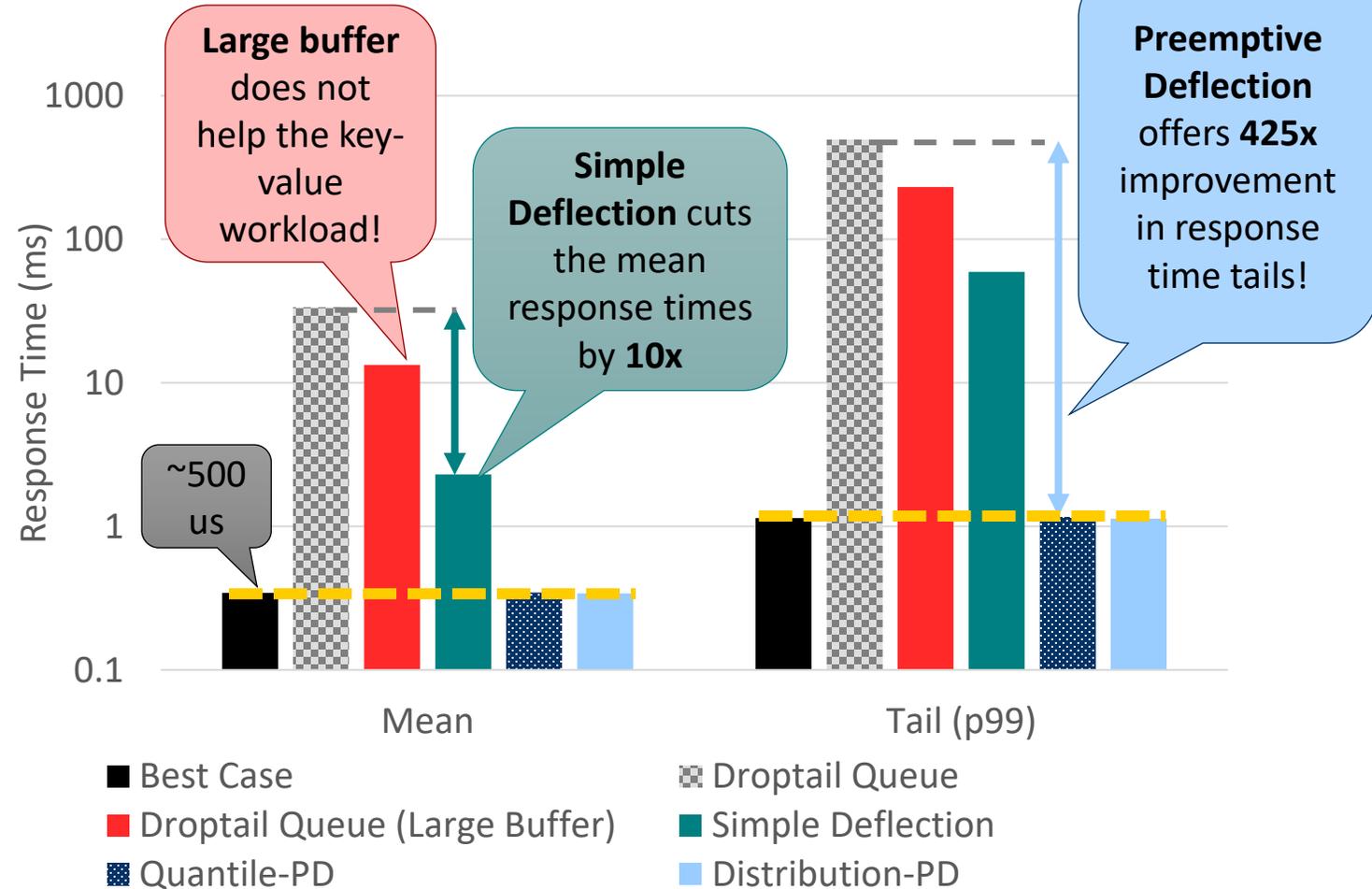
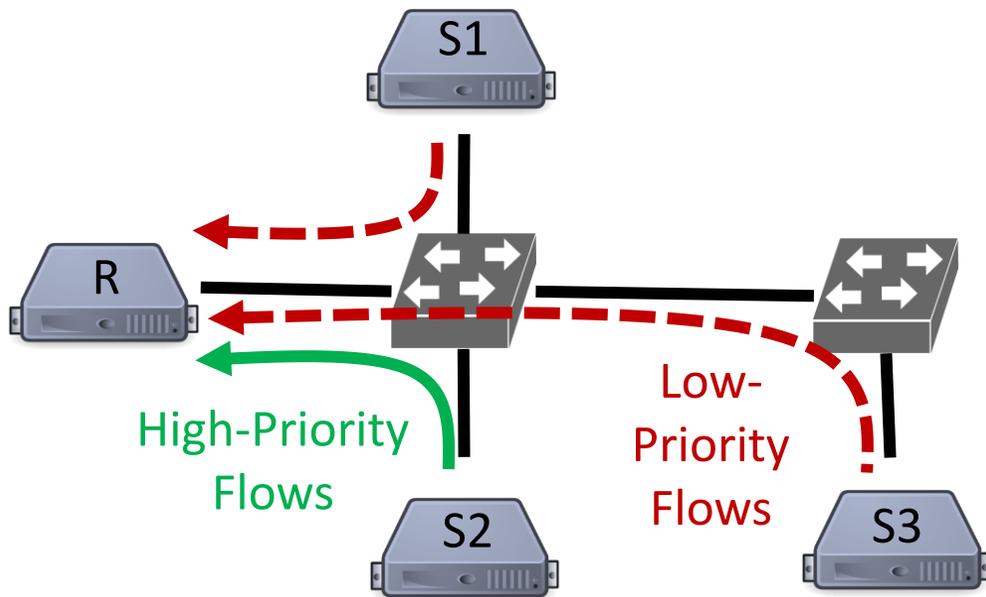
- ✓ Implementable using a sample rolling window for arriving packets
- Requires consecutive comparisons in limited PISA stages.

Putting it All Together



Implementable Deflection Improves the Performance

Physical Testbed Setup

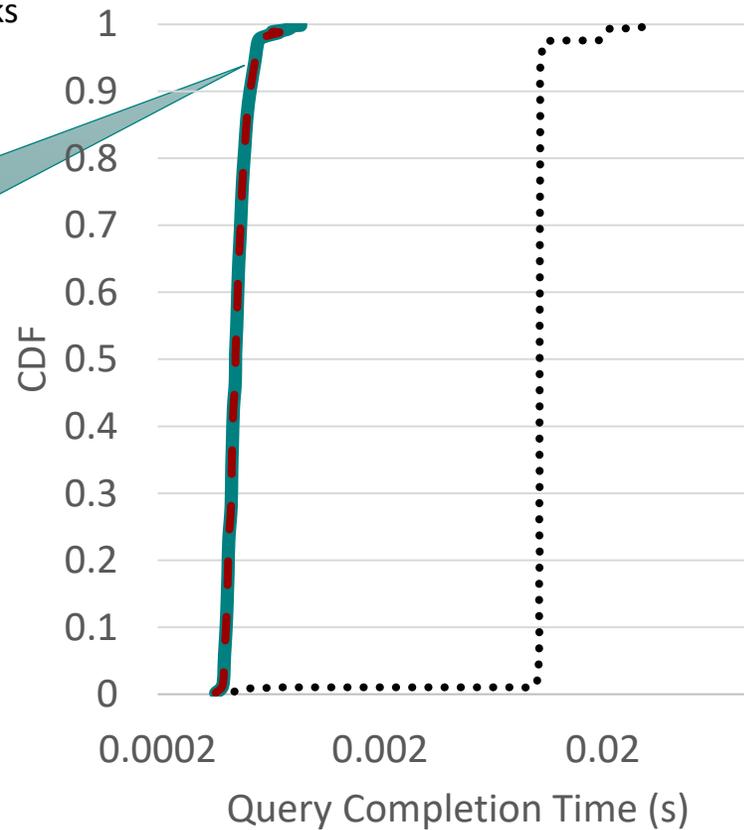


Implementable Deflection Under Large-scale Incast

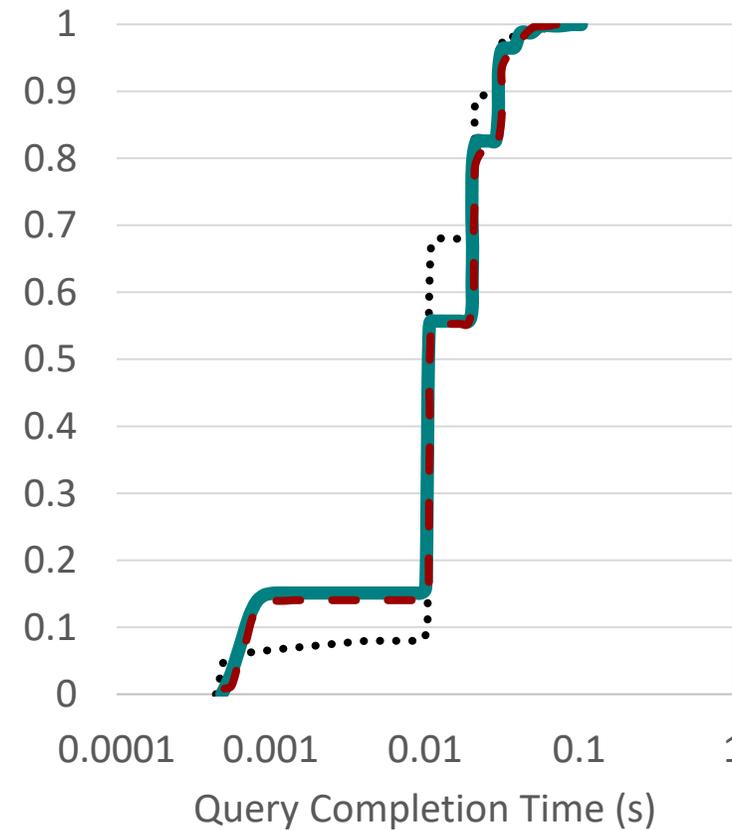
- Large-scale OMNET simulations
- 2-tier leaf-spine with 100 Gbps links
- 40 machines with all-to-all traffic
- Swift Congestion control

Simple Deflection is effective under moderate congestion!

50% Background + 5% Incast



50% Background + 35% Incast



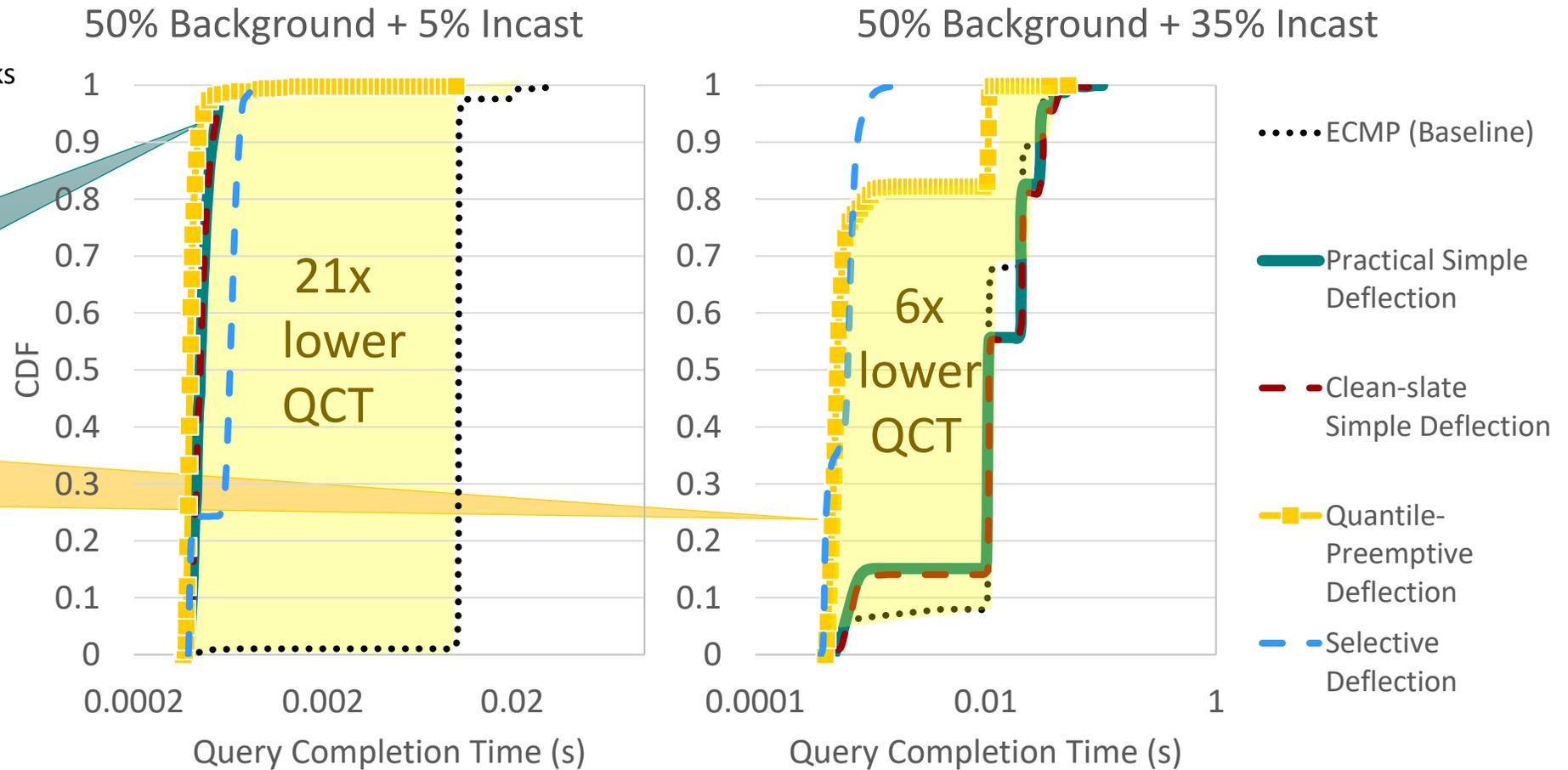
- ECMP (Baseline)
- Practical Simple Deflection
- - Clean-slate Simple Deflection

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

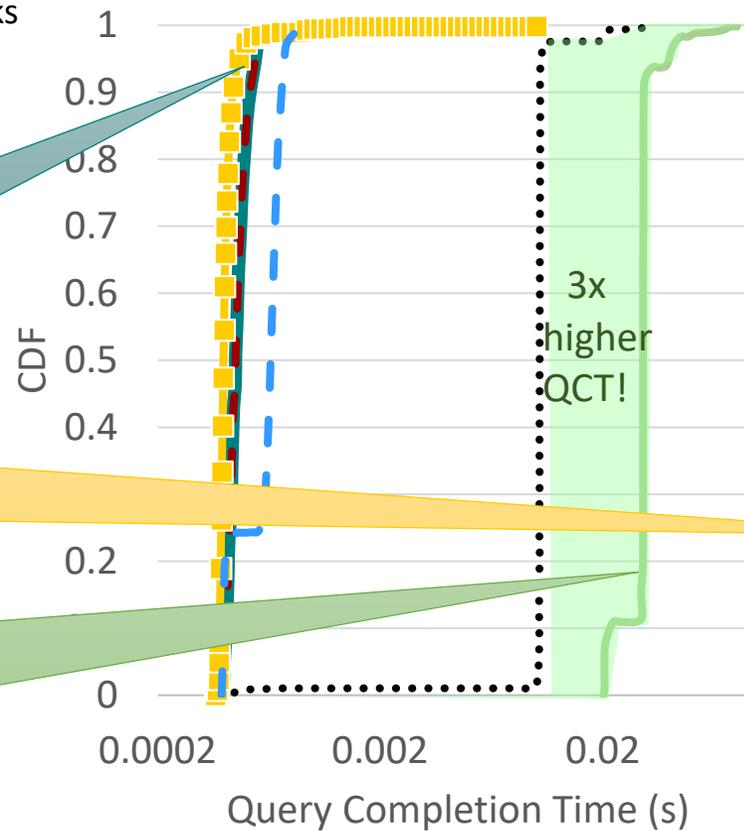
- Large-scale OMNET simulations
- 2-tier leaf-spine with 100 Gbps links
- 40 machines with all-to-all traffic
- Swift Congestion control

Simple Deflection is effective under moderate congestion!

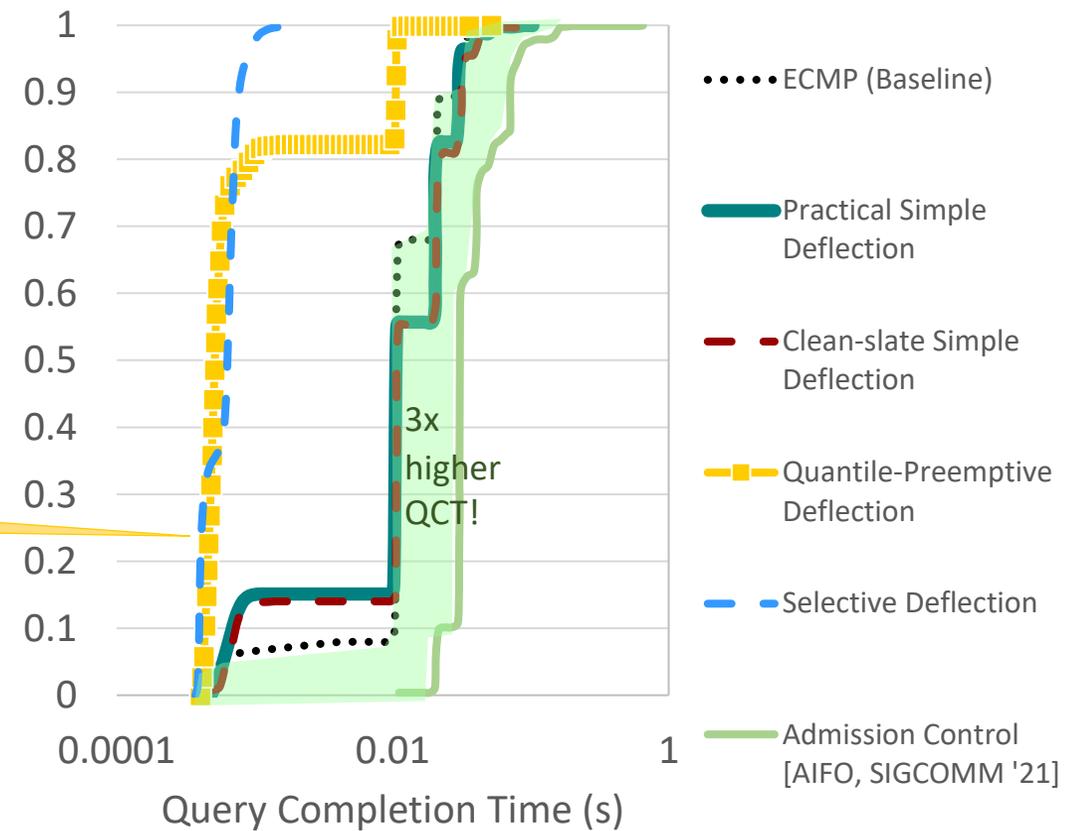
Preemptive Deflection offers superior performance under large Incast

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

50% Background + 5% Incast



50% Background + 35% Incast



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