Practical Packet Deflection in Datacenters

Sepehr Abdous, Erfan Sharafzadeh, Soudeh Ghorbani

December 2023
Datacenter Traffic is Busty!

1. Bursts are manifested at small timescales!

2. Bursts occur for different workloads

[Zhang et al., IMC ’17]
Existing Techniques Fall Short Against Bursts
Existing Techniques Fall Short Against Bursts

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

![TCP Sawtooth](image)

- ~ few milliseconds
- >> 50 µs

[CDF](#) Burst Duration (µs)

[Data Mining](#) [Web](#) [Cache](#)

[Zhang et al., IMC '17]
Existing Techniques Fall Short Against Bursts

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

[Singh et al., SIGCOMM '12]
Existing Techniques Fall Short Against Bursts

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

Filtering the potential destination ports requires many comparisons

Only deflect to free buffers!
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

Pop a low-priority packet from the buffer for deflection!
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

Simple

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

Selective

Sort packets in the queue and deflect lower-priority packets!
What Makes **Simple Deflection** Hard to Implement

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

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

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

**Priority Spectrum**

- **Low priority:** Longest remaining time
- **High priority:** Shortest remaining time

**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?

**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?

---

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

**CHALLENGE 2:** **Packet extraction** is not feasible after inserting the packet into a queue
Implementable Selective Deflection in PISA

Can we use existing traffic management capabilities to approximate **Selective 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** =

\[
\tau \times \left[ 1 - \right] \times \left[ \right]
\]
Implementable Selective Deflection in PISA

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

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

Compare its priority to a representation of existing packets

Yes, with Preemptive Deflection!

Admission policy:

\[
\tau = 8 \times \left[1 - 0.5 \times \frac{1}{6}\right] = 7.3 \\
\tau = 8 \times \left[1 - 0.5 \times \frac{5}{6}\right] = 4.6
\]

Queue Size: 4 5 2 3 2 1

Sensitivity Parameter: 

Yes, with Preemptive 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

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

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

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

Yes, with Preemptive 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

![Diagram showing network setup with high-priority and low-priority flows]

- **Large buffer does not help the key-value workload!**
- **Simple Deflection cuts the mean response times by 10x**
- **Preemptive Deflection offers 425x improvement in response time tails!**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Mean Response Time (ms)</th>
<th>Tail (p99) Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Case</td>
<td>~500 us</td>
<td></td>
</tr>
<tr>
<td>Droptail Queue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Droptail Queue (Large Buffer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Deflection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantile-PD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribution-PD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Physically Testbed Setup**: R represents the router, and S1, S2, S3 represent switches and servers.
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
```

**CDF**

Query Completion Time (s)
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

CDF of Query Completion Time (s)

- 50% Background + 5% Incast
- 50% Background + 35% Incast

- ECMP (Baseline)
- Practical Simple Deflection
- Clean-slate Simple Deflection
- Quantile-Preemptive Deflection
- Selective Deflection

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

**ECMP (Baseline)**

**Practical Simple Deflection**

**Clean-slate Simple Deflection**

**Quantile-Preemptive Deflection**

**Selective Deflection**

**Admission Control** [AIFO, SIGCOMM '21]

---

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

---

Practical Packet Deflection in Datacenters, CoNEXT 2023
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
Backup slides