Backdraft: a Lossless Virtual Switch that Prevents the Slow Receiver Problem

Alireza Sanaee
Farbod Shahinfar
Gianni Antichi
Brent E. Stephens

Queen Mary University of London
Sharif University of Technology
Queen Mary University of London
The University of Utah
Packet loss is a problem

Packet loss impacts tail latency!

Packet loss degrades throughput!

Packet loss wastes compute!
Packet loss can occur in the network
Packet loss can **ALSO** occur at the end-hosts

100 Gbps Network

Diagram showing a network setup with servers and networking components.
Vagaries of CPU performance

Unpredictable

End-host

Vswitch

Data center network
The slow receiver problem

Slow receivers are applications unable to keep up with the offered load

RDMA over Commodity Ethernet at Scale

Understanding Host Network Stack Overheads

Qizhe Cai
Cornell University

Shubham Chaudhary
Cornell University

Midhul Vuppalapati
Cornell University

Jaehyun Hwang
Cornell University

Rachit Agarwal
Cornell University

Switching Overheads 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
Backdraft is a lossless virtual switch that solves the slow receiver problem.
Lossless virtual switching is challenging

- Head-of-line Blocking
- Congestion Spreading
- Wasted Compute
Backdraft provides per flow queuing in the virtual switches

Backdraft implements Backpressure all the way to the application

Backdraft allows for higher throughput and lower tail latency
Outline

Motivation

Backdraft design

Backdraft evaluation
Insights of Backdraft

Slow receivers are pervasive!
Memcached needs 32 cores to achieve 100 Gbps with large values
Slow receivers are pervasive

Nginx needs 14 cores to achieve 100 Gbps
Slow receivers are pervasive

**IPerf3** only does networking functionality yet needs at least 6 cores to achieve 100 Gbps
Slow receivers are pervasive!

The slow receiver problem manifests at µs-scale.
Slow receivers manifest at µs-scale

40 Gbps throughput variation in 100us!
Slow receivers manifest at µs-scale

40 Gbps throughput variation in 100us!
Slow receivers are pervasive!

The slow receiver problem manifests at µs-scale

Packet loss occurs in presence of Homa
Homa experiences high RPC completion time due to the slow receiver problem.
Insights of Backdraft

Slow receivers are pervasive!

The slow receiver problem manifests at µs-scale.

Packet loss occurs in presence of Homa

Standard lossless techniques cannot be used in a virtual context
Standard lossless techniques are not practical in virtual switching realm

<table>
<thead>
<tr>
<th></th>
<th>Rate limiting</th>
<th>Backpressure</th>
<th>Credit-based</th>
<th>PicNIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown line rate</td>
<td>HOL blocking</td>
<td>Extra RTT</td>
<td>Wasted CPU</td>
<td></td>
</tr>
<tr>
<td>on CPU</td>
<td>Congestion Spreading</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

21
Slow receivers are pervasive!

The slow receiver problem manifests at short time scales.

Packet loss occurs in presence of the state-of-the art congestion controls – Homa

Standard lossless techniques cannot be used in a virtual context
Outline

Motivation

Backdraft design

Backdraft evaluation
Three components of Backdraft

1) Dynamic per flow queueing
   - Avoids HOL blocking
   - On-demand memory use

2) Doorbell queues
   - Avoids wasted CPU

3) Backdraft overlay network
   - Avoids packet loss
   - Prevents congestion spreading
Key idea behind the per flow queuing

If each flow has its own separate queue, then each flow can be paused and resumed individually without the HOL blocking problem!
Why do we lose packets?
Why do we lose packets?
Why do we lose packets?

Blue packets are gone!
Would pausing the upstream work?
Pause the upstream!
Per flow queuing to the rescue!
Per flow queuing to the rescue!
Per flow queuing to the rescue!
Design space of the Per Flow Queuing

**A large number short queues**

- **Memory is limited**
- **High flow isolation**
- **Low burst absorbance**

**A small number long queues**

- **Low flow isolation**
- **High burst absorbance**
Dynamic Per Flow Queuing

- On-demand queue allocation
- On-demand queue resizing

Wait, Whaaaat? DPFQ?????
Dynamic Per Flow Queuing

On-demand queue allocation

On-demand queue resizing
Dynamic Per Flow Queuing

On-demand queue allocation

On-demand queue resizing

Shared Memory Region

Dynamic Queue Management

Flow 1
Dynamic Per Flow Queuing

- On-demand queue allocation
- On-demand queue resizing

Shared Memory Region

Flow 1

Queue resizing

Dynamic Queue Management
Dynamic Per Flow Queuing

On-demand queue allocation

On-demand queue resizing
Dynamic Per Flow Queuing

On-demand queue allocation

On-demand queue resizing

Shared Memory Region

Dynamic Queue Management
Dynamic Per Flow Queuing

- On-demand queue allocation
- On-demand queue resizing

Shared Memory Region

Dynamic Queue Management

Queue reclamation

Flow 1

Flow 2

On-demand queue allocation
On-demand queue resizing
Three components of Backdraft

1) Dynamic per flow queueing
   - Avoids HOL blocking
   - On-demand memory use

2) Doorbell queues
   - Avoids wasted CPU

3) Backdraft overlay network
   - Avoids packet loss
   - Prevents congestion spreading
Doorbell Queues

Per flow queuing is **NOT scalable** and wastes CPU cycles

[Berkeley Extensible Software Switch (BESS): Han, et. al. NSDI ’14]
Key idea behind doorbell queues

Backdraft only busy polls one queue per CPU core regardless of the number of created queues!
How does doorbell queues work?

End-host

Vswitch

App RX

NIC RX

NIC TX

Data center network
How does doorbell queues work?
How does doorbell queues work?
How does doorbell queues work?

Data center network

End-host

Vswitch

App RX

NIC RX

NIC TX
How does doorbell queues work?
How does doorbell queues work?

End-host

App RX

Vswitch

NIC RX

NIC TX

Data center network
How does doorbell queues work?

End-host

App RX

DPFQ reclaims the empty queue

Vswitch

NIC RX NIC TX

Data center network
How does doorbell queues work?

- **End-host**
  - App RX

- **Vswitch**
  - NIC RX
  - NIC TX

- **Data center network**
Three components of Backdraft

1) Dynamic per flow queueing
   - Avoids HOL blocking
   - On-demand memory use

2) Doorbell queues
   - Avoids wasted CPU

3) Backdraft overlay network
   - Avoids packet loss
   - Prevents congestion spreading
Backdraft overlay network

Key idea

Utilizing the large buffering capacity at the end-hosts

Approach

Buffering packet at the end-hosts

Backdraft sends overlay PAUSE messages to the upstream virtual switch.

Upon receive of an overlay message, the vswitch stops reading the culprit flow messages.

Backdraft overlay network can solve congestion spreading problem due to slow receivers where even BFC cannot solve.
Outline

Motivation

Backdraft design

Backdraft evaluation
Backdraft implementation

Backdraft is built on top of BESS.

Backdraft uses TCP acceleration service (TAS) as a user level TCP library.
[TAS: Kauffmann, et. al. EuroSys ’19]

Backdraft is about ~4K LOC.
Two questions we address in this talk

Can Backdraft solve problems with existing congestion control protocols that still are not solved?

How does Backdraft impact the real workload application performance?
Two questions we address in this talk

Can Backdraft solve problems with existing congestion control protocols that still are not solved?
Homa experiment setup

![Diagram showing Homa DPDK Server and Client connected through BESS and Network]

- Homa DPDK Server
- Homa DPDK Client
- BESS
- Network
Homa experiment setup
Backdraft complements Homa – RPC Completion time

Homa (small RPC) achieves higher throughput when it runs on top of Backdraft.

(a) 200B message size

- BESS
- BD
Backdraft complements Homa – RPC
Completion time

Homa (large RPC) achieves higher throughput when it runs on top of Backdraft.
Can Backdraft solve problems with existing congestion control protocols that still are not solved?

How does Backdraft impact the real workload application performance?
Experiment setup

- **VSwitch**
  - F1-50
  - F51-60

- **UDP App**
  - F1-50

- **Mutilate**
  - F51-F60

- **UDP Receiver**
  - F1-50

- **Memcached**
  - F51-60

End-host
Evaluating different components of Backdraft - Request completion time

![Graph showing request completion time for Fast Receiver (0 cyc), Slow Receiver (0.5K cyc), and Slow Receiver (5K cyc). The graph indicates that the completion time increases with decreasing receiver speed, with additional cyclicity affecting the performance.]
Evaluating different components of Backdraft - Request completion time
Evaluating different components of Backdraft - Request completion time

![Graphs showing request completion time](image)
Evaluating different components of Backdraft - Request completion time
Evaluating different components of Backdraft - Request completion time
Evaluating different components of Backdraft - Goodput

- Fast Receiver (0 cyc) 3 Mpps
- Slow Receiver (0.5K cyc) 2 Mpps
- Slow Receiver (5K cyc) 0.5 Mpps

Legend: Lossy
Evaluating different components of Backdraft - Goodput

![Chart showing Goodput in Mpps for Fast Receiver (0 cyc), Slow Receiver (0.5K cyc), and Slow Receiver (5K cyc). The chart compares Lossy and BP scenarios.]
Evaluating different components of Backdraft - Goodput
Evaluating different components of Backdraft - Goodput
Evaluating different components of Backdraft - Goodput
Backdraft Takeaways

Slow receivers are pervasive

1. Dynamic per-flow queuing
2. Doorbell queues
3. Overlay network

Backdraft can achieve up to 20x better tail latency compared to lossy approach

We ❤️ open source

https://github.com/Lossless-Virtual-Switching/Backdraft