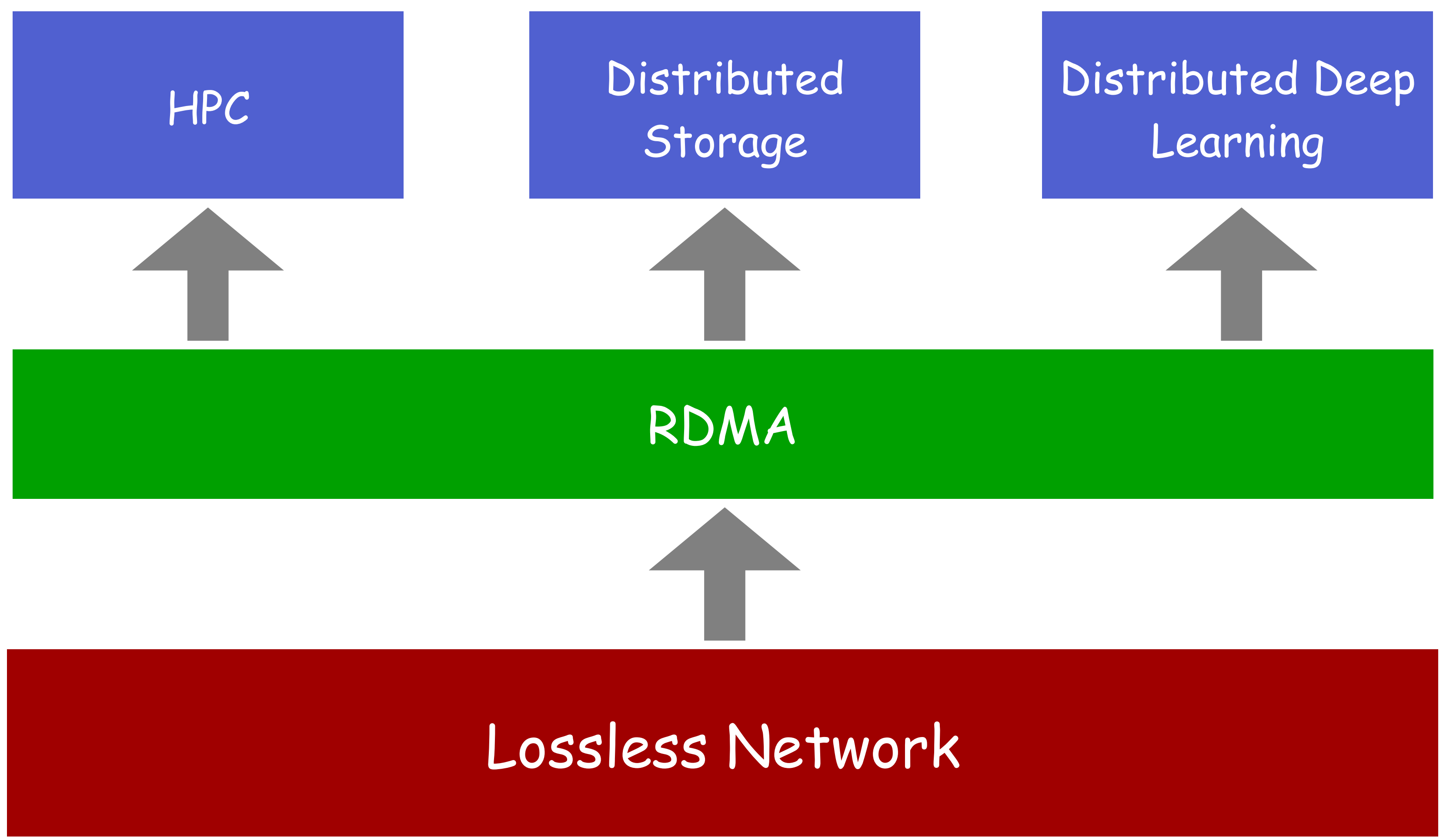
Less is More:
Dynamic and Shared Headroom Allocation in PFC-enabled Datacenter Networks

Danfeng Shan, Yuqi Liu, Tong Zhang, Yifan Liu,
Yazhe Tang, Hao Li, and Peng Zhang

西安交通大學 XI'AN JIAOTONG UNIVERSITY
南京航空航天大學 NUAA
清華大學 Tsinghua University

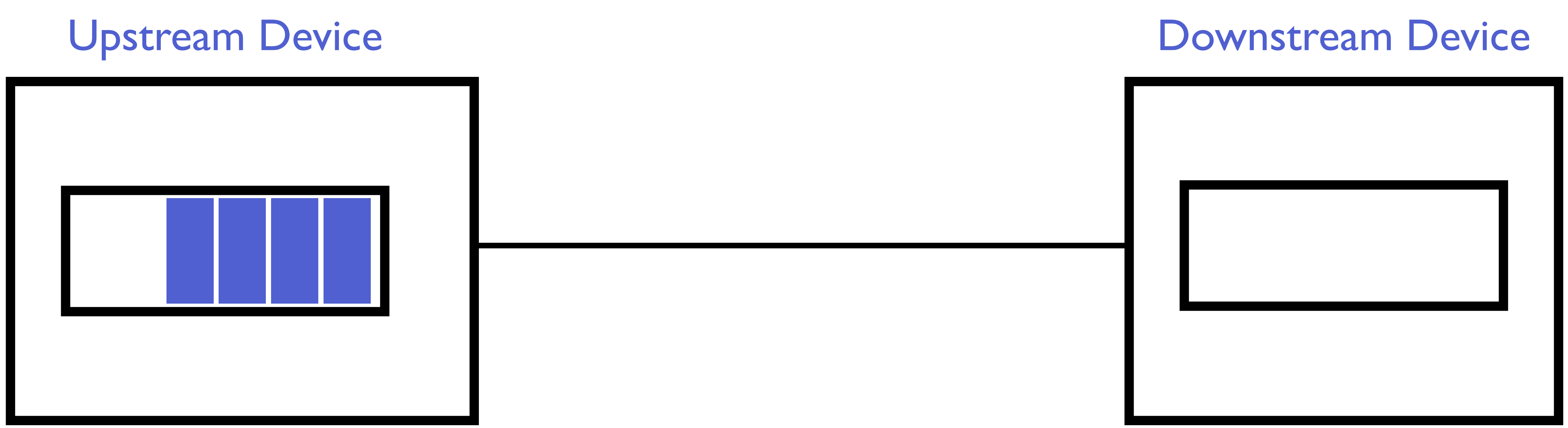
Background: Lossless Network

- Lossless network is very attractive in DCN
  - Ultra-low latency and high throughput



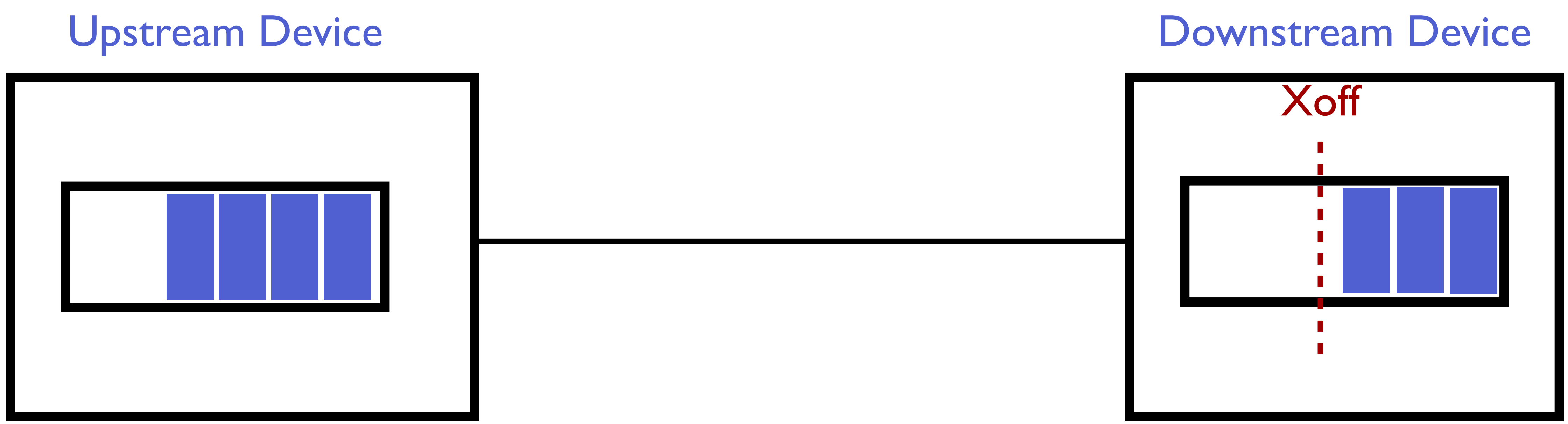
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow



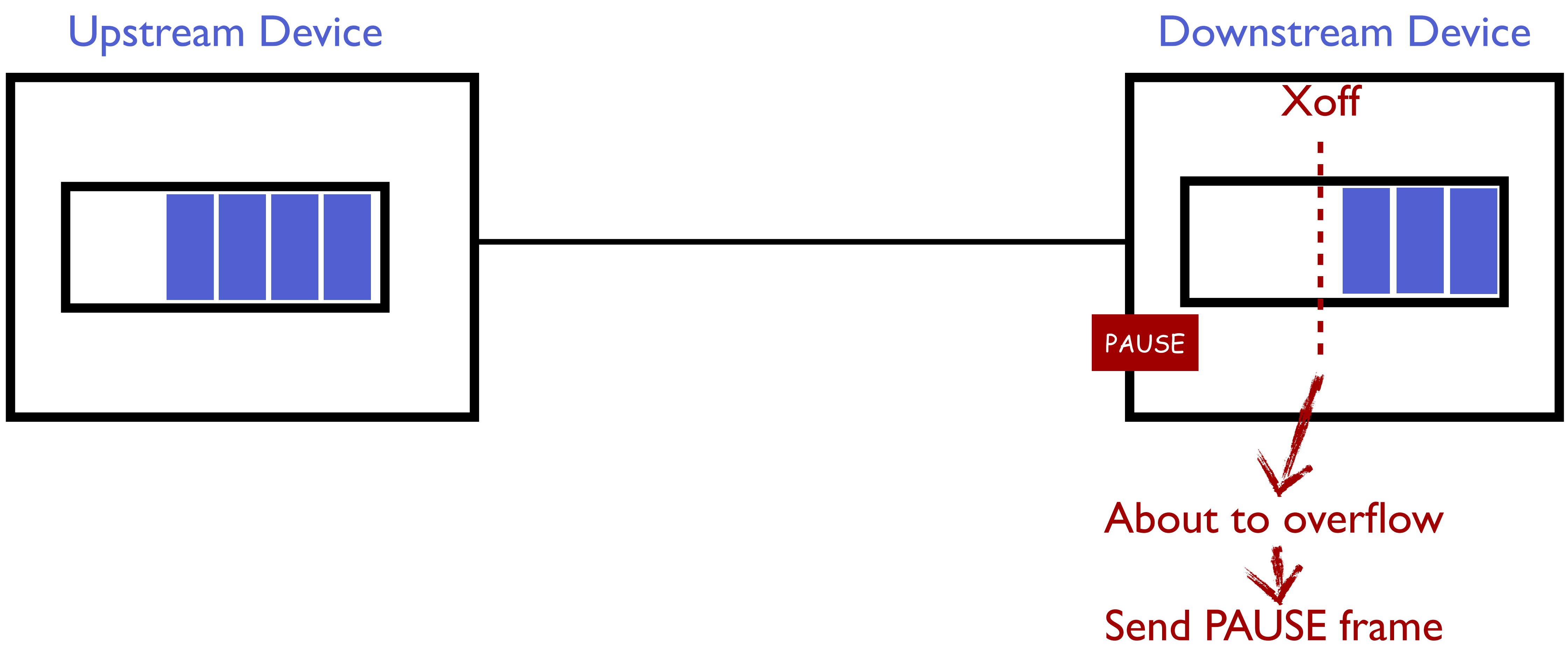
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow



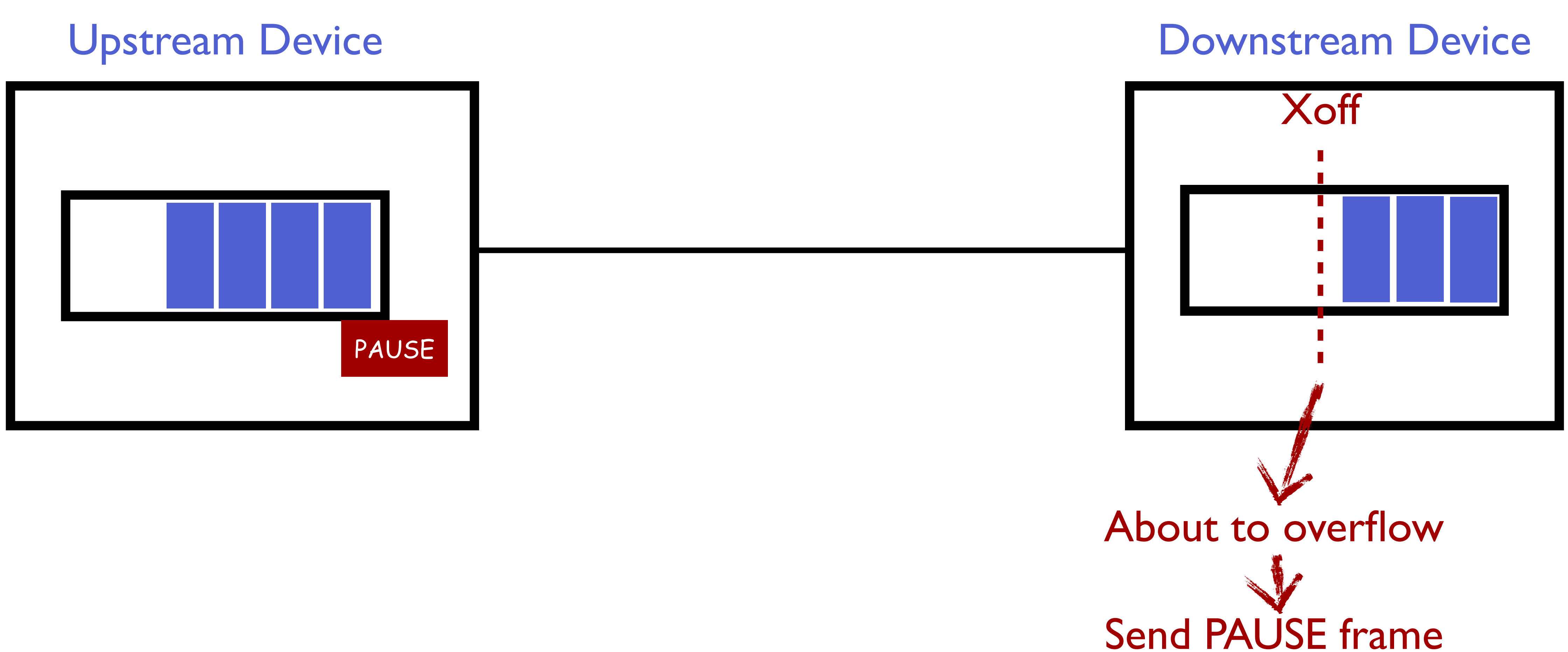
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow



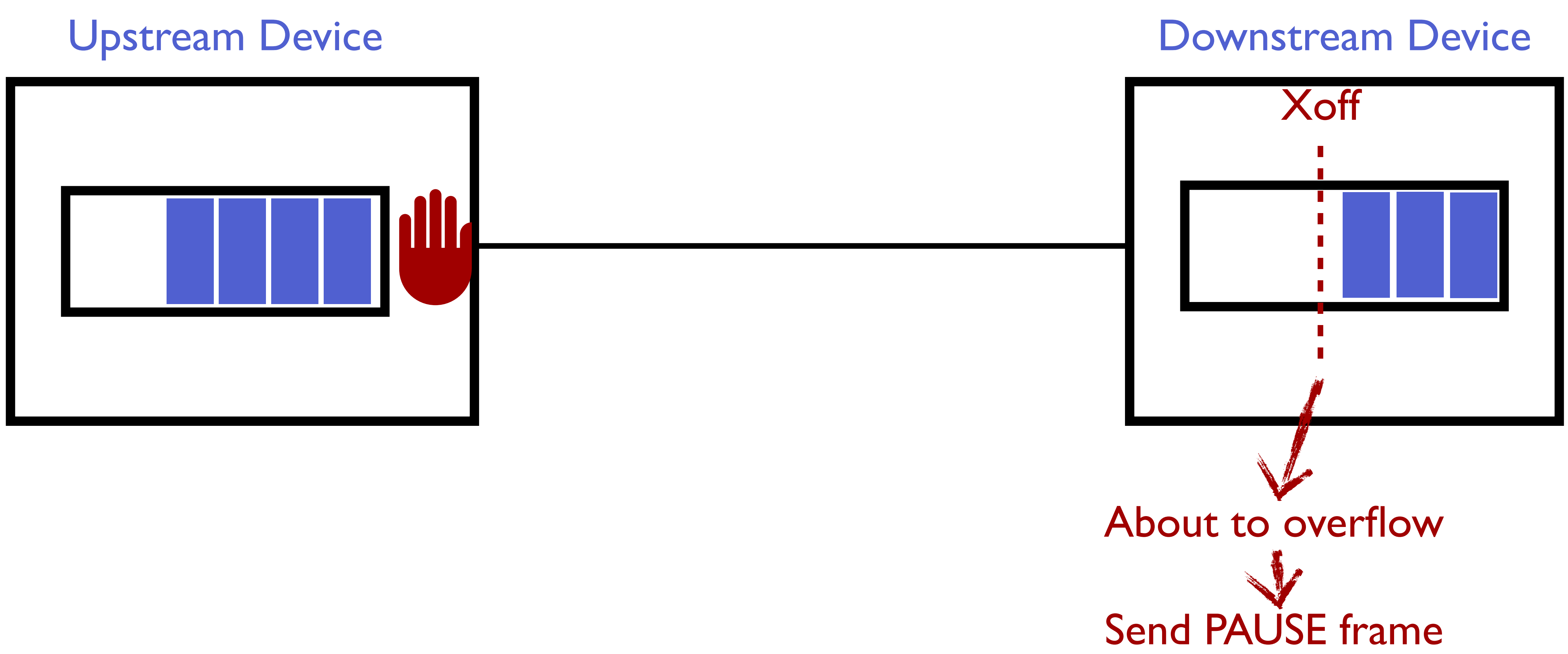
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow



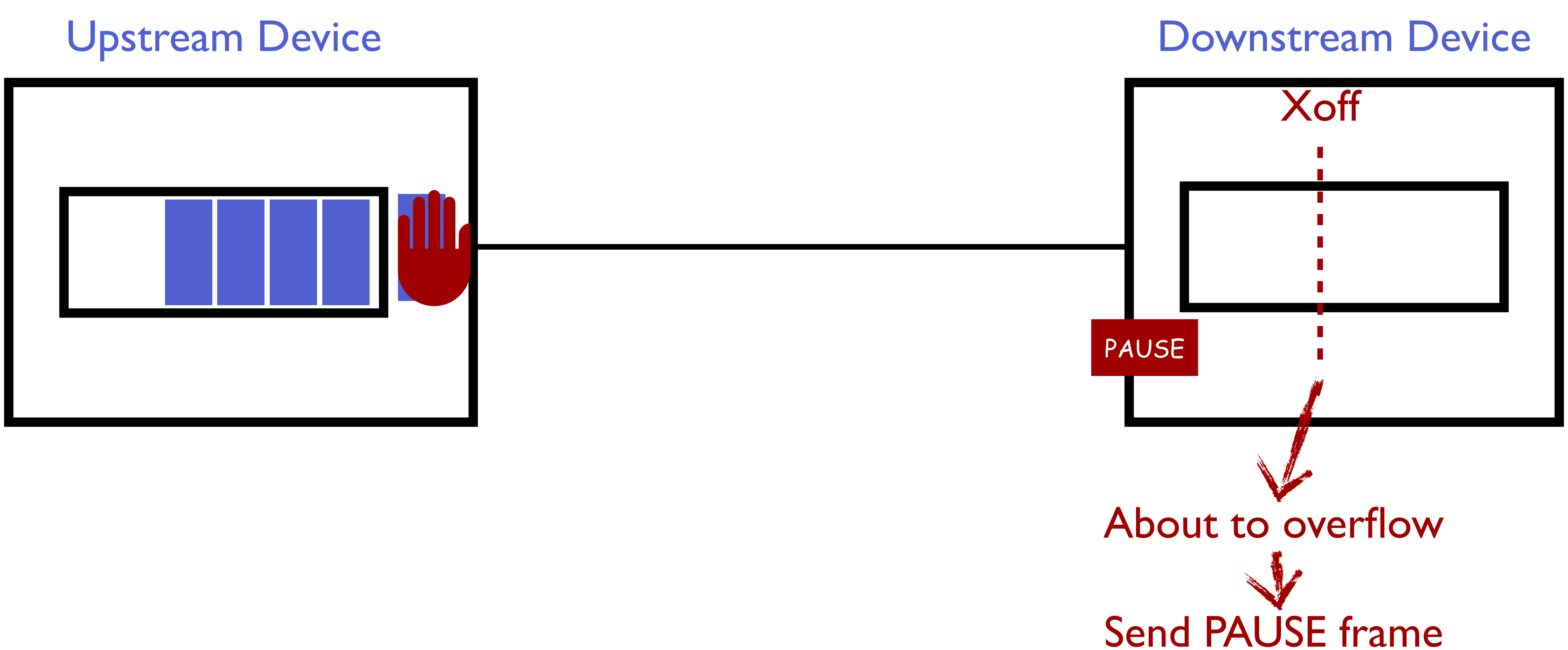
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow



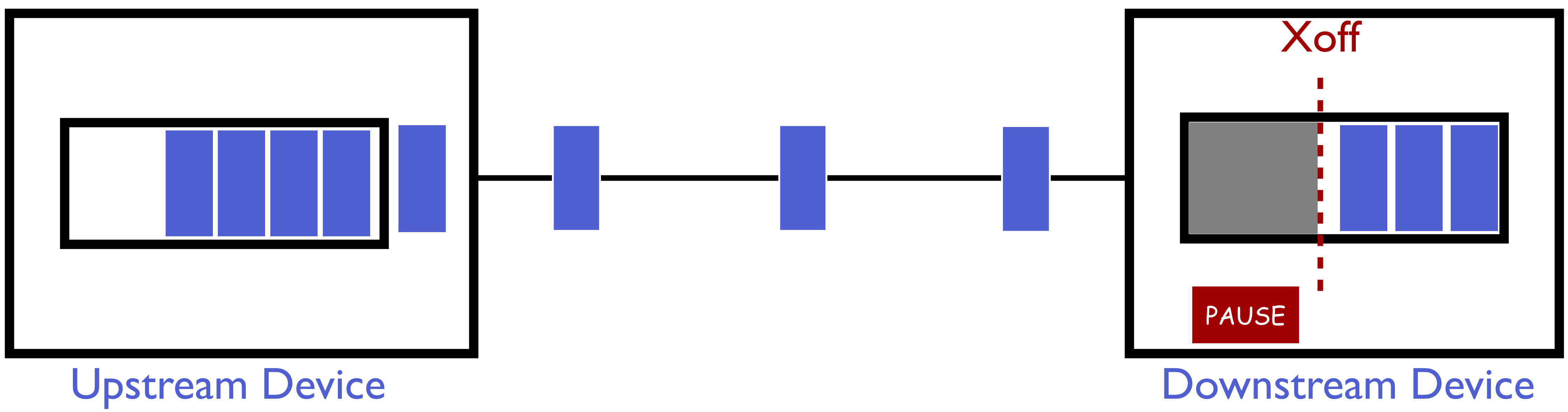
Background: PFC

- Ethernet: Priority-based Flow Control
  - Hop-by-hop flow control
  - Pause upstream devices when buffer is about to overflow
  - PFC messages are harmful
    - HoL blocking, congestion spreading, collateral damage, deadlock
    - We should avoid PFC messages as much as possible



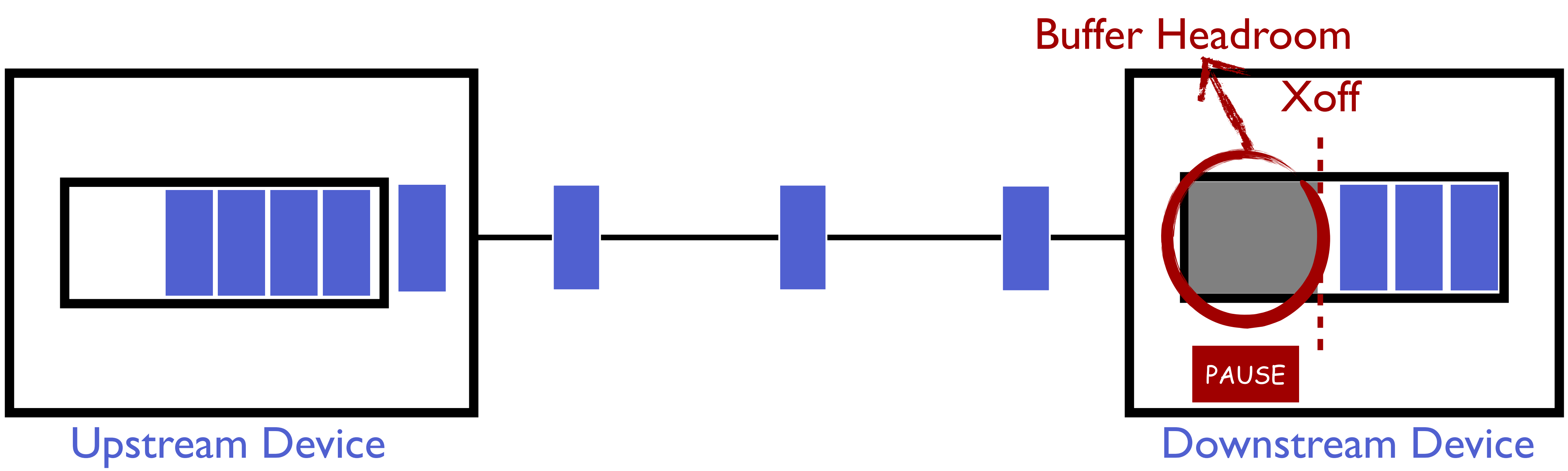
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving packets during this time



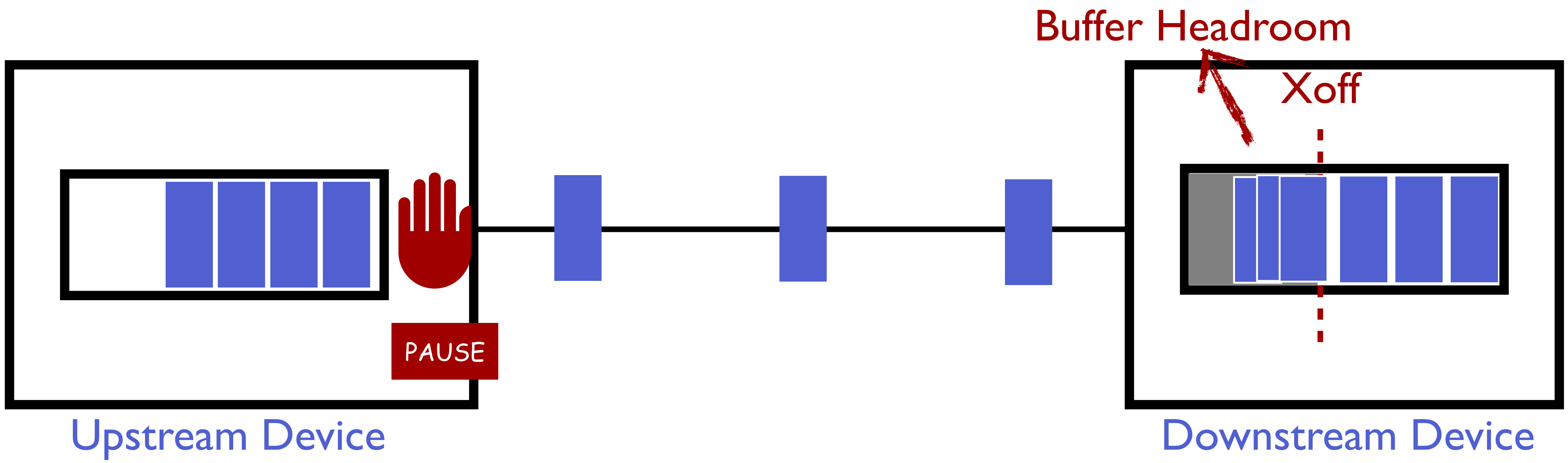
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving packets during this time



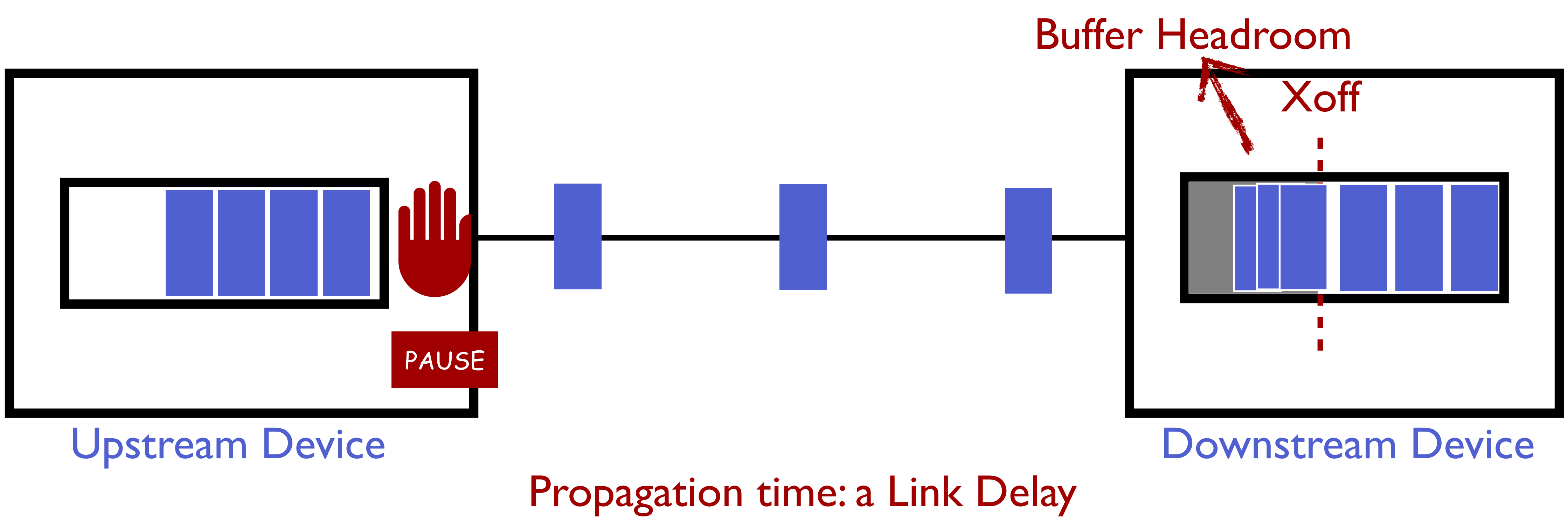
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving packets during this time



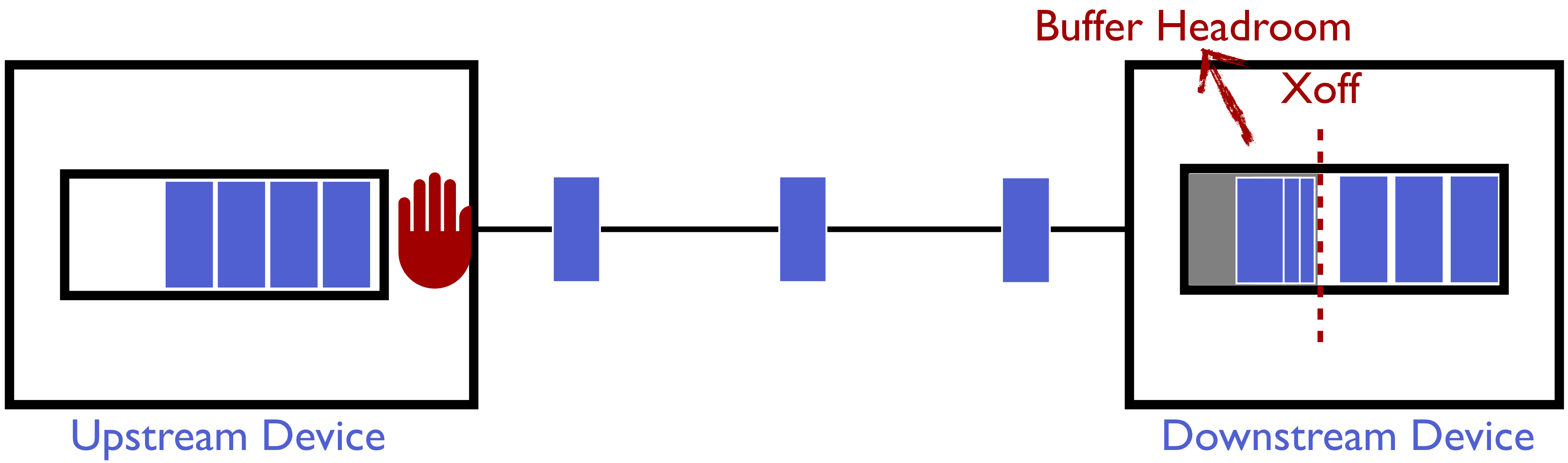
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving packets during this time



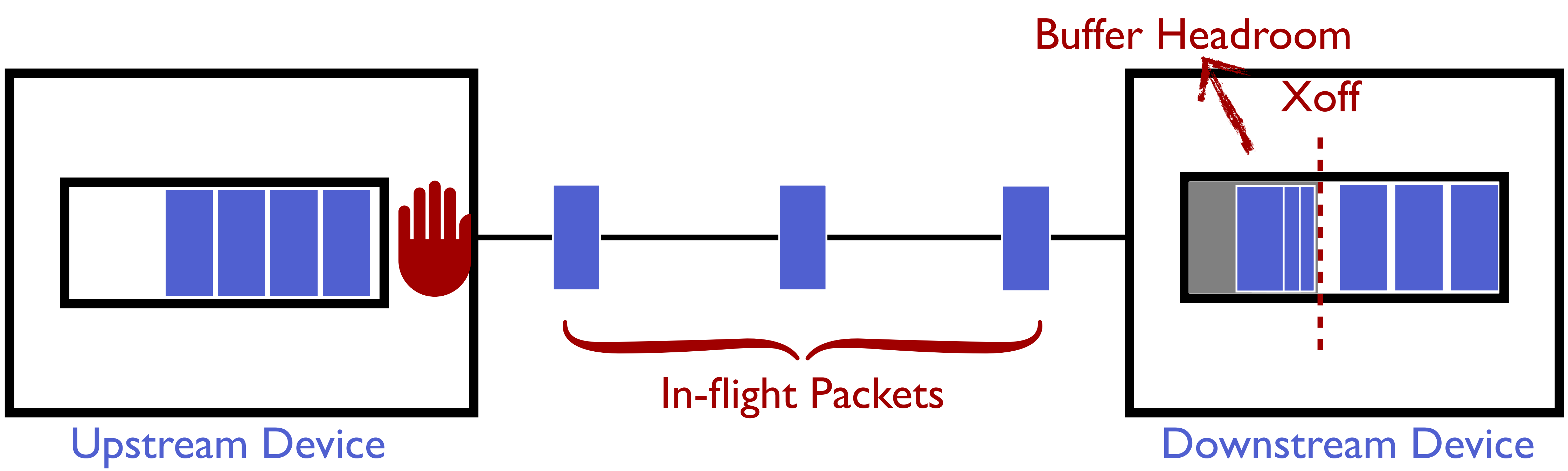
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving traffic during this time



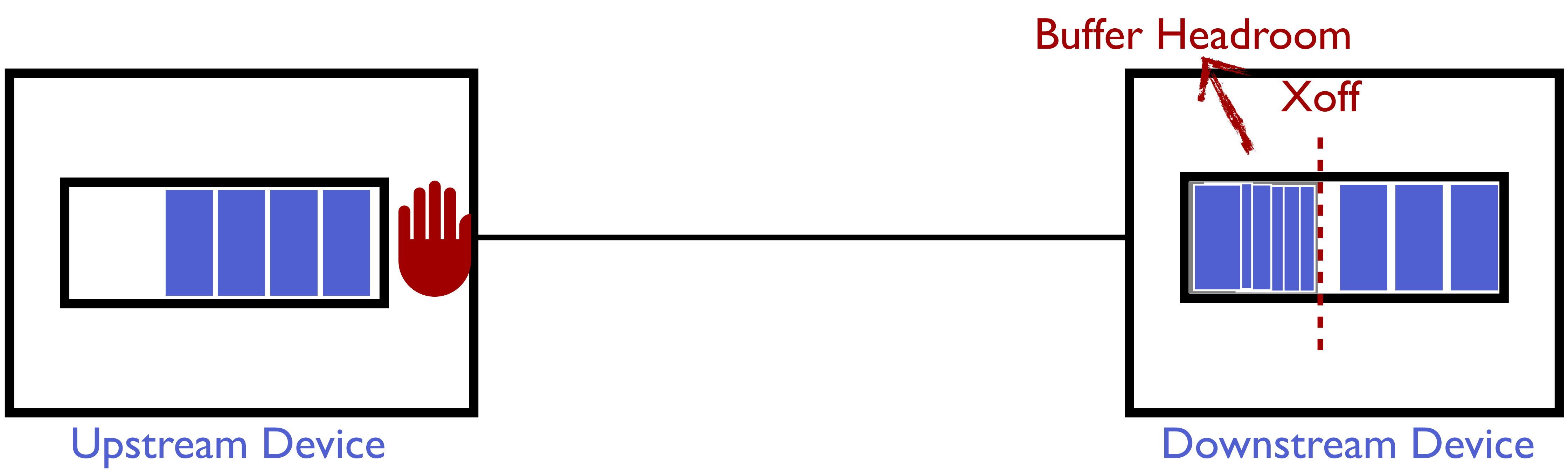
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving traffic during this time



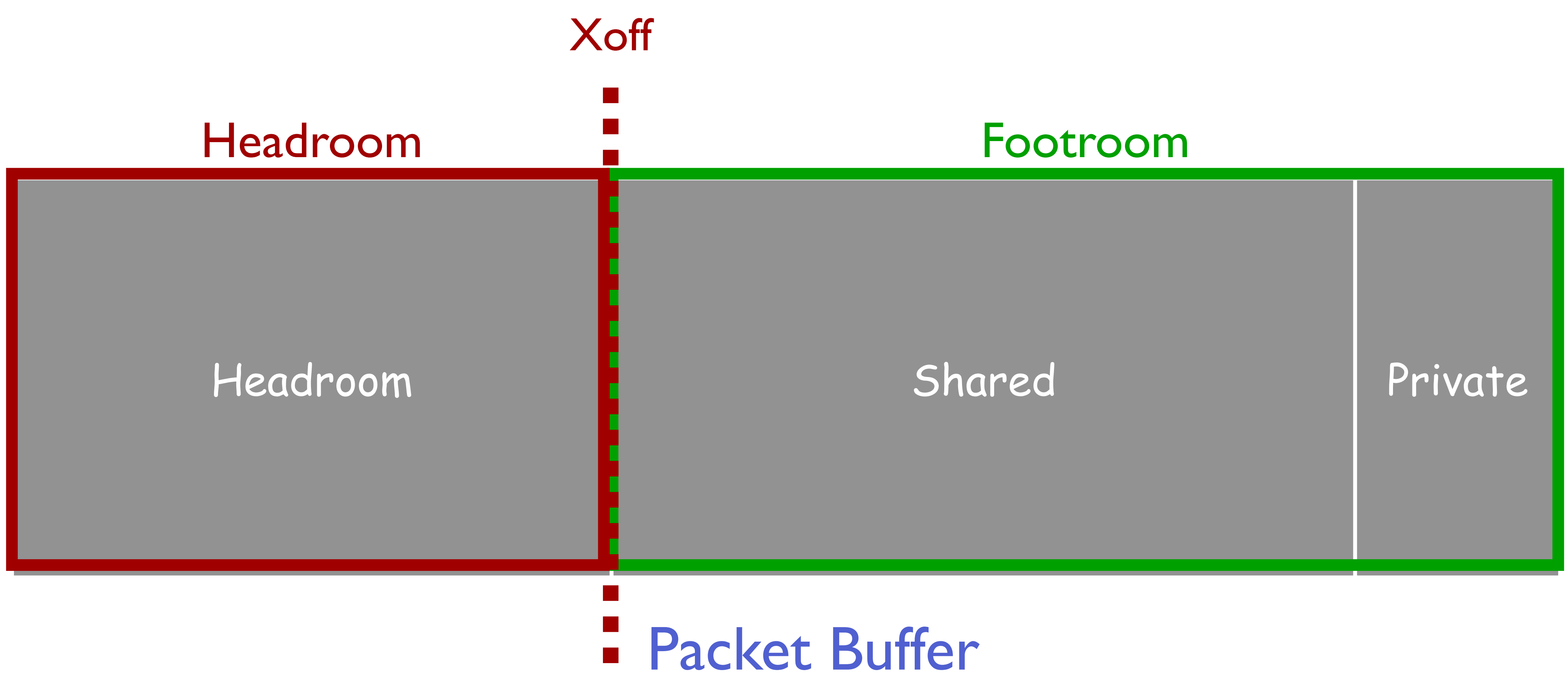
Background: Headroom

- Buffer Headroom
  - It takes time for the PAUSE frame to take effect
  - Buffer headroom: absorb arriving traffic during this time



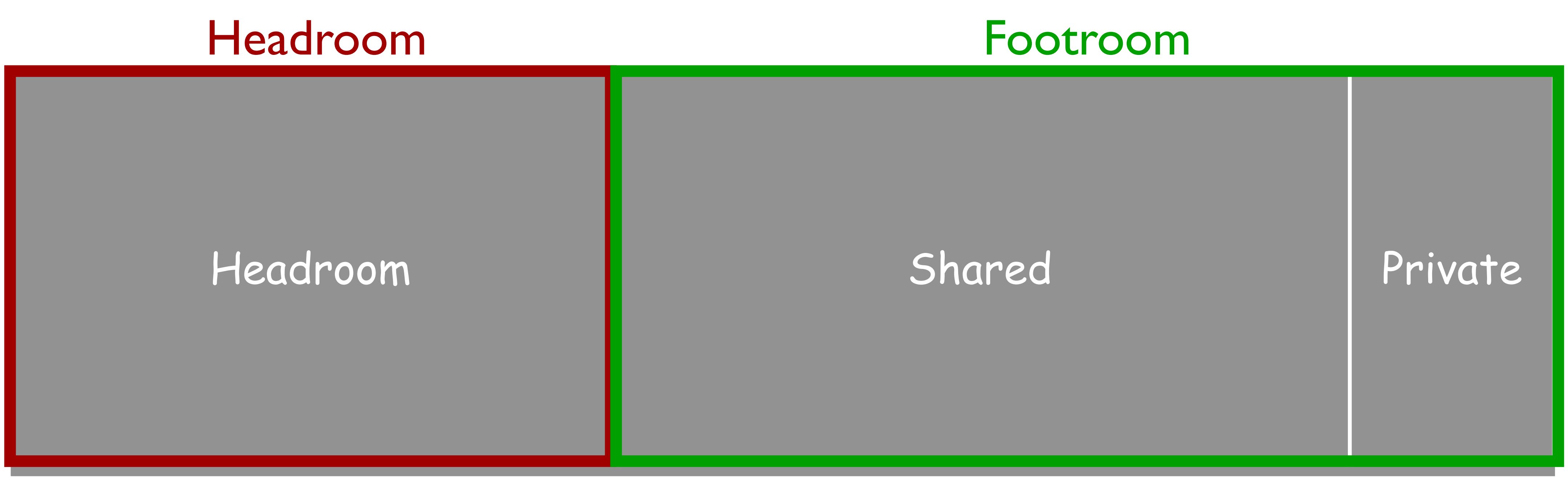
Background: Buffer Organization

- Buffer structure
  - Headroom buffer: absorb in-flight packets after sending PAUSE frame
  - Shared buffer: shared among all queues
  - Private buffer: dedicate buffer for each queue



Background: Buffer Organization

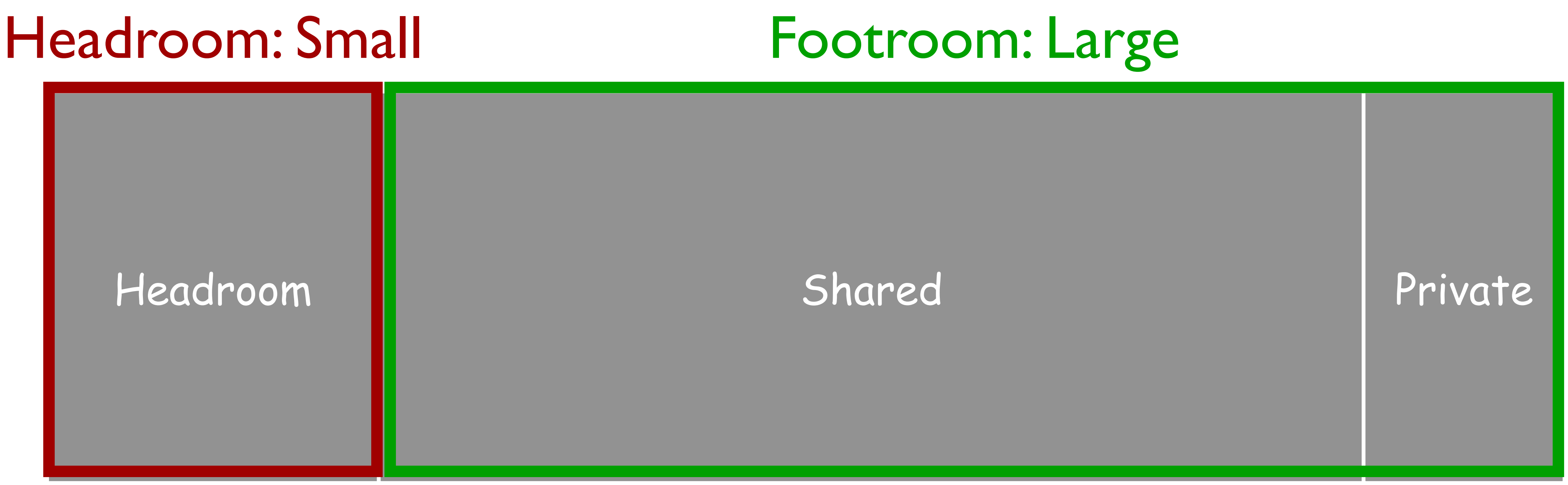
- Buffer allocation
  - Headroom buffer (for each ingress queue)
    - Link capacity * Delay for PAUSE to take effect
  - Shared buffer: dynamically allocated
  - Private buffer: statically configured



Packet Buffer

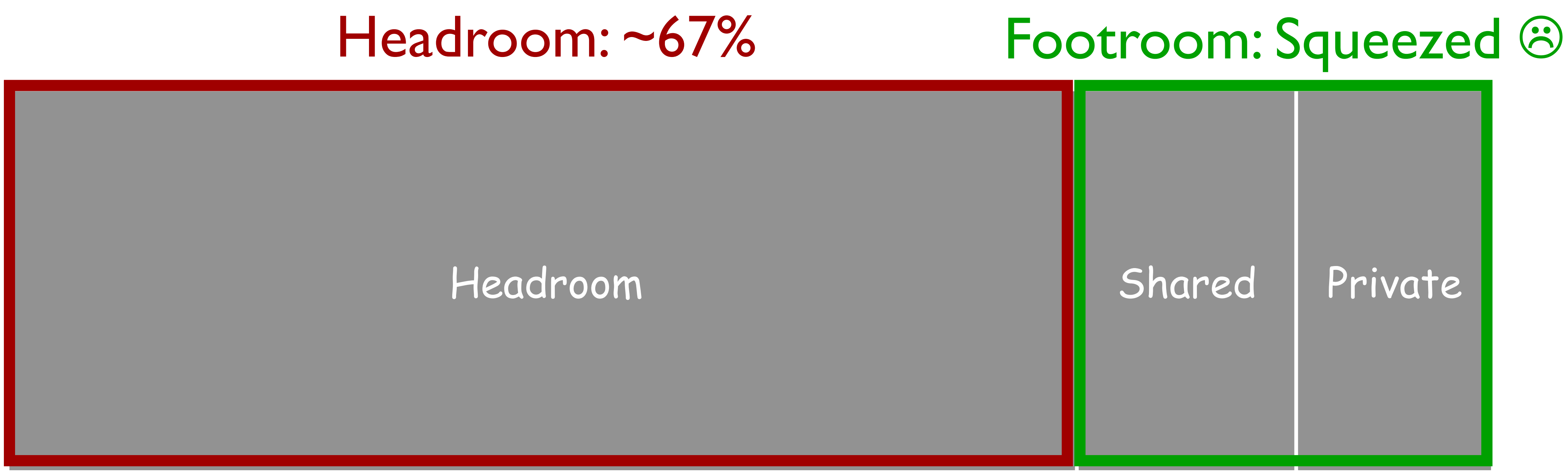
Motivation

- What we expect
  - Headroom
    - A small fraction
  - Footroom
    - Most buffer
    - Aborb burst without triggering PFC messages



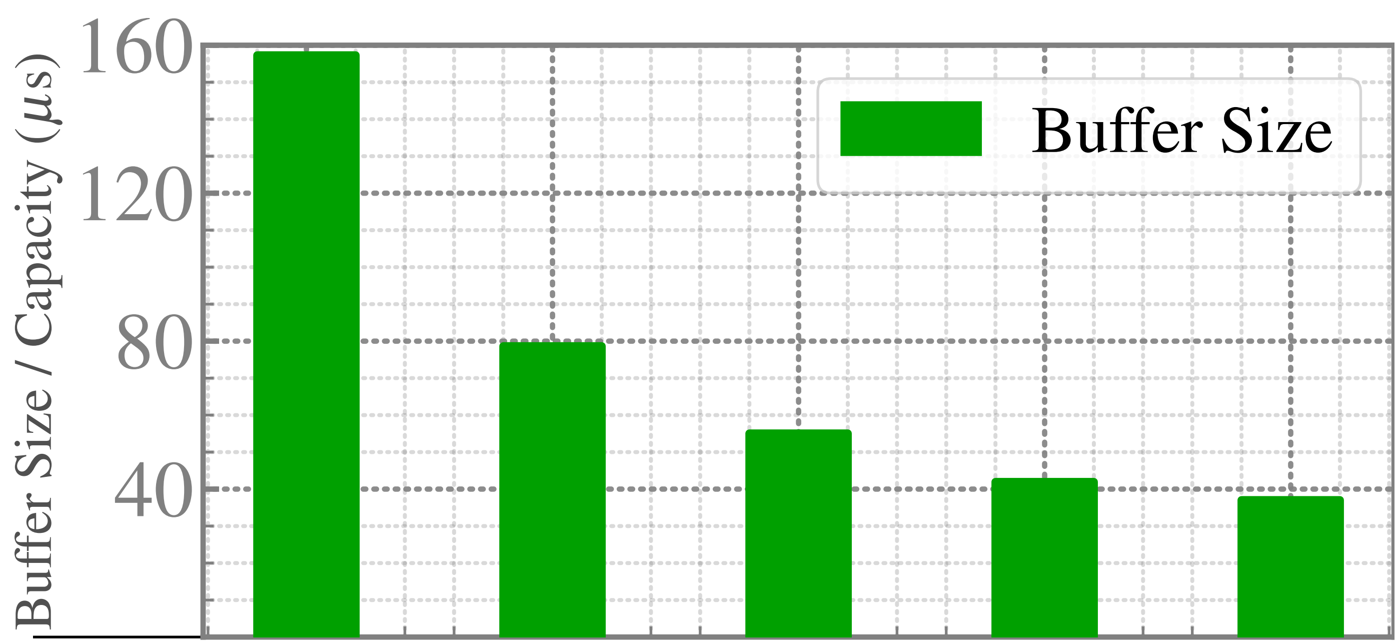
Motivation

- What the reality is
  - Headroom
    - As large as ~67%
  - Footroom
    - Only a small fraction
    - PFC messages can be frequently triggered



Motivation

- Why?
  - Reason 1: Buffer is increasingly insufficient
    - Buffer is integrated on the chip
    - Buffer size is limited by the chip area



Chip	Trident+	Trident2	Tomahawk2	Tomahawk3	Tomahawk4
Capacity	480Gbps	1.28Tbps	6.4Tbps	12.8Tbps	25.6Tbps
Year	2010	2012	2016	2017	2019

Buffer trends in Broadcom's switching chip:
Buffer size per unit of capacity has decreased by 4×

Motivation

- Why?
  - Reason 1: Buffer is increasingly insufficient
    - Buffer is integrated on the chip
    - Buffer size is limite

Inevitable ☹

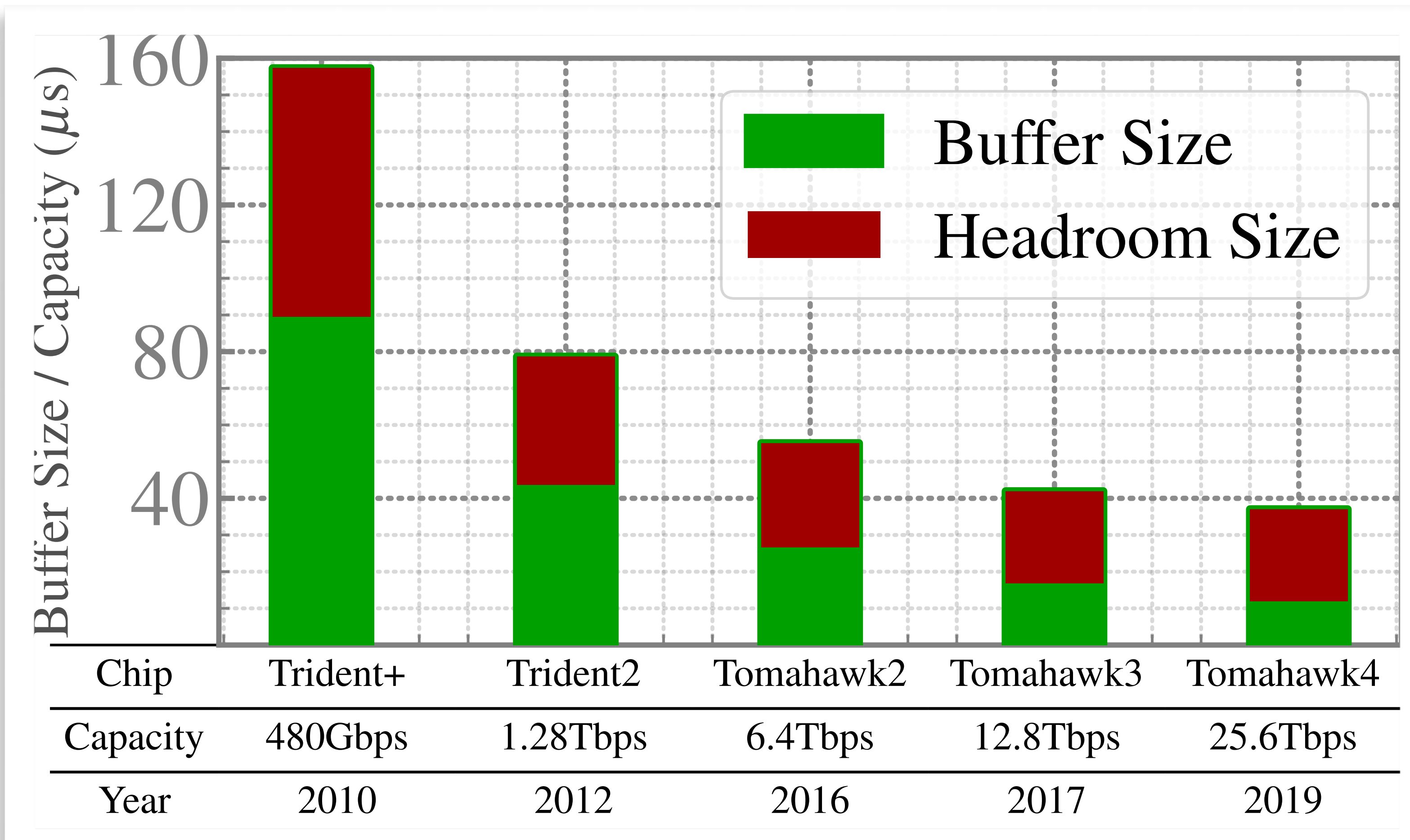
Buffer trends in Broadcom's switching chip:
Buffer size per unit of capacity has decreased by 4×

Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method (SIH): static and independent
      - Reserve a static fraction of buffer beforehand
        - Higher link capacity, larger headroom buffer
      - Independently reserve buffer for every ingress queue

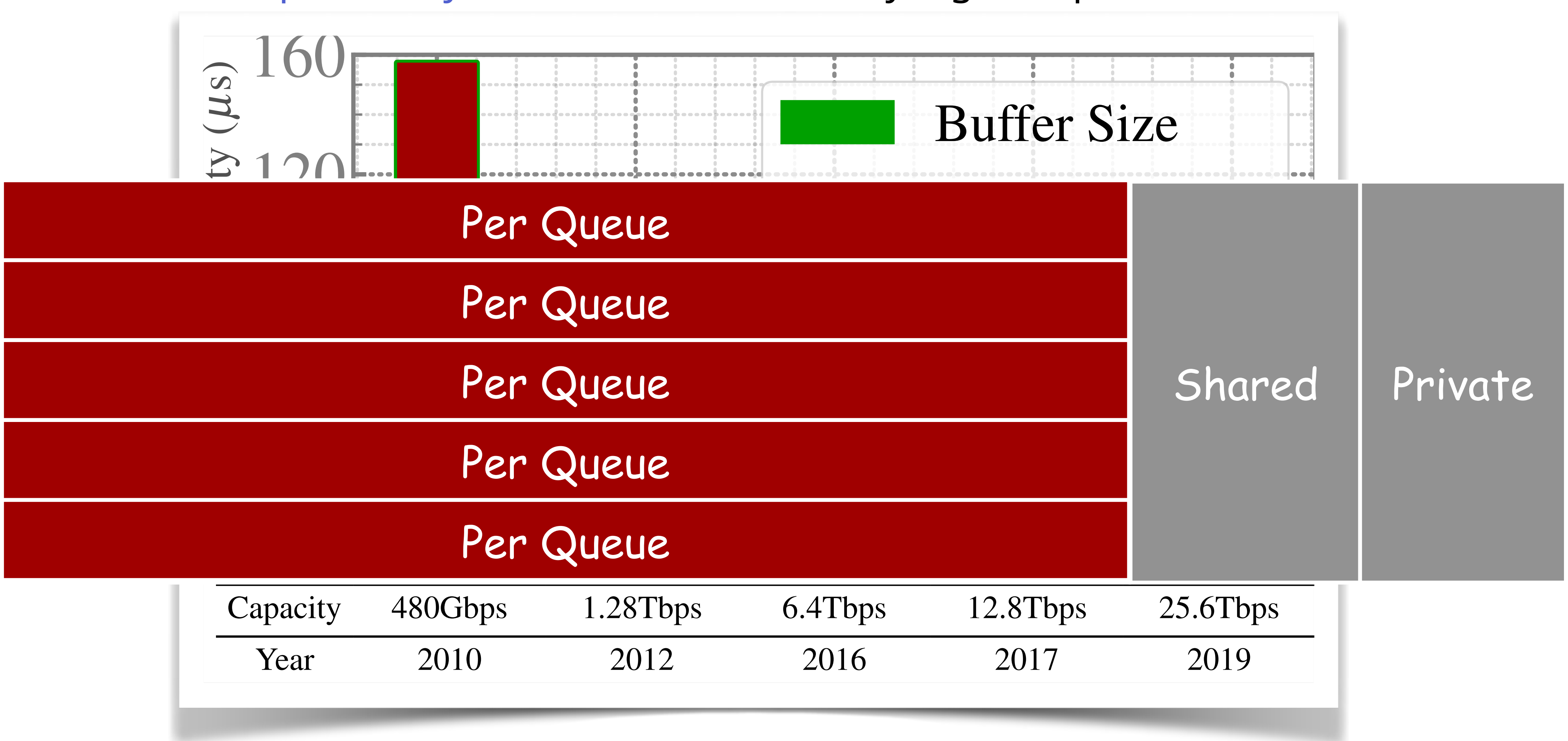
Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method (SIH): static and independent
      - Reserve a static fraction of buffer beforehand
        - Higher link capacity, larger headroom buffer
      - Independently reserve buffer for every ingress queue



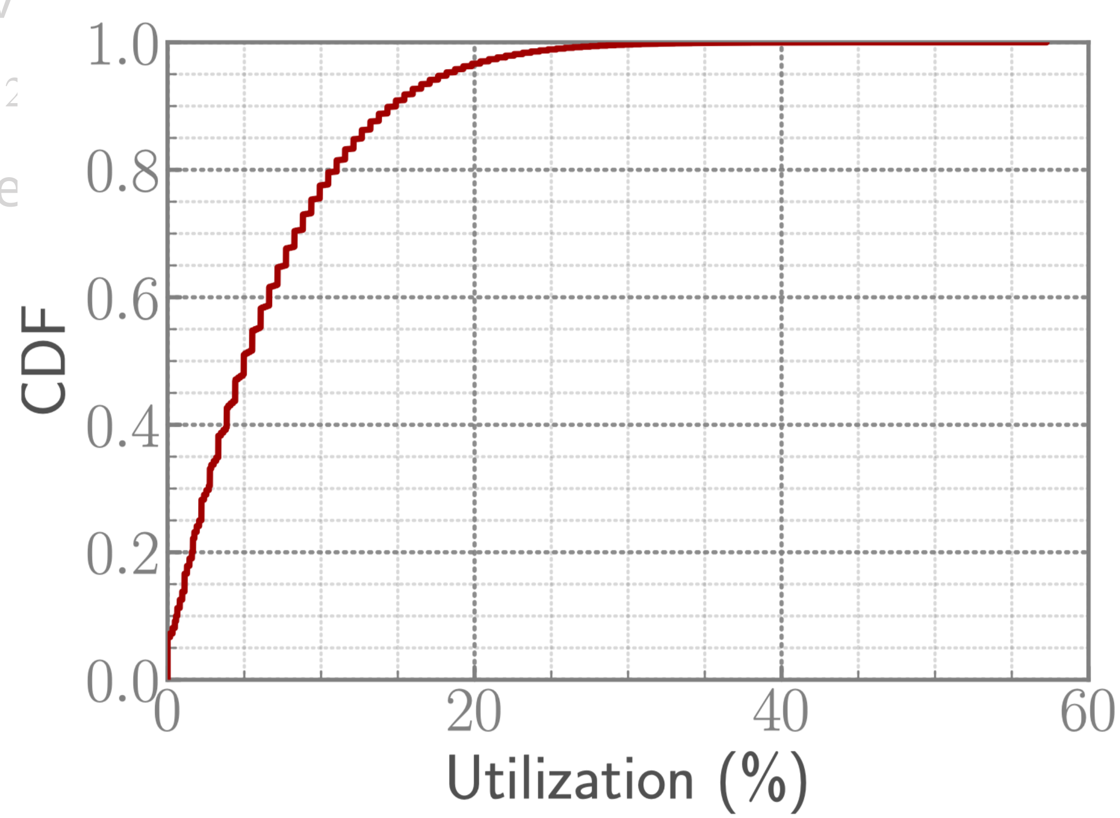
Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method (SIH): static and independent
      - Reserve a static fraction of buffer beforehand
        - Higher link capacity, larger headroom buffer
      - Independently reserve buffer for every ingress queue



Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method (SIH): static and independent
      - Reserv
        - $\eta = 2$ ... eadroom buffer
      - Indepe ... ue



Headroom buffer utilization at the local maximum point:

Only 4.96% at the median

Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method (SIH): static and independent
      - Reserve a static fraction of buffer
        - $\eta = 2(C \cdot D_{prop} + L_{MTU}) + 3840B$: Higher link capacity, larger headroom buffer
      - Independently reserve buffer for every ingress queue
    - Why this is inefficient?
      - Not all queues need to occupy headroom
      - Different ingress queues in the same port share the uplink capacity
      - Upstream devices is not always sending traffic at full rate

Motivation

- Why?
  - Reason 2: headroom allocation method is inefficient
    - Current headroom allocation method: static and independent
      - Reserve a static fraction of buffer
      - Independently reserve buffer for every ingress queue
    - Why this is inefficient?
      - Not all queues need to occupy headroom
      - Different ingress queues naturally share the uplink capacity
      - Upstream devices is not always sending traffic at full rate

Headroom allocation scheme should be improved

Our Approach:
Dynamic Shared Headroom

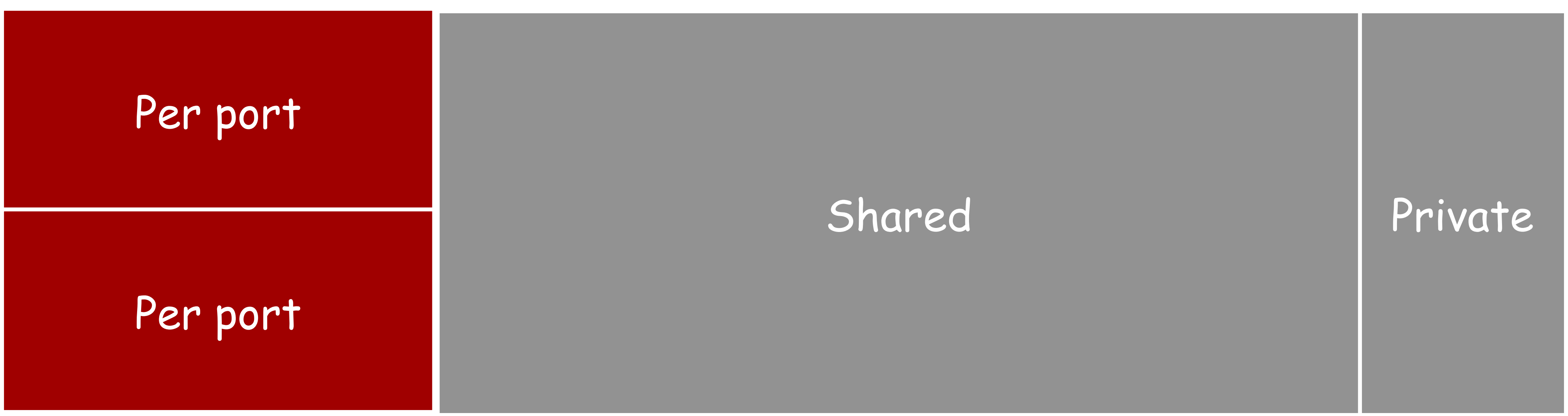
Design

Observation 1: Different ingress queues share the uplink capacity

No need to independently reserve buffer for each queue

Idea 1 (Shared): Reserve enough buffer for each port

Insurance Headroom



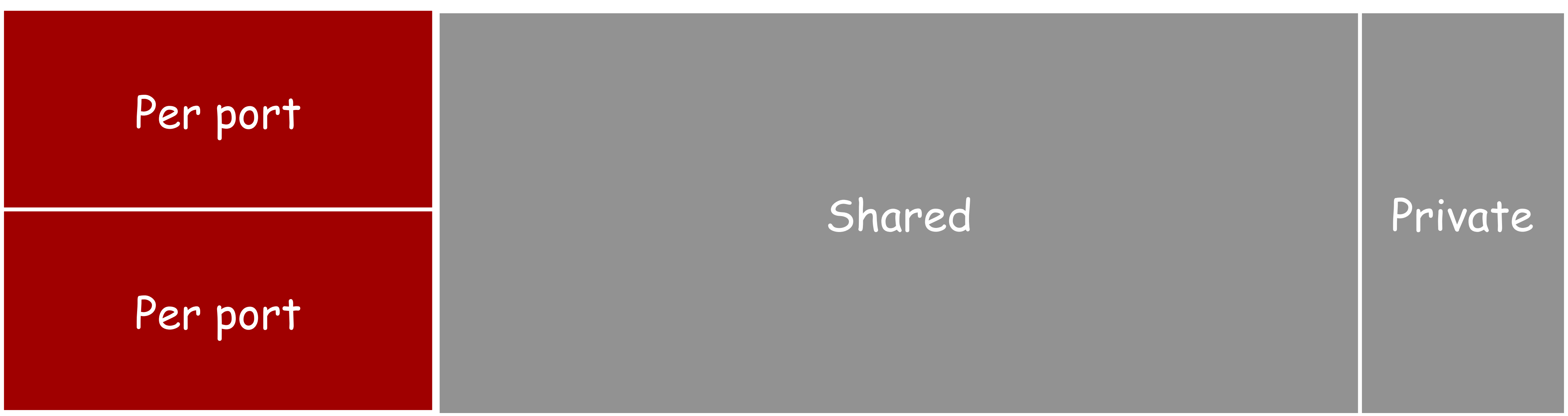
Design

Observation 1: Different ingress queues share the uplink capacity

No need to independently reserve buffer for each queue

Idea 1 (Shared): Reserve enough buffer for each port

Insurance Headroom



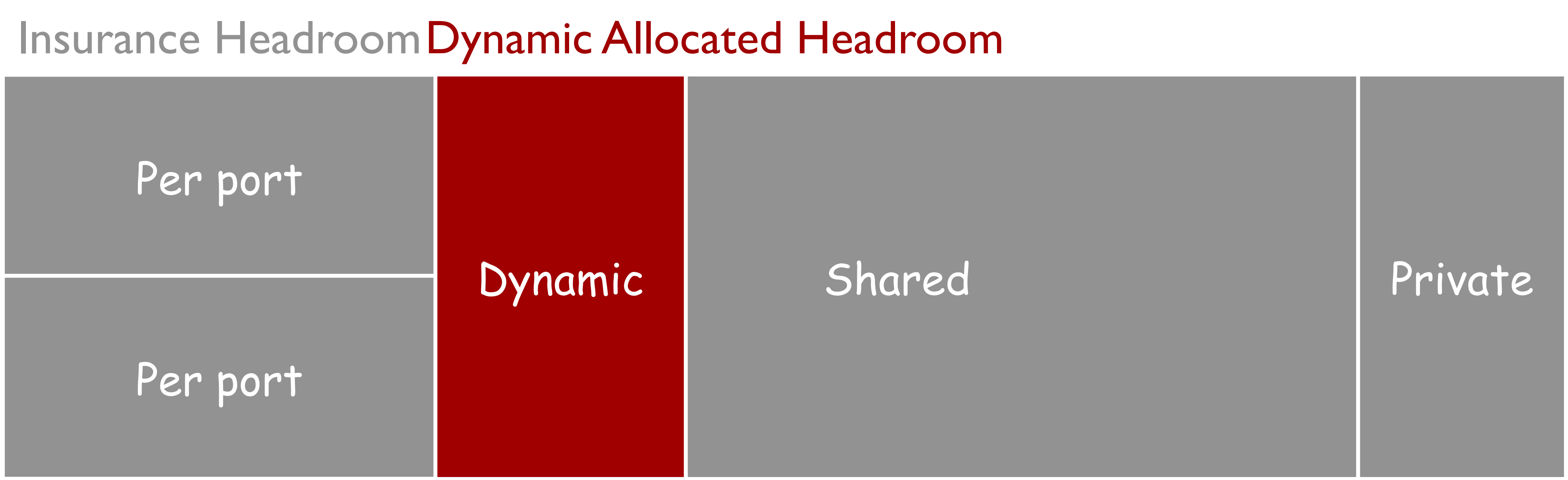
Only insurance headroom? → Performance isolation is violated

Design

Observation 2: Not all queues need to occupy headroom

A queue needs headroom only when congested

Idea 2 (Dynamic): Allocate headroom based on the congestion status



Design

Observation 3: Upstream devices are not always sending traffic at full rate

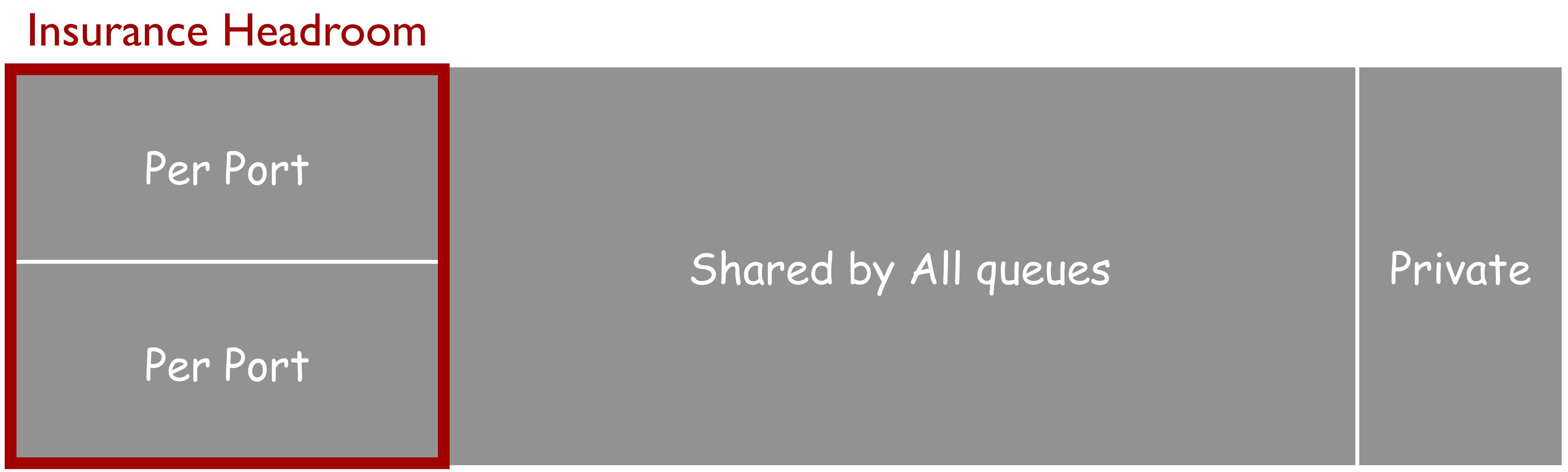
No need to allocate worst-case buffer for all queues

Idea 3 (Shared): let all queues share the allocated buffer

Insurance Headroom Shared Headroom, Dynamic Allocated

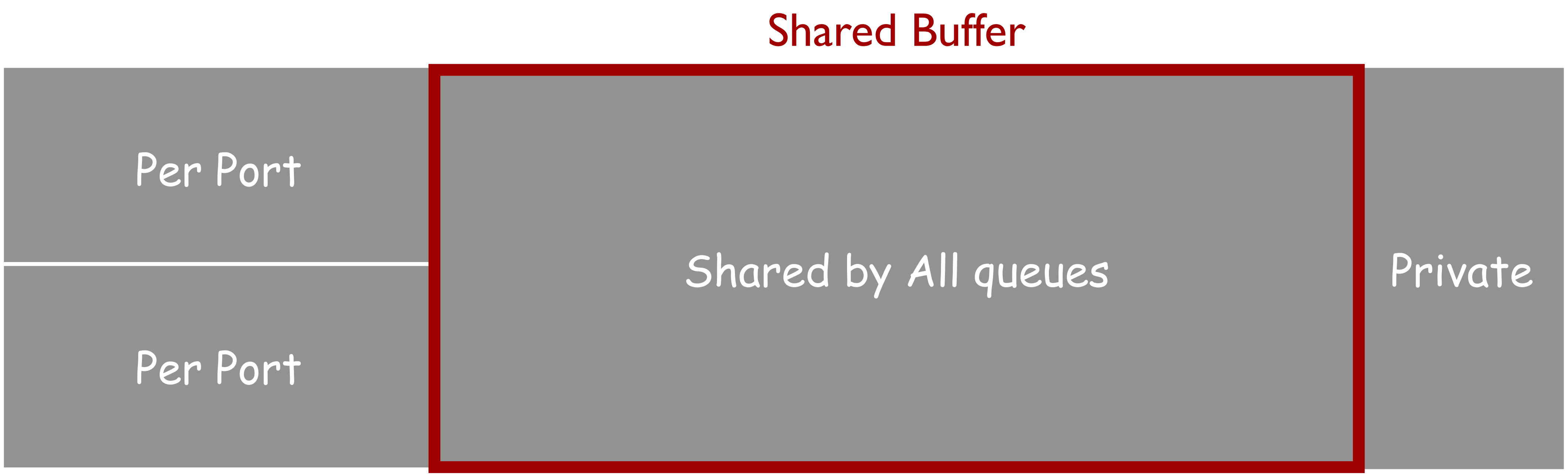
Design

- Buffer structure with DSH
  - Insurance Headroom
    - Ensure lossless forwarding
    - Per-port allocated (rather than per-queue)
    - Shared by queues in the same port



Design

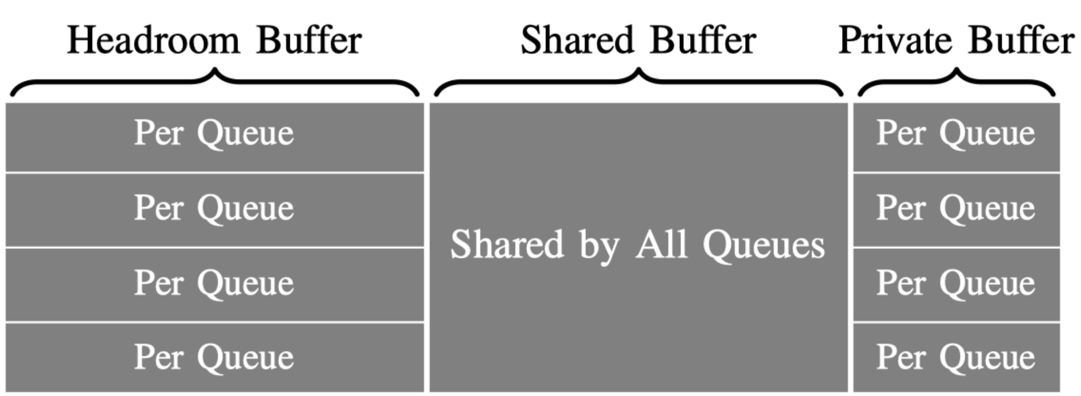
- Buffer structure with DSH
  - Shared Buffer = Shared Headroom + Shared Footroom
    - Shared headroom
      - Prevent performance isolation issue
      - Dynamically allocated as needed
      - Shared by all queues
    - Shared footroom: the same as traditional



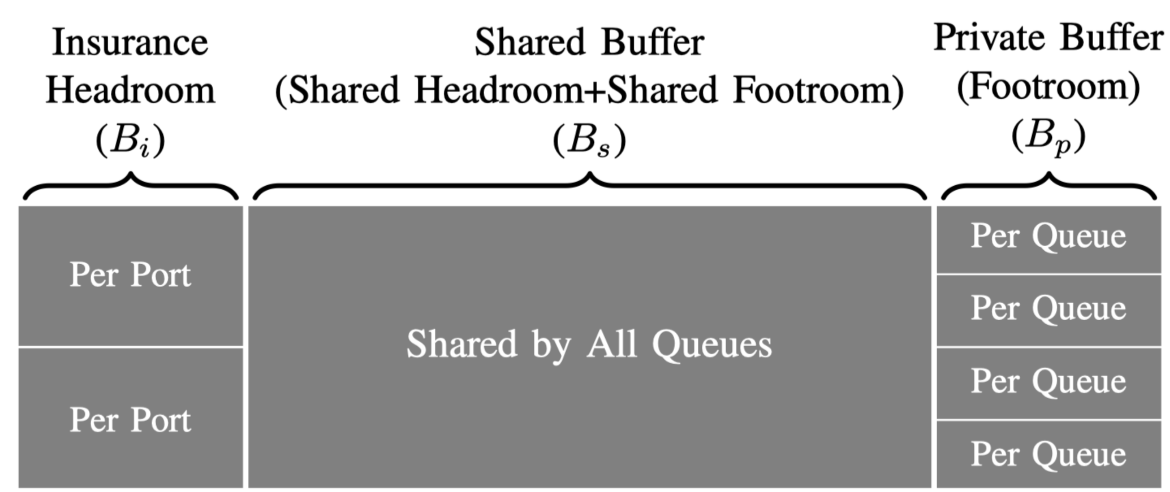
Design

- Benefits: readily realizing
  - DSH does not modify the buffer structure

Existing

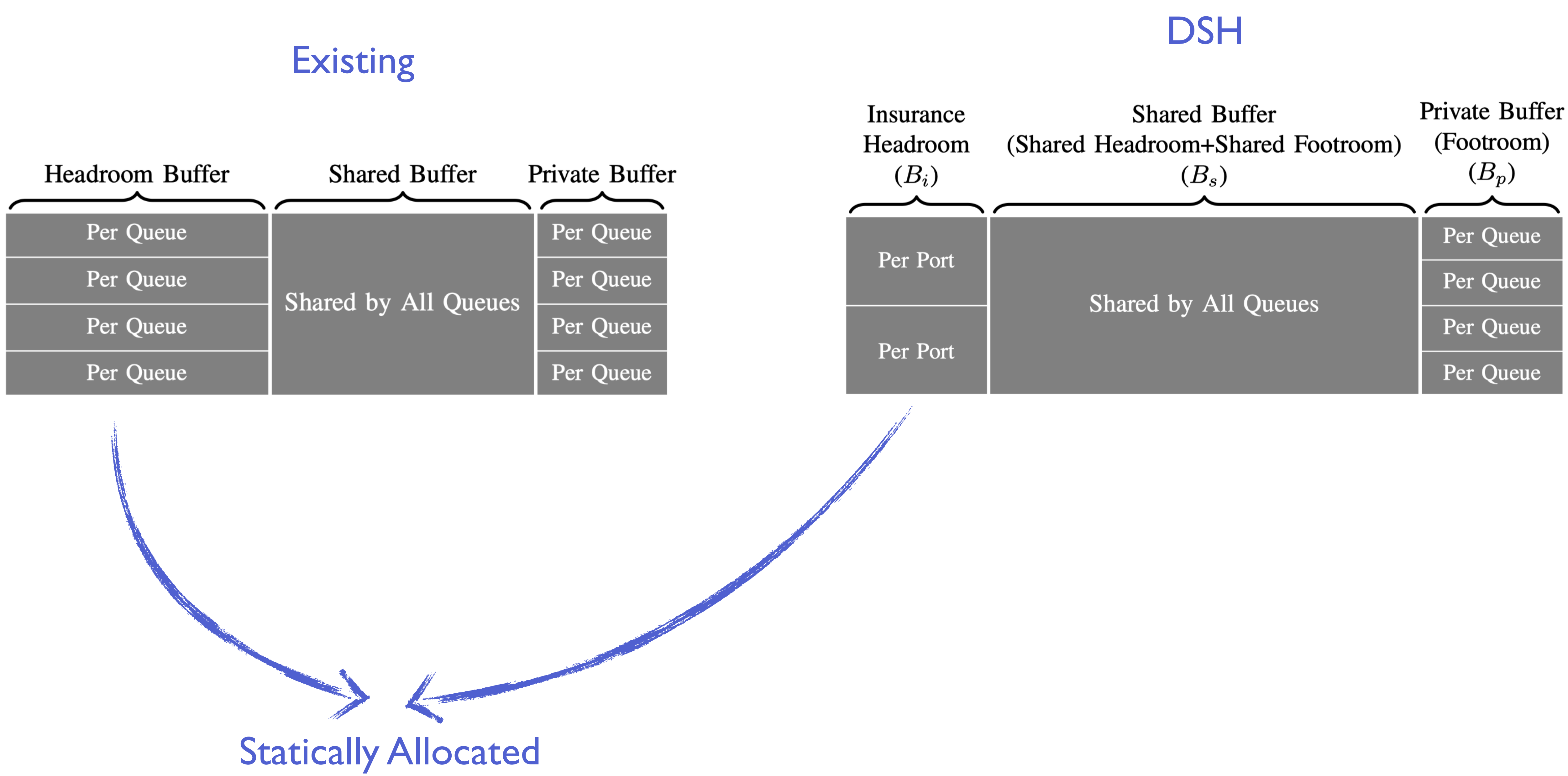


DSH



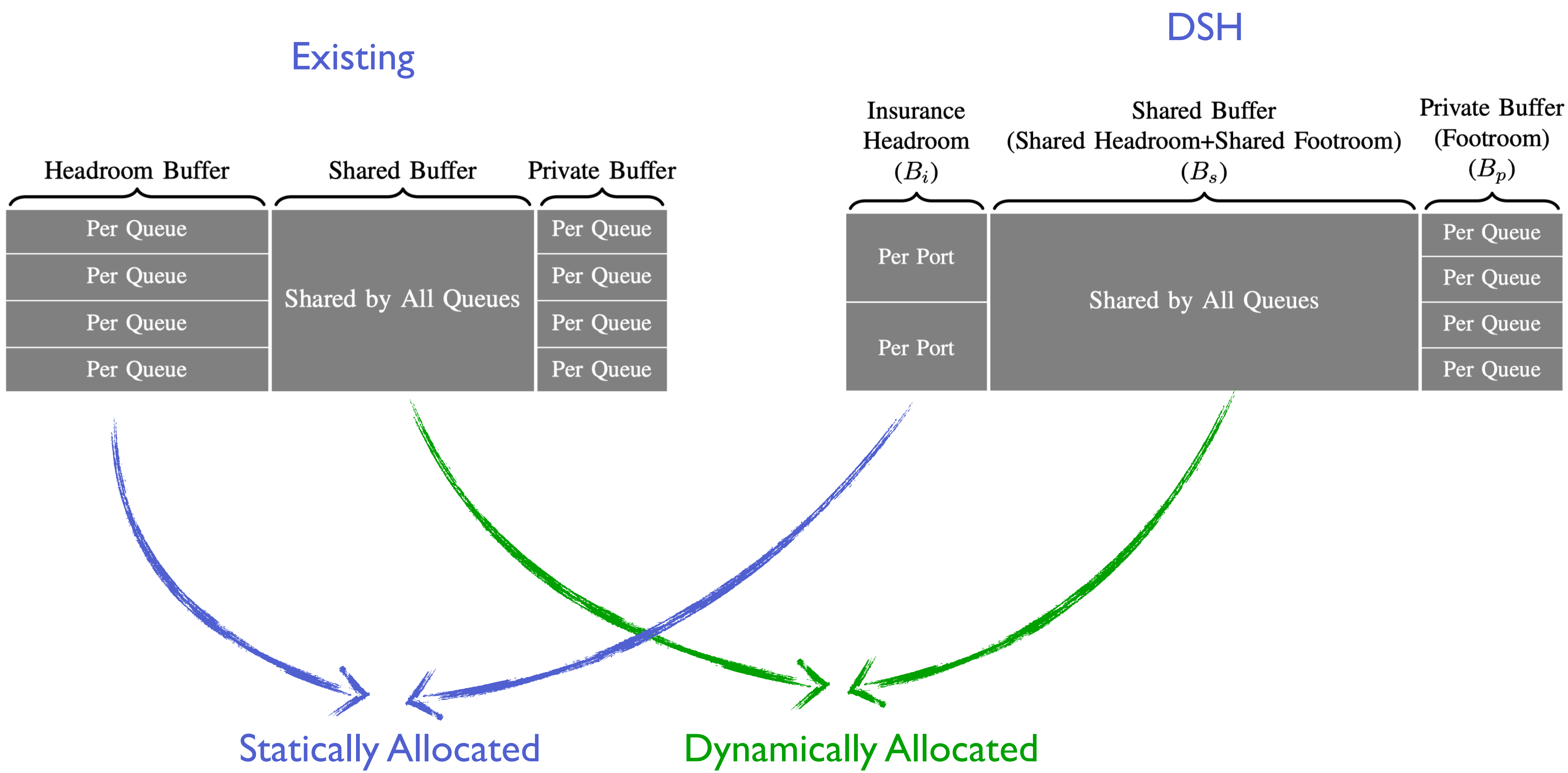
Design

- Benefits: readily realizing
  - DSH does not modify the buffer structure



Design

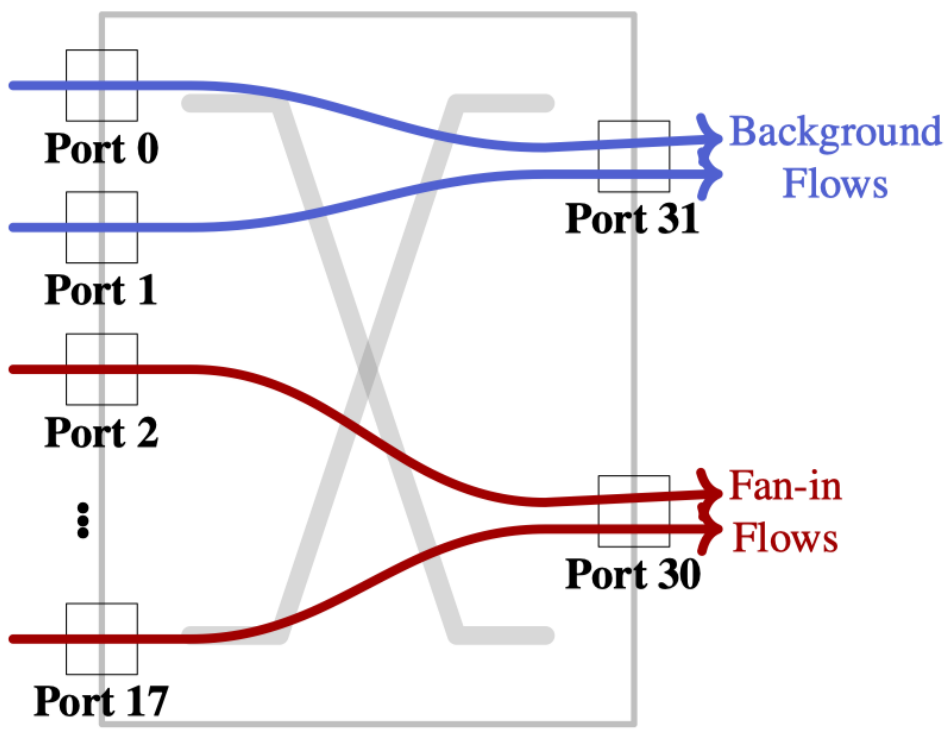
- Benefits: readily realizing
  - DSH does not modify the buffer structure



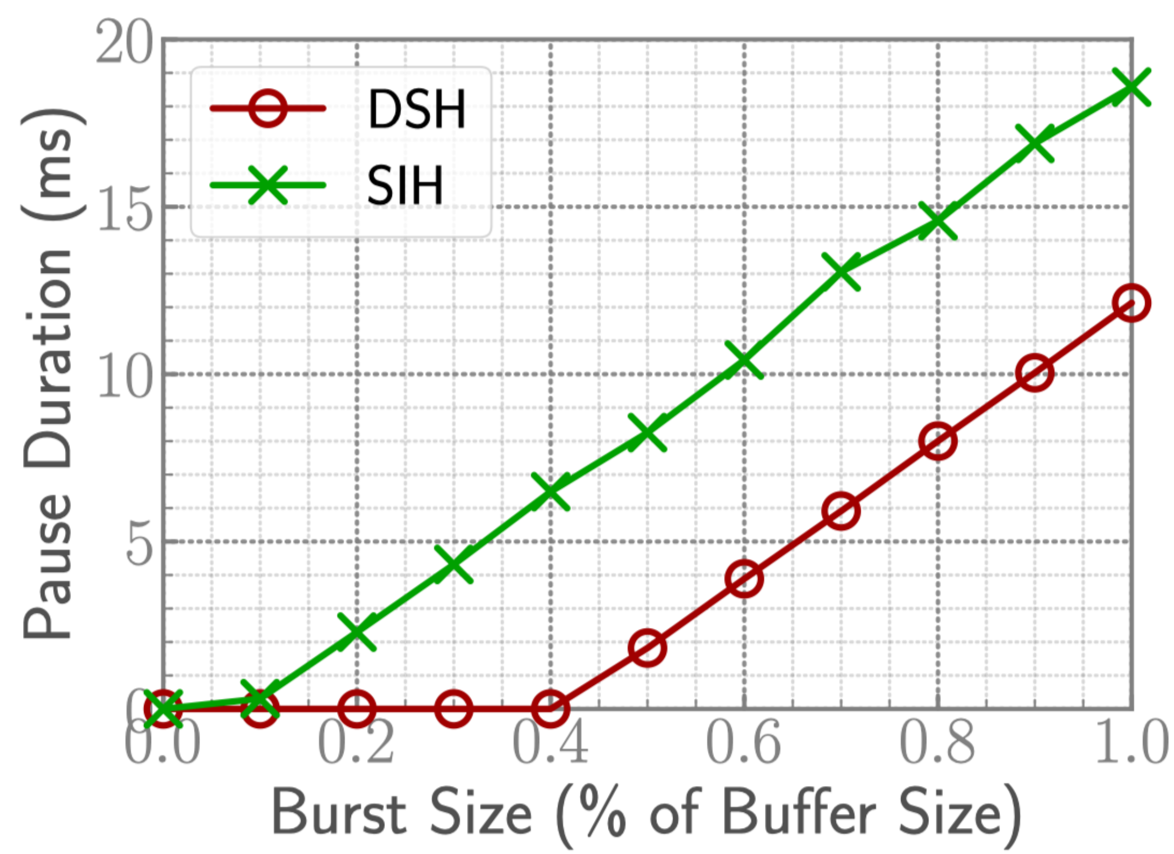
Design

- Benefits: readily realizing
  - DSH does not modify the buffer structure
  - DSH can be simply realized by flow control
    - [Queue-level] Buffer occupancy of a *queue* > X_{qoff}
      - Occupy the shared headroom
      - Send PAUSE frame to pause the queue
    - [Port-level] Buffer occupancy of a *port* > X_{poff}
      - Occupy the insurance headroom
      - Send PAUSE frame to pause the entire port

Evaluation — PFC Avoidance



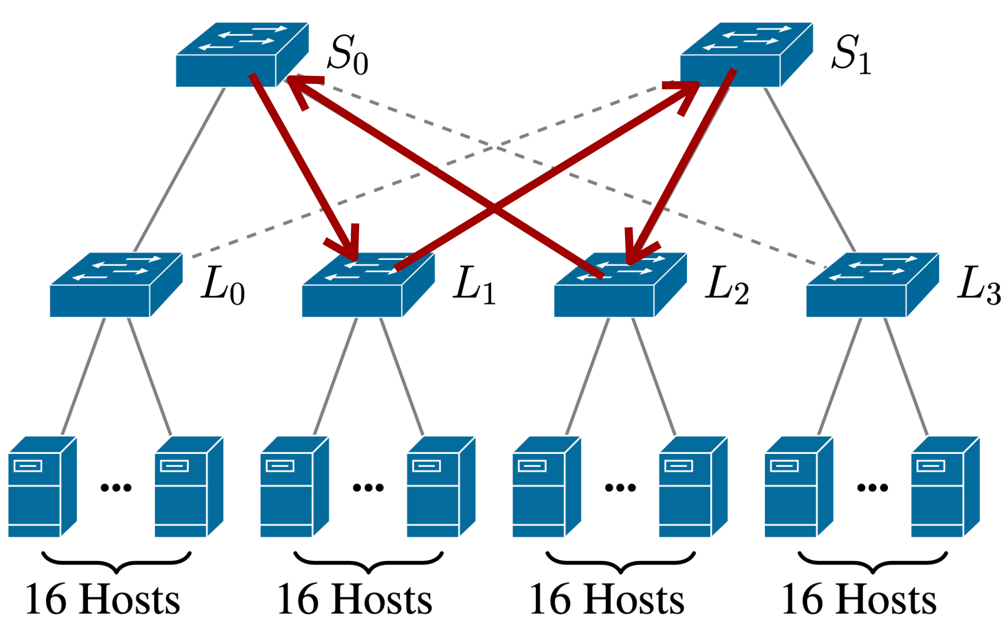
Scenario



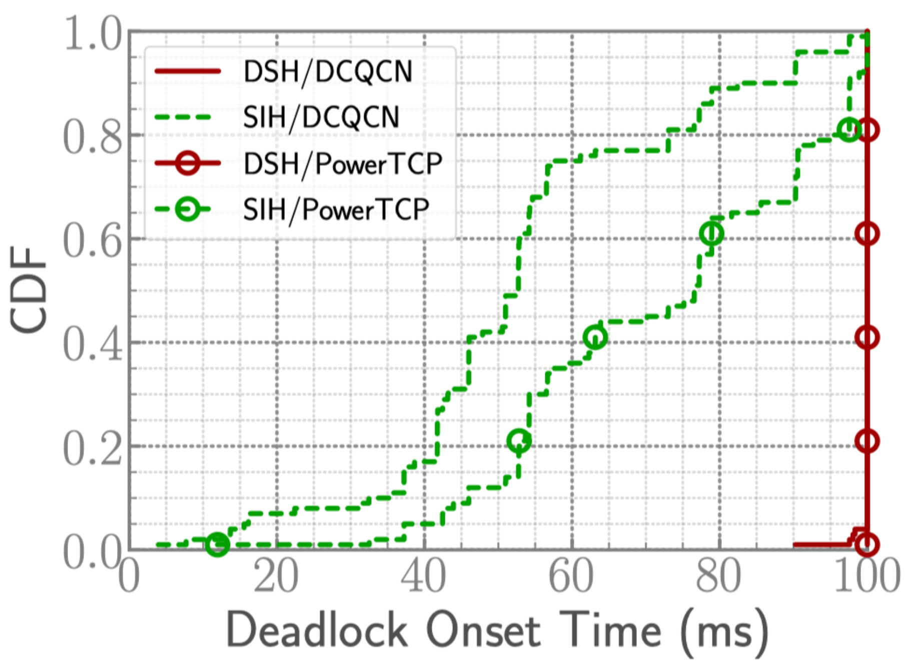
Total Pause Time

DSH can absorb 4× more bursty traffic without triggering PFC messages

Evaluation — Deadlock Avoidance



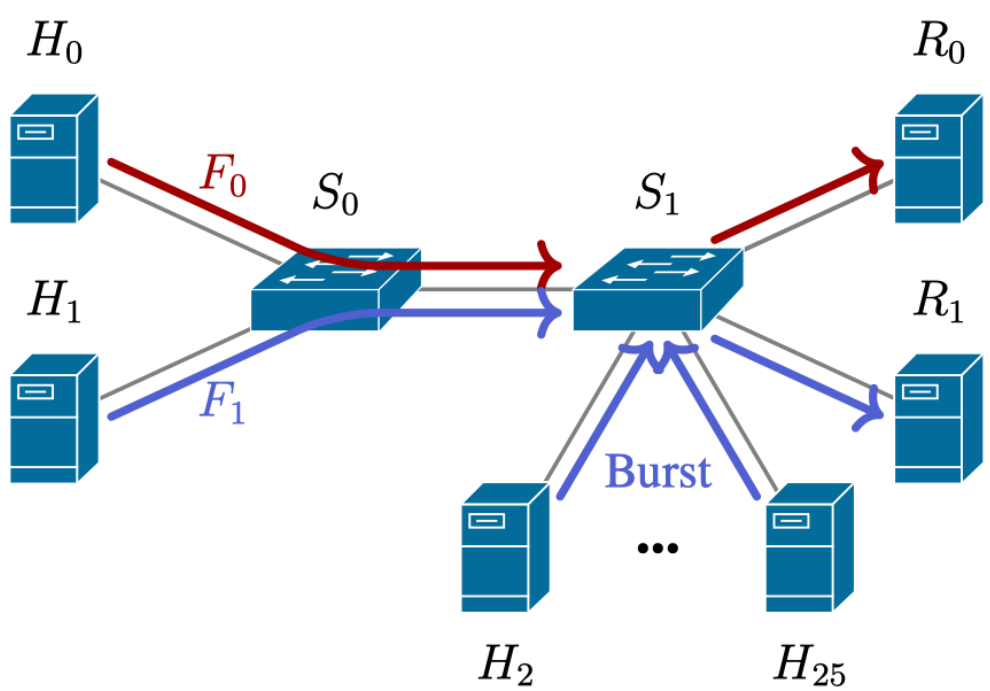
Scenario



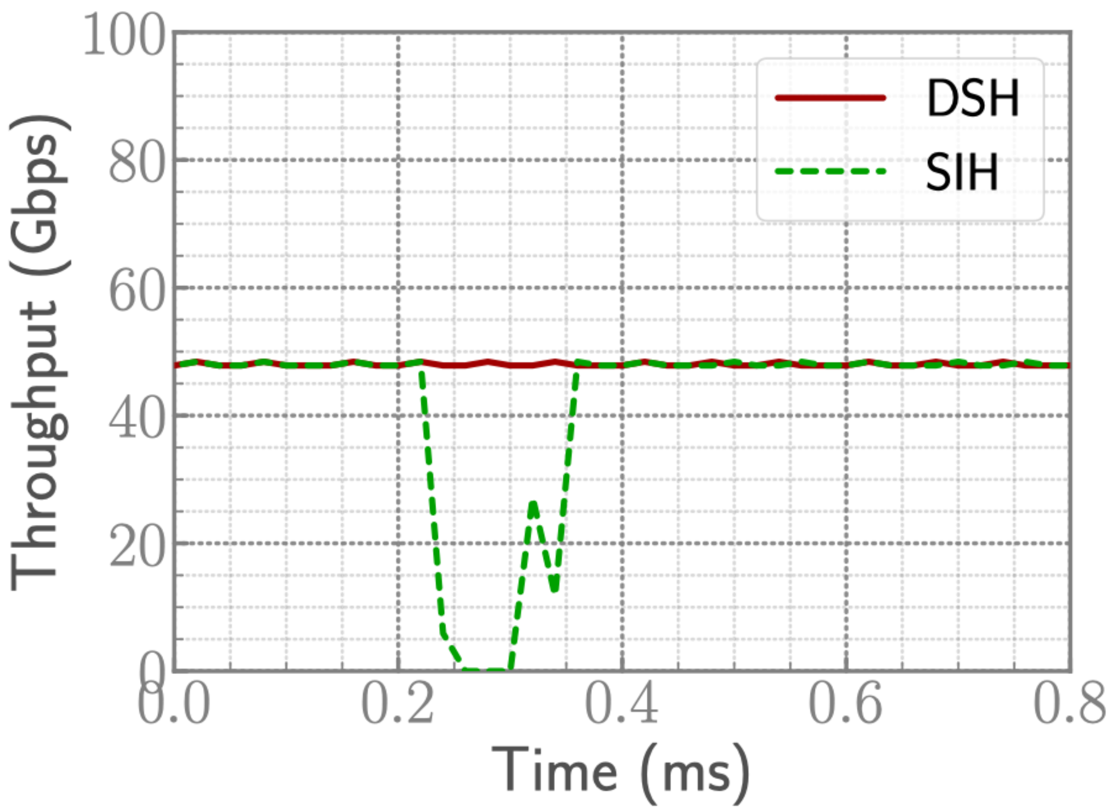
CDF of deadlock onset time

DSH can avoid 96% deadlocks with DCQCN and all deadlocks with PowerTCP

Evaluation — Collateral Damage Mitigation



Scenario



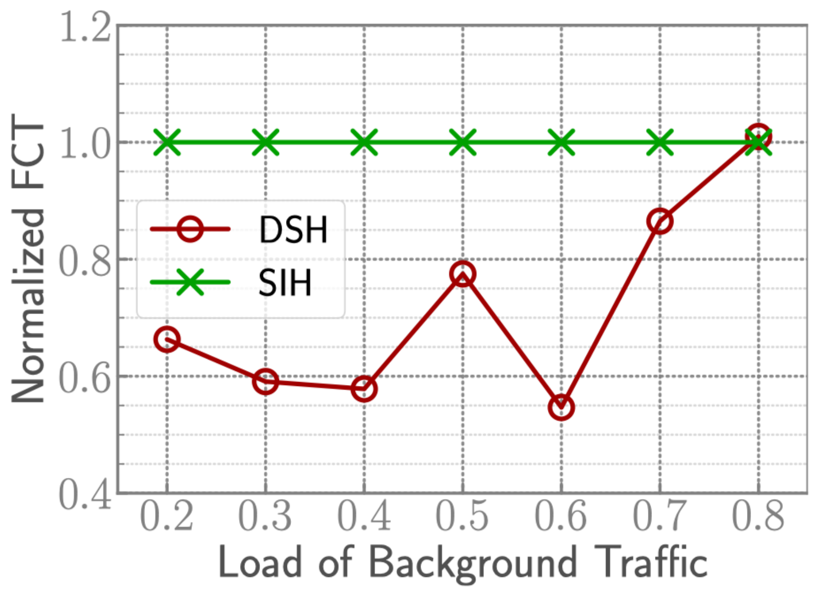
Throughput of F_0 (w/o CC)

DSH can effectively avoid performance degradation of the innocent flow

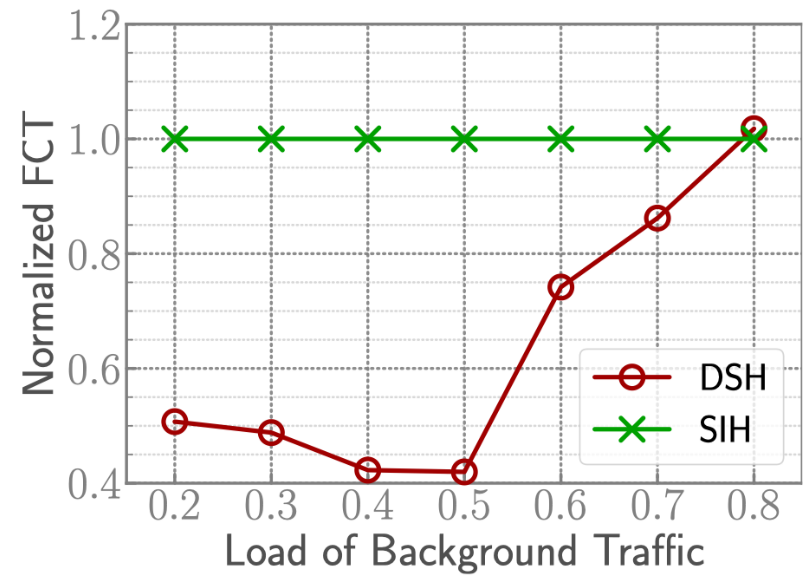
Evaluation

- Large-scale benchmark traffic
  - 100Gbps leaf-spine topology
    - 16 leaf switches, 16 spine switches, 256 servers
  - 16MB buffer (emulating Broadcom Tomahawk)
  - Transport
    - DCQCN
    - PowerTCP
  - Workload
    - Fan-in flows: 16 senders send 64KB data to 1 receiver
    - Background flows
      - Flow size: web search traffic
      - Flow arrival: Poisson
    - Total network load: 90%

Evaluation — Large-scale Benchmark

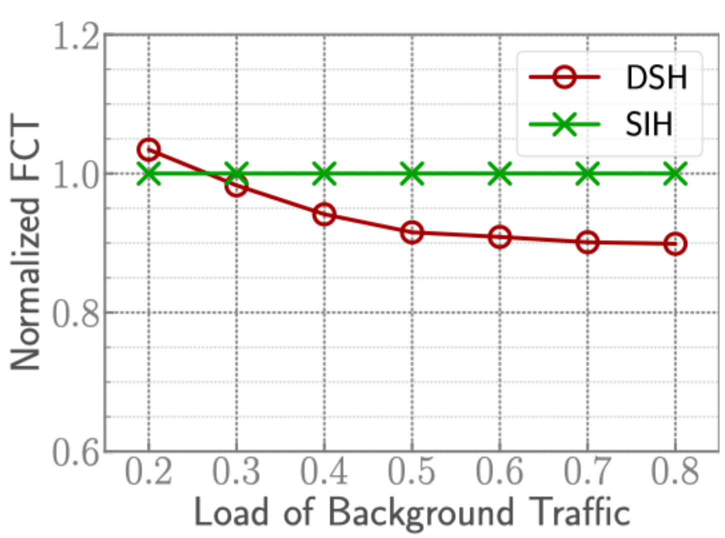


FCT of fan-in traffic (DCQCN)

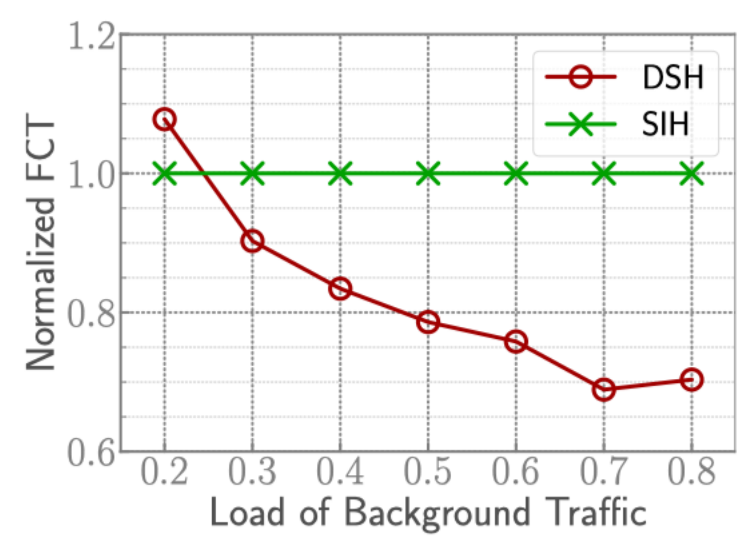


FCT of fan-in traffic (PowerTCP)

Fan-in traffic: DSH can reduce the FCT by up to 57.7%



FCT of background traffic (DCQCN)



FCT of background traffic (PowerTCP)

Background traffic: DSH can reduce the FCT by up to 31.1%

Conclusion

- Buffer becomes increasingly insufficient while current headroom allocation scheme is quite inefficient
- DSH is an efficient headroom allocation scheme, which dynamically allocates headroom and enables headroom to be shared
- DSH can significantly reduce PFC messages

Thank you!