PURDUE UNIVERSITY

STyGIANet

Datacenter Switch Buffer Sharing

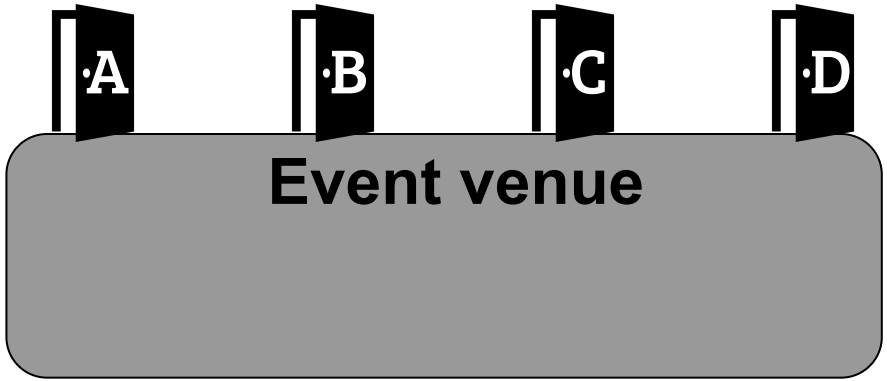
CS 422 / CS 536

April 16, 2026

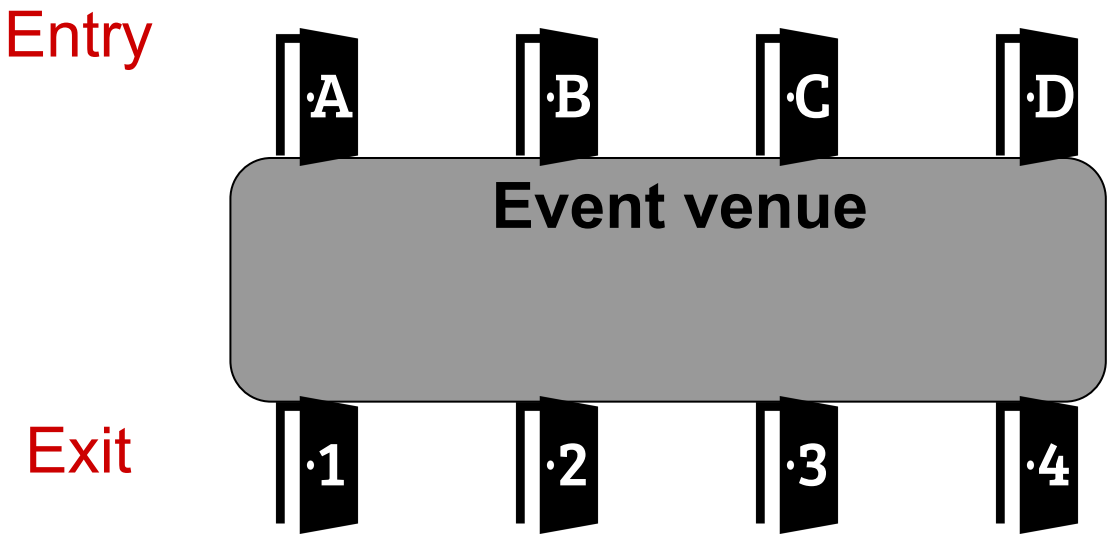
Vamsi Addanki
vaddank@purdue.edu

Let's Play a Game

Entry



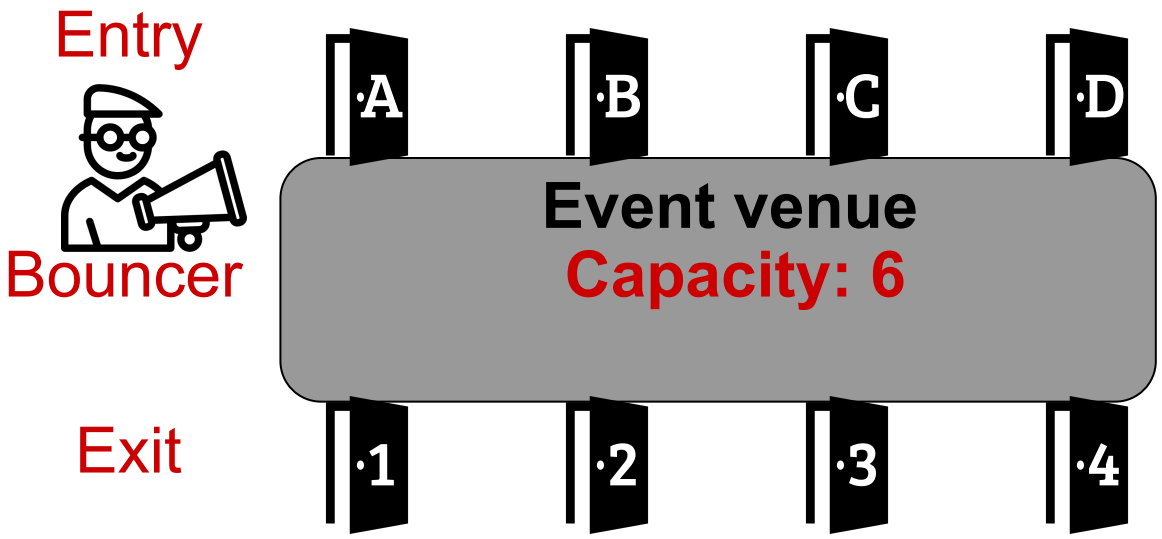
Let's Play a Game



PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

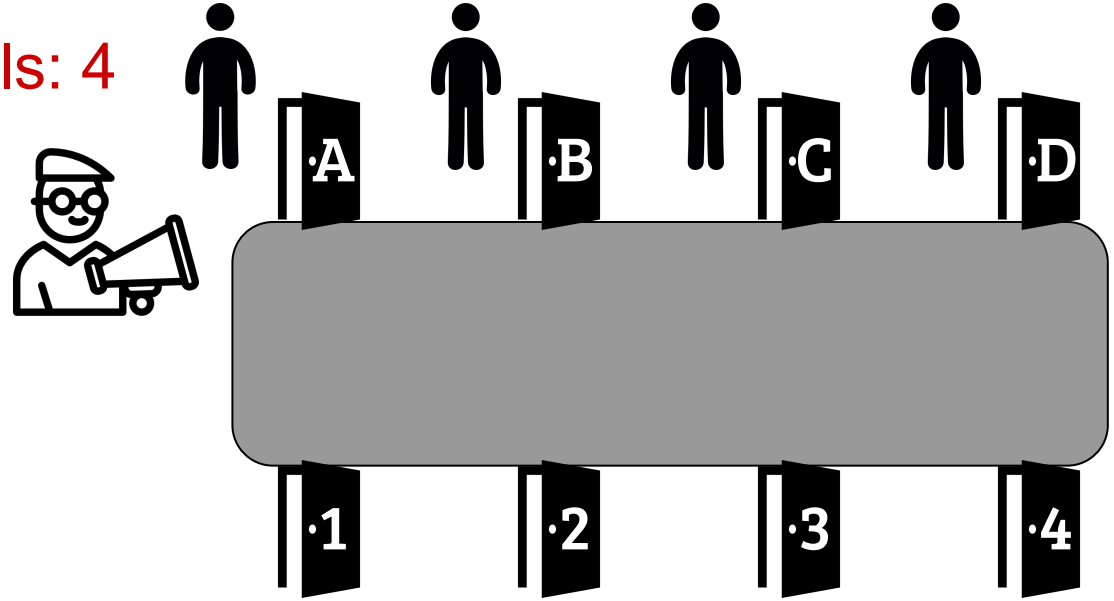


PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

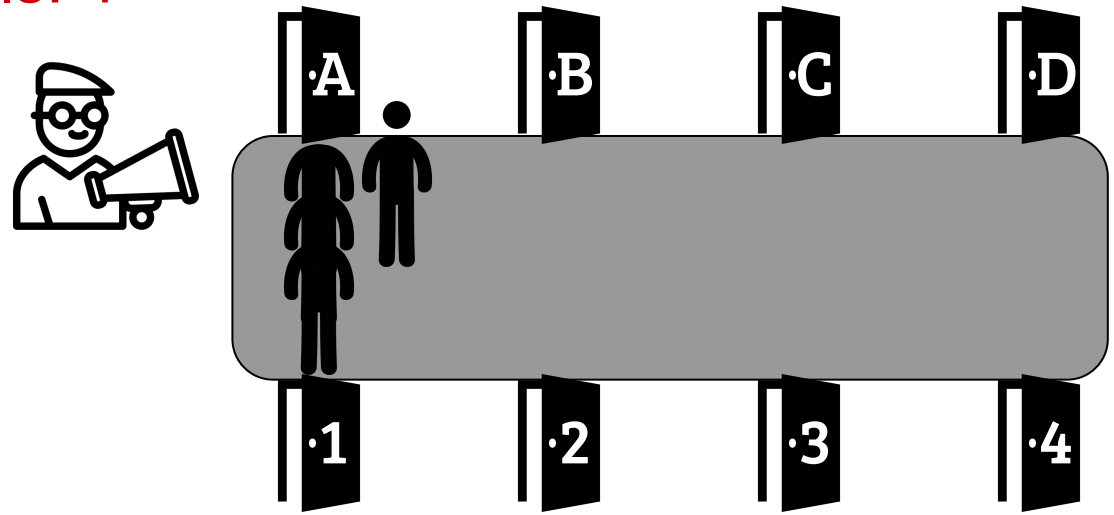
Arrivals: 4



Capacity: 6

Let's Play a Game

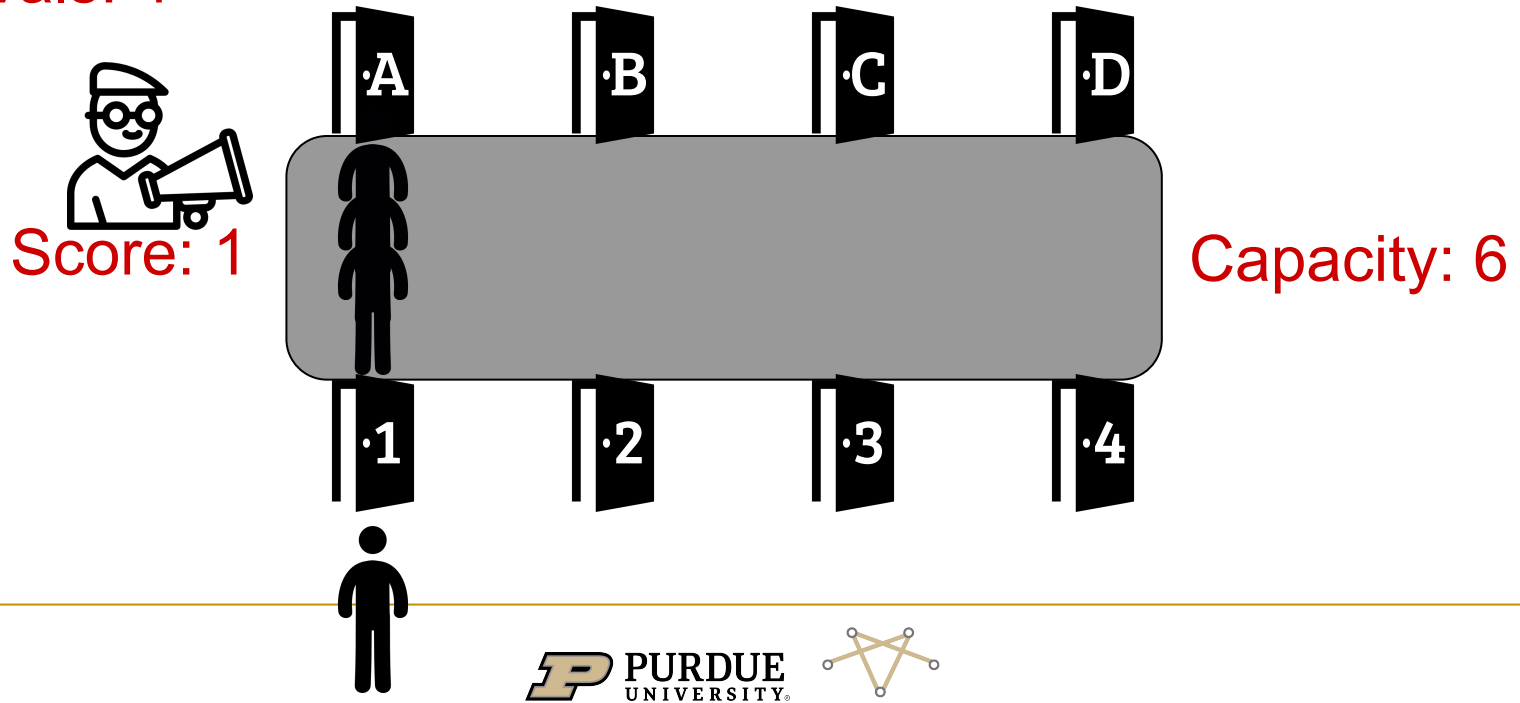
Arrivals: 4



Capacity: 6

Let's Play a Game

Arrivals: 4



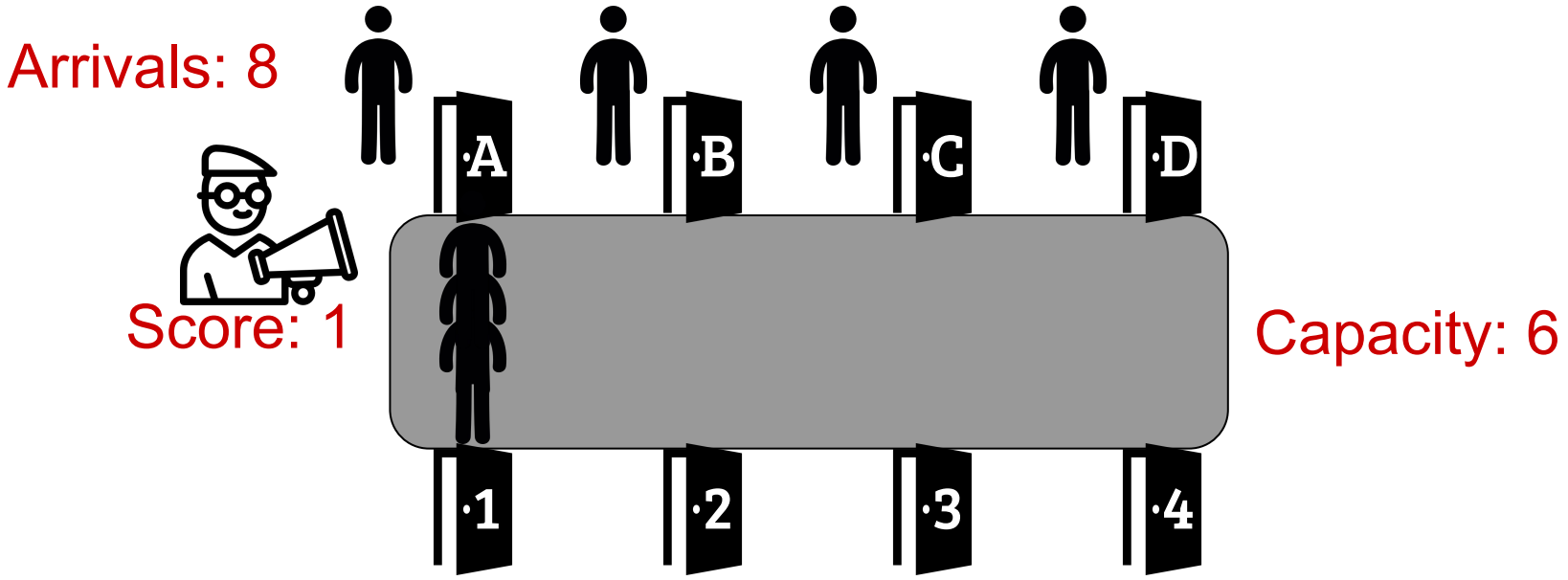
Score: 1

Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

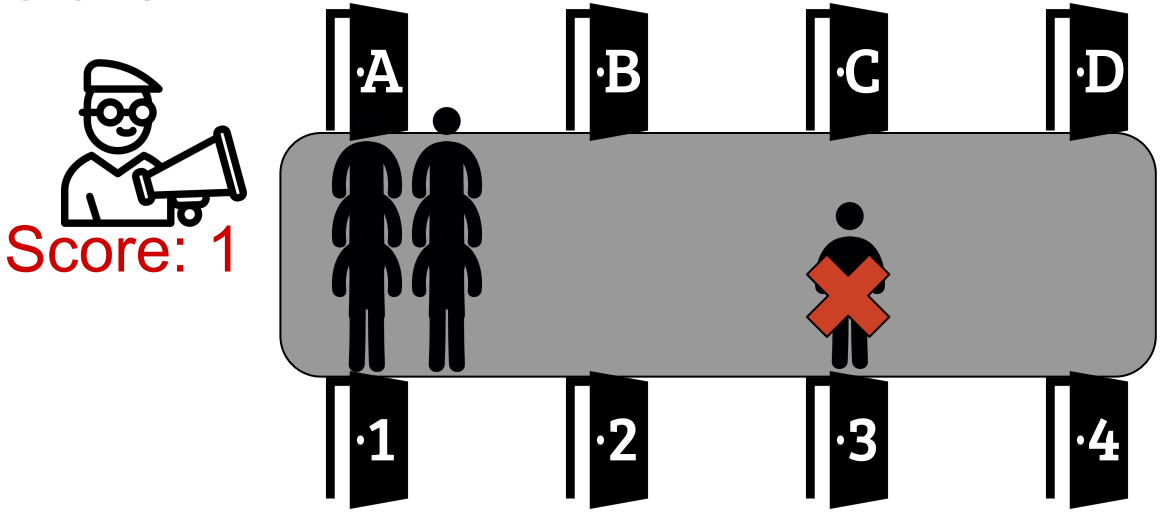


PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 8



Score: 1

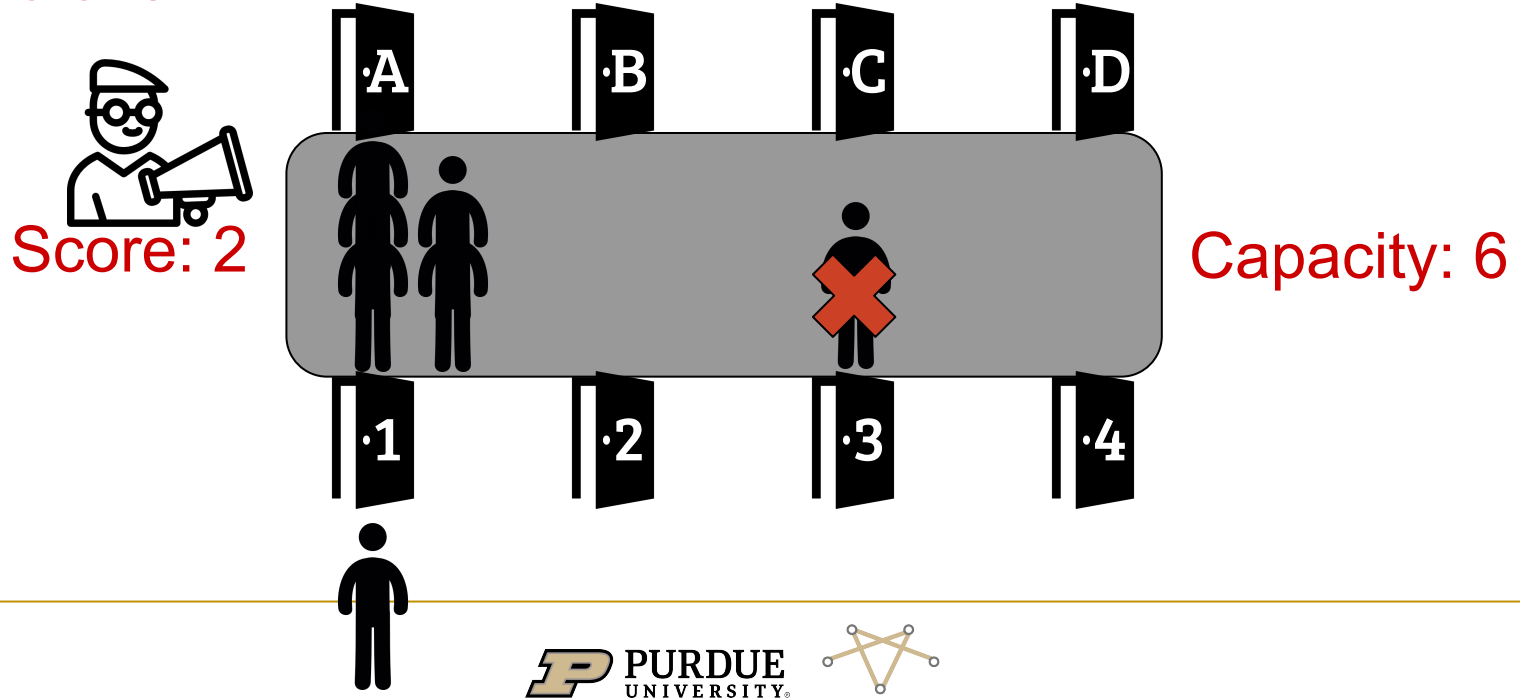
Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 8



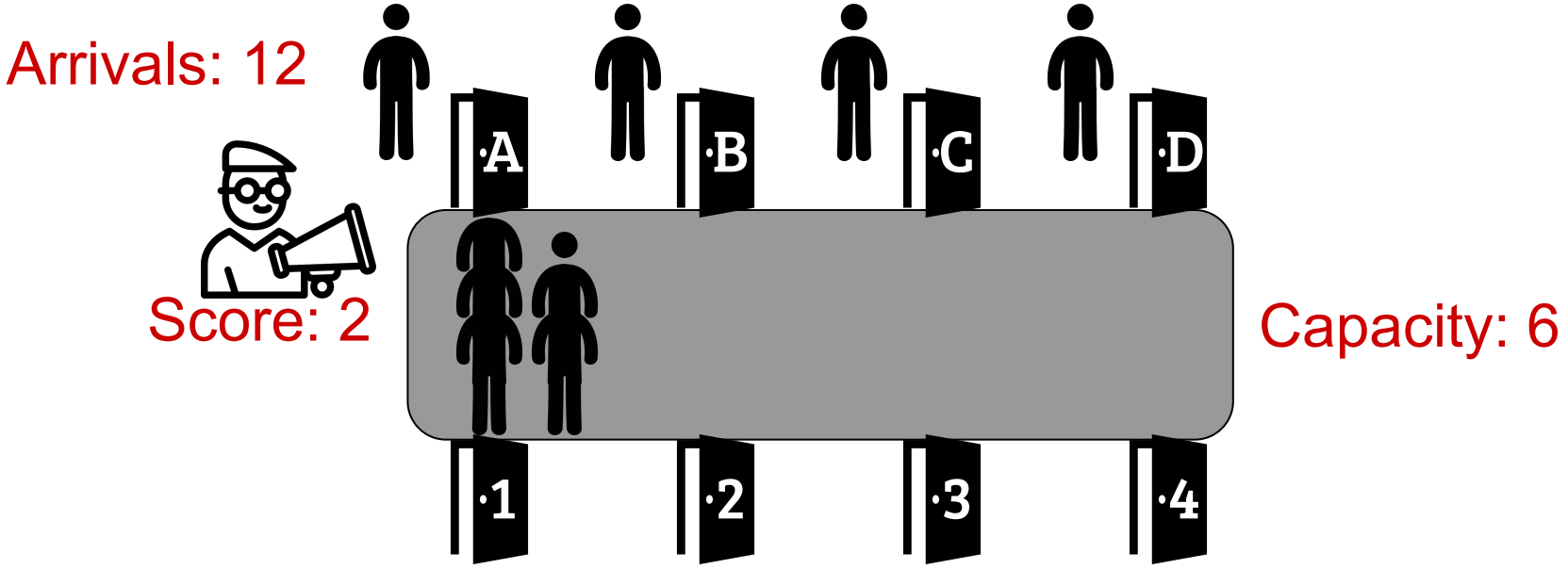
Score: 2

Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

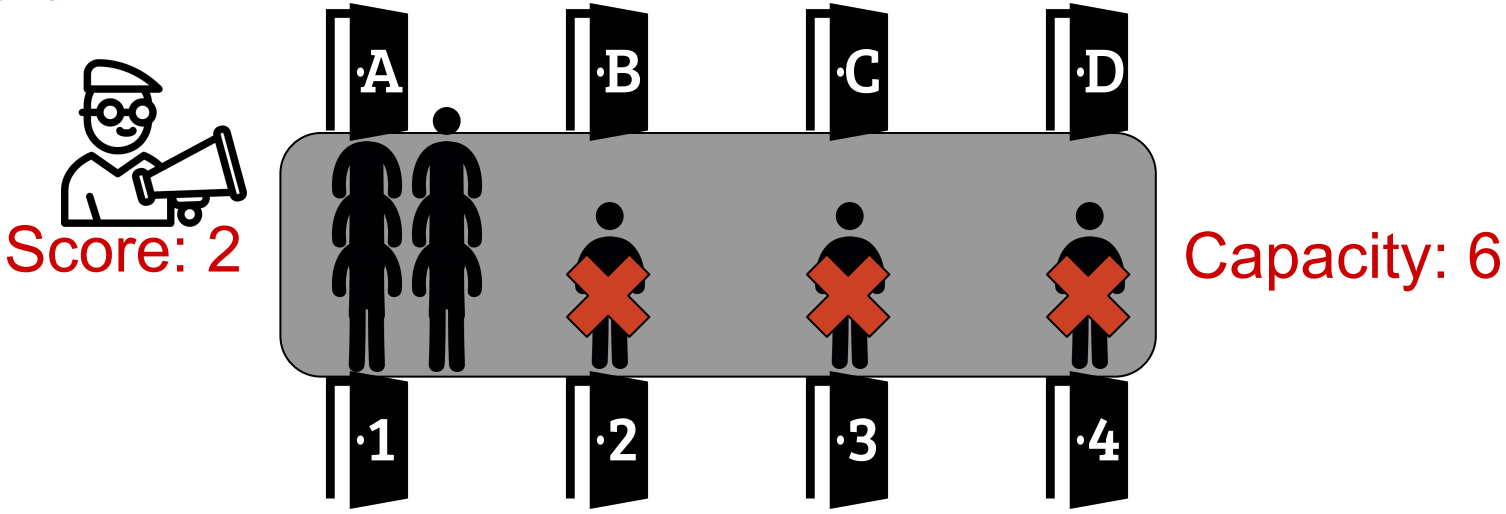


PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 12



Score: 2

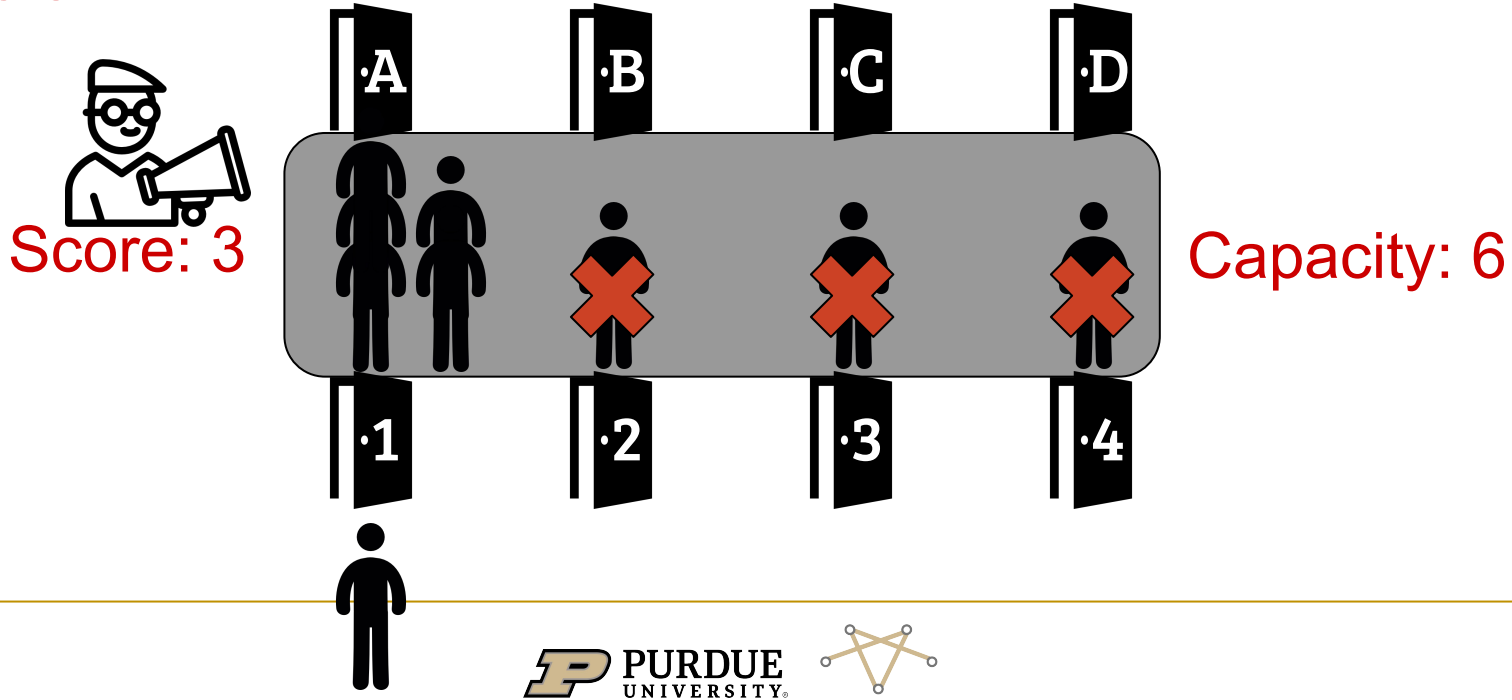
Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 12



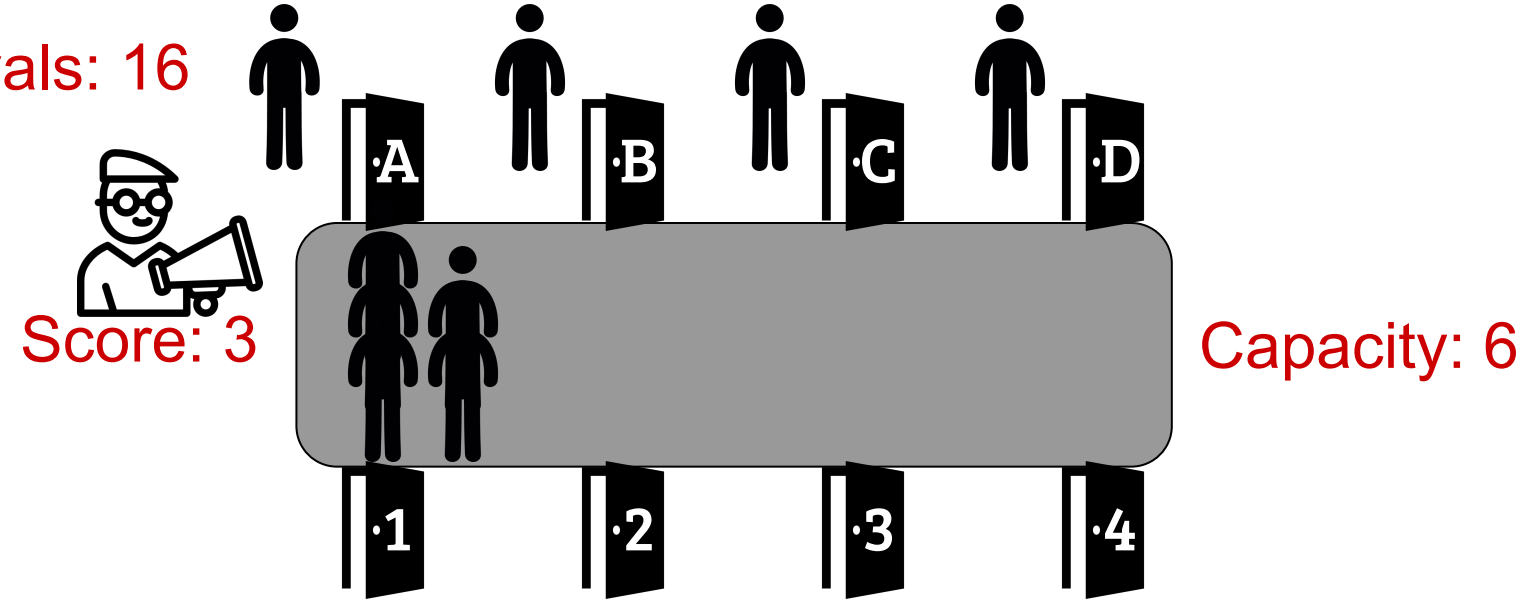
Score: 3

Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

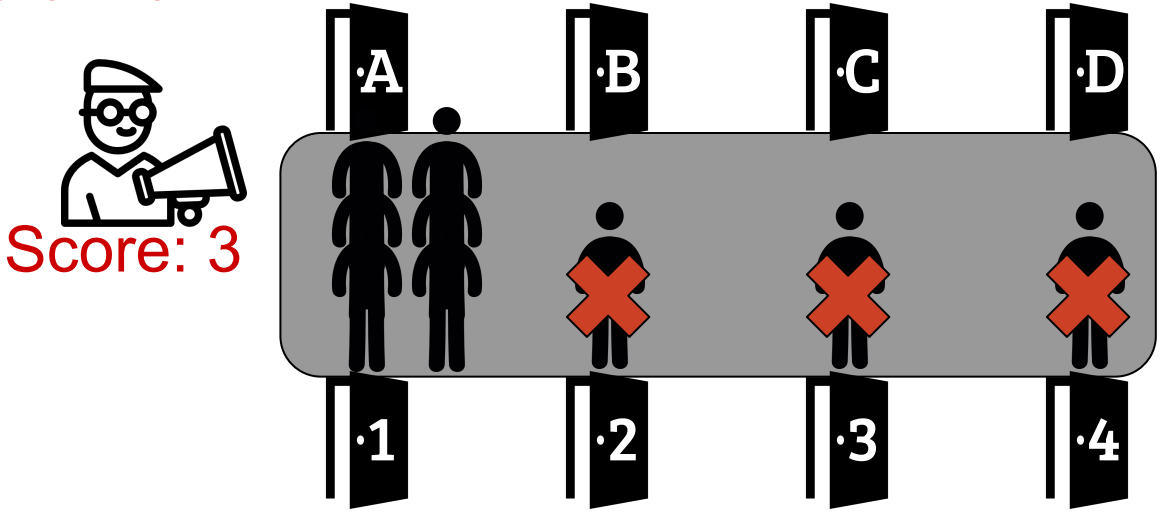


PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 16



Score: 3

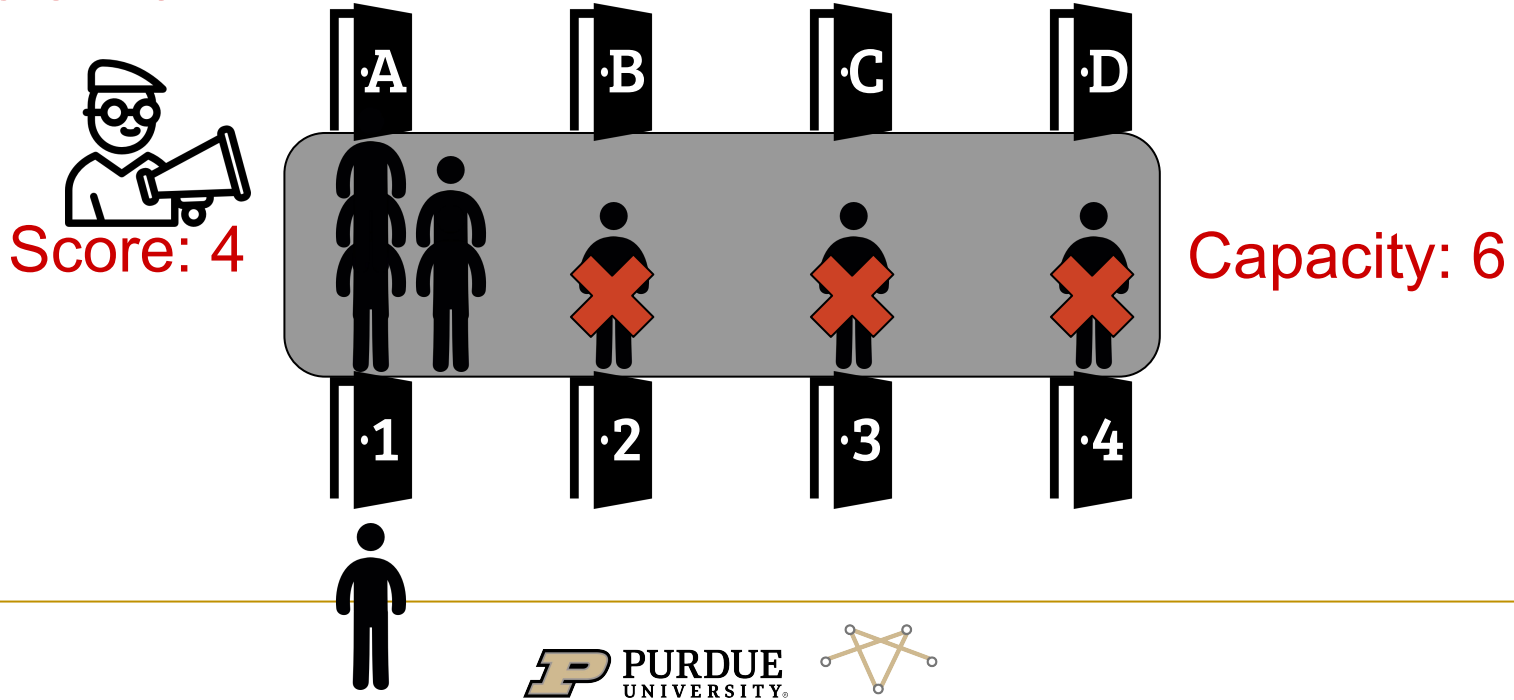
Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 16



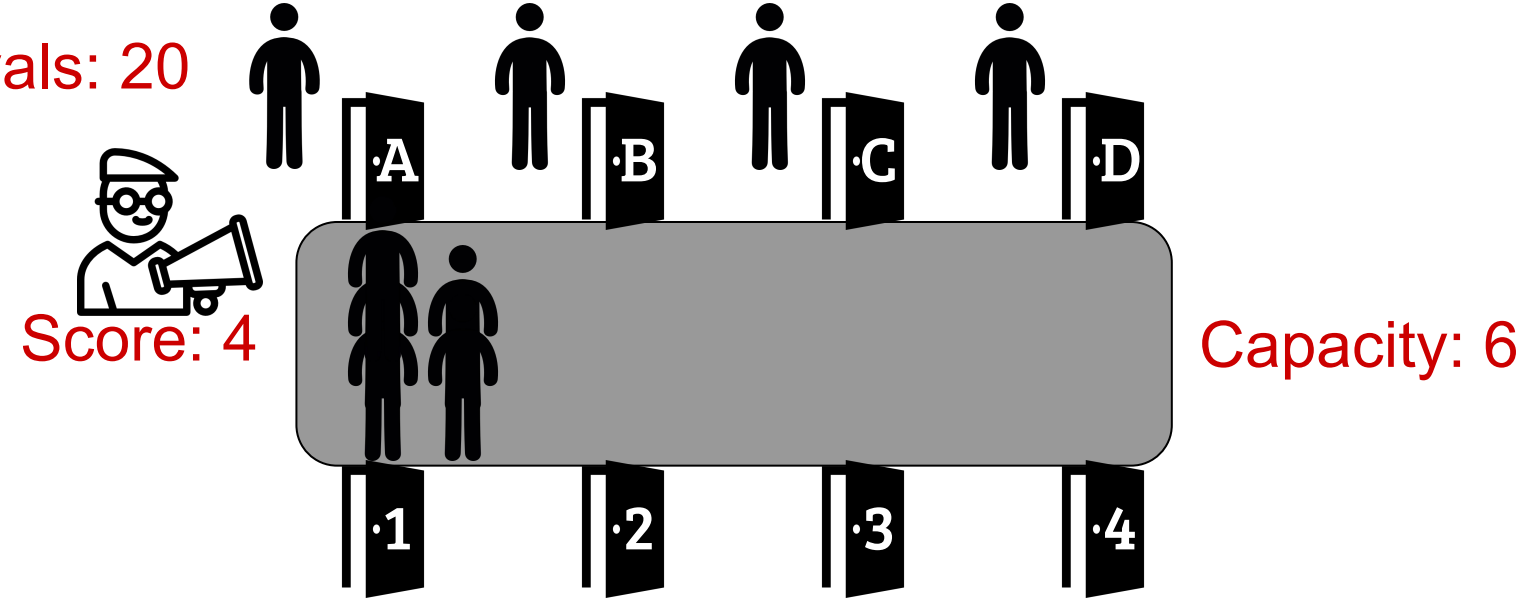
Score: 4

Capacity: 6

PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

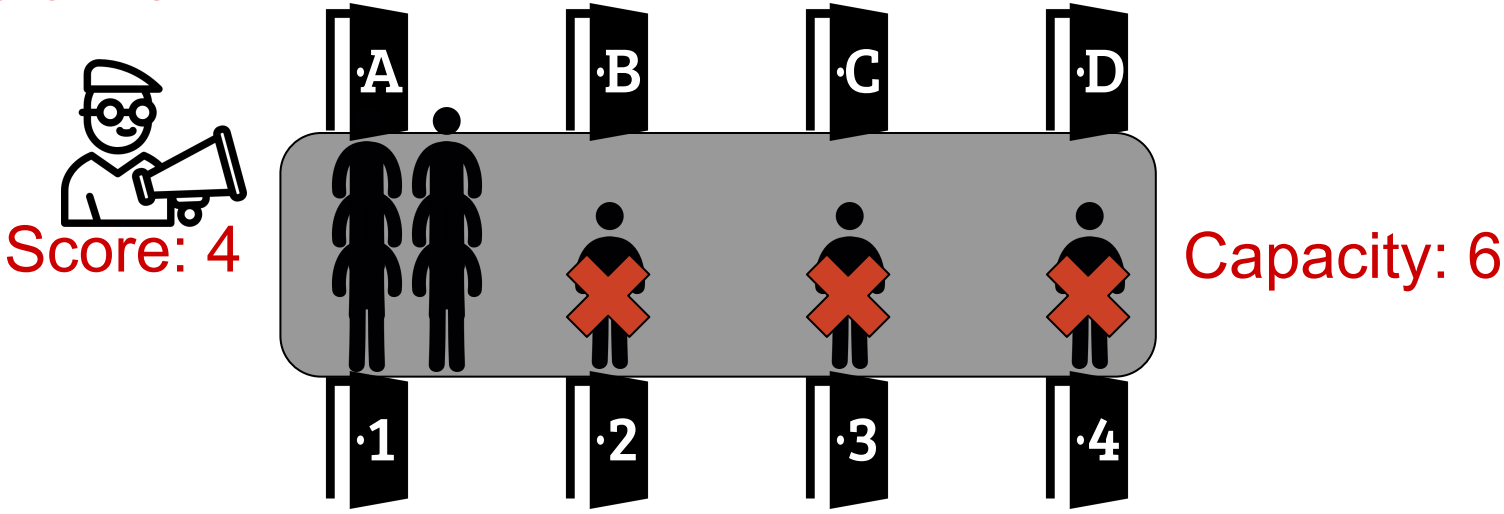


PURDUE UNIVERSITY

STyGIANet

Let's Play a Game

Arrivals: 20

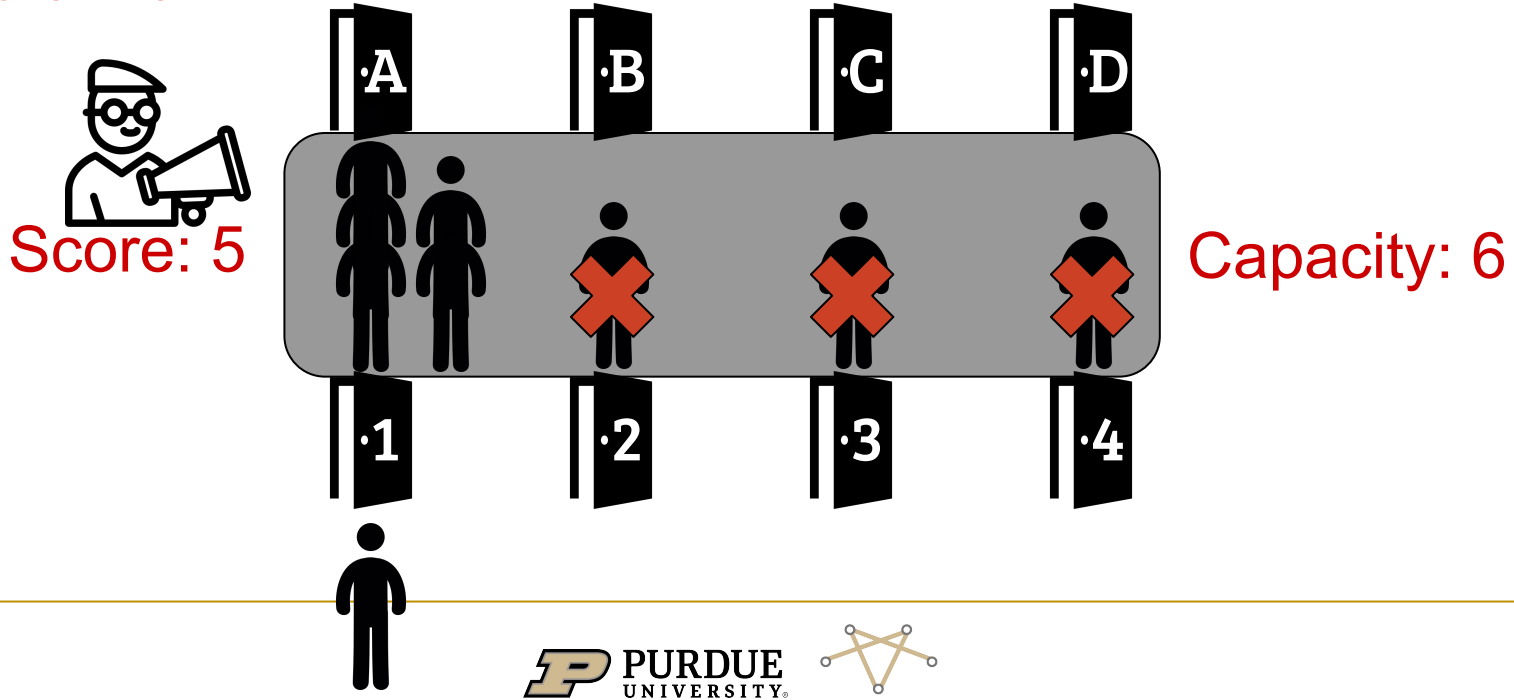


Score: 4

Capacity: 6

Let's Play a Game

Arrivals: 20

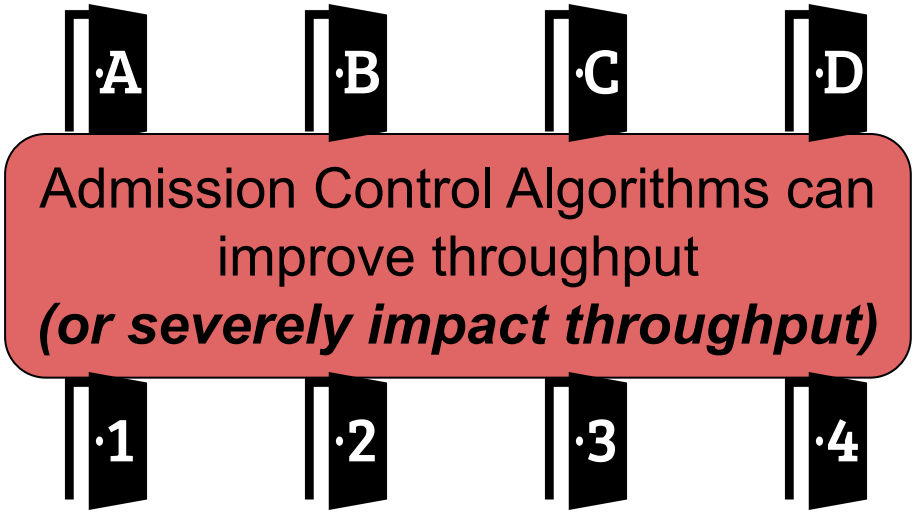


Score: 5

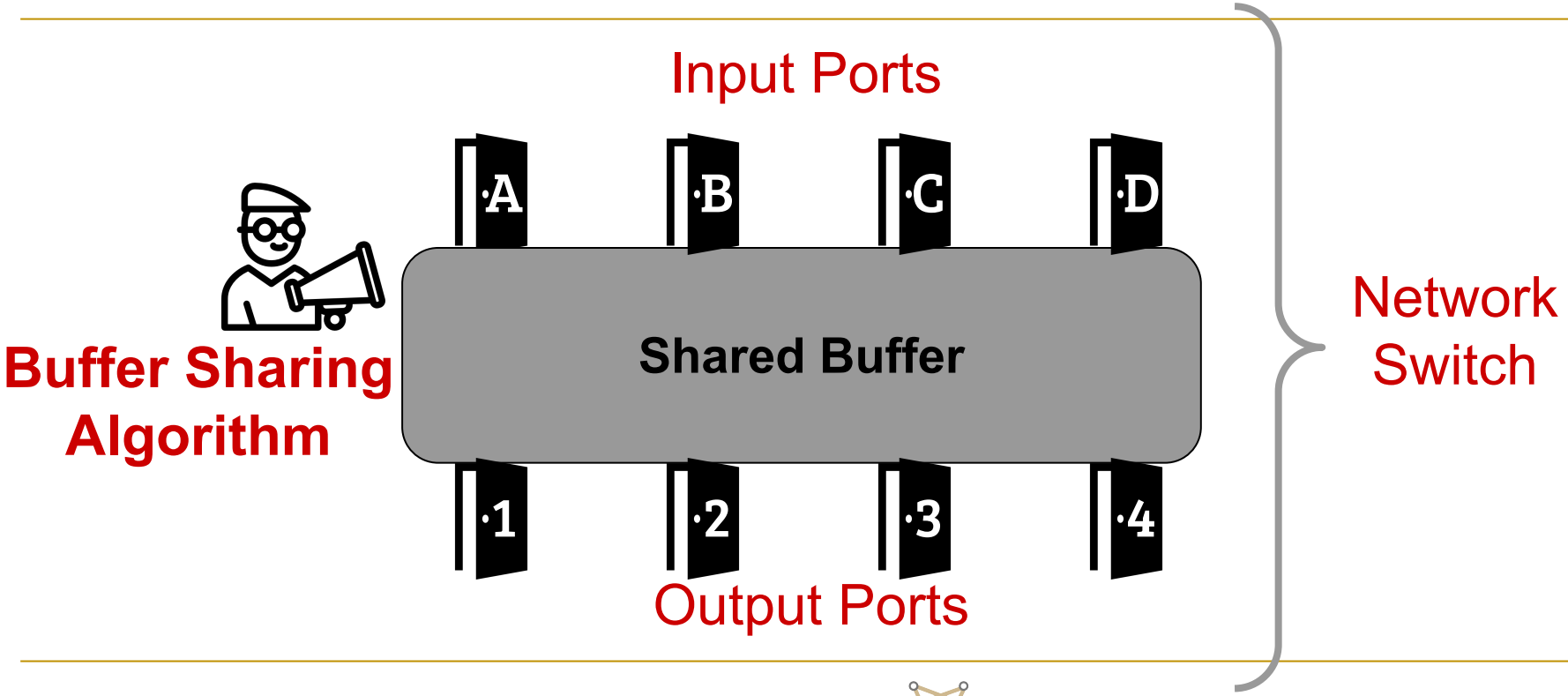
Capacity: 6

PURDUE UNIVERSITY

STyGIANet


A
B
C
D
Admission Control Algorithms can improve throughput
(or severely impact throughput)
1
2
3
4

Buffer Sharing



PURDUE UNIVERSITY

STyGIANet

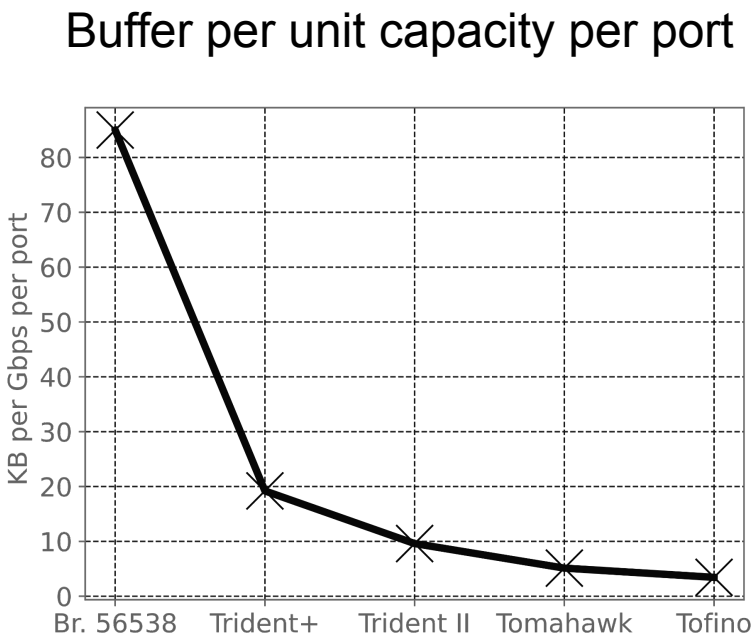
Traditional Datacenter Networks

- TCP-based applications
- Host-networking consumes CPU clock cycles (a lot!!)
- Loss-tolerant traffic

Buffer Sharing: An Emerging Critical Problem

- Traffic is bursty: Requires buffers to avoid packet losses
- Stringent performance requirements eg., Job completion time
  - Requires low latency, high throughput and high burst absorption
- **But** buffer sizes are unable to scale with capacity increase

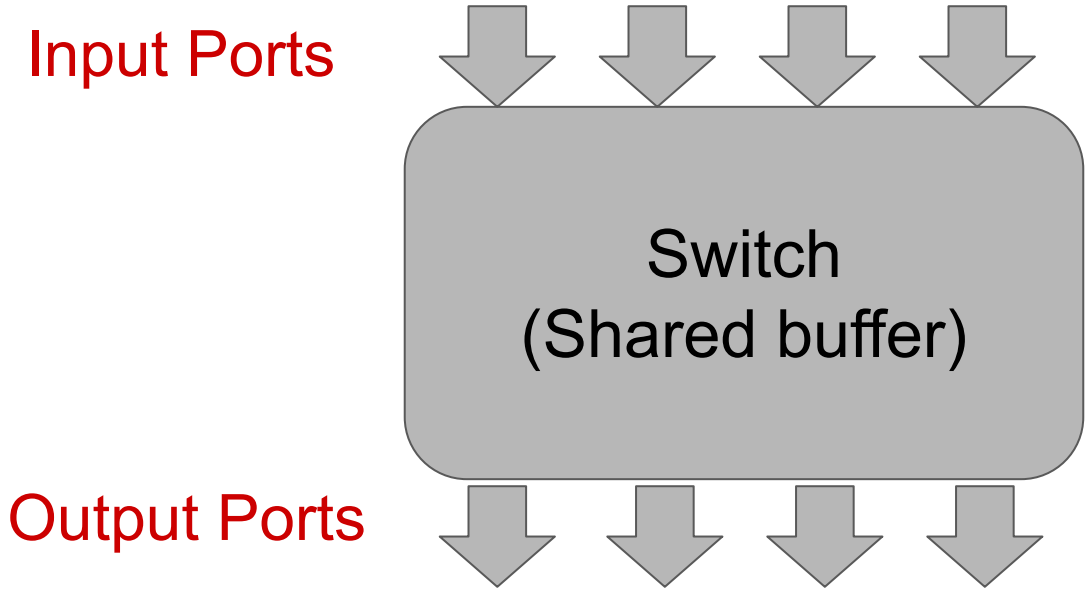
Shrinking Buffer Sizes



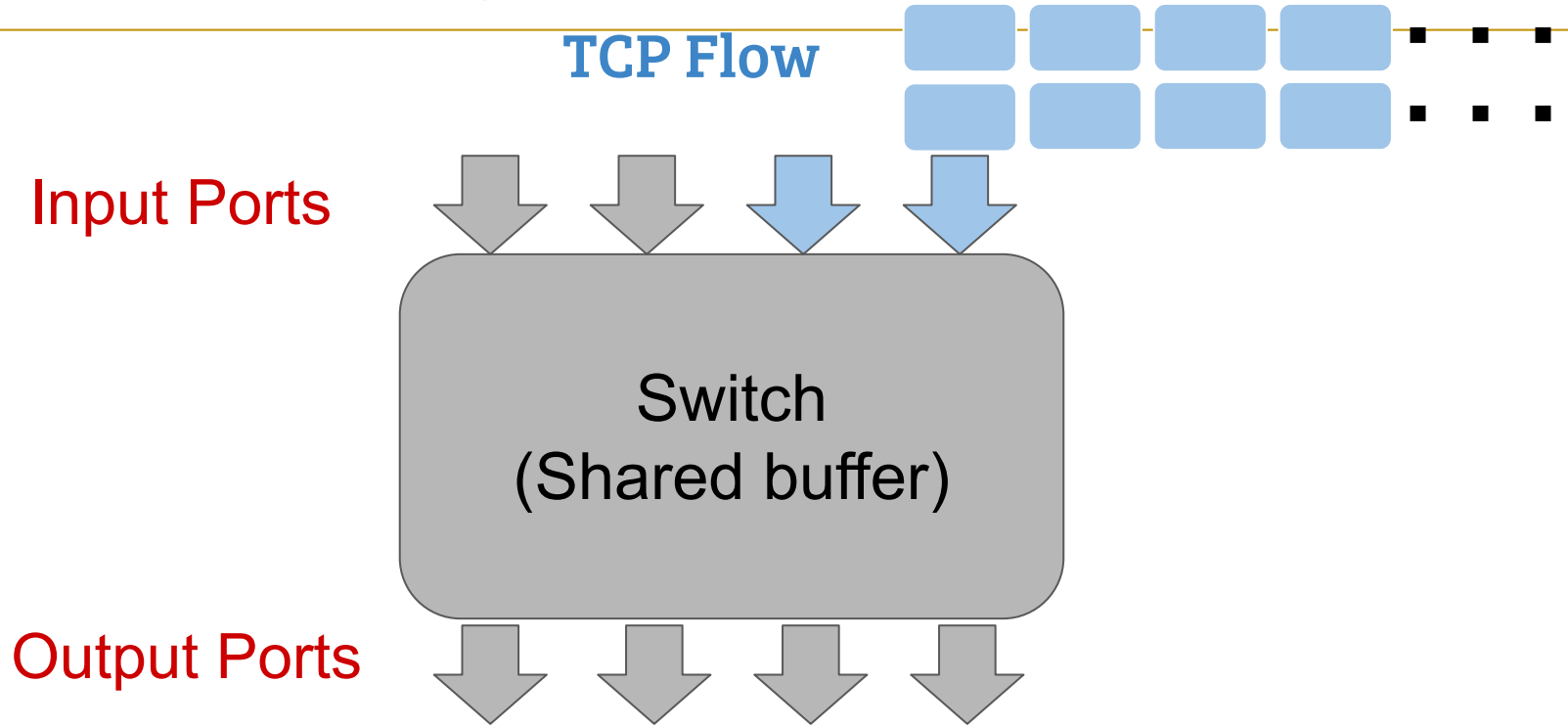
PURDUE UNIVERSITY

STyGIANet

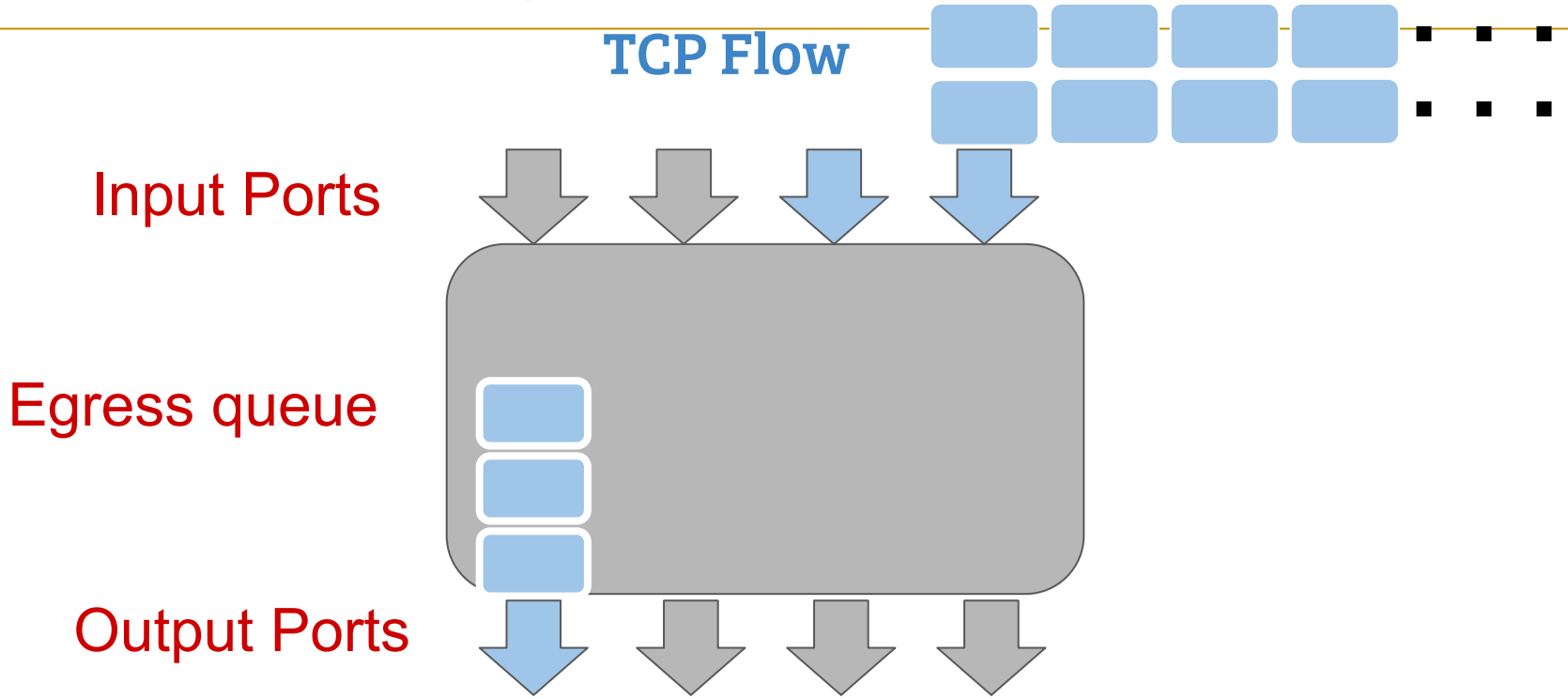
Switch Buffer Sharing with TCP



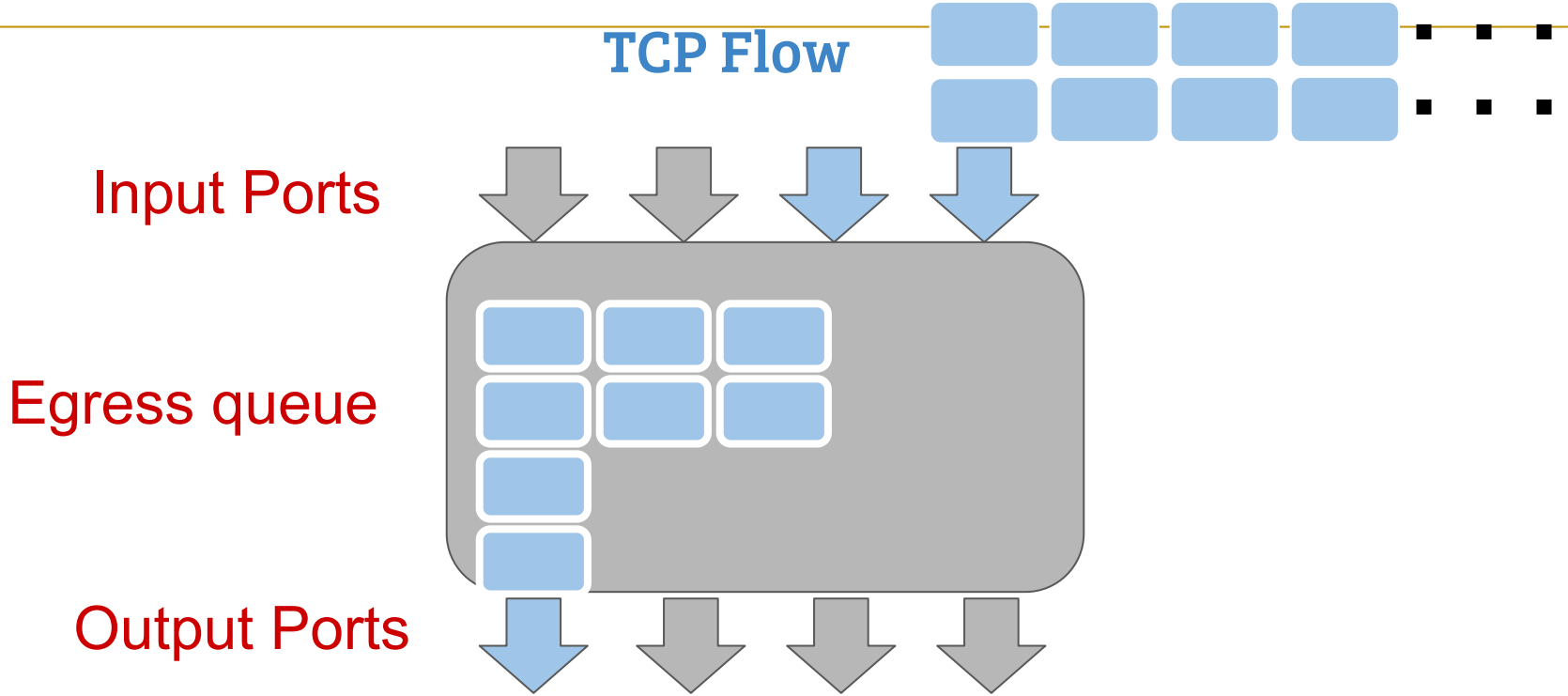
Switch Buffer Sharing with TCP



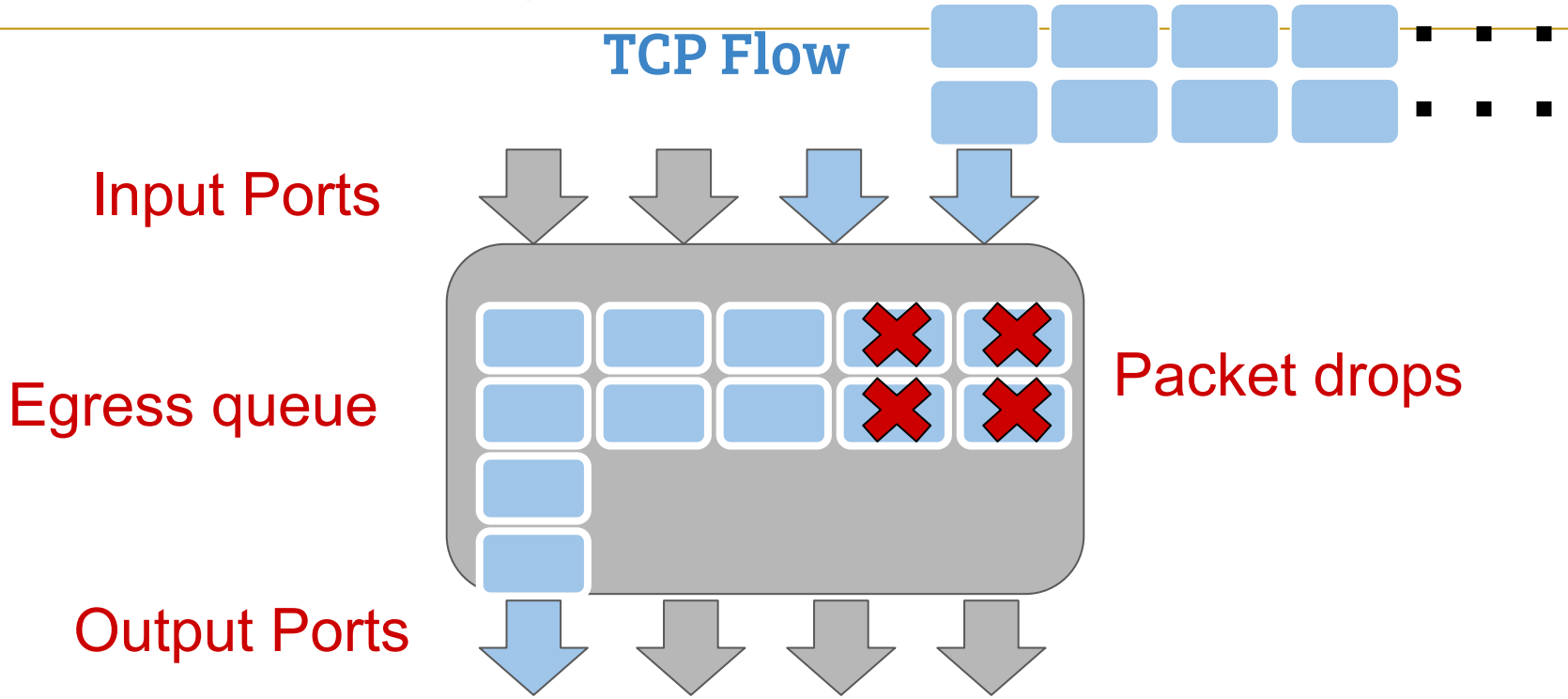
Switch Buffer Sharing with TCP



Switch Buffer Sharing with TCP



Switch Buffer Sharing with TCP



PURDUE UNIVERSITY

STyGIANet

Switch Buffer Sharing with TCP

- Based on egress queue lengths and packet drops
- A buffer sharing algorithm assigns a threshold for each egress queue in a switch
- Packet accepted: Threshold > Queue length (egress)
- Packet dropped: Threshold < Queue length (egress)

Dynamic Thresholds (State-of-the-art)

Threshold = alpha x (Remaining shared buffer)

$$T_p^i(t) = \alpha_p \cdot \underbrace{(B - Q(t))}_{Remaining}$$

Dynamic Thresholds (State-of-the-art)

Threshold = alpha x (Remaining shared buffer)

$$T_p^i(t) = \alpha_p \cdot \underbrace{(B - Q(t))}_{Remaining}$$

Threshold for a queue of priority *p* at port *i*

Dynamic Thresholds (State-of-the-art)

Threshold = alpha x (Remaining shared buffer)

$$T_p^i(t) = \alpha_p \cdot \underbrace{(B - Q(t))}_{Remaining}$$

Configurable parameter

eg.,
config buffer profile update ingress_profile_lossy --pool ingress_pool **--shared-dynamic-alpha 2**

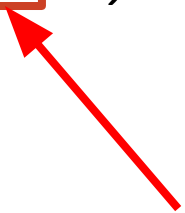
Dynamic Thresholds (State-of-the-art)

Threshold = alpha x (Remaining shared buffer)

$$T_p^i(t) = \alpha_p \cdot \underbrace{(B - Q(t))}_{Remaining}$$

Overall occupancy : (priority p)

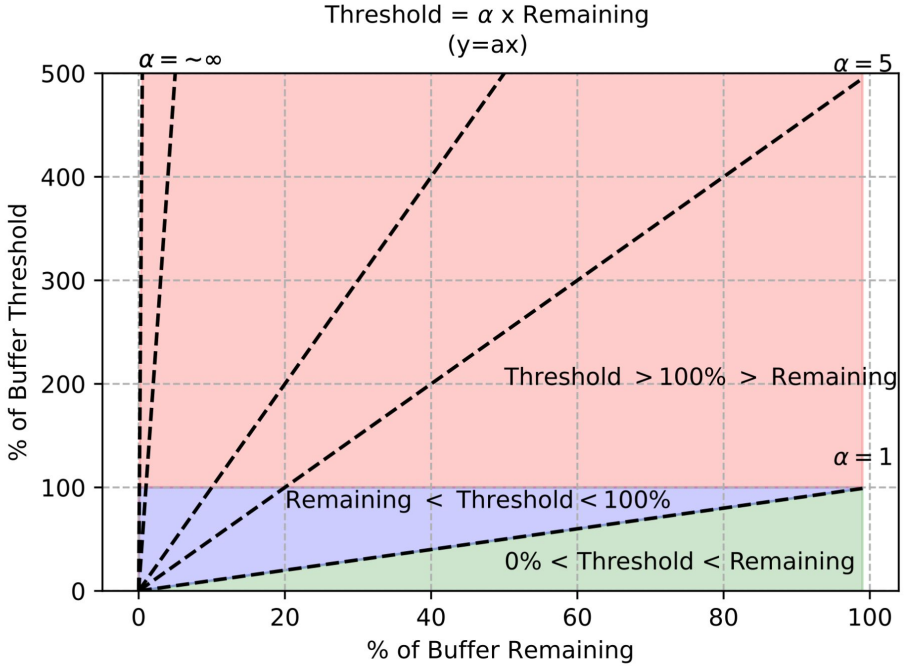
$$Q_p = B \cdot \left(\frac{n_p \alpha_p}{1 + \sum_p n_p \alpha_p} \right)$$



PURDUE UNIVERSITY

STyGIANet

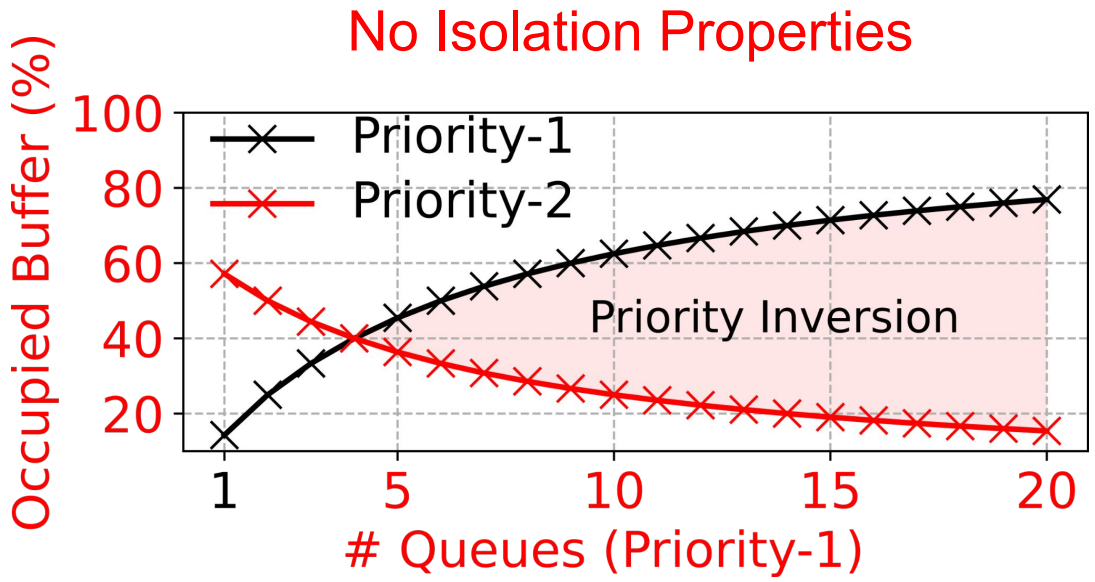
Dynamic Thresholds (State-of-the-art)



PURDUE UNIVERSITY

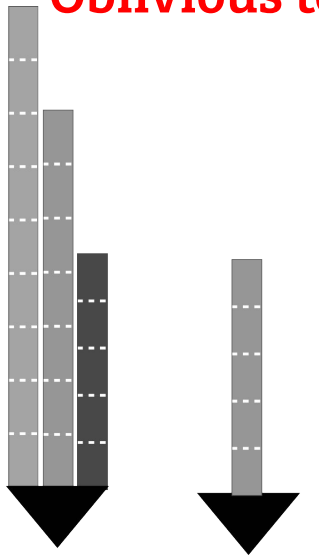
STyGIANet

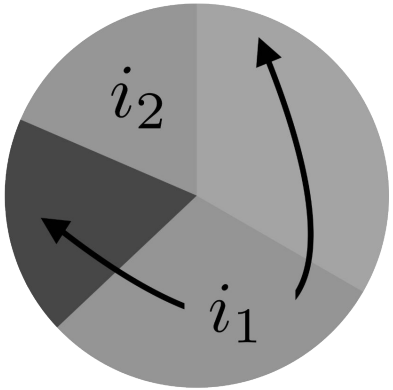
Drawbacks of Dynamic Thresholds



Drawbacks of Dynamic Thresholds

Oblivious to drain rate



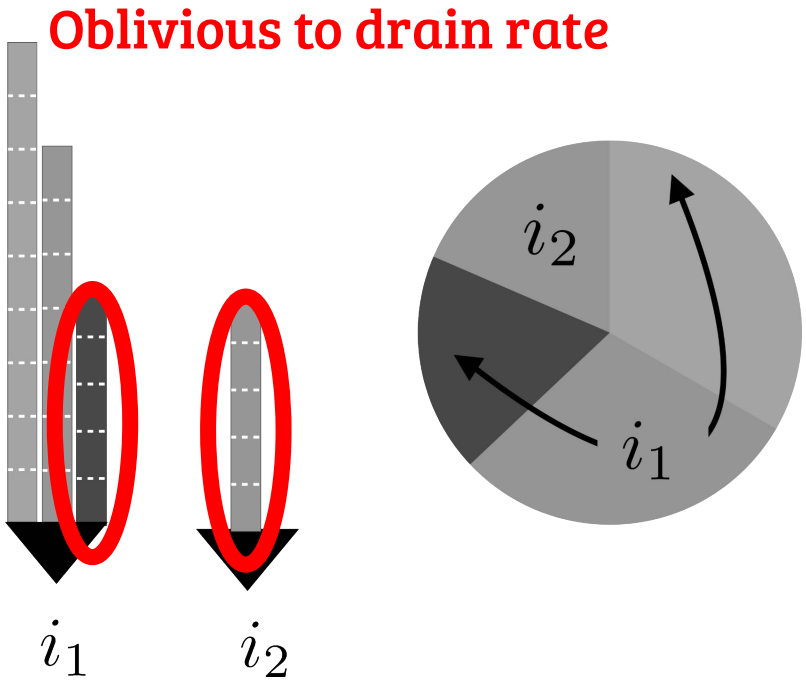


Ports: i_1 i_2

PURDUE UNIVERSITY

STyGIANet

Drawbacks of Dynamic Thresholds



Active Buffer Management in Datacenters

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

Threshold per queue
port i, priority p

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

Parameter

To be set for each priority

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

congested queues of priority p

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

Remaining shared buffer

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

Normalized dequeue rate

ABM: Active Buffer Management

$$T_p^i(t) = \alpha_p \cdot \frac{1}{n_p} \cdot (B - Q(t)) \cdot \frac{\mu_p^i}{b}$$

$$\sum_i \alpha_p \cdot \frac{1}{n_p} \cdot \mu_p^i \leq \alpha_p$$

Properties of ABM

- Upper bounds the buffer allocated to a priority **(Prevents monopoly)**

$$B_p^{max} \leq \frac{B \cdot \alpha_p}{1+\alpha_p}$$

Depends only on the parameter set for the corresponding priority

Properties of ABM

- Lower bounds the buffer allocated to a priority (Minimum buffer guarantee)

$$B_p^{min} \geq \frac{B \cdot \alpha_p}{1 + \sum_{p \in \mathcal{P}} \alpha_p}$$

Depends only on the parameter set for all priorities

Properties of ABM

- Upper bounds the drain time for each priority (Bounded queuing delays)

$$\Gamma \leq \frac{B \cdot \alpha_p}{(1+\alpha_p) \cdot b}$$

Depends only on the parameter set for the corresponding priority and the port bandwidth

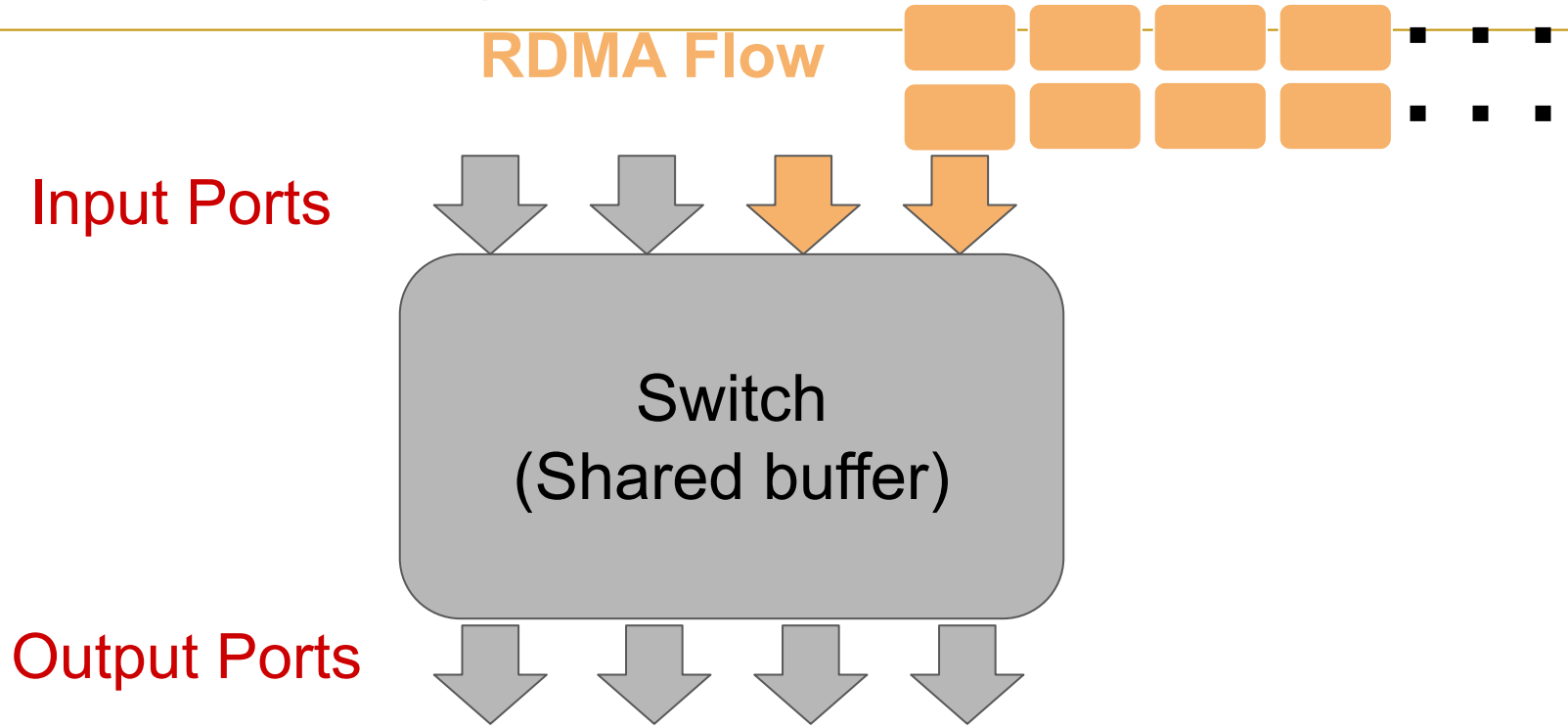
Exercise on Buffer Sharing with TCP

- Simulate datacenter workloads
  - Measure Flow completion times
  - Compare Dynamic Thresholds and ABM
  - Source code: https://github.com/inet-tub/ns3-datacenter

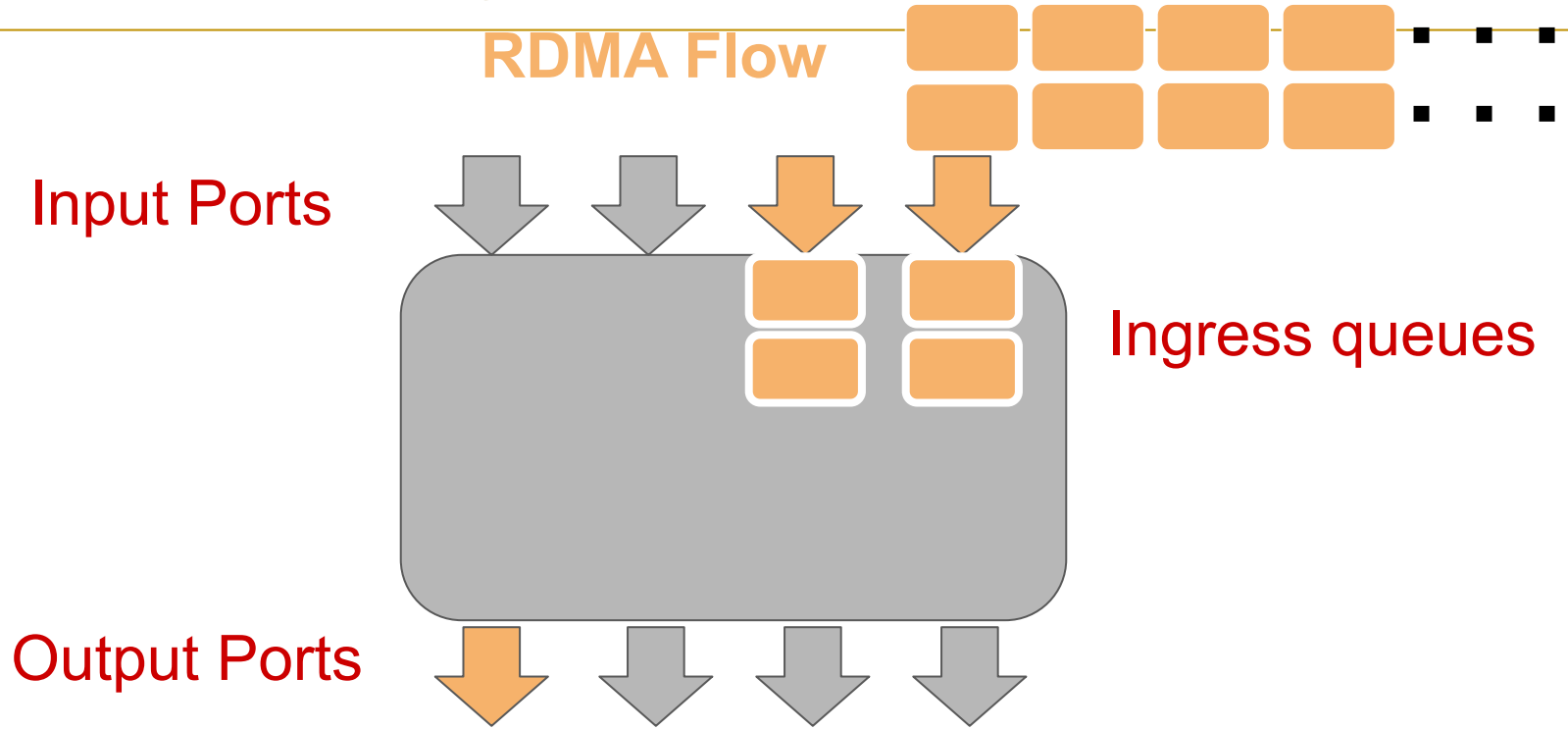
Modern Datacenter Networks

- ~~TCP-based applications~~
  - RDMA-based applications
- ~~Host-networking consumes CPU clock cycles (a lot!!)~~
  - Host networking is offloaded to the NIC
  - NIC implements the entire networking stack
- ~~Loss-tolerant traffic~~
  - Lossless traffic
  - Requires Priority Flow Control (PFC)

Switch Buffer Sharing with RDMA



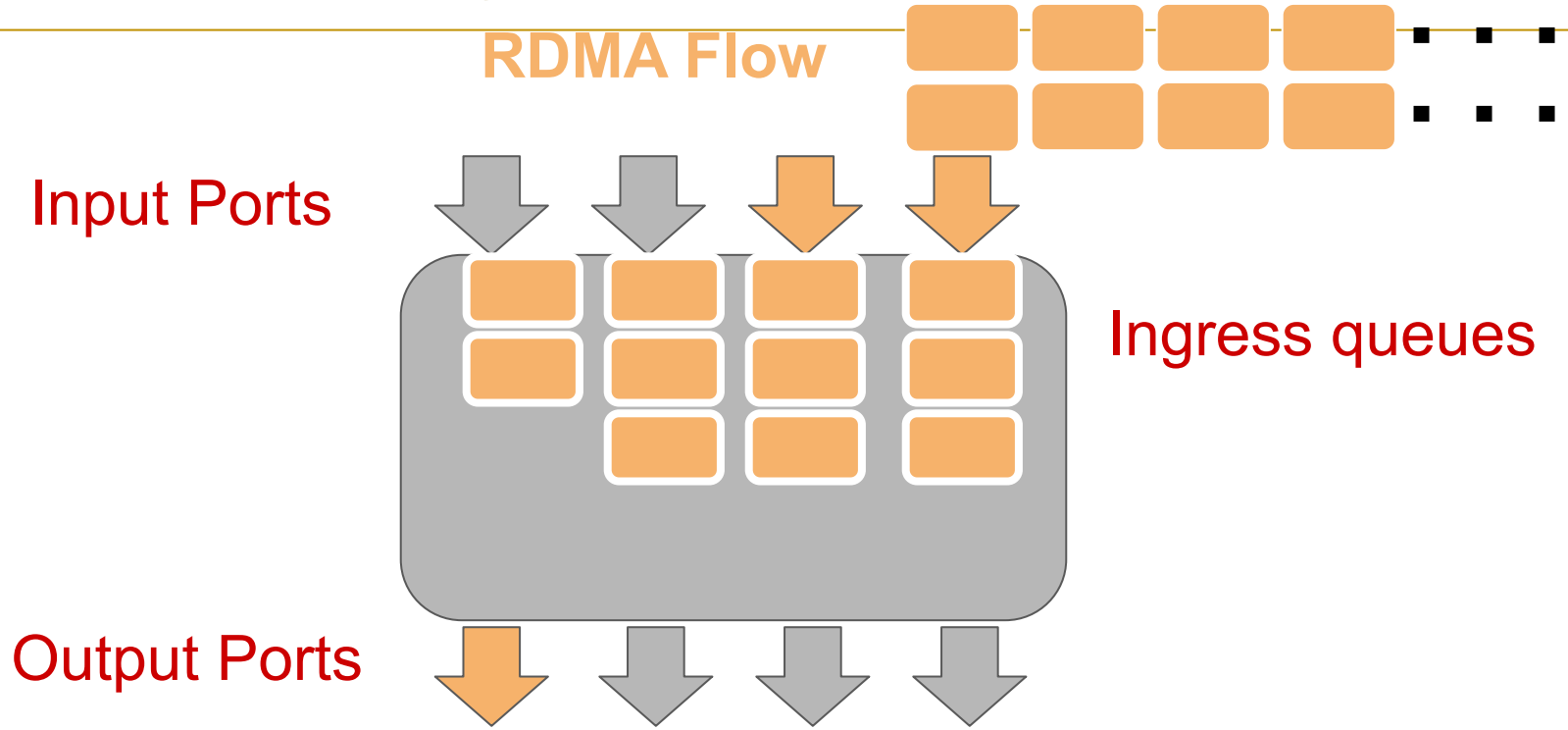
Switch Buffer Sharing with RDMA



PURDUE UNIVERSITY

STyGIANet

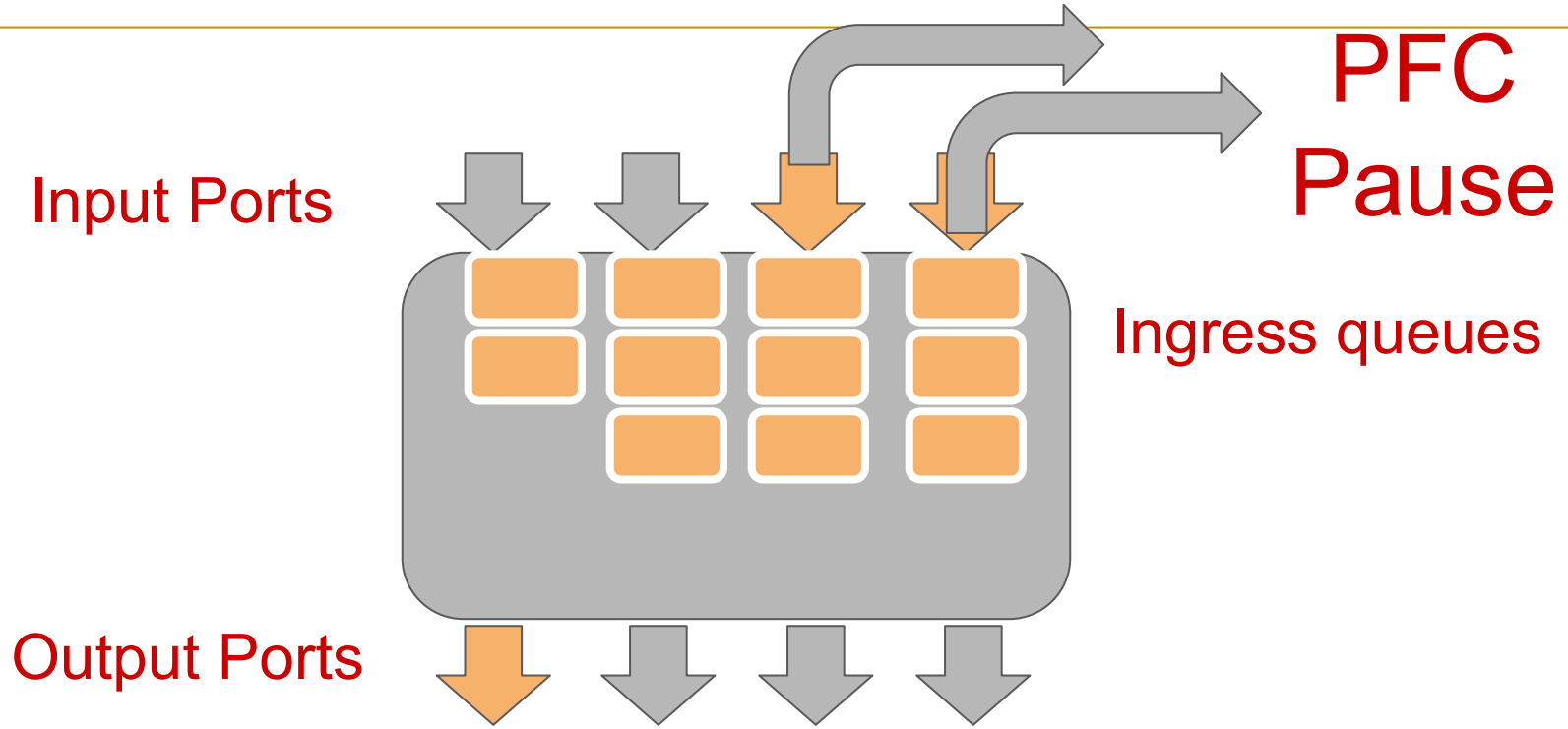
Switch Buffer Sharing with RDMA



PURDUE UNIVERSITY

STyGIANet

Switch Buffer Sharing with RDMA

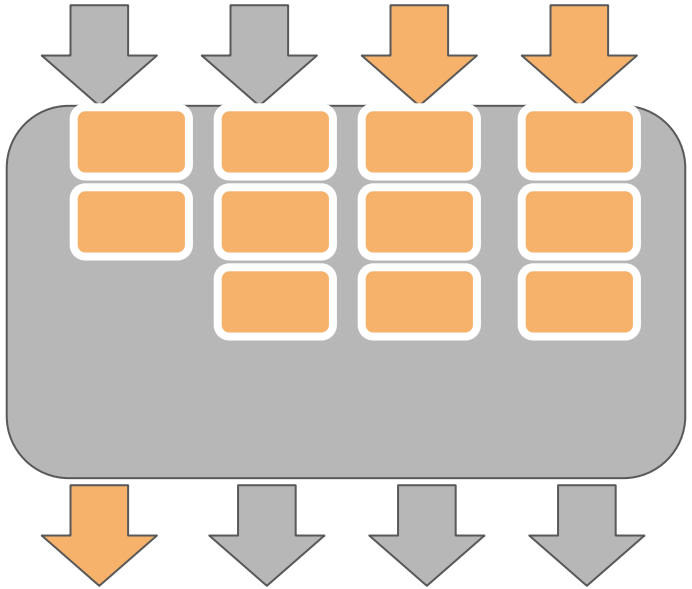


PURDUE UNIVERSITY

STyGIANet

Switch Buffer Sharing with RDMA

Input Ports



Ingress queues

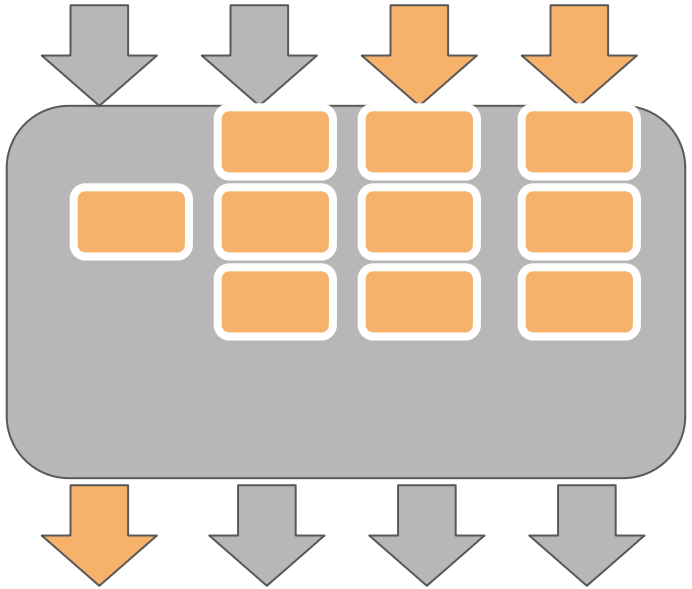
Output Ports

PURDUE UNIVERSITY

STyGIANet

Switch Buffer Sharing with RDMA

Input Ports



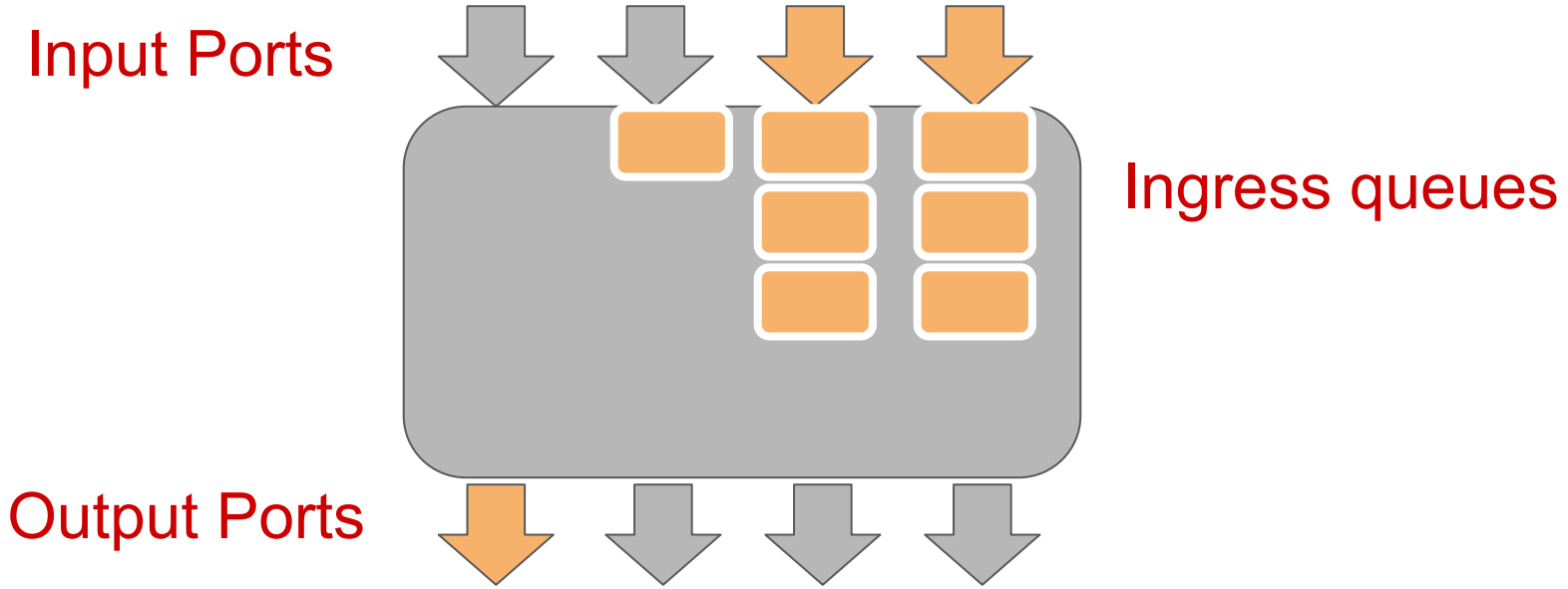
Ingress queues

Output Ports

PURDUE UNIVERSITY

STyGIANet

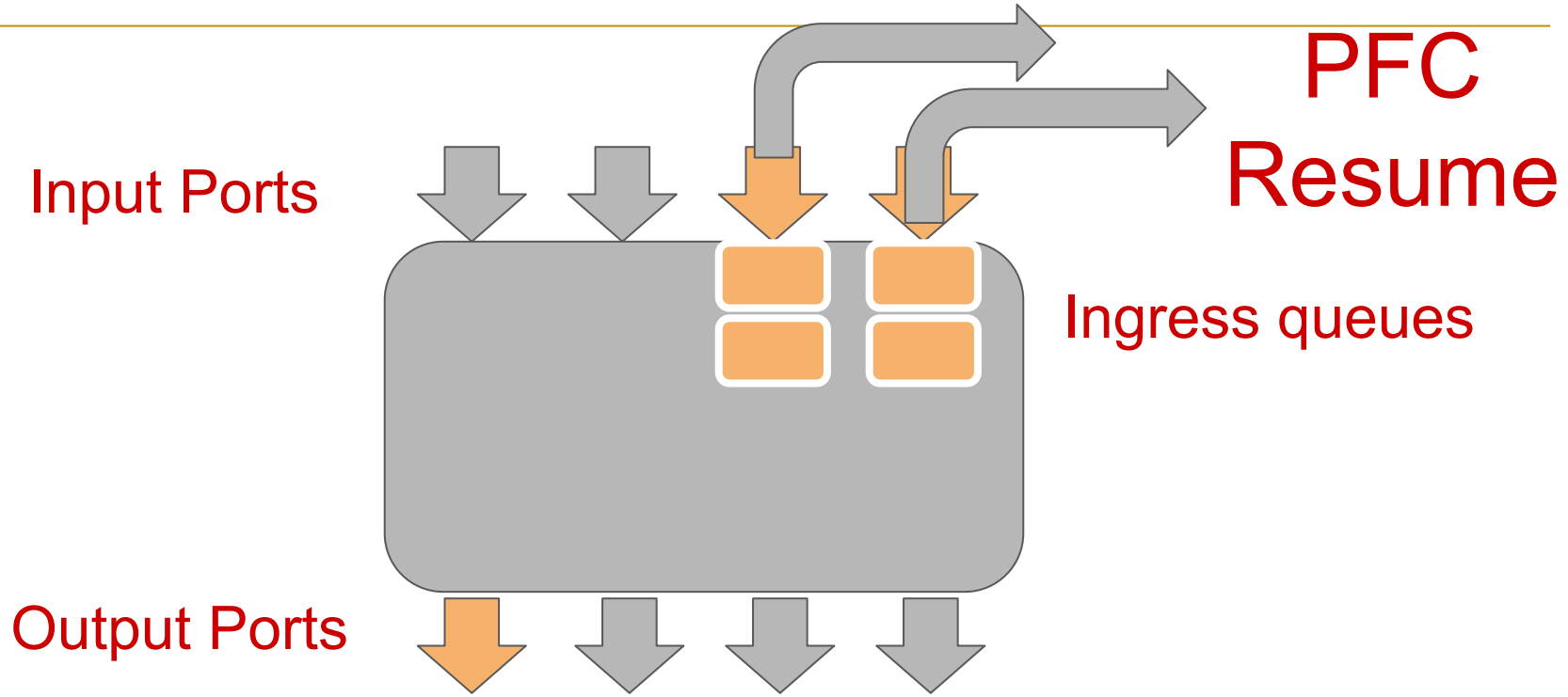
Switch Buffer Sharing with RDMA



PURDUE UNIVERSITY

STyGIANet

Switch Buffer Sharing with RDMA



PURDUE UNIVERSITY

STyGIANet

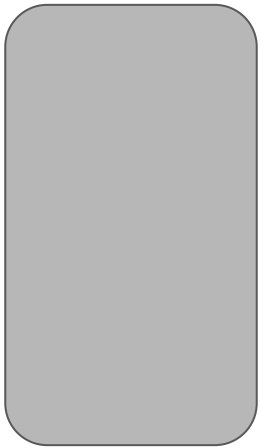
Switch Buffer Sharing with RDMA

- Based on ingress queue lengths and PFC
- A buffer sharing algorithm assigns a threshold for each ingress queue in a switch
- Packets are always accepted
- PFC Pause: Threshold < Queue length (ingress)

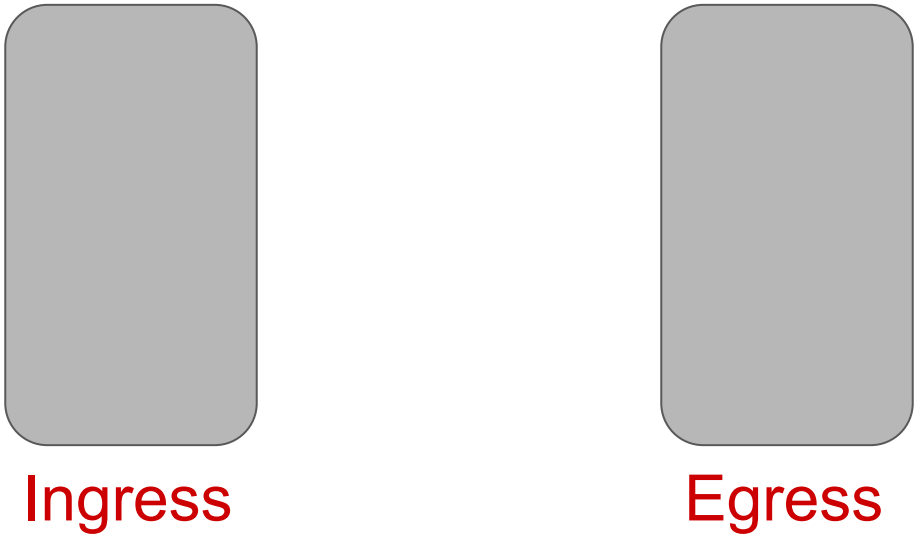
Production Datacenter Networks

- A mix of RDMA and TCP traffic
- Switches use shared buffers
- Both RDMA and TCP *share* the limited buffer space at each switch in the network

SONiC Buffer Model (RDMA + TCP)

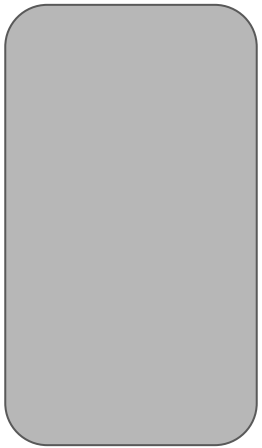


SONiC Buffer Model (RDMA + TCP)

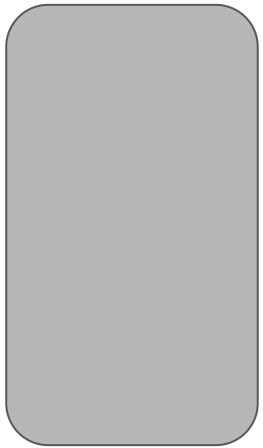


SONiC Buffer Model (RDMA + TCP)

- Every packet is accounted both in ingress and egress



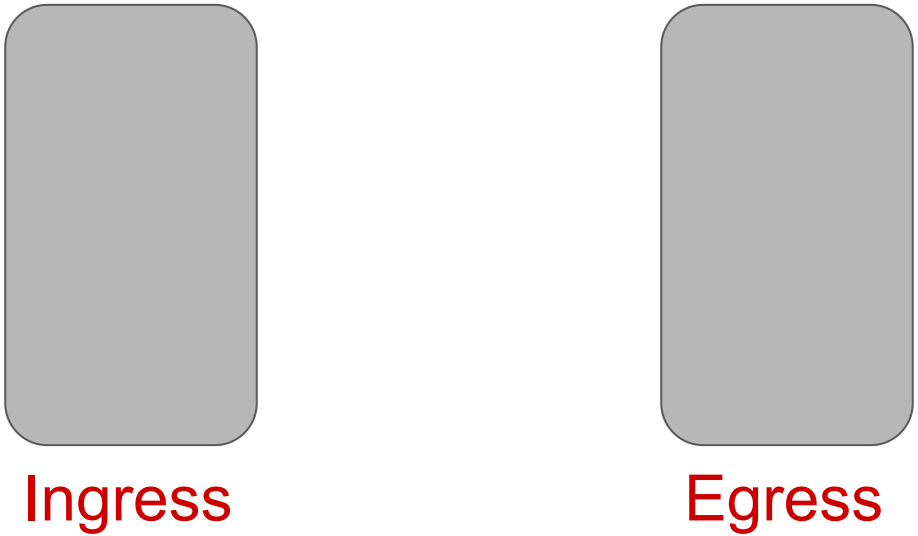
Ingress



Egress

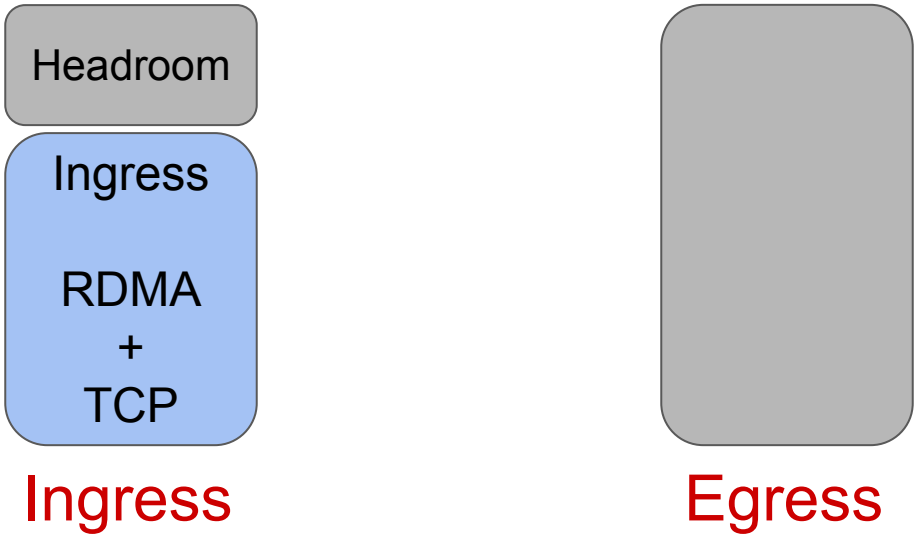
SONiC Buffer Model (RDMA + TCP)

- Buffer is logically divided into pools



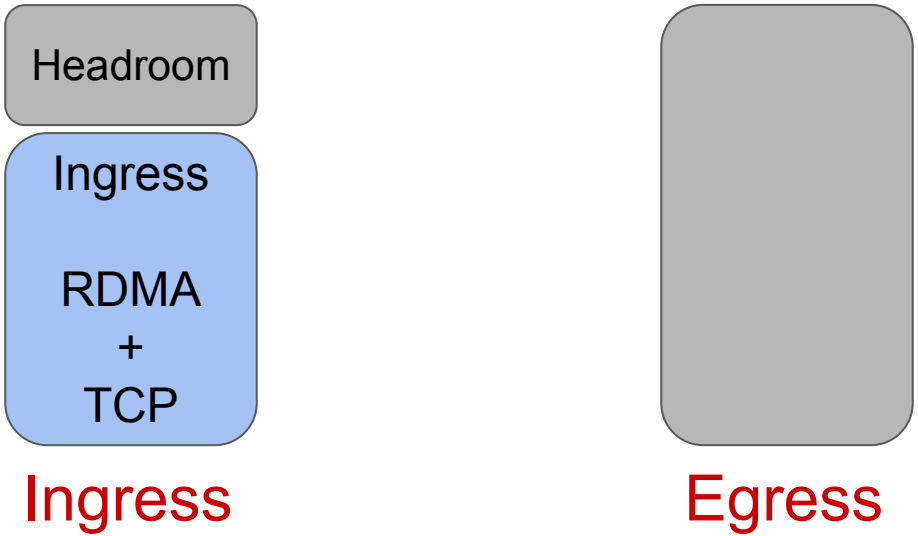
SONiC Buffer Model (RDMA + TCP)

- Ingress pool is shared by both RDMA and TCP



SONiC Buffer Model (RDMA + TCP)

- Headroom pool in the ingress is reserved for RDMA

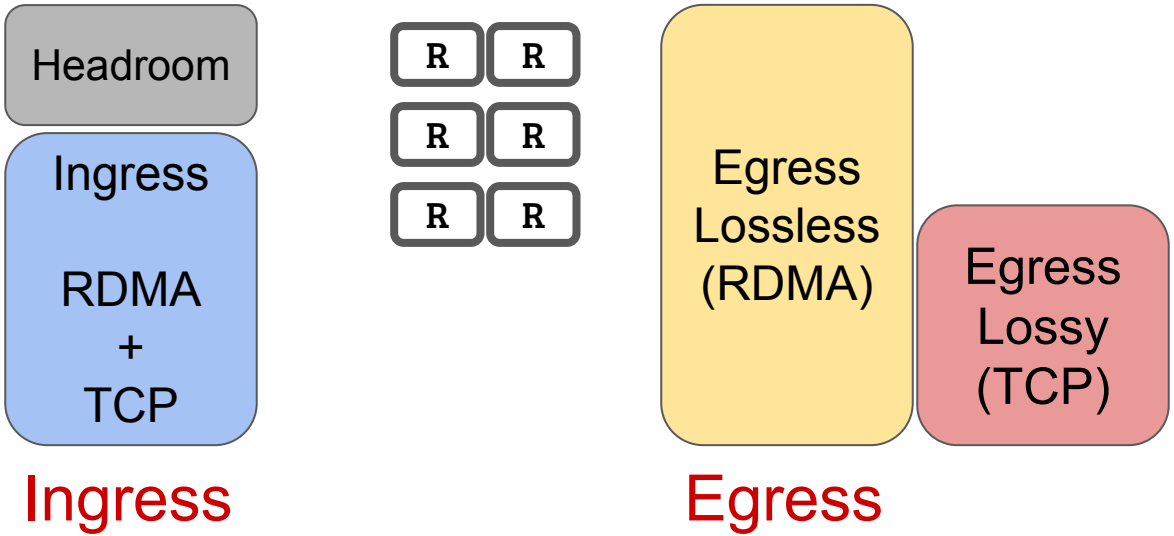


SONiC Buffer Model (RDMA + TCP)

- Egress lossless (RDMA) and Egress lossy (TCP) pools

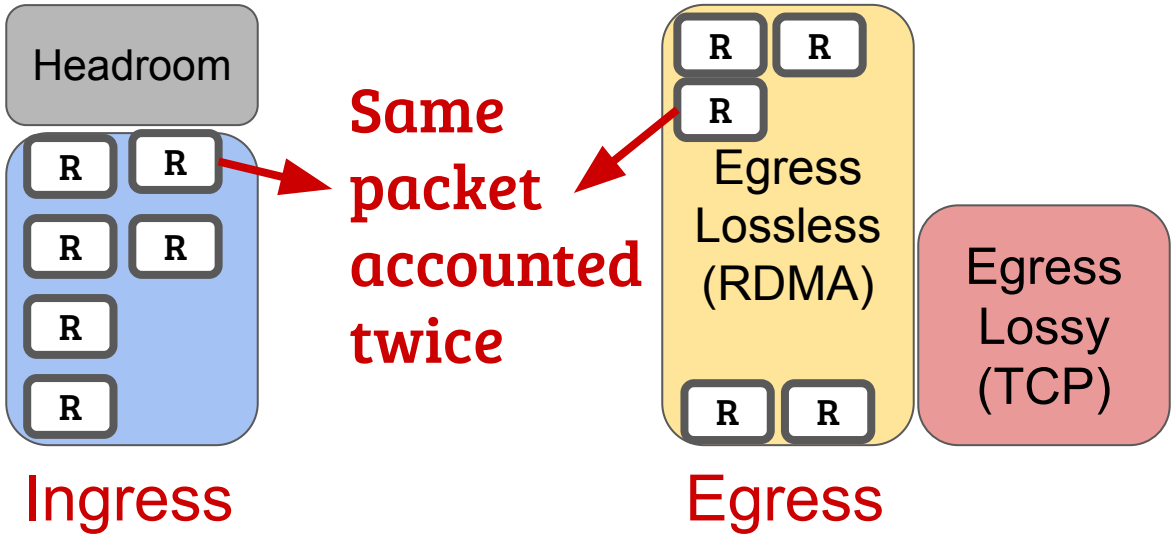
SONiC Buffer Model (RDMA + TCP)

- Example: RDMA packets



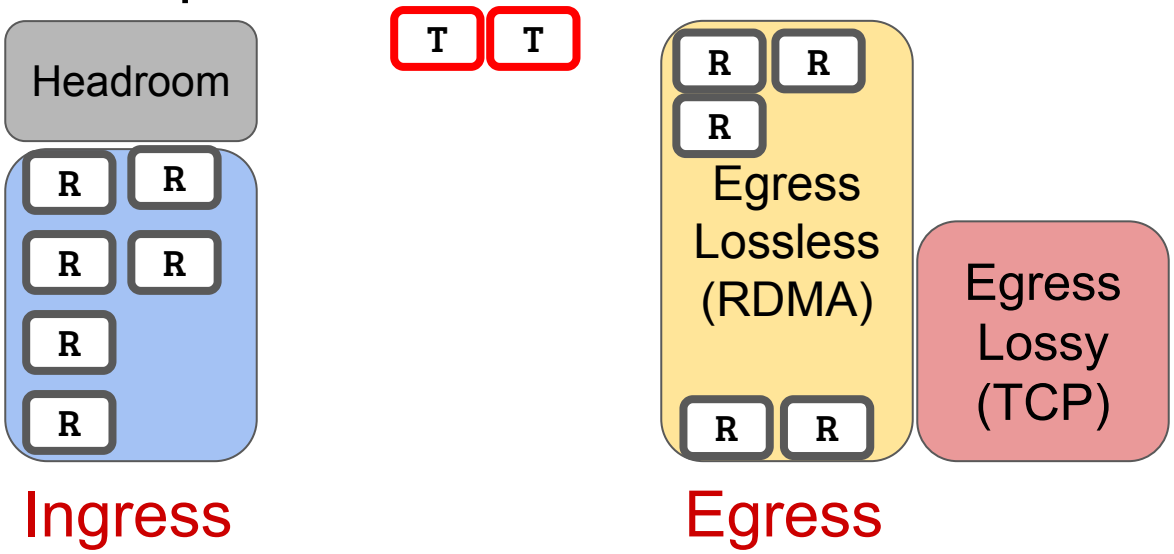
SONiC Buffer Model (RDMA + TCP)

- Egress lossless (RDMA) and Egress lossy (TCP) pools



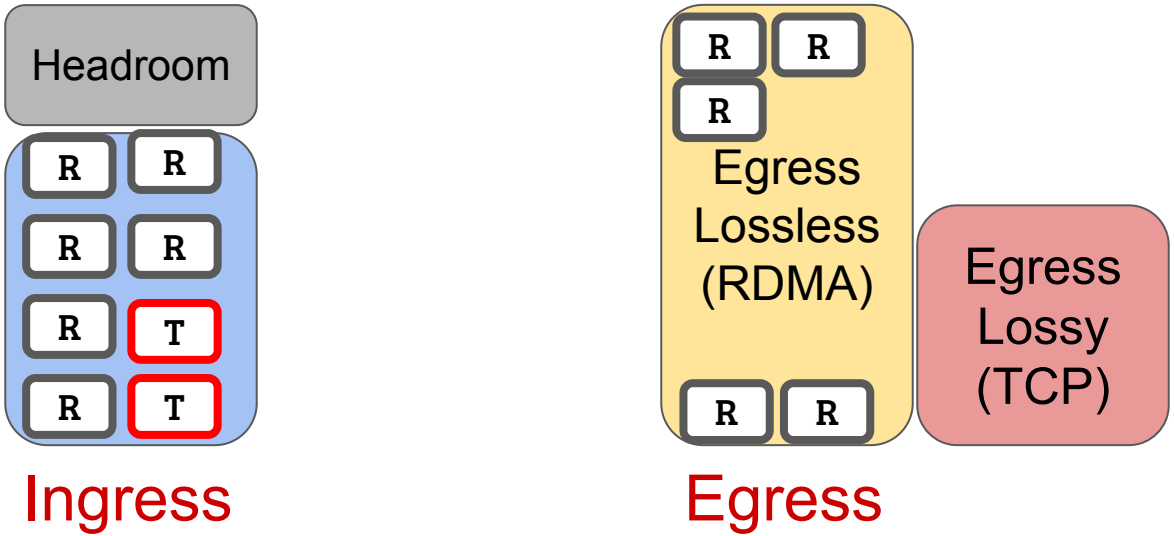
SONiC Buffer Model (RDMA + TCP)

- Example: TCP packets



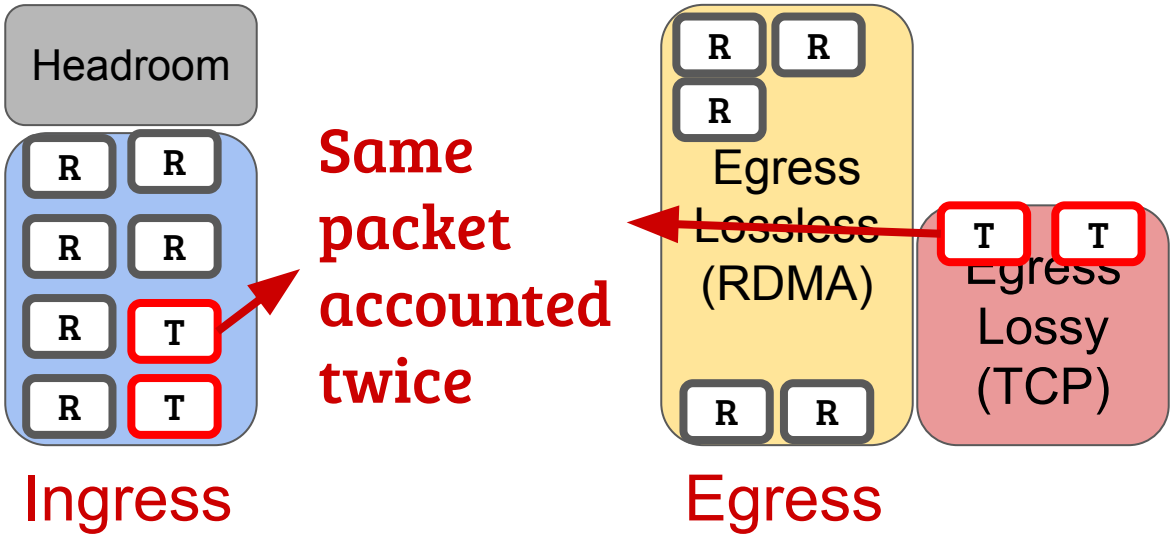
SONiC Buffer Model (RDMA + TCP)

- Example: TCP packets



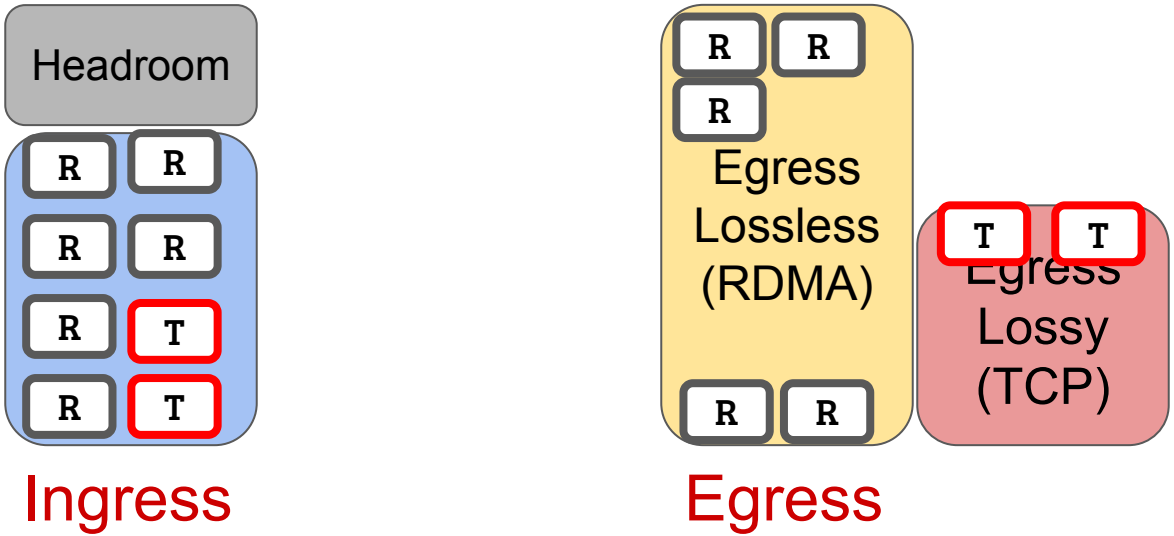
SONiC Buffer Model (RDMA + TCP)

- Example: TCP packets



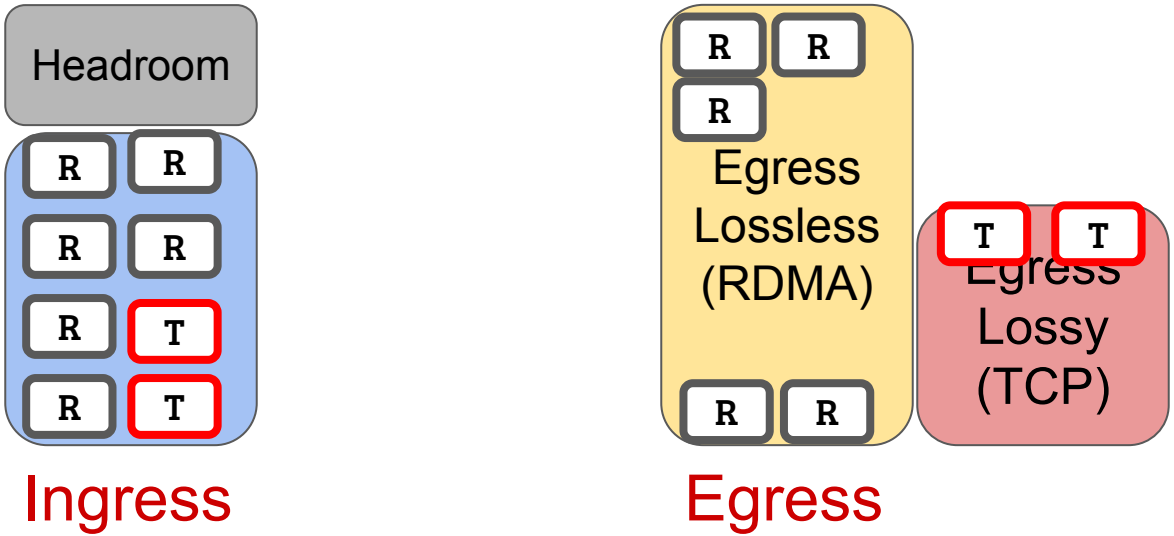
SONiC Buffer Model (RDMA + TCP)

- Buffer Sharing: Dynamic Thresholds in each pool

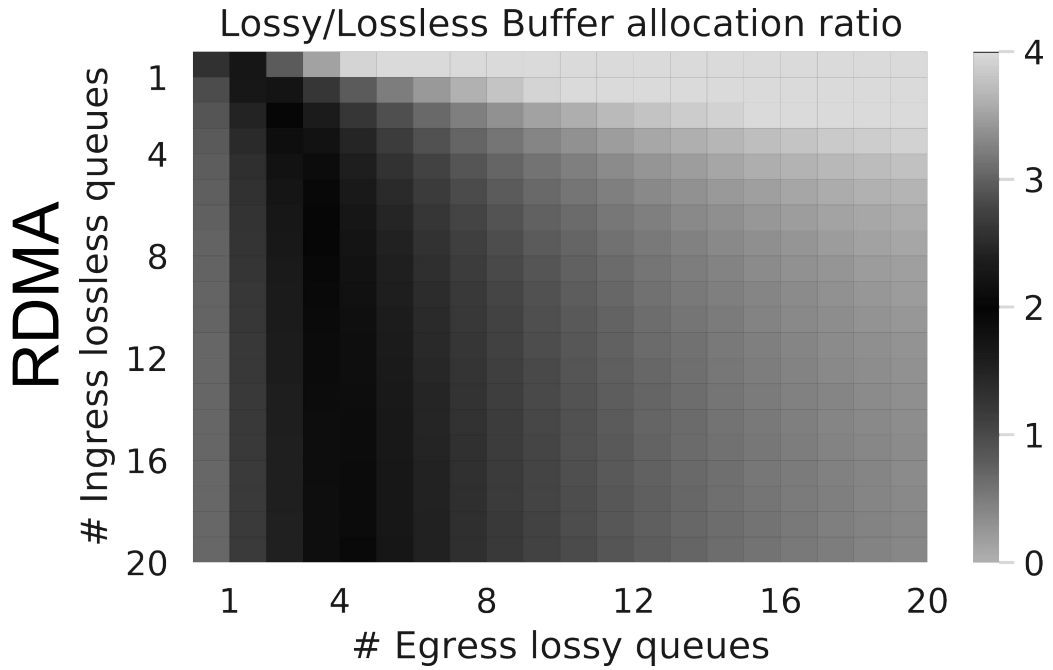


SONiC Buffer Model (RDMA + TCP)

- Buffer Sharing: $\alpha \times$ Remaining pool size



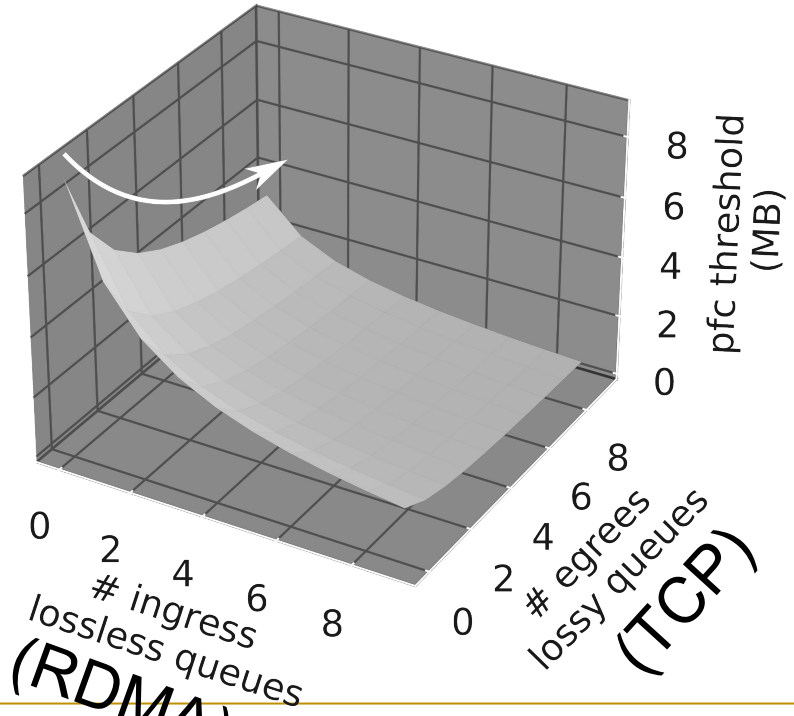
Drawbacks of SONiC Buffer Model



PURDUE UNIVERSITY

STyGIANet

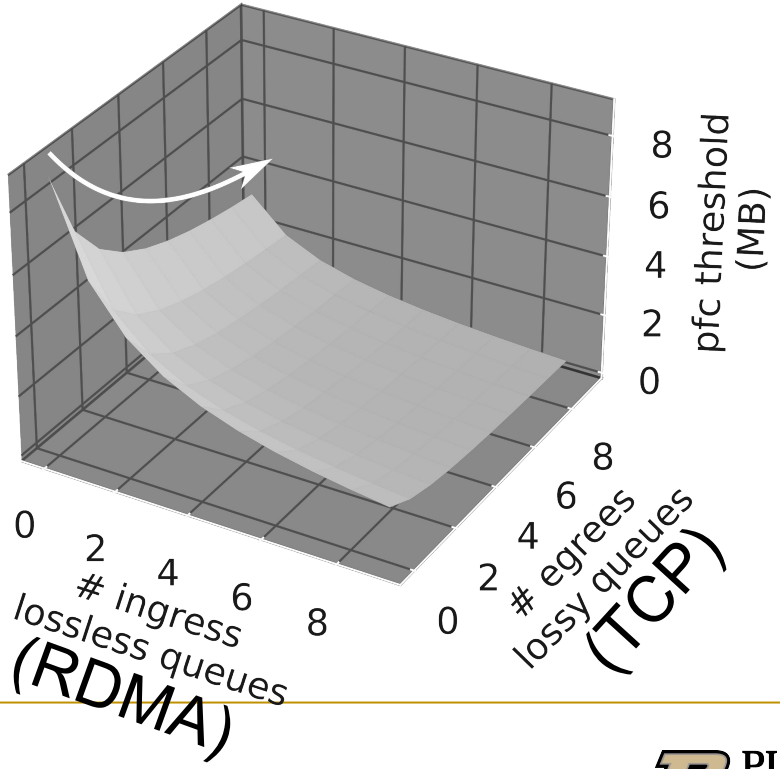
Drawbacks of SONiC Buffer Model

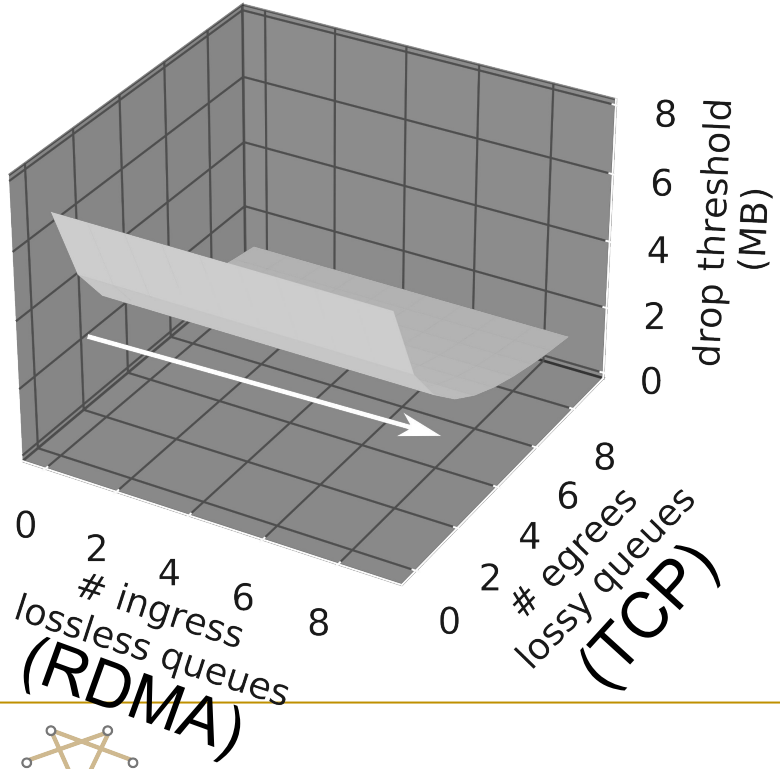


PURDUE UNIVERSITY

STyGIANet

Drawbacks of SONiC Buffer Model

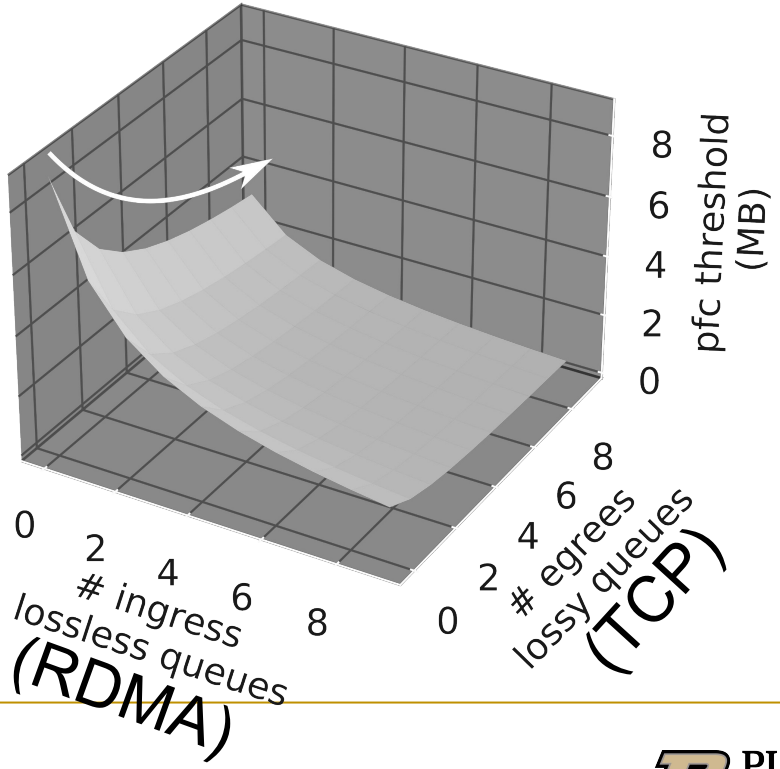


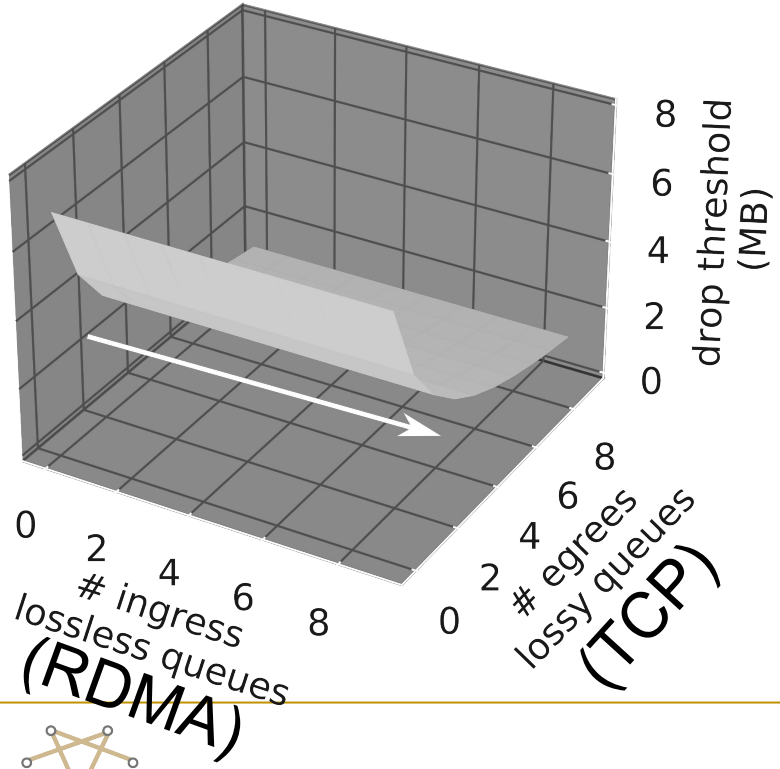


PURDUE UNIVERSITY

STyGIANet

Drawbacks of SONiC Buffer Model

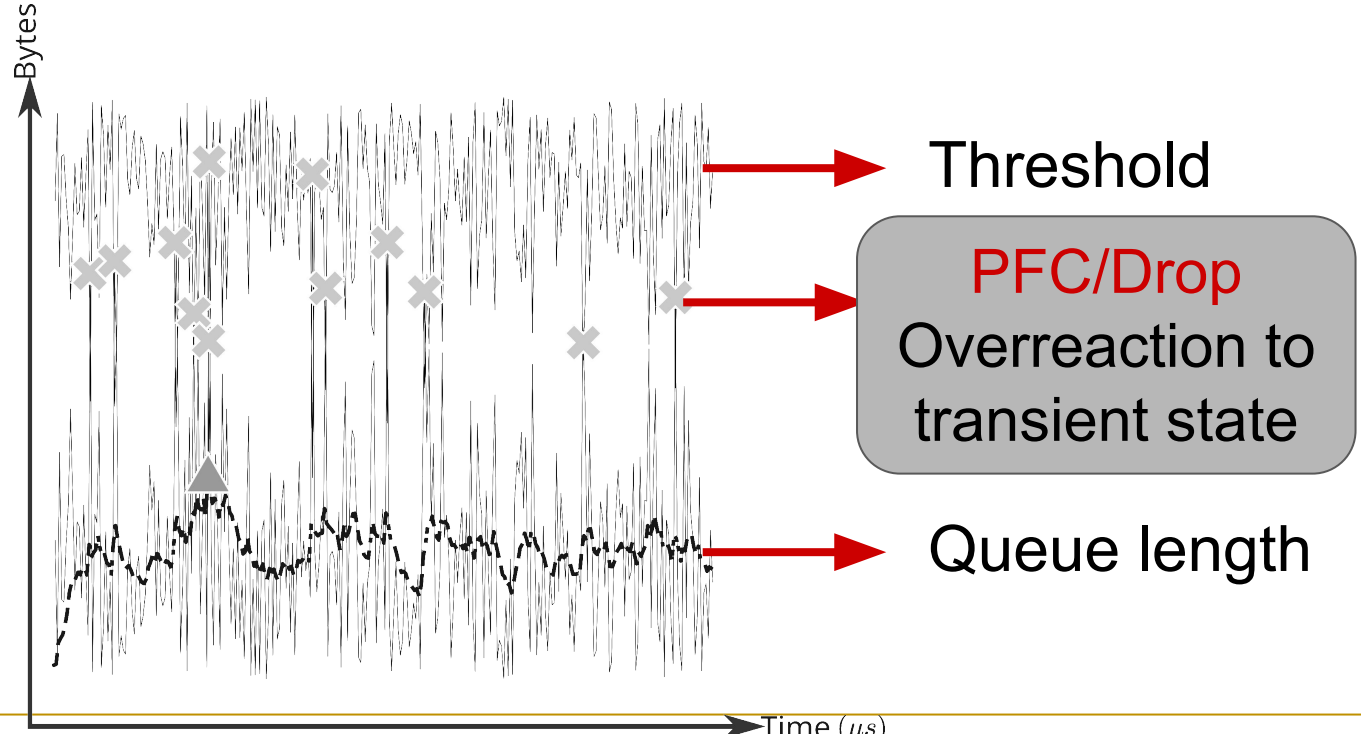




PURDUE UNIVERSITY

STyGIANet

Drawbacks of SONiC Buffer Model



PURDUE UNIVERSITY

STyGIANet

Can we isolate RDMA and TCP while improving burst absorption?

REVERIE

Low Pass Filter-Based Switch Buffer Sharing for Datacenters with RDMA and TCP Traffic

PURDUE UNIVERSITY

STyGIANet

Reverie

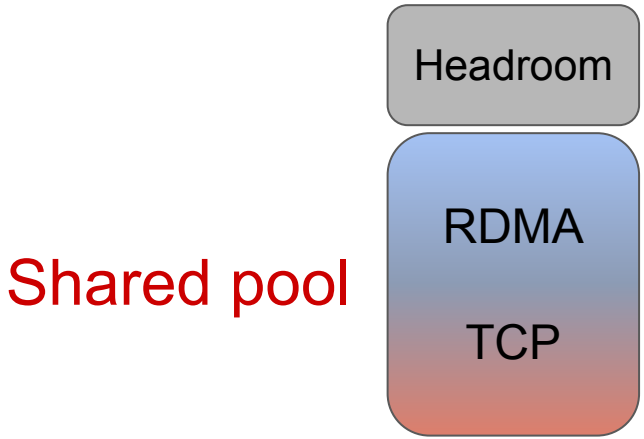
- Achieves isolation across RDMA and TCP
- Improves burst absorption

Reverie

- Single shared buffer pool for RDMA and TCP

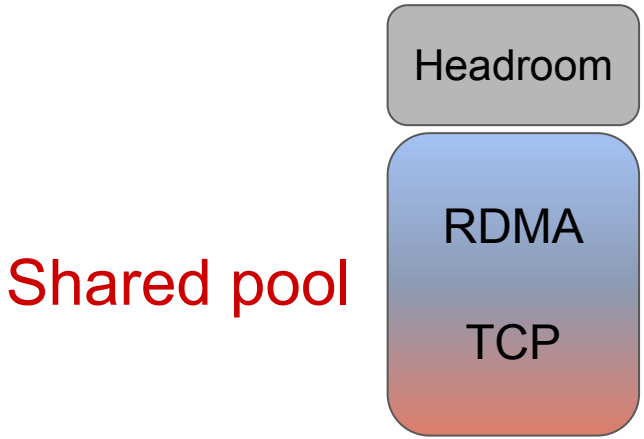
Reverie

- Single shared buffer pool for RDMA and TCP



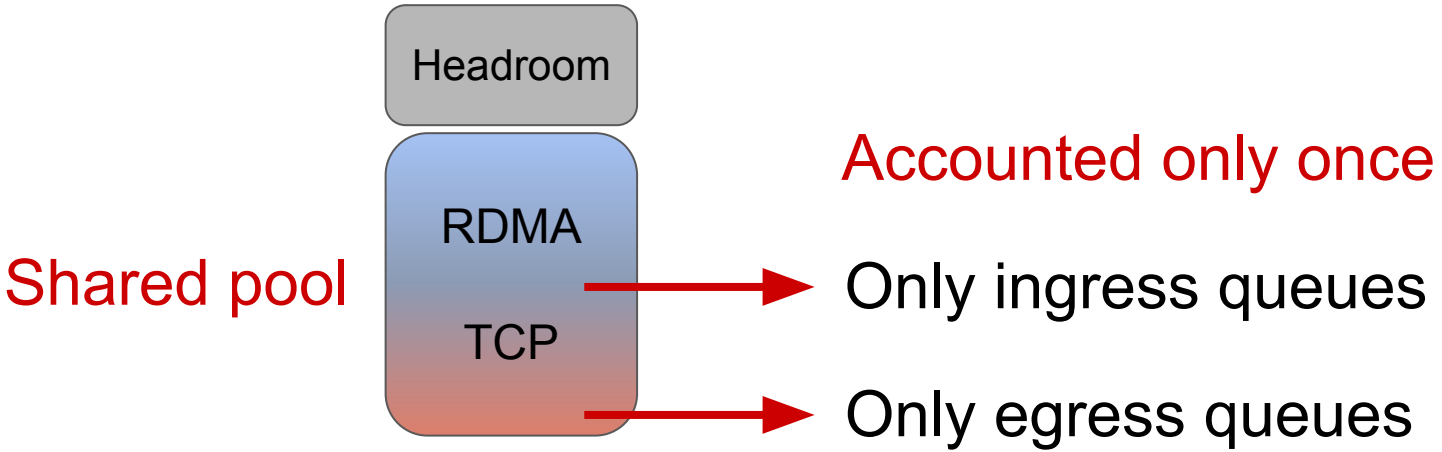
Reverie

- Consolidated ingress and egress buffer views
  - Birds-eye view of the buffer



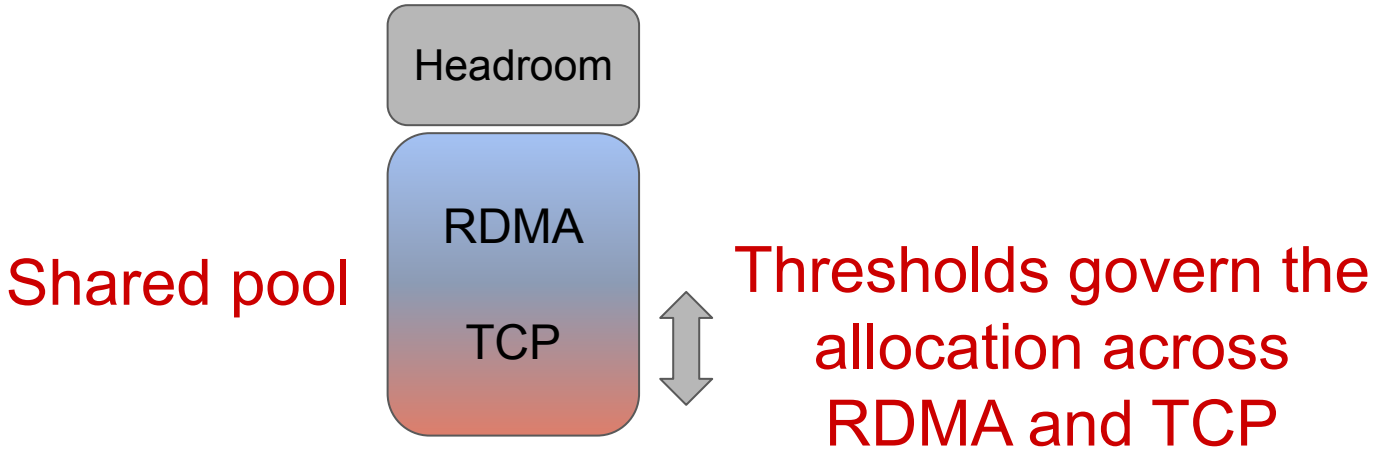
Reverie

- Consolidated ingress and egress buffer views
  - Birds-eye view of the buffer



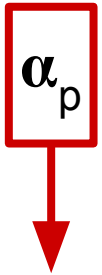
Reverie

- Consolidated ingress and egress buffer views
  - Birds-eye view of the buffer



Reverie

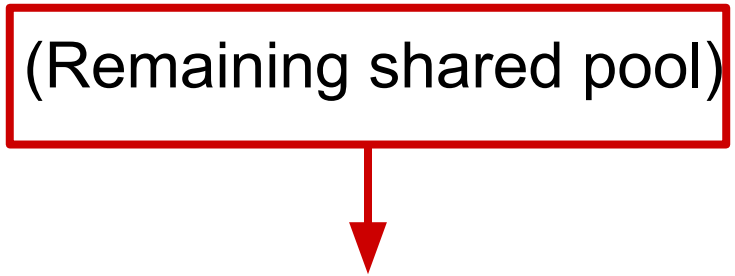
- Threshold: α_p □ (Remaining shared pool) □ $\frac{1}{n_p}$



Configurable parameter for each queue
e.g., α_r for RDMA (ingress queues) and
α_t for TCP (egress queues)

Reverie

- Threshold: α_p □ (Remaining shared pool) □ $\frac{1}{n_p}$



Shared pool size — total shared occupancy

Reverie

- Threshold: α_p □ (Remaining shared pool) □ $\frac{1}{n_p}$

Number of congested queues of type p

Reverie

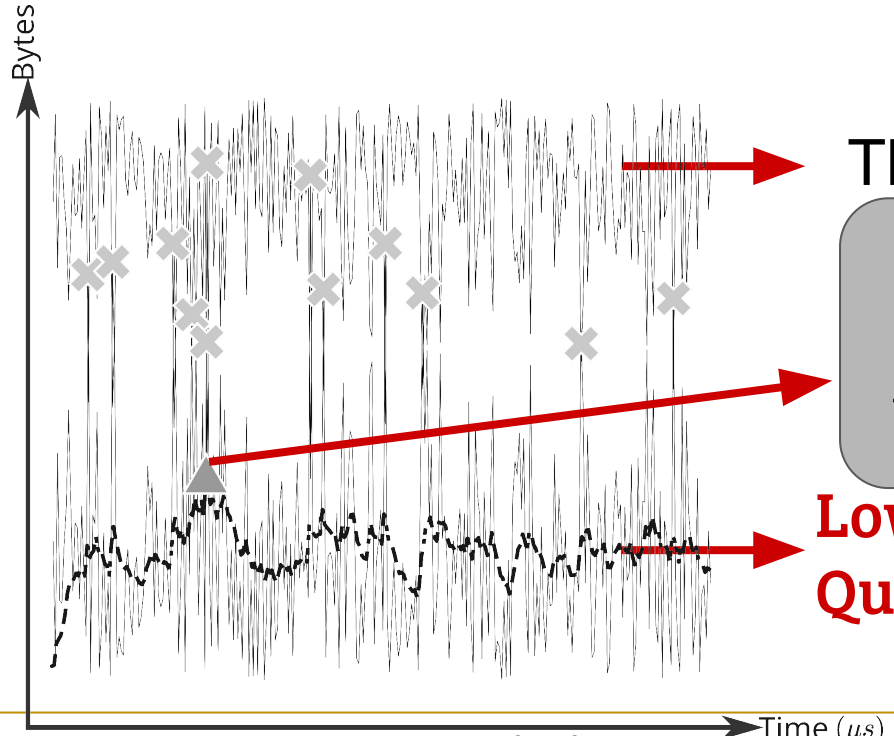
- Threshold: α_p □ (Remaining shared pool) □ $\frac{1}{n_p}$

RDMA vs TCP

Isolation ✅

Fair allocation ✅

Low-pass Filtered Queue length



Threshold

PFC/Drop
Smooth reaction
to transient state

Low-pass filtered Queue length

PURDUE UNIVERSITY

STyGIANet

How can we improve buffer sharing further?

Buffer Sharing (An Online Perspective)

- Goal: Maximize the number of transmitted packets
  - Throughput maximization
- Online algorithm (ALG) takes spontaneous decisions upon every packet arrival
- Offline optimal algorithm (OPT) has prior knowledge of the entire arrival sequence and performs optimally

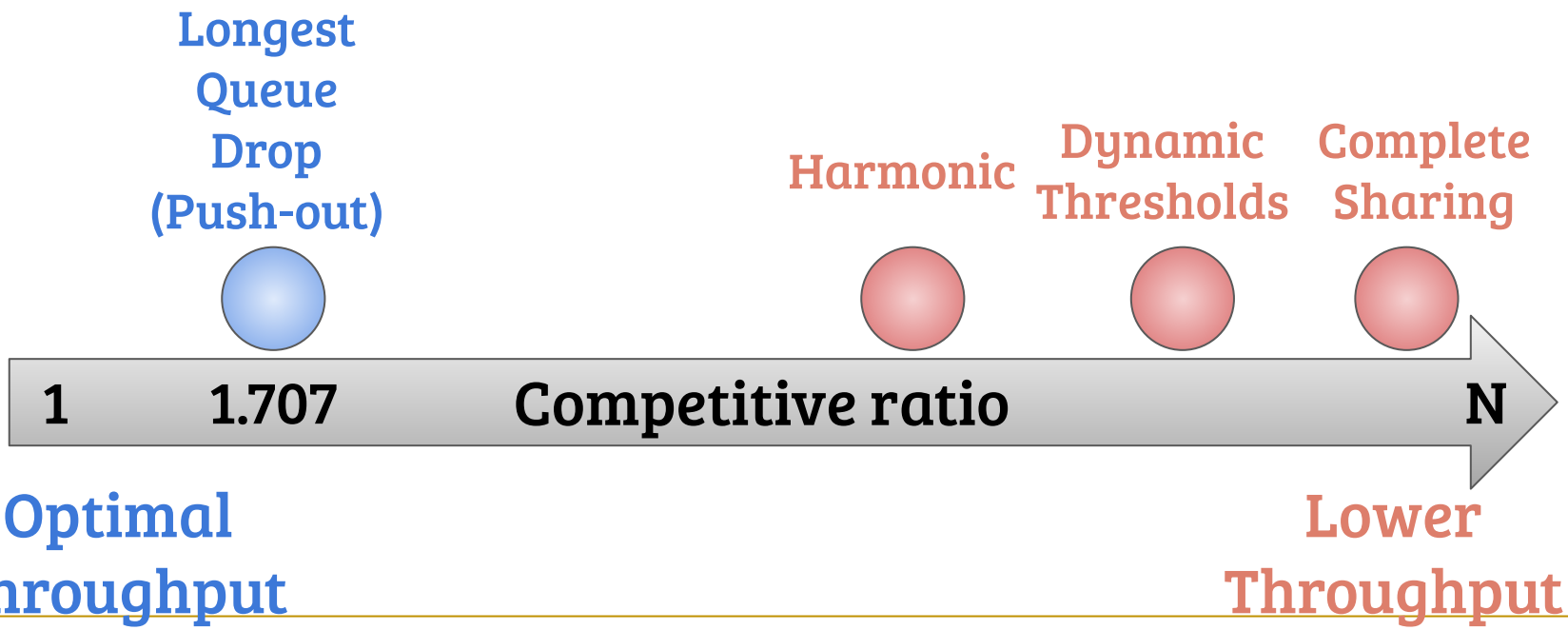
Buffer Sharing (An Online Perspective)

- ALG is *C* competitive if OPT transmits no more than *C* times that of ALG
  - $OPT \leq C \cdot ALG$

Competitive Ratio

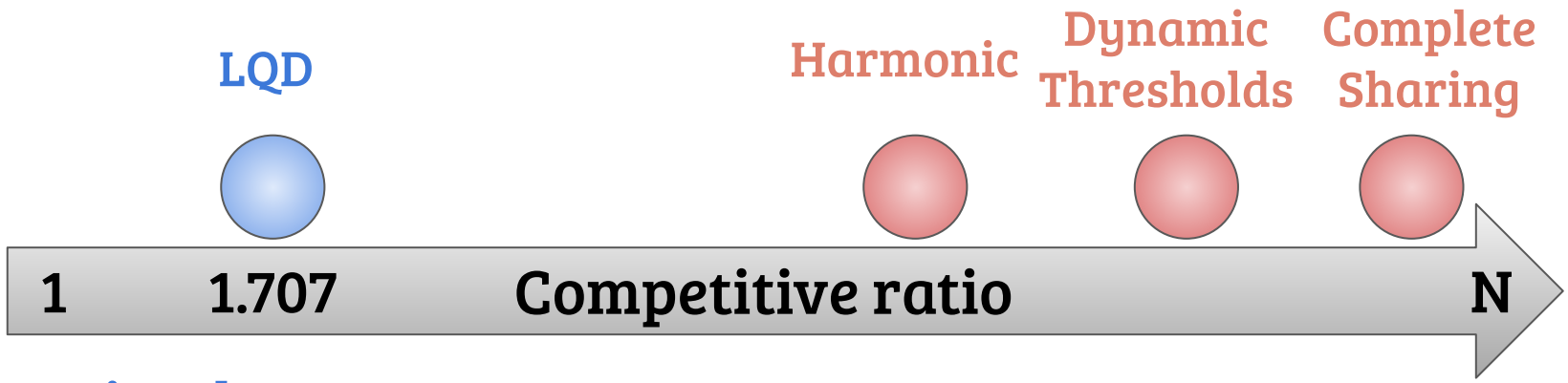
Online Buffer Sharing Algorithms

- Drop-tail: Drop on arrival or accept
  - All commodity switches support drop-tail buffers
- Push-out: Accept all packets and push a packet out when the buffer is full
  - *Not supported in hardware*


Longest
Queue
Drop
(Push-out)
Harmonic
Dynamic
Thresholds
Complete
Sharing
1
1.707
Competitive ratio
N
Optimal
Throughput
Lower
Throughput

Not supported in hardware!

All commodity switches support drop-tail

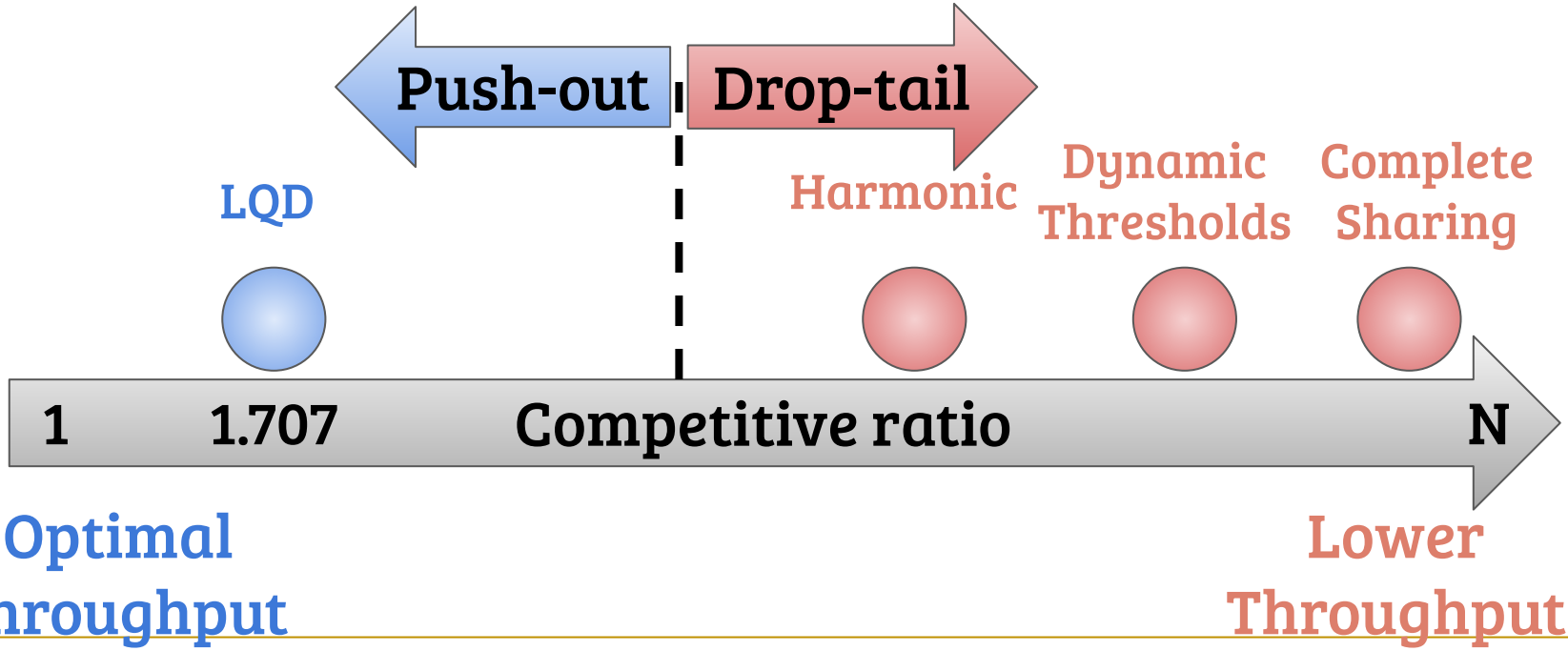


Optimal Throughput

Lower Throughput

PURDUE UNIVERSITY

STyGIANet



PURDUE UNIVERSITY

STyGIANet

Predictions: A Hope for Competitive Buffer Sharing

- Predict the actions of a push-out algorithm (LQD)
- Augment drop-tail algorithms with predictions
  - Peek into the future

Predictions: A Hope for Competitive Buffer Sharing

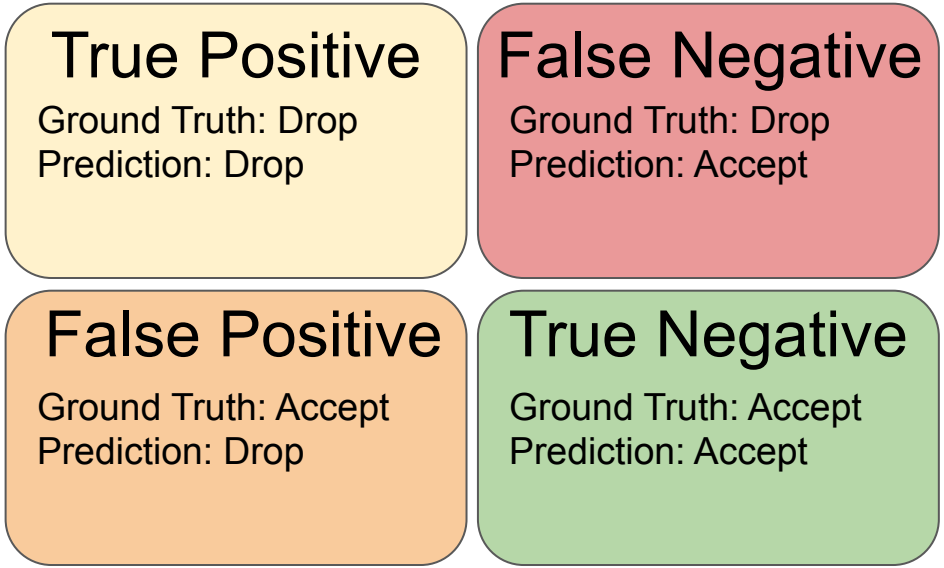
- Predict the actions of a push-out algorithm (LQD)
- Augment drop-tail algorithms with predictions
  - Peek into the future

Can predictions improve drop-tail's competitive ratio?

Naive Approach

- Upon a packet arrival
  - Predict LQD's action
  - If prediction is to accept, then accept
  - If prediction is to drop, then drop

Challenge: Imperfect Predictions



Challenge: Imperfect Predictions

- Excessive false positives can lead to starvation
  - eg., every prediction is “drop”
- Even a single false negative can hurt throughput forever

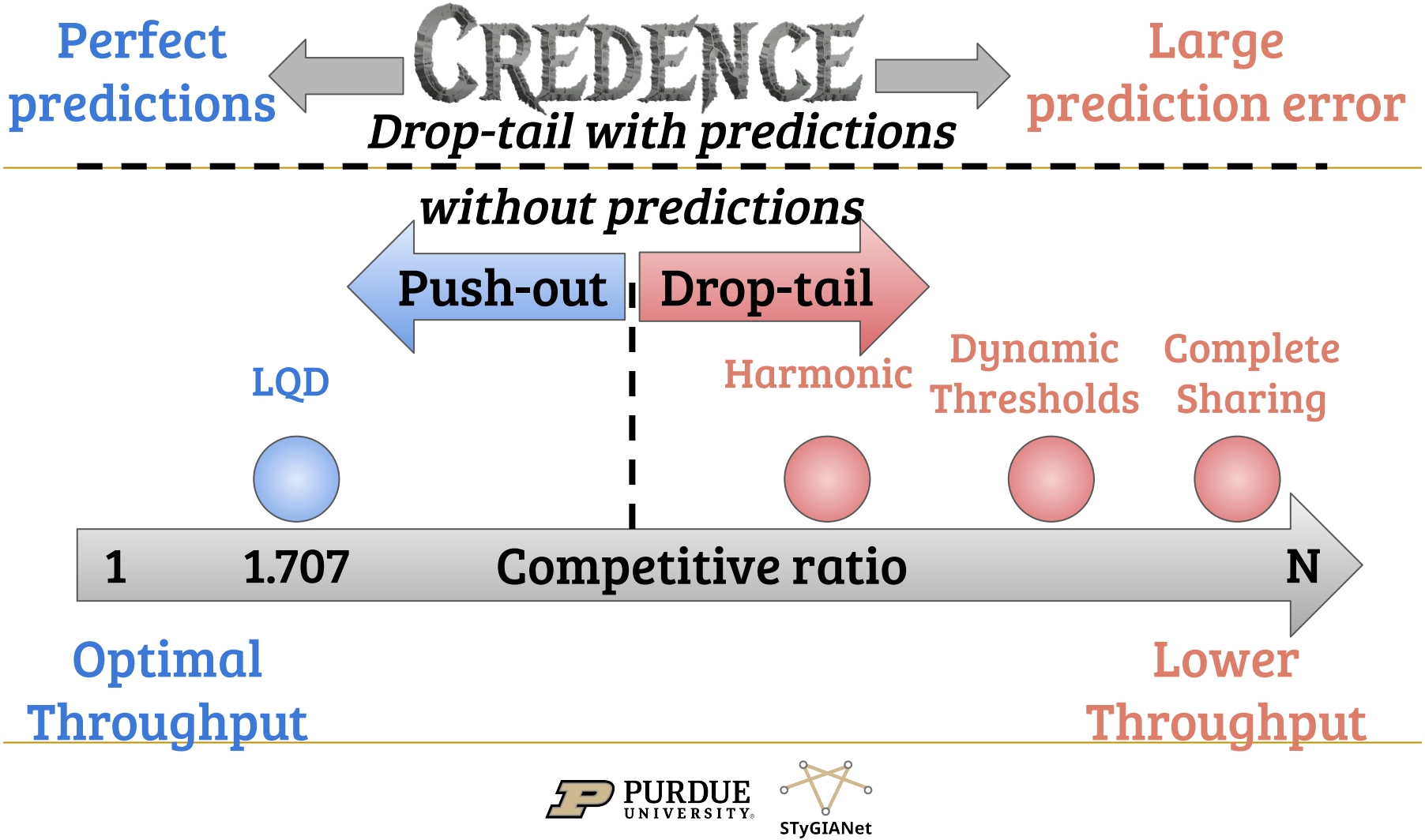
Objectives

- Consistency (under perfect predictions)
  - Competitive ratio close to push-out
- Robustness (with large prediction error)
  - Competitive ratio close to existing algorithms
- Smoothness
  - Competitive ratio smoothly degrades with prediction error

Augmenting Datacenter Switch Buffer Sharing with ML Predictions

PURDUE UNIVERSITY

STyGIANet



PURDUE UNIVERSITY

STyGIANet

Credence

- Per-queue thresholds
  - Thresholds are incremented and decremented based on Longest Queue Drop (Push-out) algorithm
- A packet is rejected immediately if the queue length is greater than its corresponding threshold
- A prediction is obtained *only if* the queue length is lower than its corresponding threshold

Credence

- Thresholds enable tackling false negative errors
  - Prevents accepting too many packets eg., if all the predictions are “accept”
- Safe guard criterion to tackle false positive errors
  - Always accept a packet if the longest queue is lower than fair-share of buffer partition
  - Prevents dropping too many packets eg., if all the predictions are “drop”

Further Reading

[1] Choudhury AK, Hahne EL. Dynamic queue length thresholds for shared-memory packet switches. IEEE/ACM Transactions On Networking. 1998 Apr;6(2):130-40.

[2] Addanki V, Apostolaki M, Ghobadi M, Schmid S, Vanbever L. ABM: Active buffer management in datacenters. InProceedings of the ACM SIGCOMM 2022 Conference 2022 Aug 22 (pp. 36-52).

[3] Addanki V, Bai W, Schmid S, Apostolaki M. Reverie: Low Pass {Filter-Based} Switch Buffer Sharing for Datacenters with {RDMA} and {TCP} Traffic. In21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24) 2024 (pp. 651-668).

[4] Addanki V, Pacut M, Schmid S. Credence: Augmenting Datacenter Switch Buffer Sharing with {ML} Predictions. In21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24) 2024 (pp. 613-634).

[5] Apostolaki M, Vanbever L, Ghobadi M. Fab: Toward flow-aware buffer sharing on programmable switches. InProceedings of the 2019 Workshop on Buffer Sizing 2019 Dec 2 (pp. 1-6).

[6] Kesselman A, Mansour Y. Harmonic buffer management policy for shared memory switches. Theoretical Computer Science. 2004 Sep 20;324(2-3):161-82.

The End