Information-Agnostic Flow Scheduling for Commodity Data Centers

Wei Bai, Li Chen, Kai Chen, Dongsu Han (KAIST), Chen Tian (NJU), Hao Wang

Sing Group @ Hong Kong University of Science and Technology

USENIX NSDI 2015, Oakland, USA

Data Center Transport

- Cloud applications
  - Desire low latency for short messages
- Goal: Minimize flow completion time (FCT)
  - Many flow scheduling proposals…

SQL

MEMCACHED

The State-of-the-art Solutions

- PDQ [SIGCOMM'12]
- pFabric [SIGCOMM'13]
- PASE [SIGCOMM'14]
- …

All assume prior knowledge of flow size information to approximate ideal preemptive Shortest Job First (SJF) with customized network elements

The State-of-the-art Solutions

- PDQ [SIGCOMM'12]
- pFabric [SIGCOMM'13]
- PASE [SIGCOMM'14]
- …

All assume ~~prior knowledge of flow size information~~ to approximate ideal preemptive Shortest Job First (SJF) with customized network elements

Not feasible for some applications

The State-of-the-art Solutions

- PDQ [SIGCOMM’12]
- pFabric [SIGCOMM’13]
- PASE [SIGCOMM’14]
- …

All assume ~~prior knowledge of flow size information~~ to approximate ideal preemptive Shortest Job First (SJF) with ~~customized network elements~~

Hard to deploy in practice

Question

Without prior knowledge of flow size information, how to minimize FCT in commodity data centers?

Design Goal 1

Without prior knowledge of flow size information, how to minimize FCT in commodity data centers?

Information-agnostic: not assume a priori knowledge of flow size information available from the applications

Design Goal 2

Without prior knowledge of flow size information, how to minimize FCT in commodity data centers?

FCT minimization: minimize average and tail FCTs of short flows & not adversely affect FCTs of large flows

Design Goal 3

Without prior knowledge of flow size information, how to minimize FCT in commodity data centers?

Readily-deployable: work with existing commodity switches & be compatible with legacy network stacks

Question

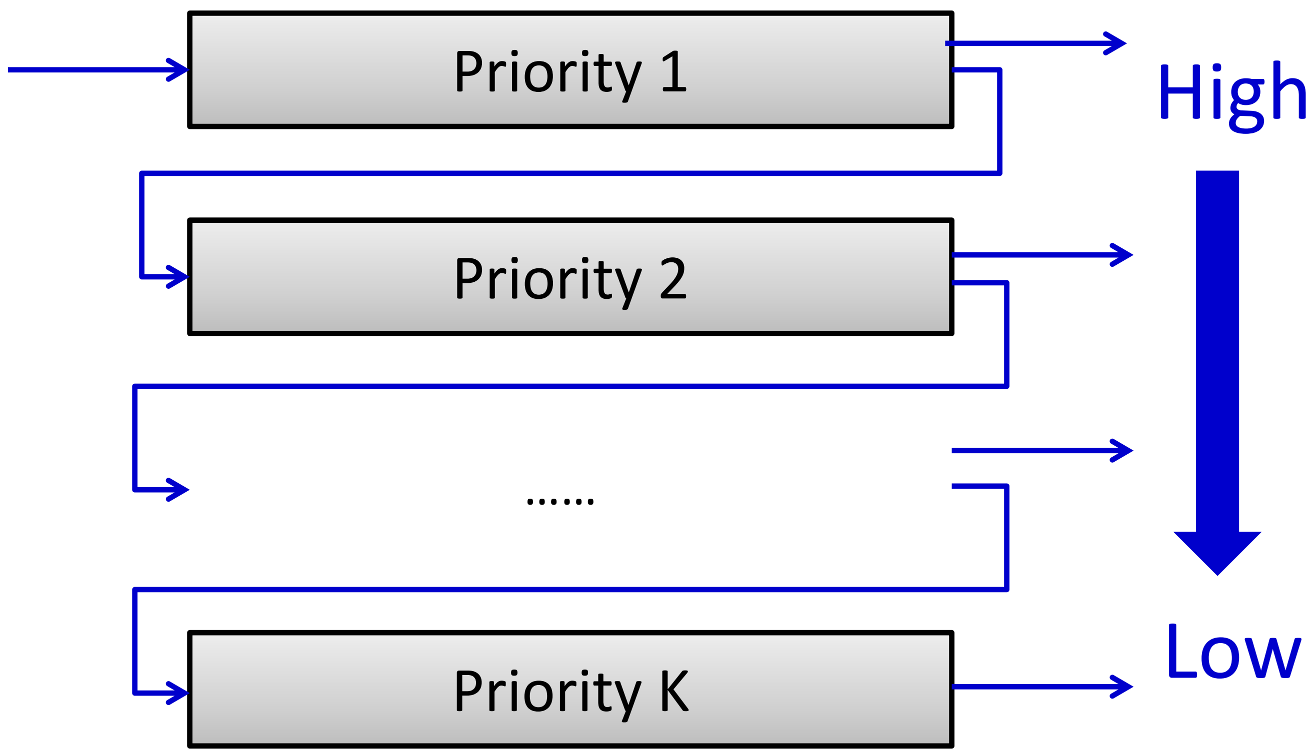
Without prior knowledge of flow size information, how to minimize FCT in commodity data centers?

Our answer: PIAS

PIAS’S DESIGN

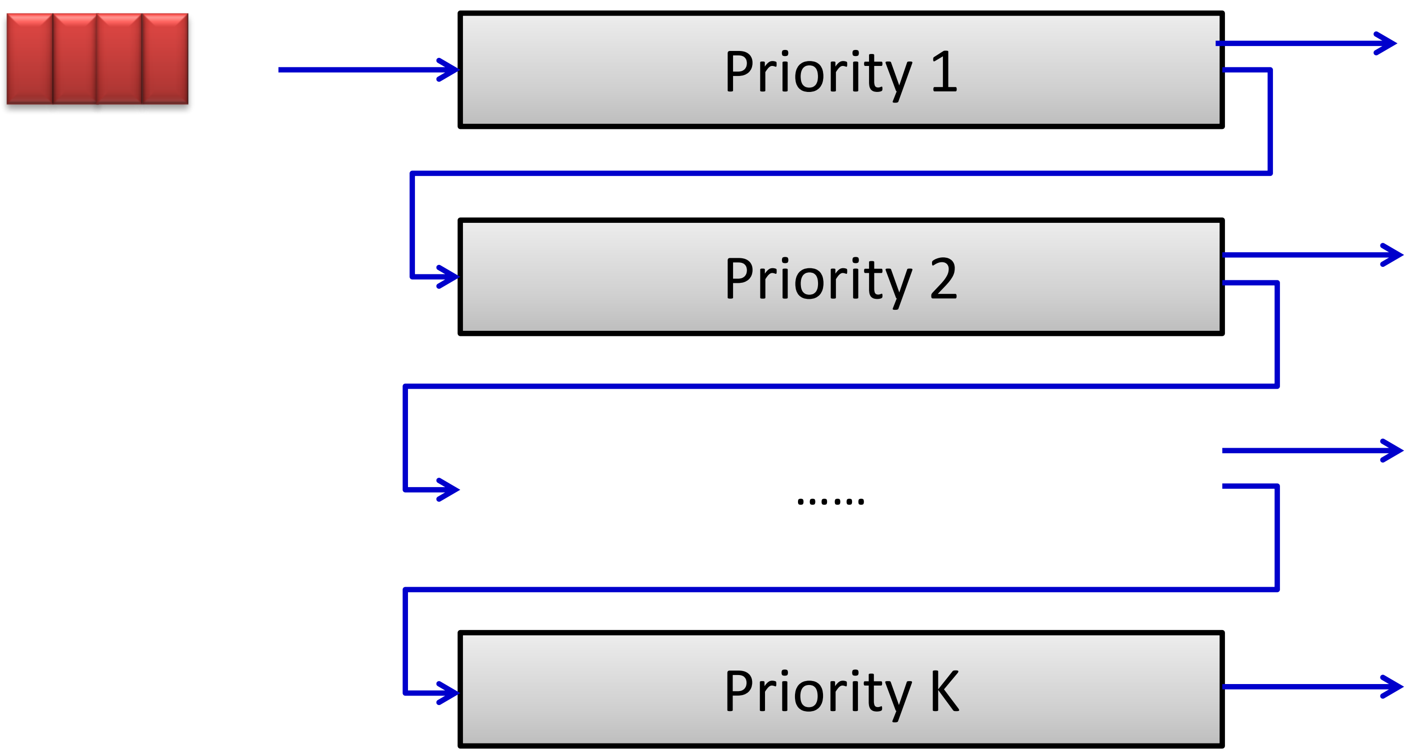
Design Rationale

- PIAS performs Multi-Level Feedback Queue (MLFQ) to emulate Shortest Job First

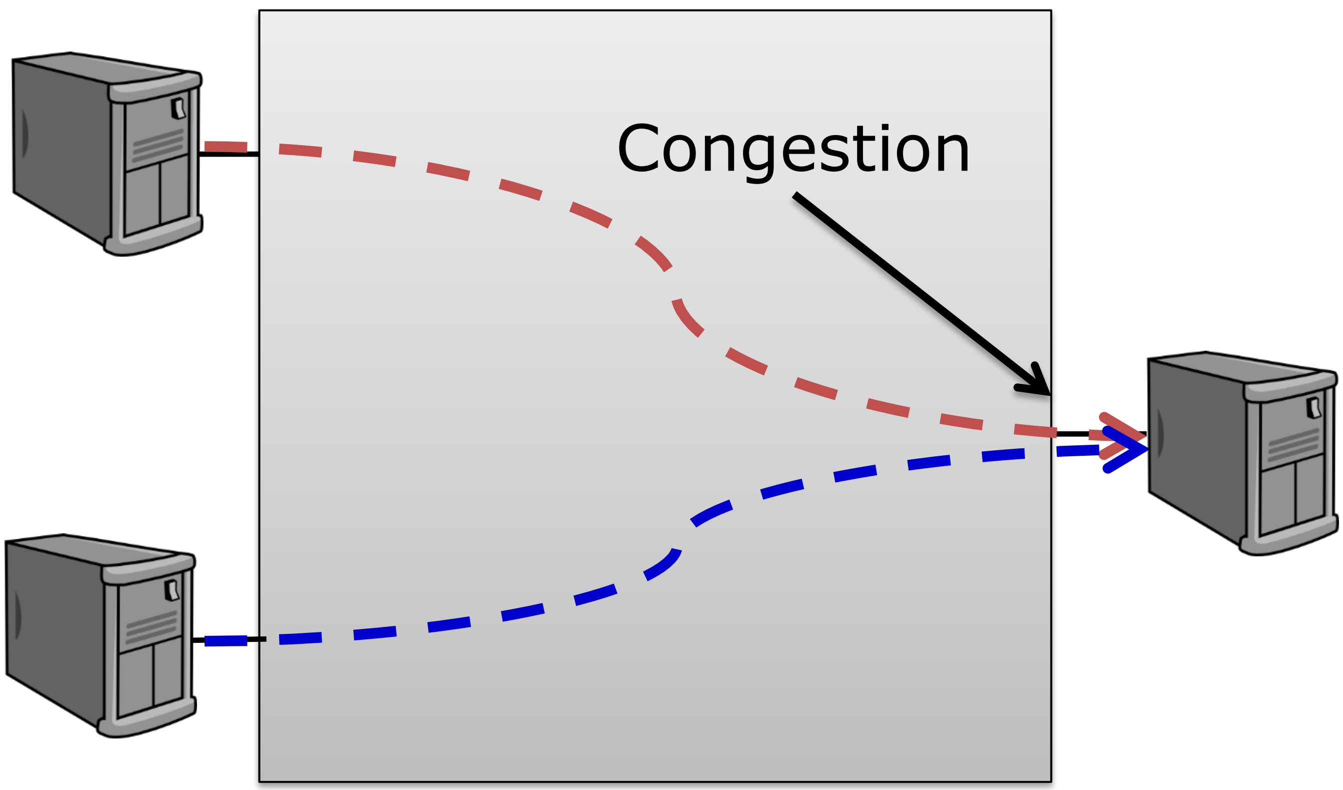


Design Rationale

- PIAS performs Multi-Level Feedback Queue (MLFQ) to emulate Shortest Job First

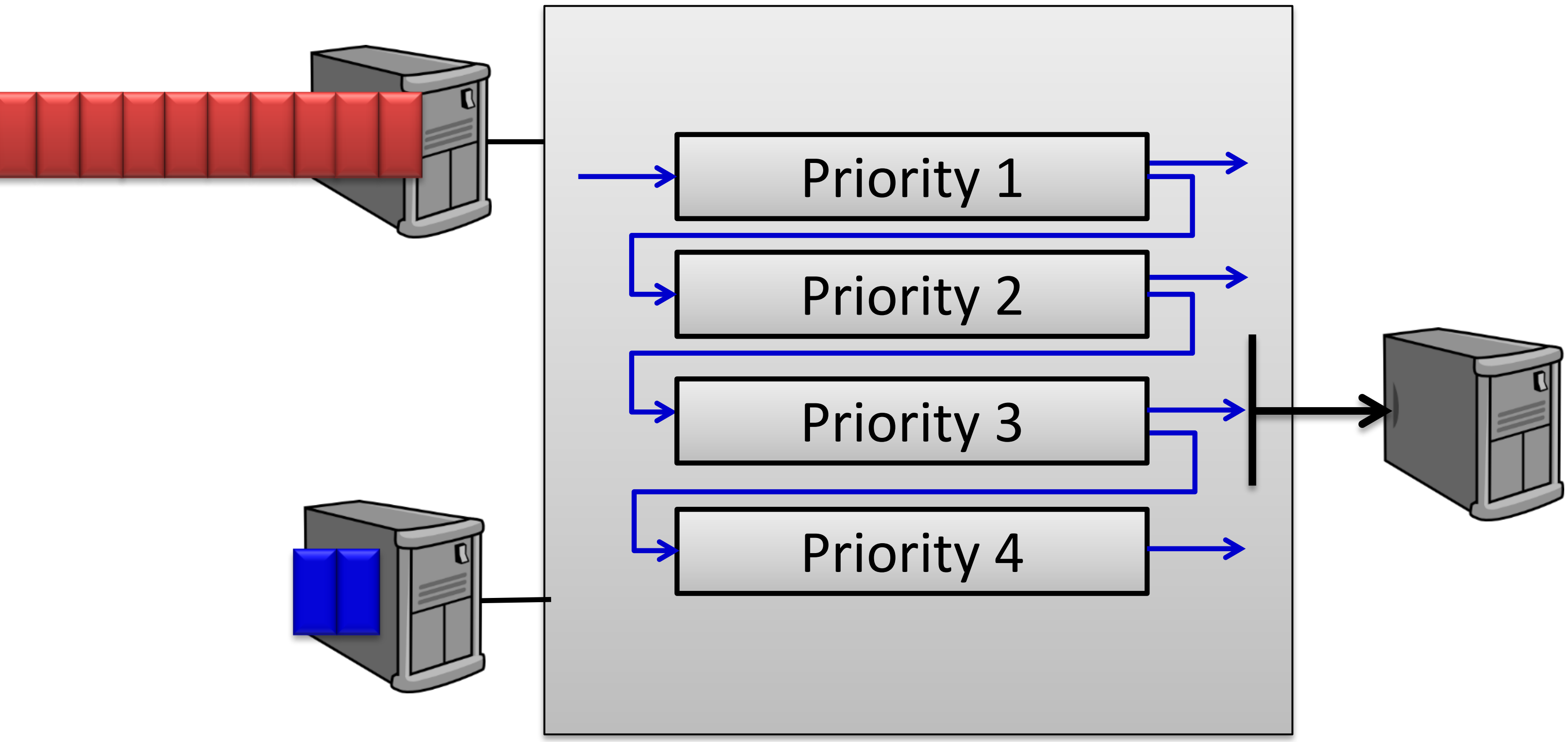


Simple Example Illustrating PIAS



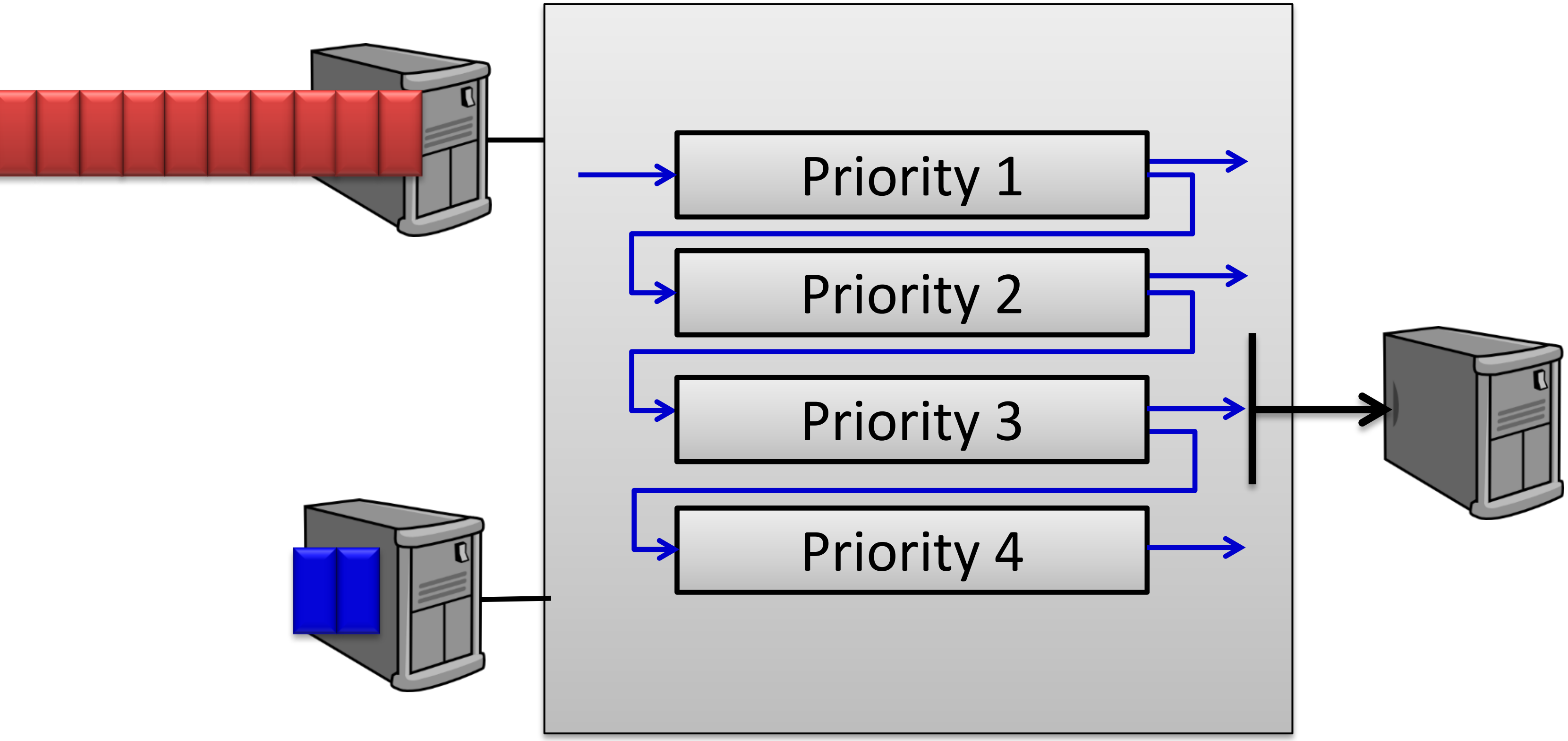
Simple Example Illustrating PIAS

Flow 1 with 10 packets and flow 2 with 2 packets arrive



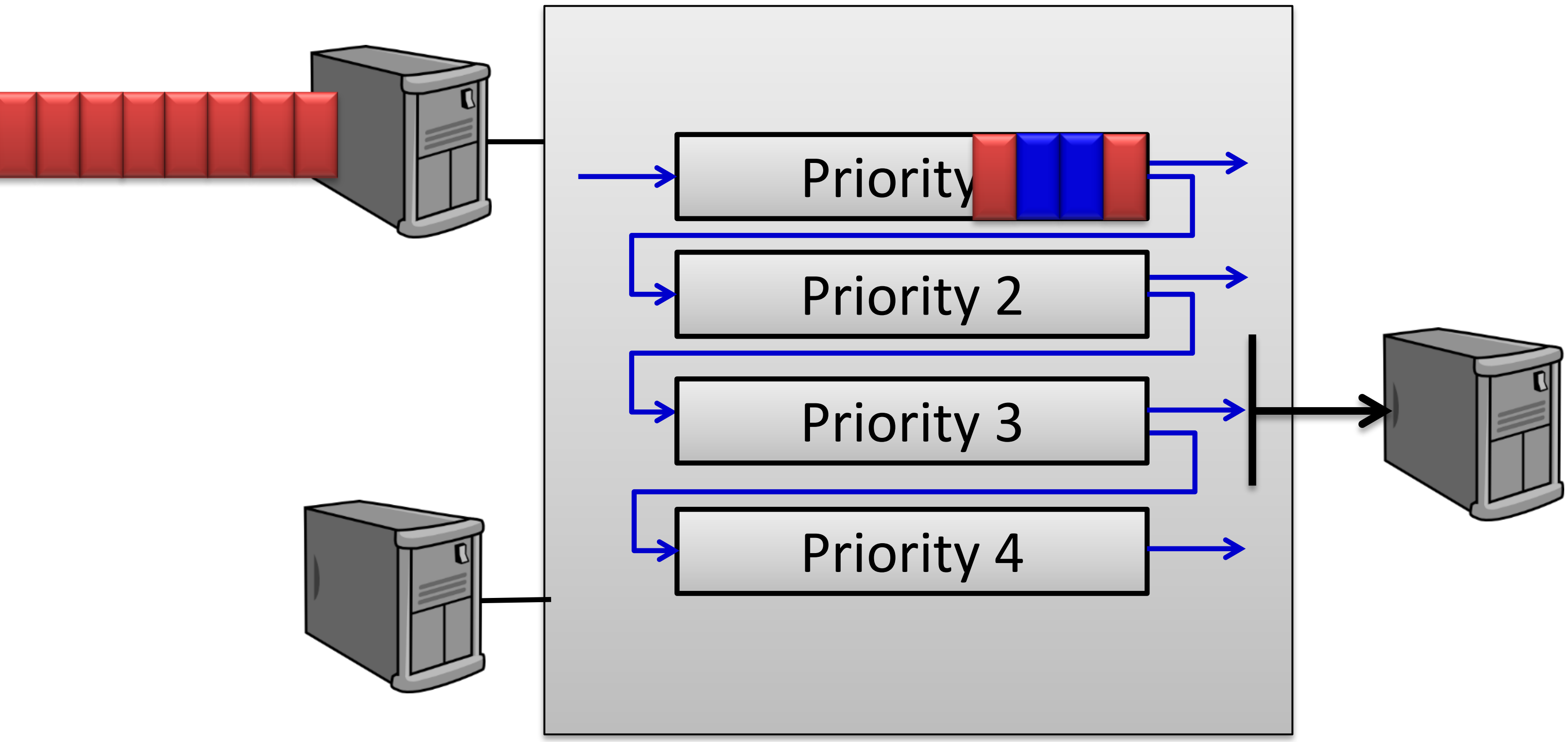
Simple Example Illustrating PIAS

Flow 1 and 2 transmit simultaneously



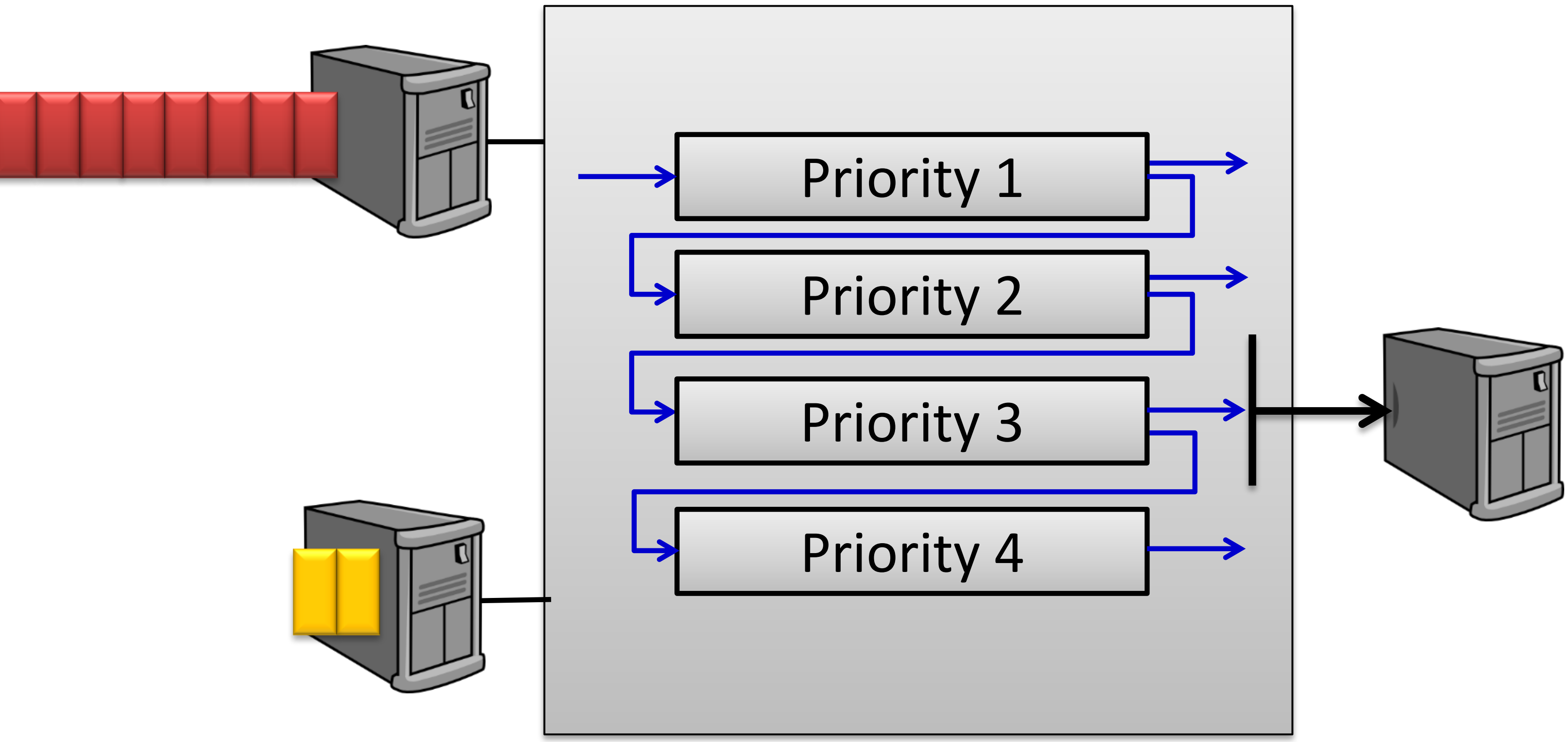
Simple Example Illustrating PIAS

Flow 2 finishes while flow 1 is demoted to priority 2



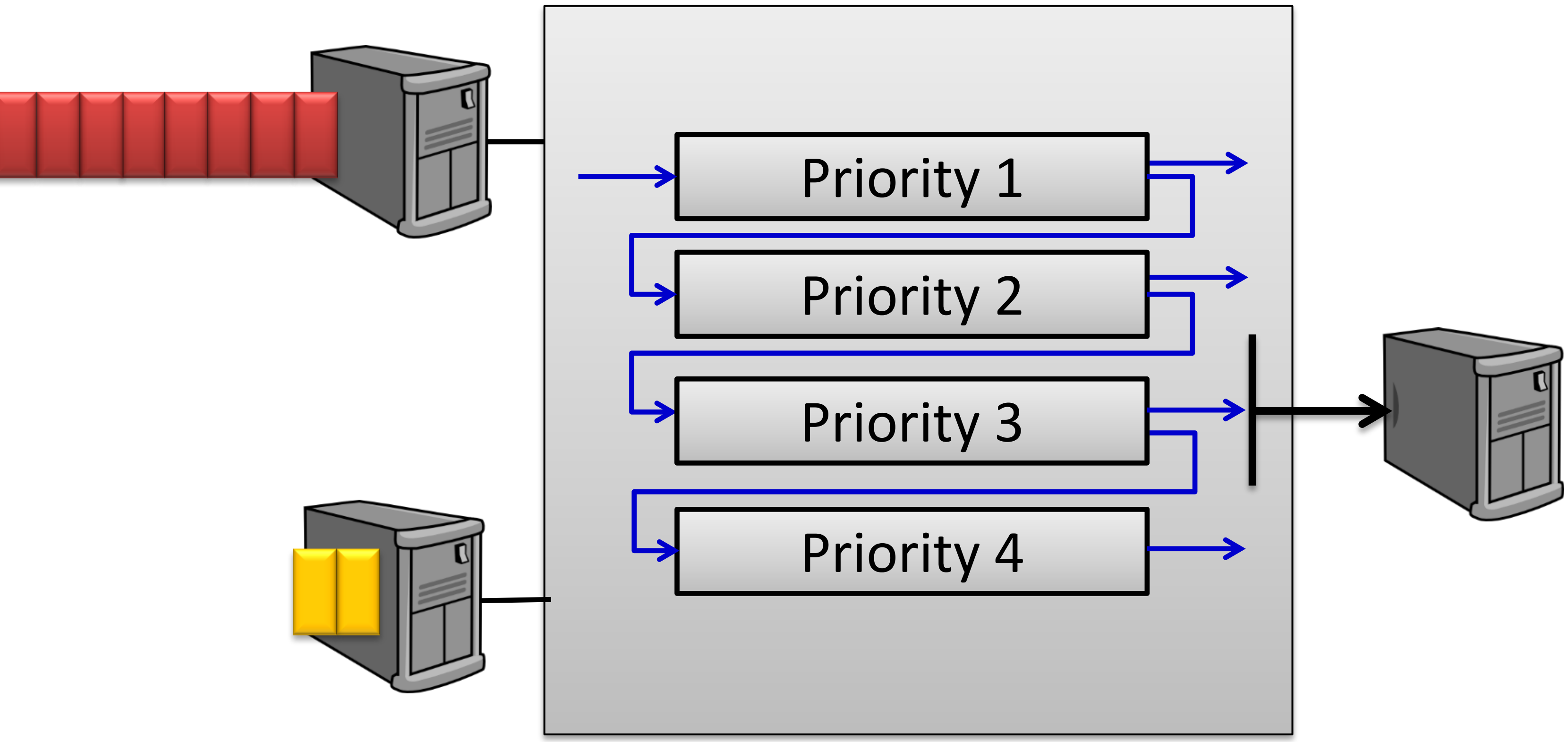
Simple Example Illustrating PIAS

Flow 3 with 2 packets arrives



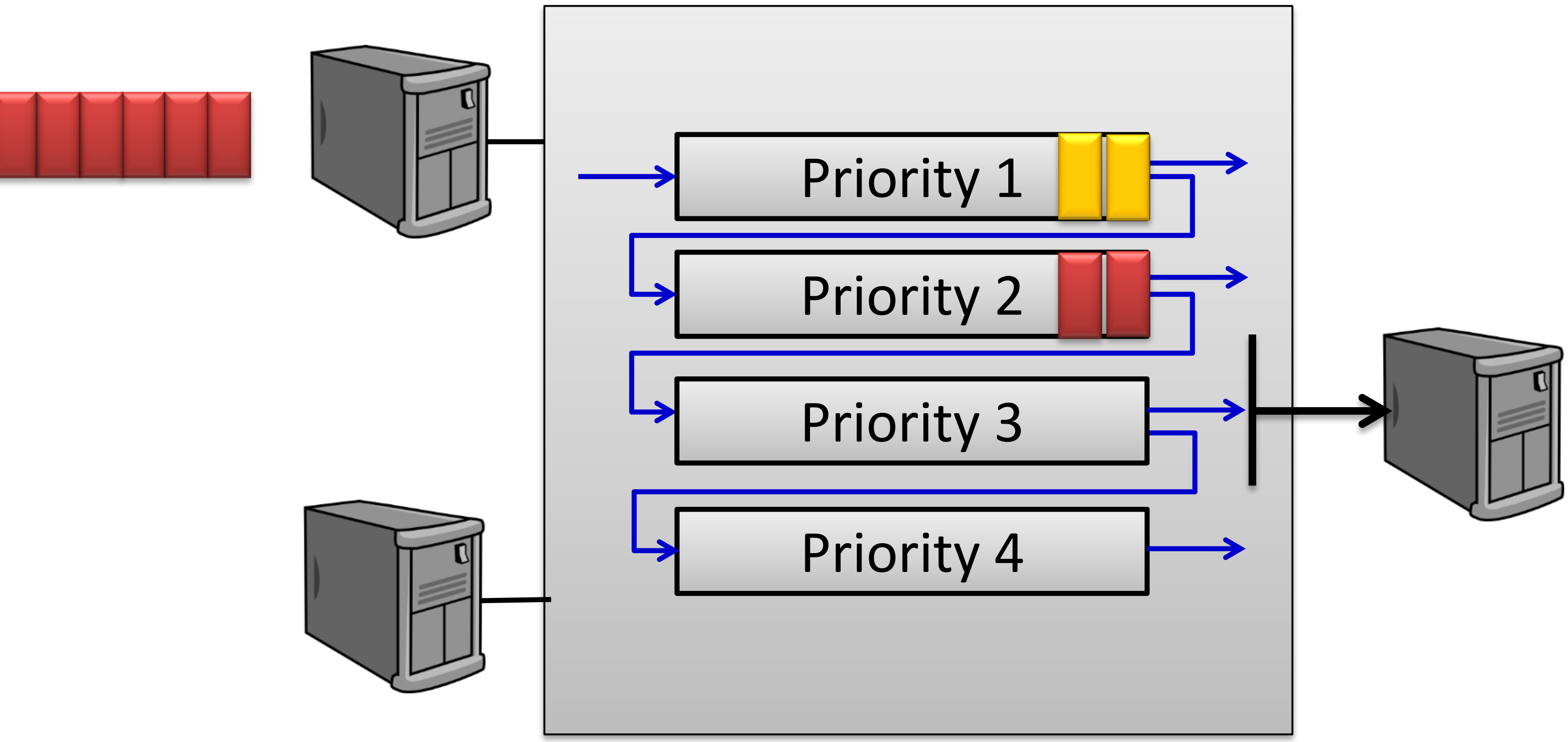
Simple Example Illustrating PIAS

Flow 3 and 1 transmit simultaneously



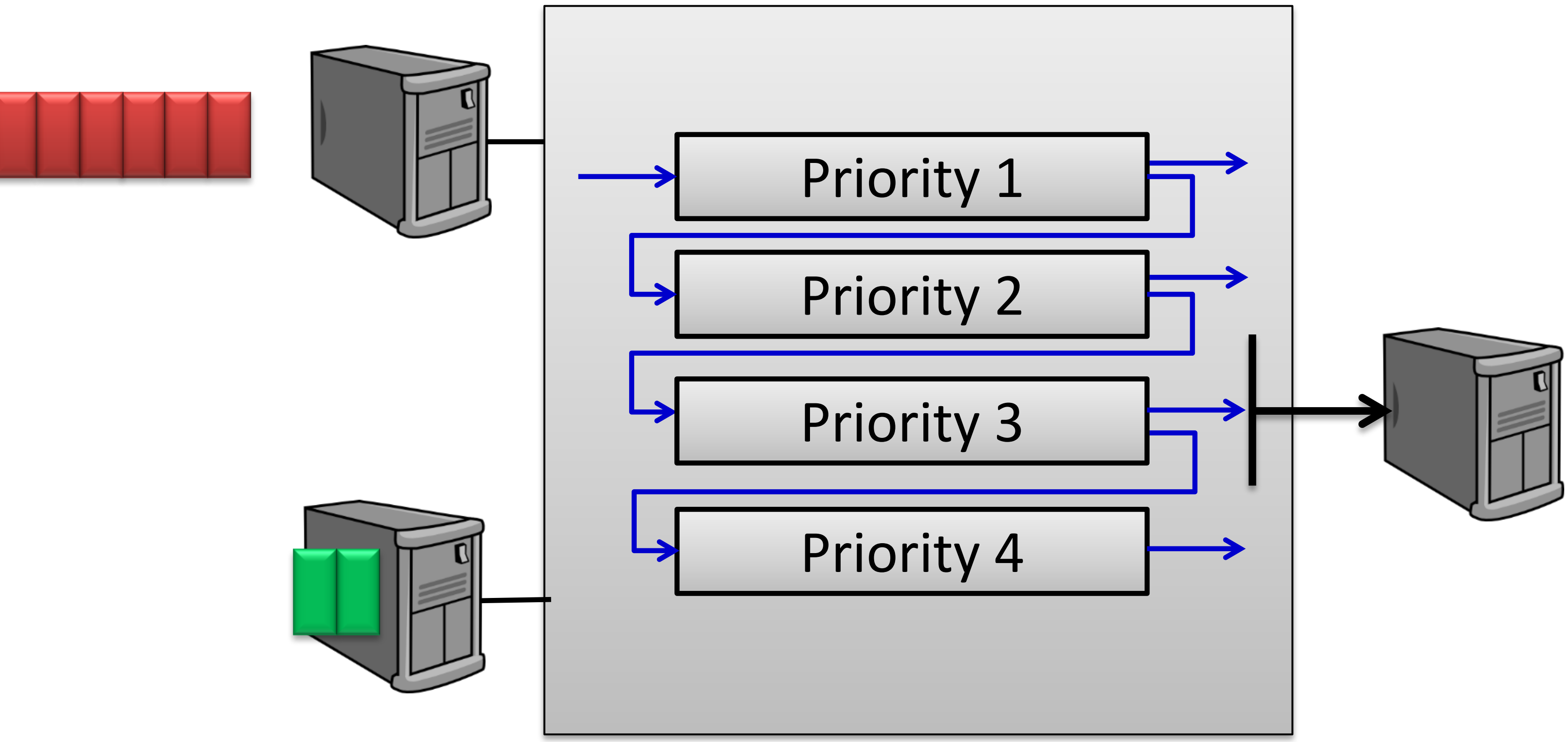
Simple Example Illustrating PIAS

Flow 3 finishes while flow 1 is demoted to priority 3



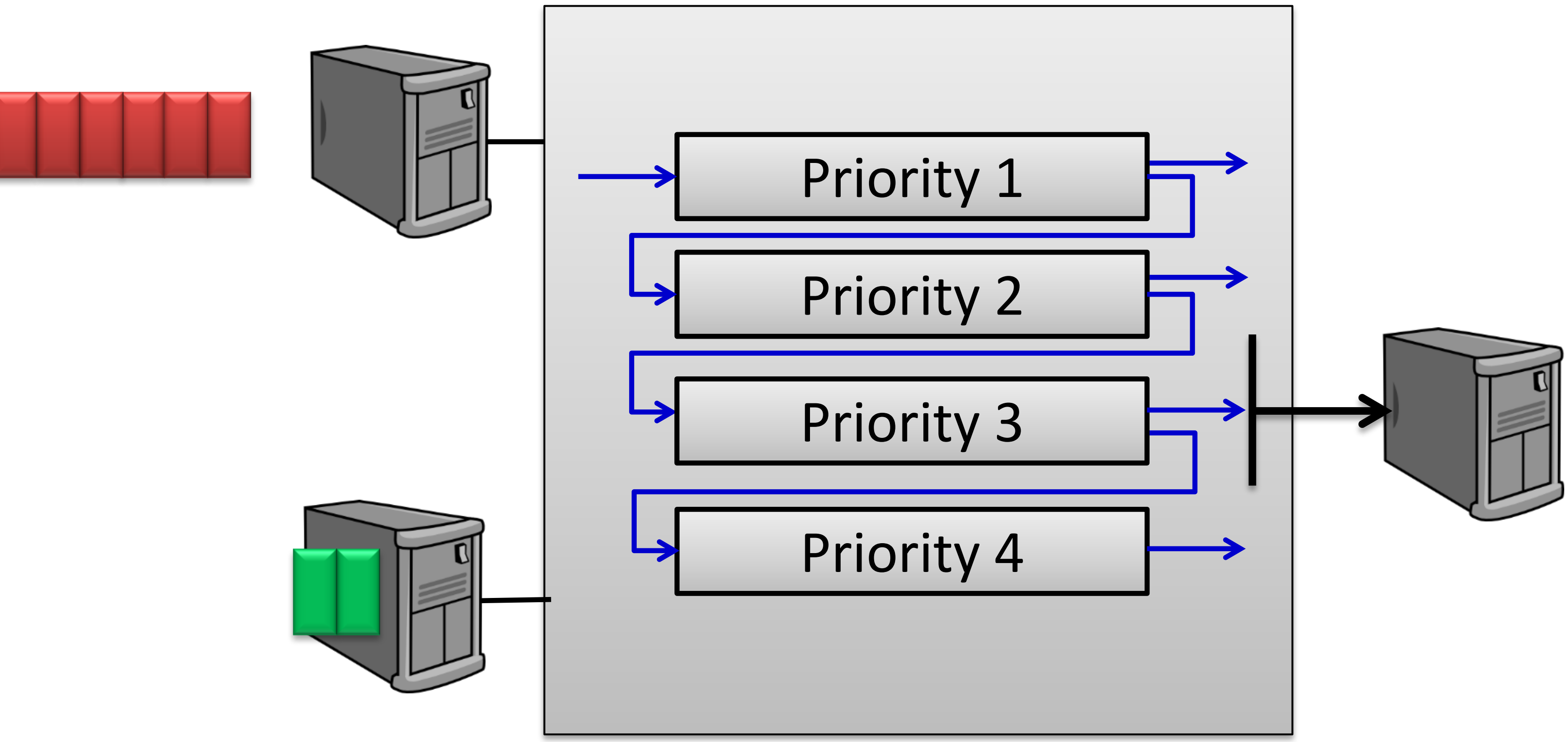
Simple Example Illustrating PIAS

Flow 4 with 2 packets arrives



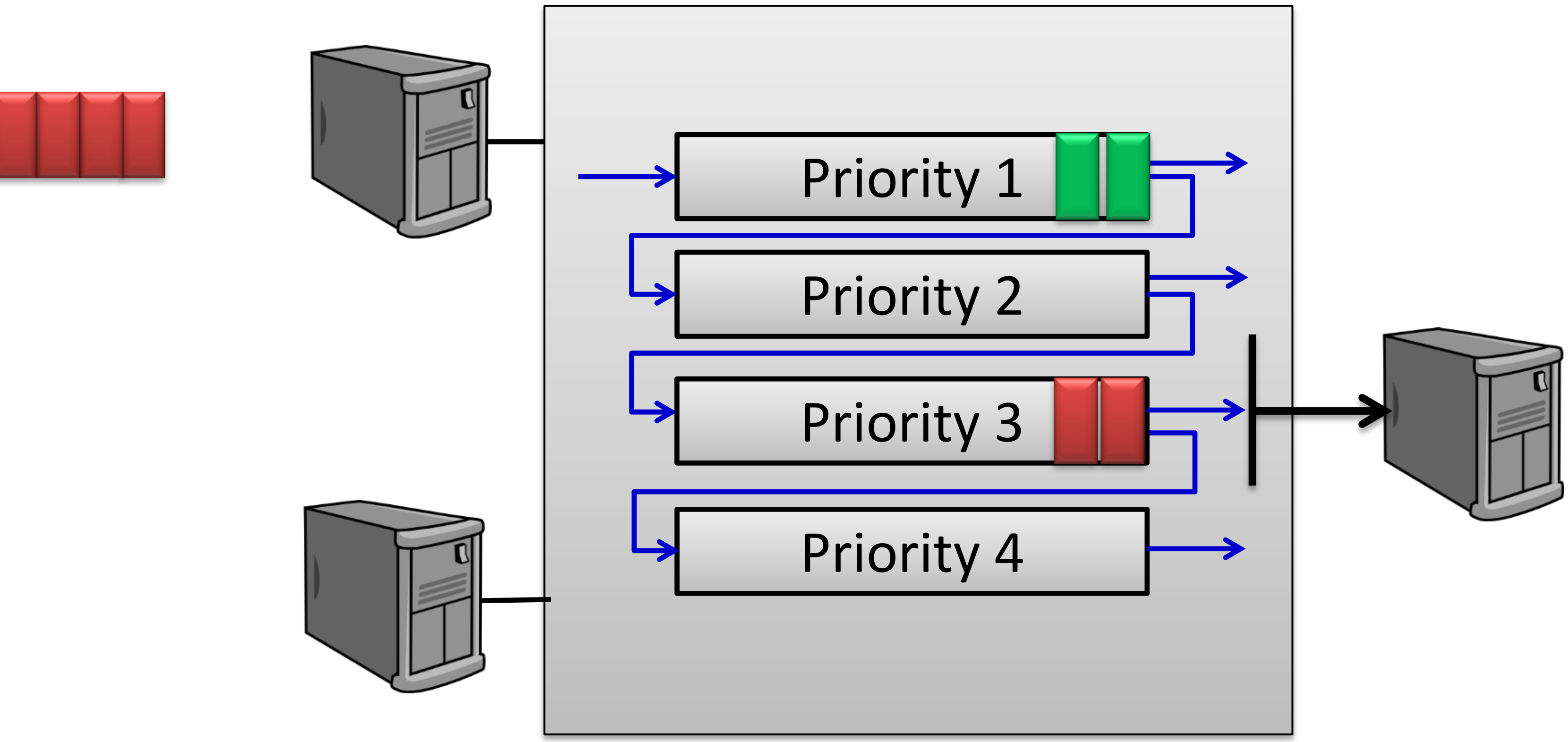
Simple Example Illustrating PIAS

Flow 4 and 1 transmit simultaneously



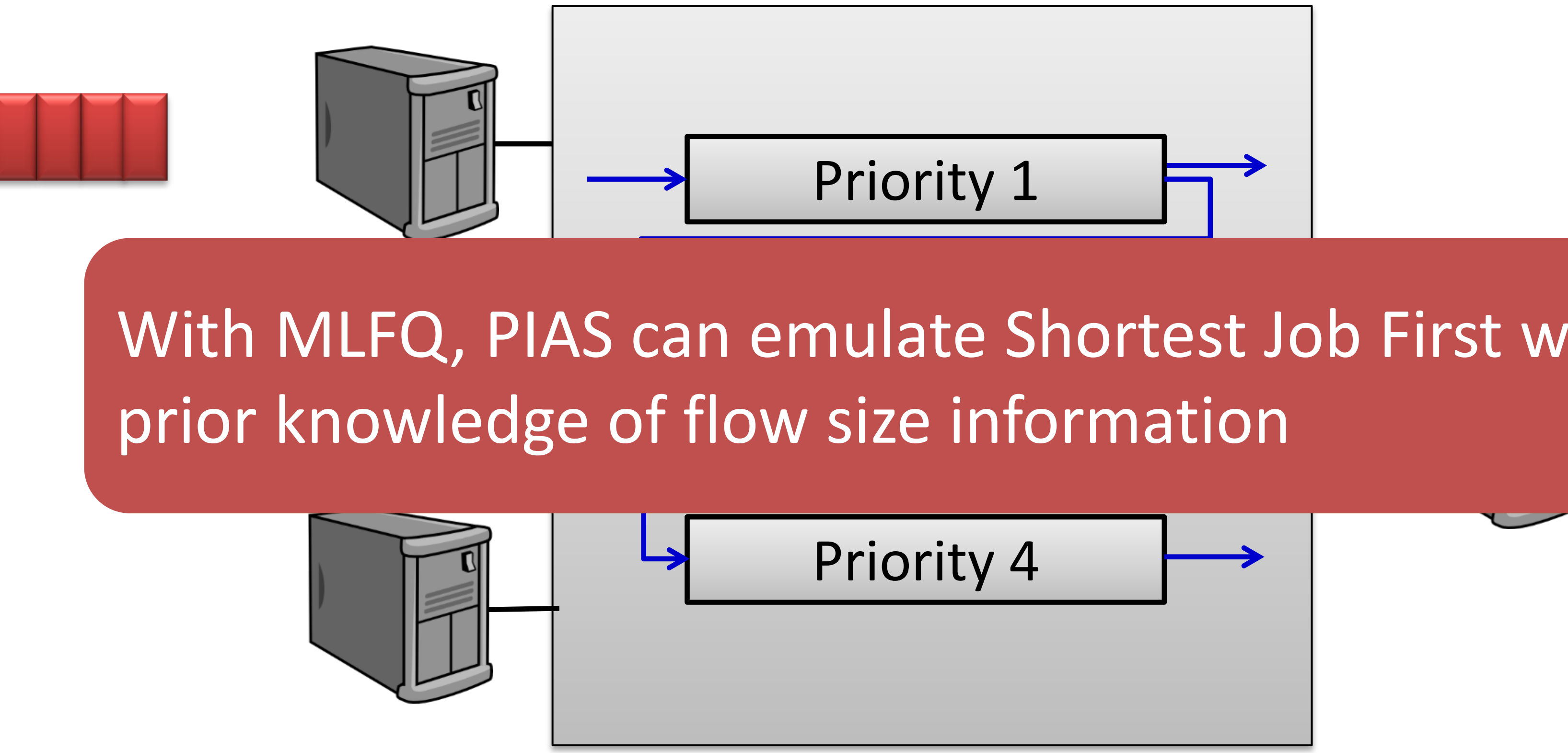
Simple Example Illustrating PIAS

Flow 4 finishes while flow 1 is demoted to priority 4



Simple Example Illustrating PIAS

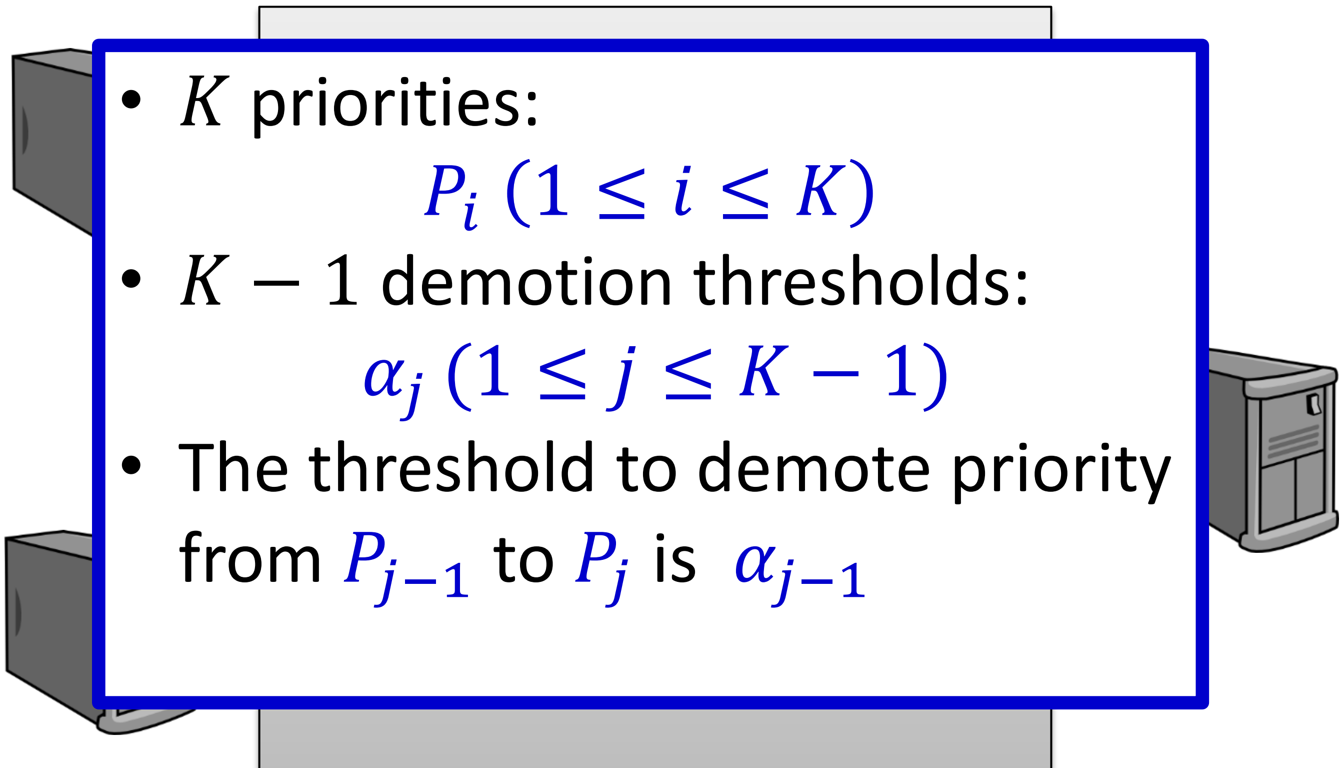
Eventually, flow 1 finishes in priority 4



With MLFQ, PIAS can emulate Shortest Job First without prior knowledge of flow size information

How to implement?

- Strict priority queueing on switches
- Packet tagging as a shim layer at end hosts



- K priorities:

$$P_i \; (1 \leq i \leq K)$$

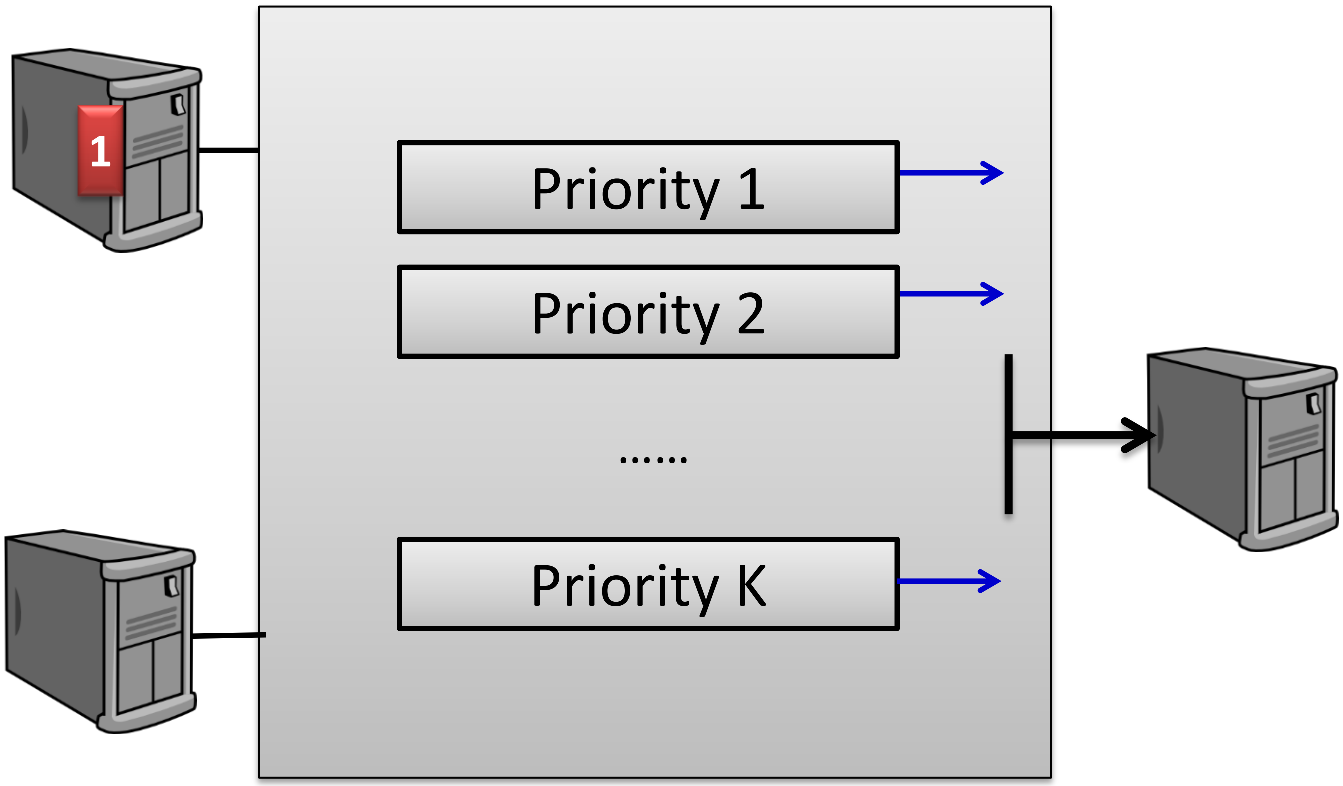
- $K - 1$ demotion thresholds:

$$\alpha_j \; (1 \leq j \leq K-1)$$

- The threshold to demote priority from P_{j-1} to P_j is α_{j-1}

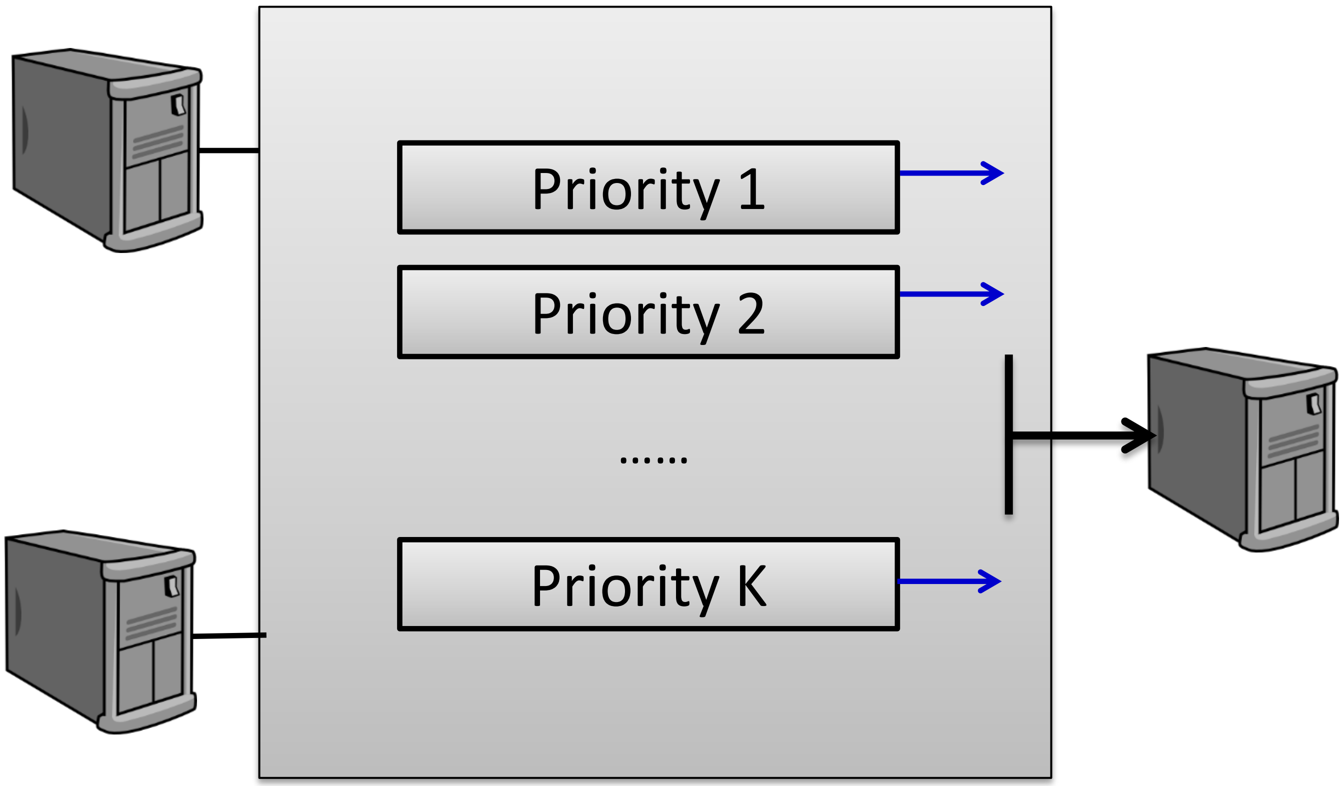
How to implement?

- Strict priority queueing on switches
- Packet tagging as a shim layer at end hosts



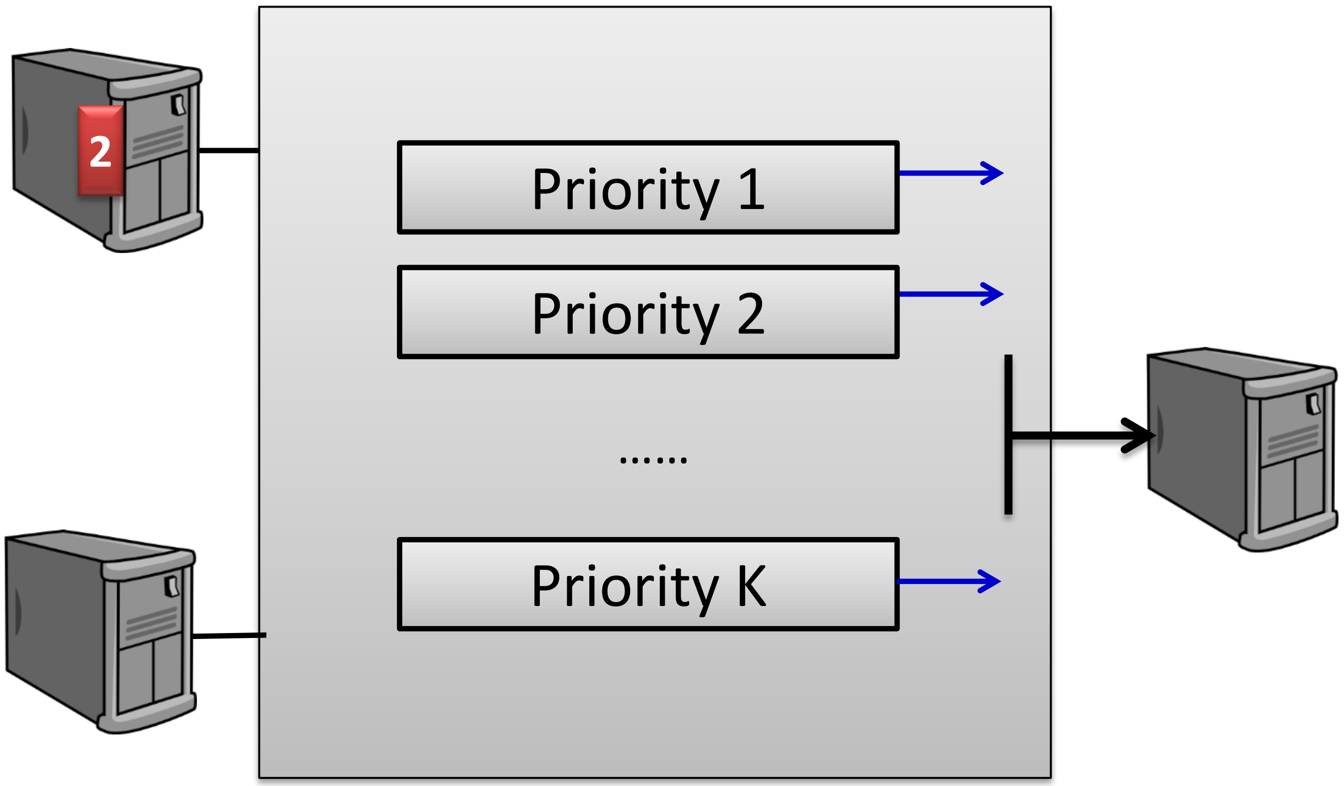
How to implement?

- Strict priority queueing on switches
- Packet tagging as a shim layer at end hosts



How to implement?

- Strict priority queueing on switches
- Packet tagging as a shim layer at end hosts



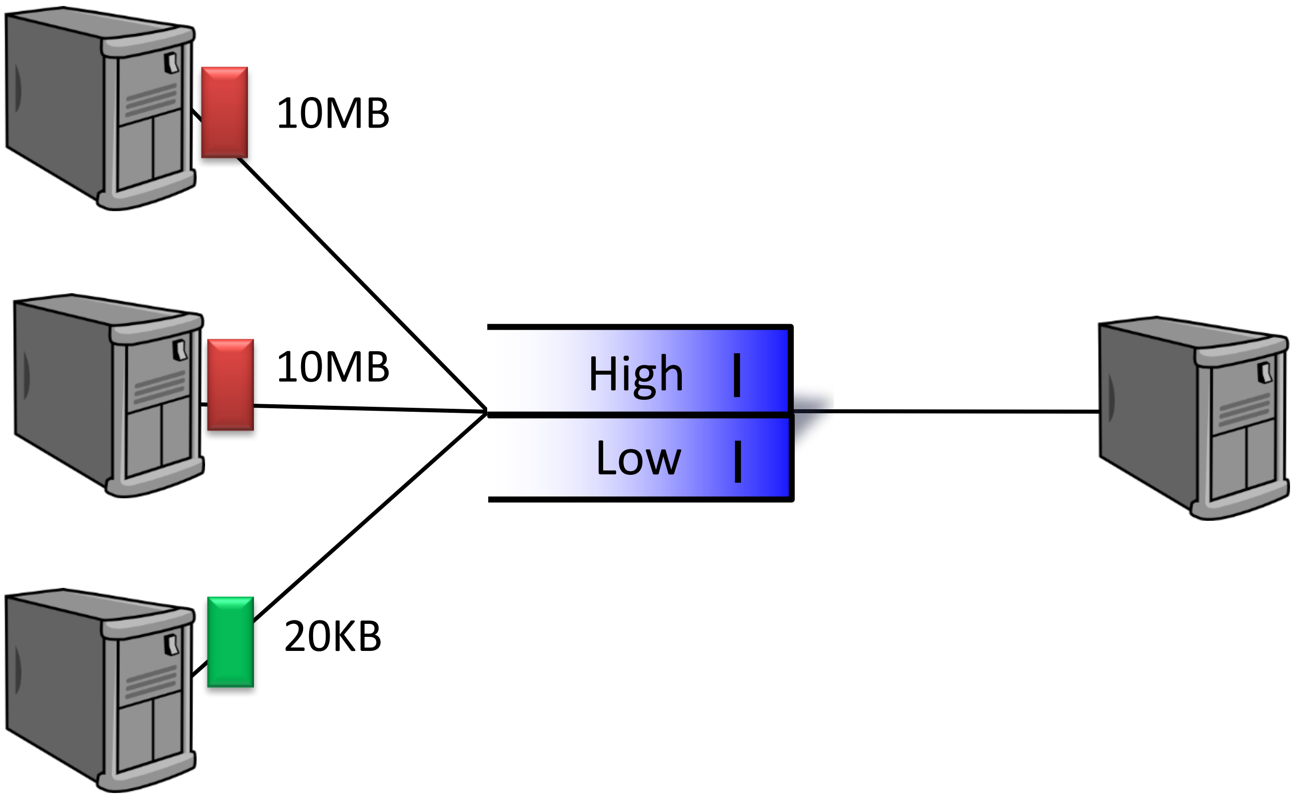
Determine Thresholds

- Thresholds depend on:
  - Flow size distribution
  - Traffic load

Traffic variations -> Mismatched thresholds

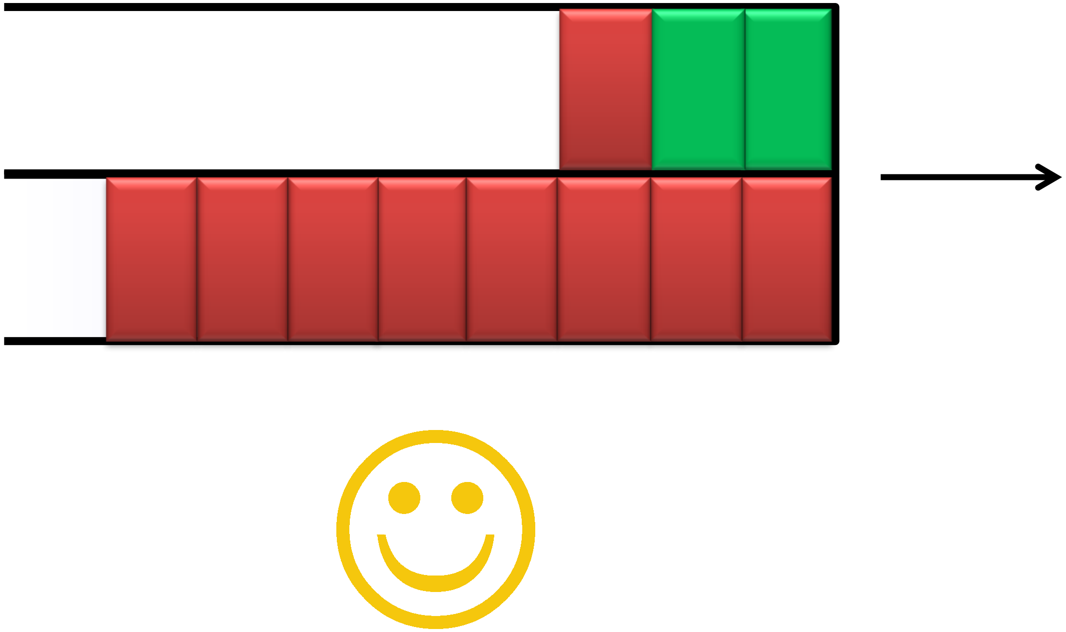
  - Solve a FCT minimization problem to calculate demotion thresholds
- Problem:
  - Traffic is highly dynamic

Impact of Mismatches



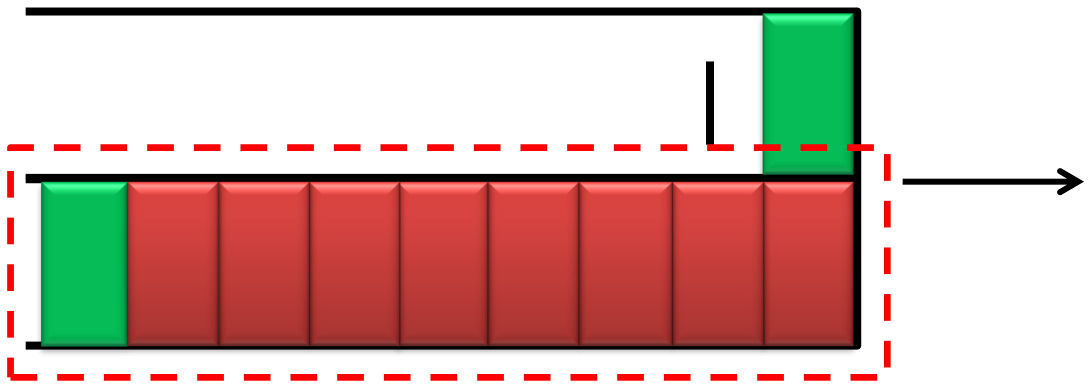
Impact of Mismatches

- When the threshold is perfect (20KB)



Impact of Mismatches

- When the threshold is too small (10KB)



Increased latency for short flows

Impact of Mismatches

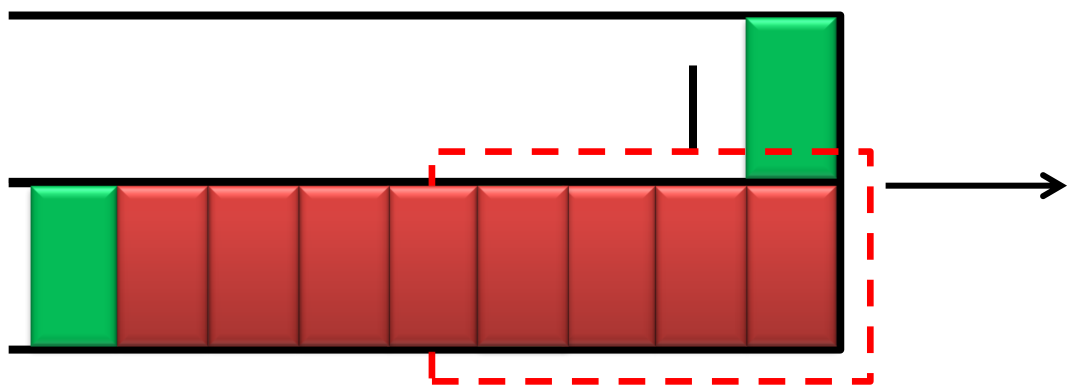
- When the threshold is too large (1MB)

Leverage ECN to keep low buffer occupation

Increased latency for short flows

Handle Mismatches

- When the threshold is too small (10KB)

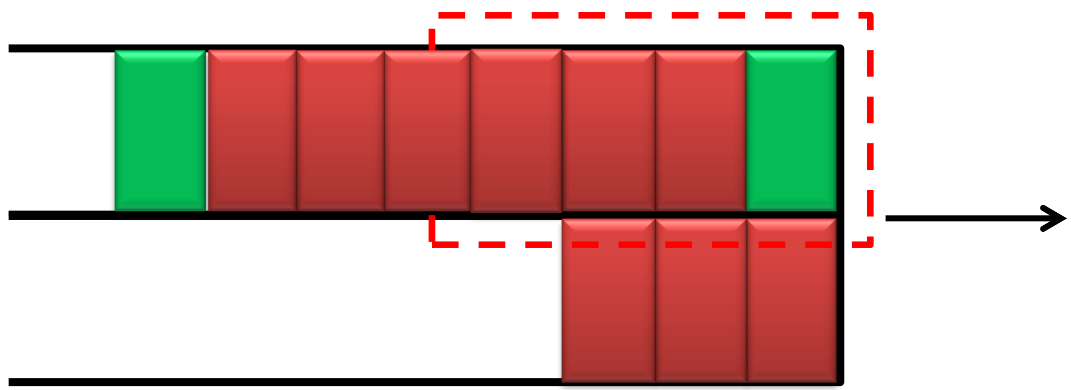


If we enable ECN

ECN can keep low latency

Handle Mismatches

- When the threshold is too large (1MB)



If we enable ECN

ECN can keep low latency

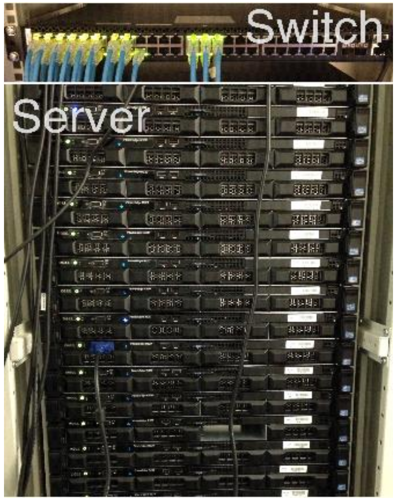
PIAS in 1 Slide

- PIAS packet tagging
  - Maintain flow states and mark packets with priority
- PIAS switches
  - Enable strict priority queueing and ECN
- PIAS rate control
  - Employ Data Center TCP to react to ECN

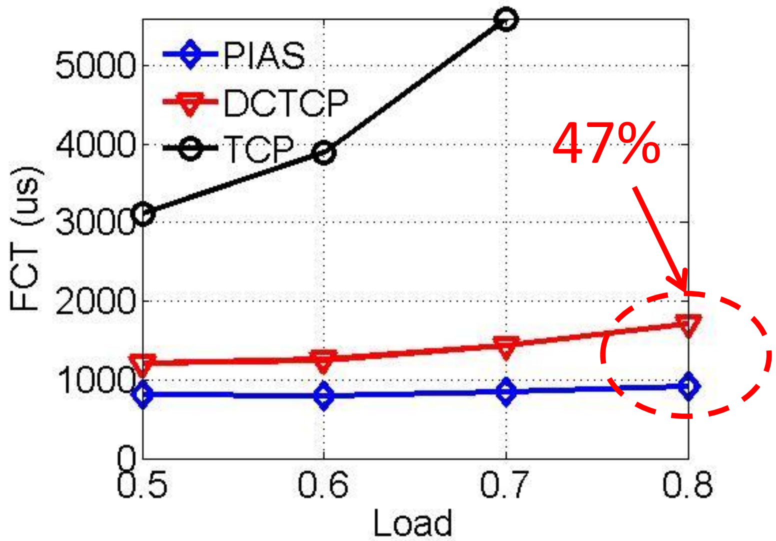
Testbed Experiments

- PIAS prototype
  - http://sing.cse.ust.hk/projects/PIAS
- Testbed Setup
  - A Gigabit Pronto-3295 switch
  - 16 Dell servers
- Benchmarks
  - Web search (DCTCP paper)
  - Data mining (VL2 paper)
  - Memcached

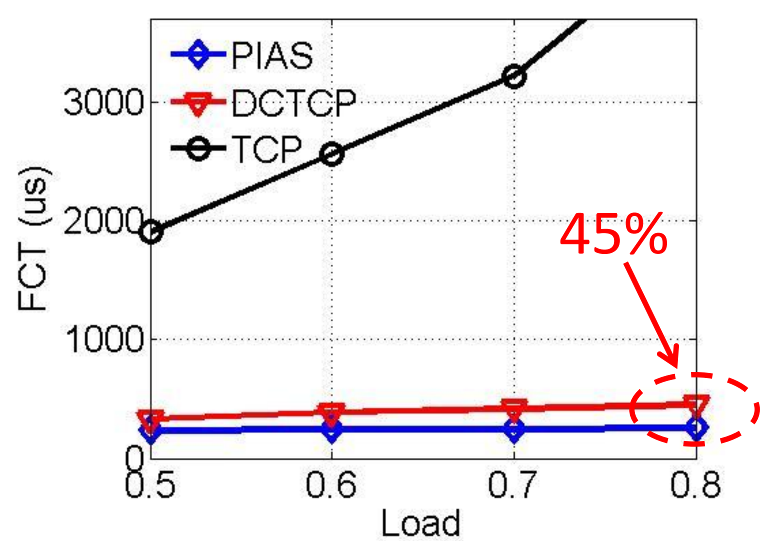
open source



Small Flows (<100KB)



Web Search



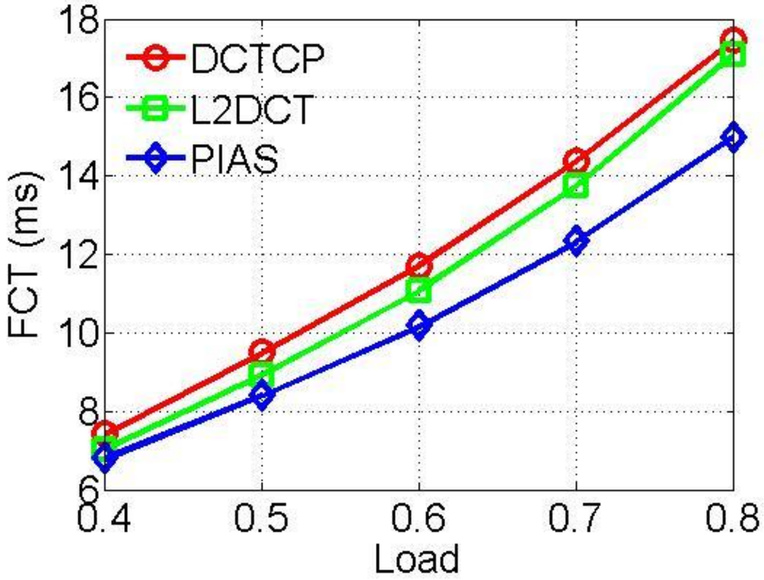
Data Mining

Compared to DCTCP, PIAS reduces average FCT of small flows by up to 47% and 45%

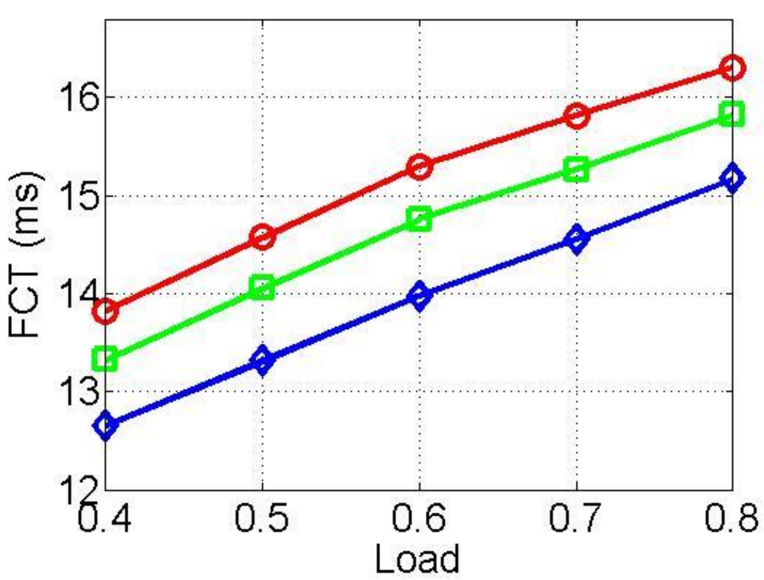
NS2 Simulation Setup

- Topology
  - 144-host leaf-spine fabric with 10G/40G links
- Workloads
  - Web search (DCTCP paper)
  - Data mining (VL2 paper)
- Schemes
  - Information-agnostic: PIAS, DCTCP and L2DCT
  - Information-aware: pFabric

Overall Performance



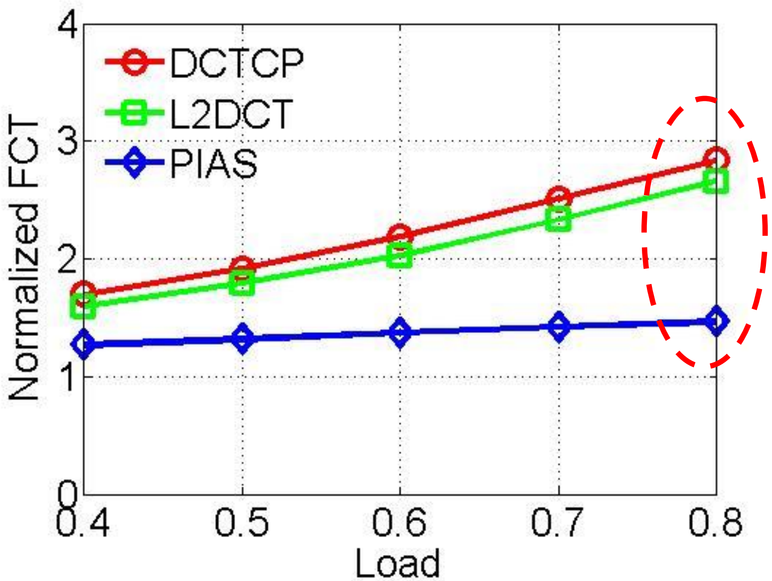
Web Search



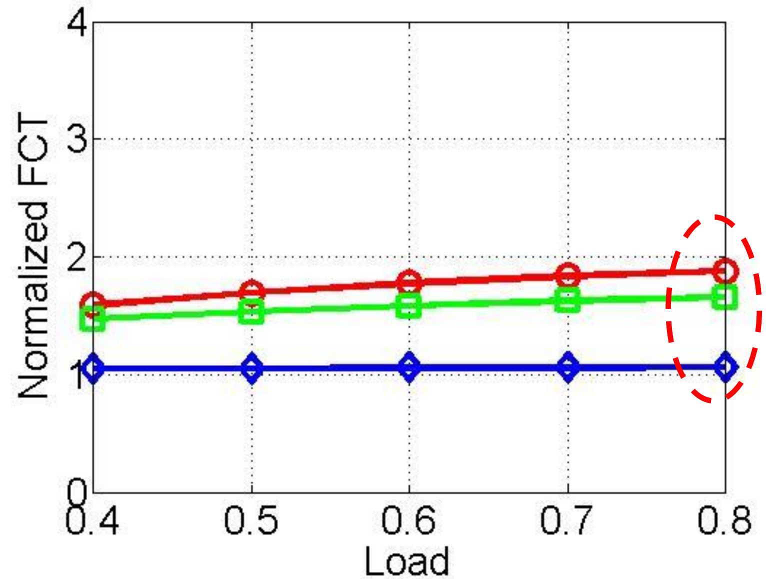
Data Mining

PIAS has an obvious advantage over DCTCP and L2DCT in both workloads.

Small Flows (<100KB)



Web Search

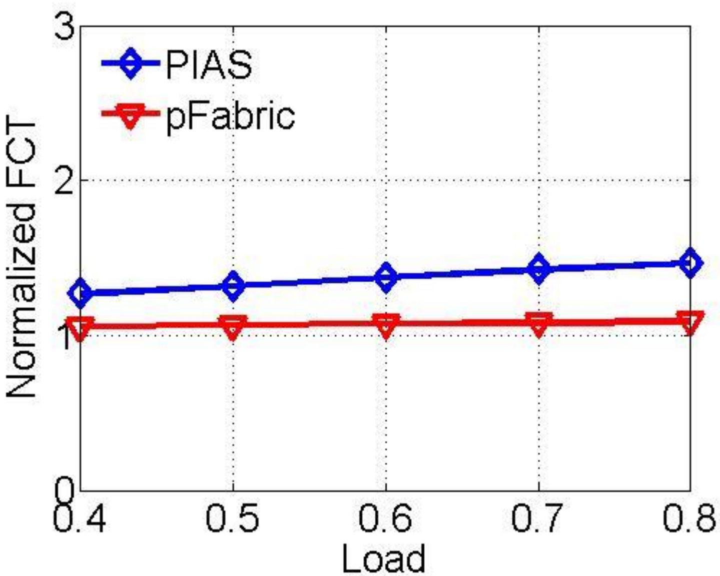


Data Mining

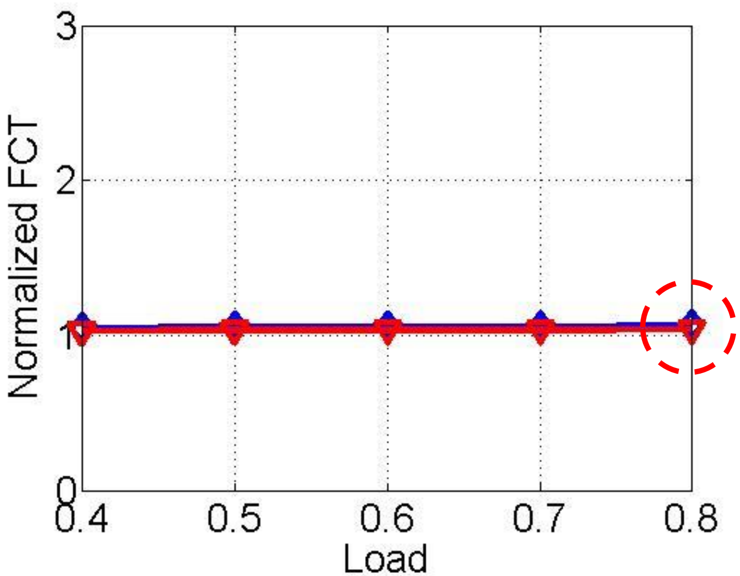
Simulations confirm testbed experiment results

40%-50% improvement

Comparison with pFabric



Web Search



Data Mining

PIAS only has 4.9% performance gap to pFabric for small flows in data mining workload

Conclusion

- PIAS: practical and effective
  - Not assume flow information from applications
    - **Information-agnostic**
  - Enforce Multi-Level Feedback Queue scheduling
    - **FCT minimization**
  - Use commodity switches & legacy network stacks
    - **Readily deployable**

Thanks!

Starvation

- Measurement
  - 5000 flows, 5.7 million MTU-sized packets
  - 200 timeouts, 31 two consecutive timeouts
- Solutions
  - Per-port ECN pushes back high priority flows when many low priority flow get starved
  - Treating a long-term starved flow as a new flow

Persistent Connections

- Solution: periodically reset flow states based on more behaviors of traffic
  - When a flow idles for some time, we reset the bytes sent of this flow to 0.
  - Define a flow as packets demarcated by incoming packets with payload within a single connection