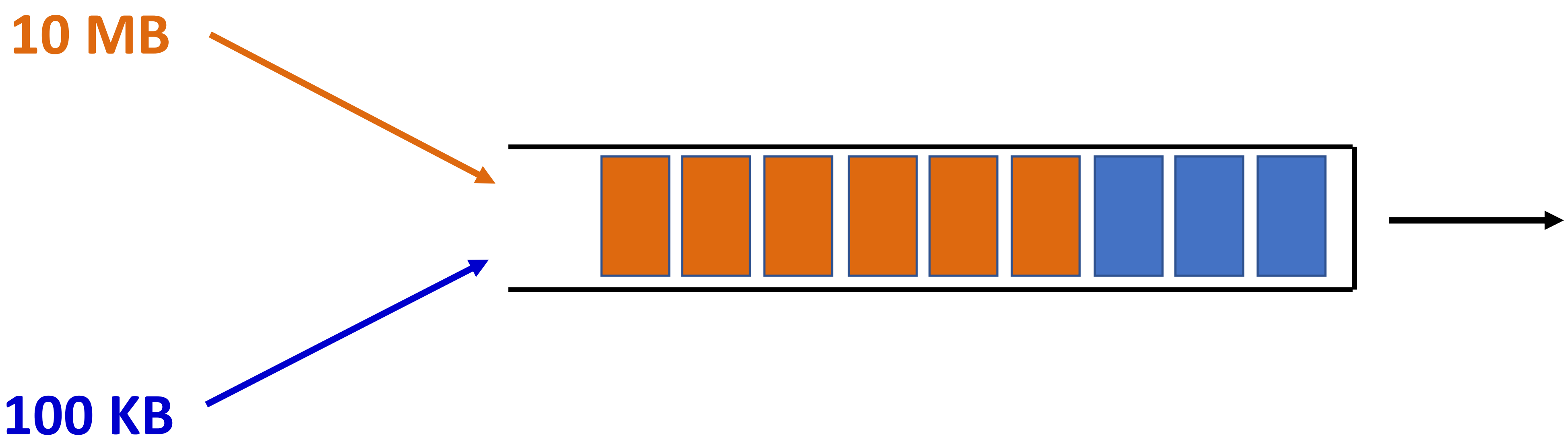
The 21th USENIX Symposium on Networked Systems Design and Implementation (NSDI'24), April, 2024

Flow Scheduling with Imprecise Knowledge

Wenxin Li, Xin He, Yuan Liu, Keqiu Li, Kai Chen, Zhao Ge,

Zewei Guan, Heng Qi, Song Zhang, Guyue Liu

Flow Scheduling



- **Flow scheduling: an effective scheme for datacenter transport**
  - Goal: Minimize flow completion time (FCT)
  - Measure: Prioritize pkts. of small flows over large ones in switch buffer

Existing solutions

Existing flow scheduling proposals

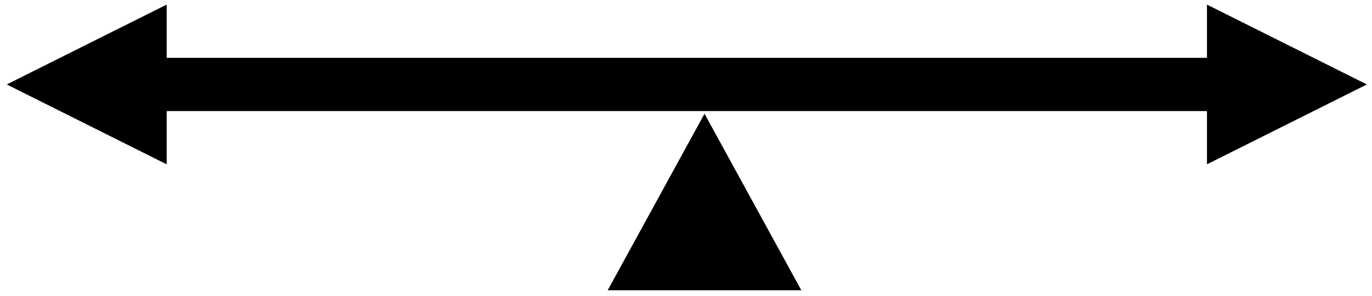
- **Clairvoyant schedulers**
  - E.g., pFabric [SIGCOMM'17]
  - E.g., pHost [CoNext'15]
  - E.g., Homa [SIGCOMM'18]
  - …

Require too many priorities or refactoring TCP/IP network stack

- **Non-Clairvoyant schedulers**
  - E.g., PIAS [NSDI'15]
  - …

Readily-deployable but has limited ability in minimizing FCT

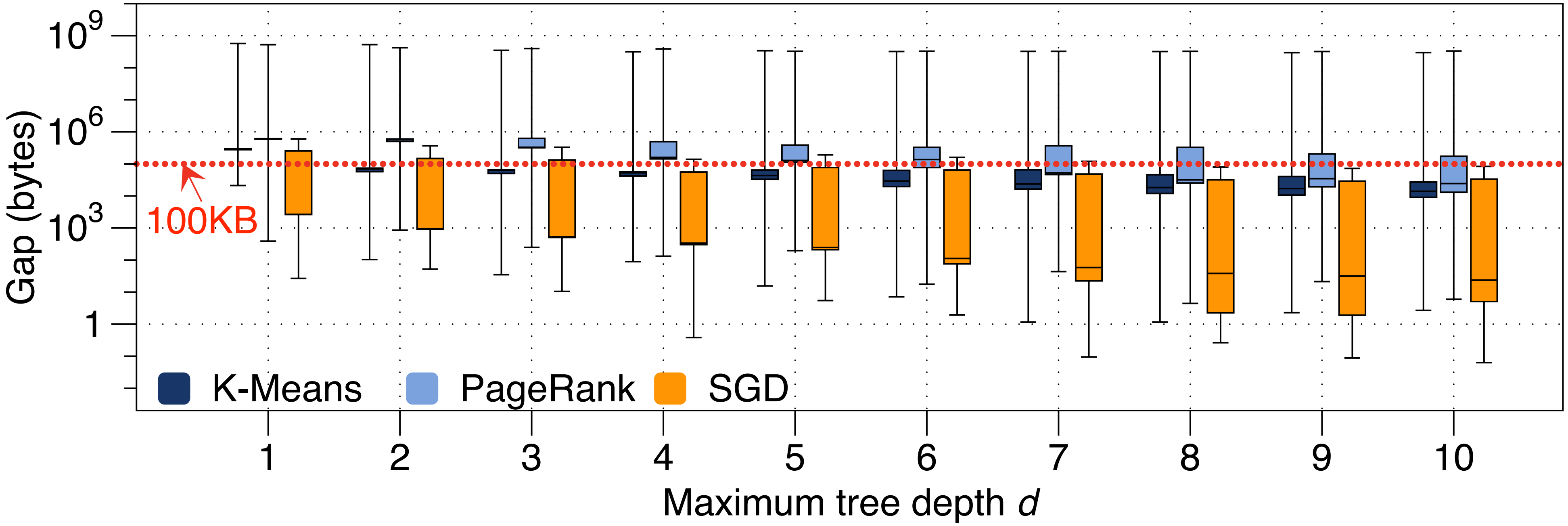
Question



A middle-point design space:

Can we use *imprecise flow information* to minimize FCT with commodity switches?

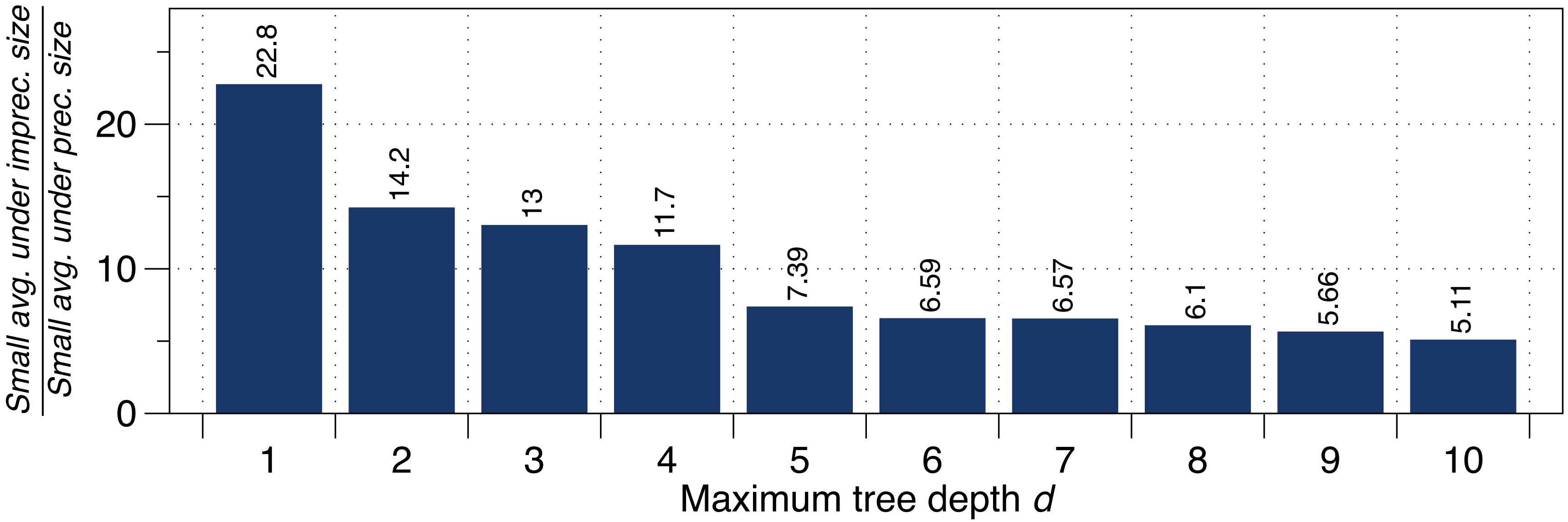
Imprecise Flow Information



- **Estimated flow sizes of ML models are often imprecise**
  - There is a gap between actual and estimated flow sizes
  - E.g., at least 34% of flows have a gap of over 100KB to their estimated sizes

How to Use Imprecise Flow Information

Directly taking it as scheduling input, e.g., Flux [NSDI'19]



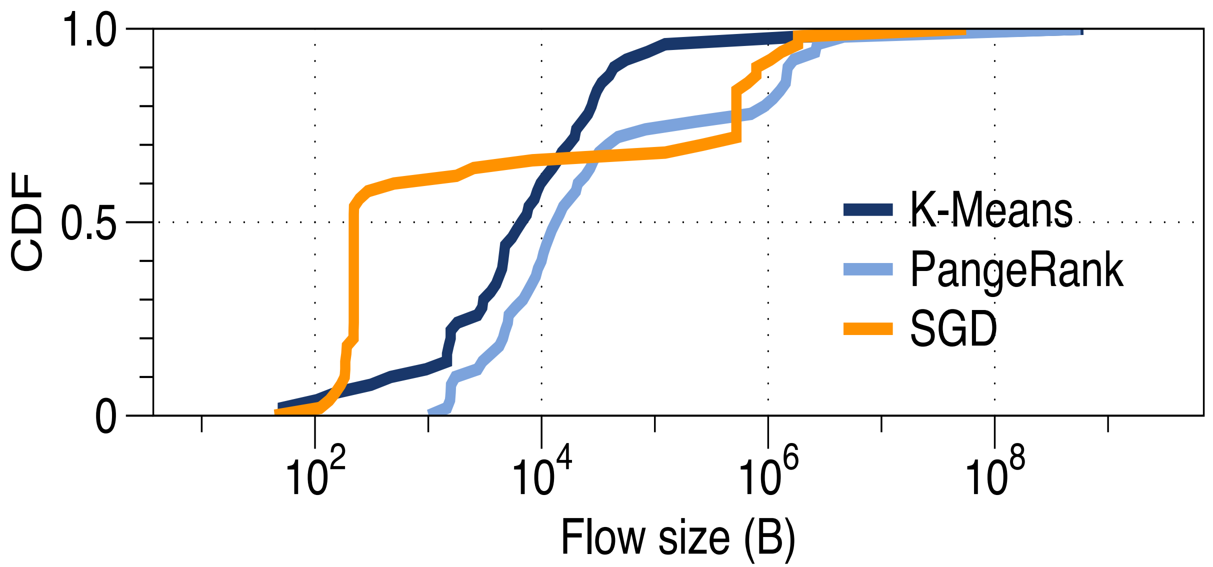
The avg. FCT of small flows can be slowed down by up to ~23x

The lower bound part of imprecise flow info. is accurate

Lower Bound: APP Perspective

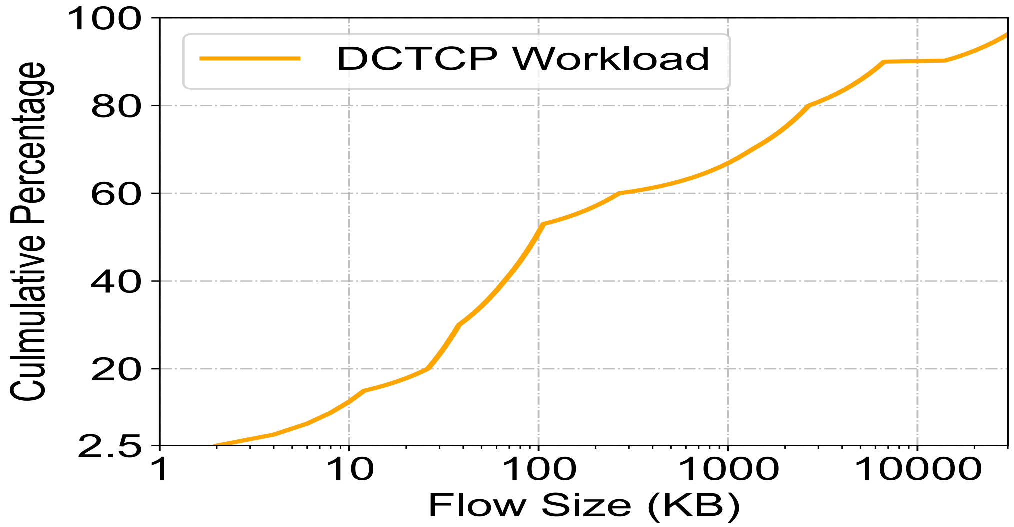
Distributed ML:

Transfer at least one parameter, **$\geq$40 B**



Large-scale web search

Transfer at least one page index, **$\geq$ 1.6 KB**



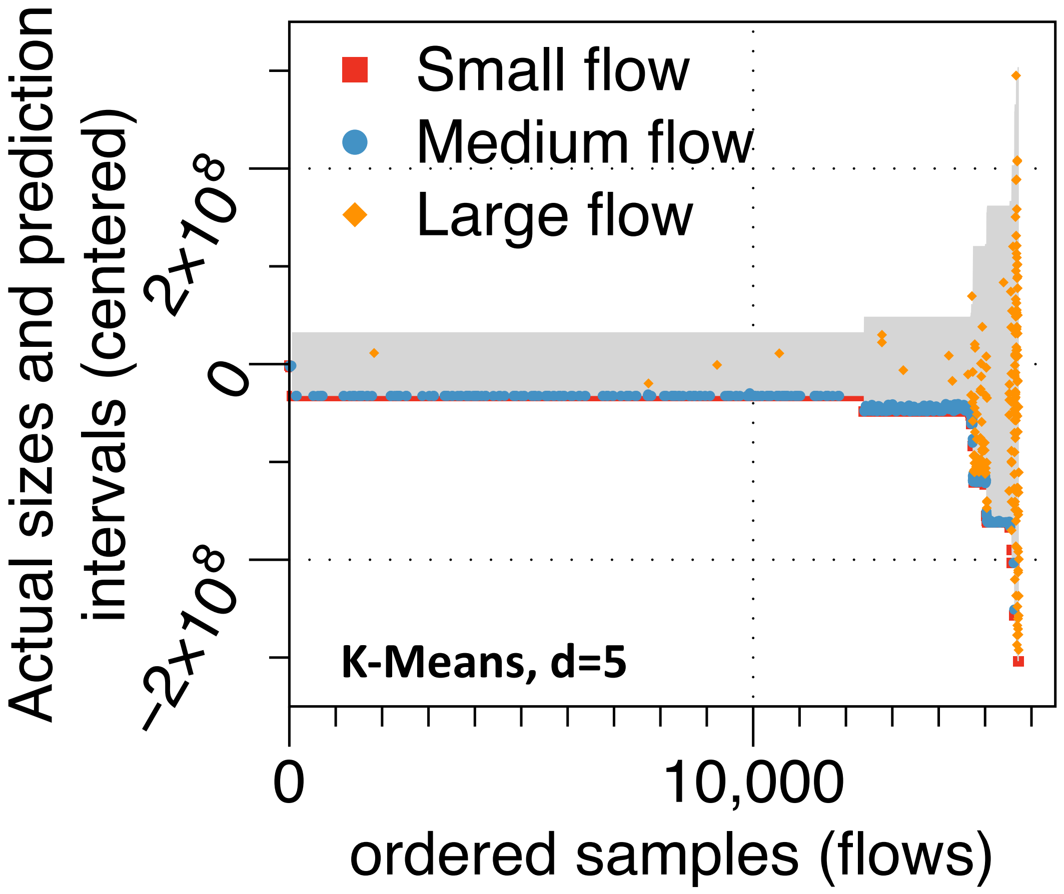
Lower Bound: Experimental Observation

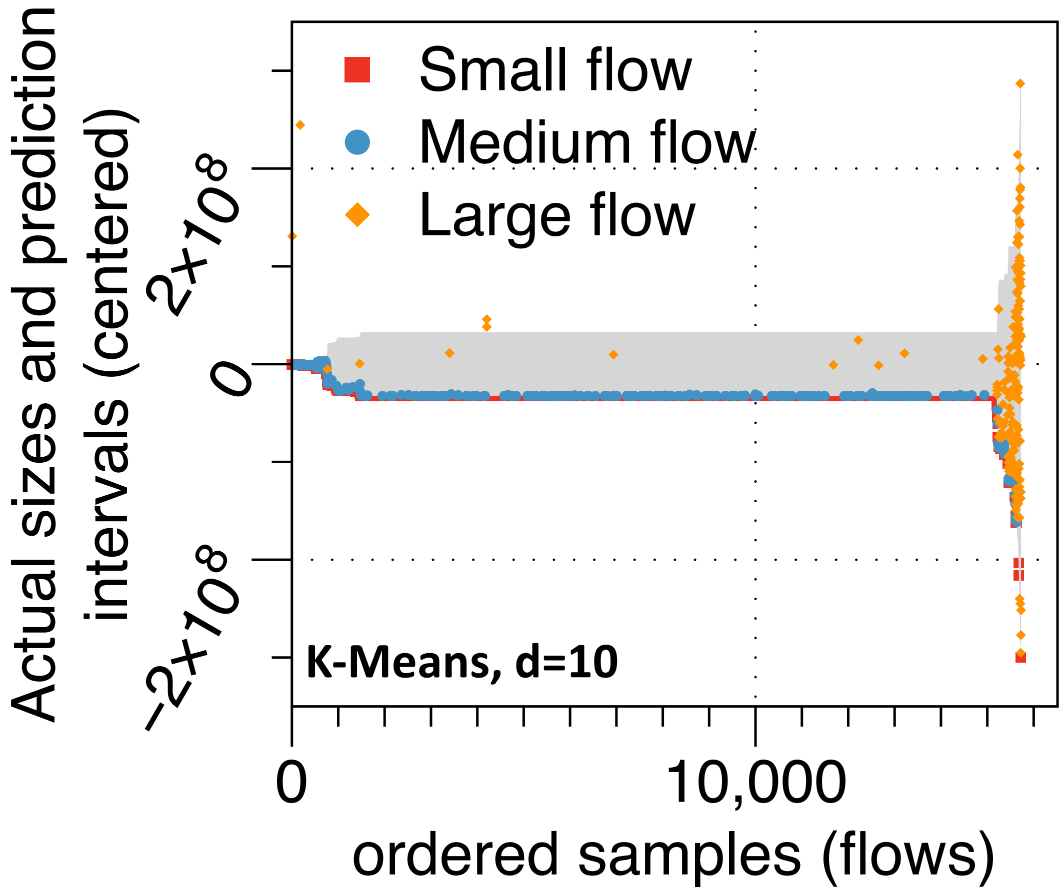
#1: with RF model, > 99.9% flows can be accurately bounded

Workloads		The ratio of bounded flows	The Ratio of out-bound flows
K-Means	d=5	99.9936%	0.0064%
	d=10	99.9301%	0.0699%
PageRank	d=5	100%	0
	d=10	99.9055%	0.0945%
SGD	d=5	100%	0
	d=10	99.9877%	0.0123%

Lower Bound: Experimental Observation

#2: Small flow's actual sizes are mostly close to their lower bounds





…see our paper for more observations

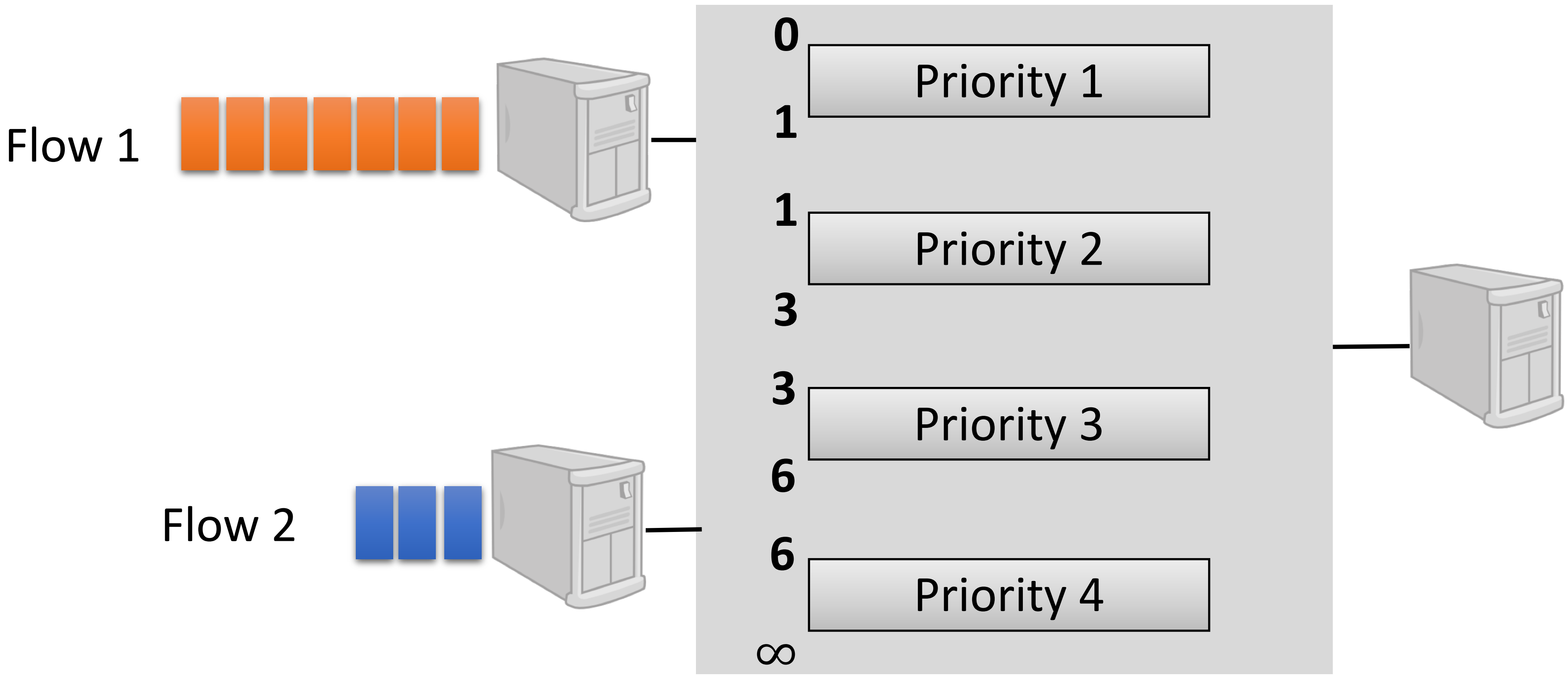
QCLIMB's Design

Lower-bound-based Scheduling

- **Queue-climbing-up phase**
  - Gradually promoting a flow's priority based on its remaining data size relative to the lower bound
- **Queue-climbing-down phase**
  - Gradually demoting a flow's priority based on its bytes sent

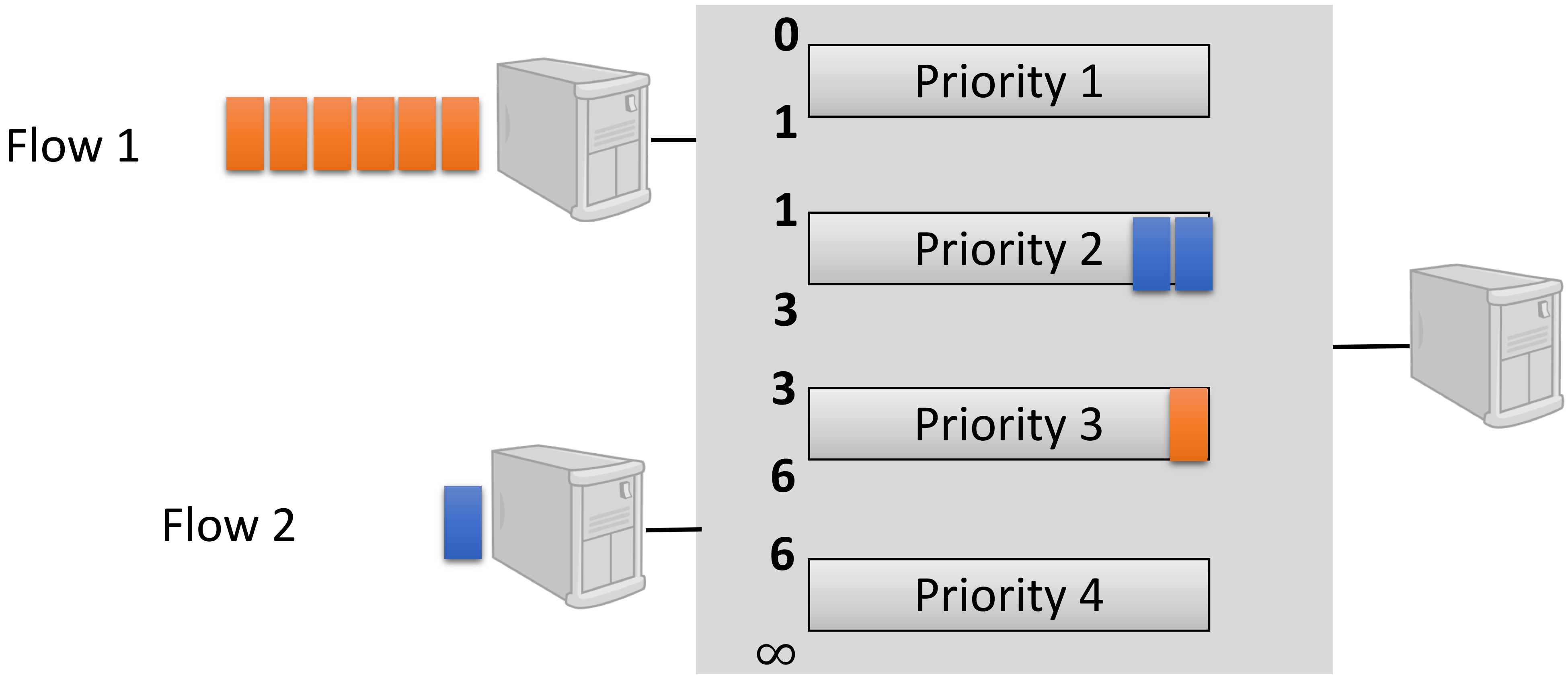
Lower-bound-based Scheduling

Flow 1 with 7 packets (LB=4) and flow 2 with 3 packets (LB=3) arrive



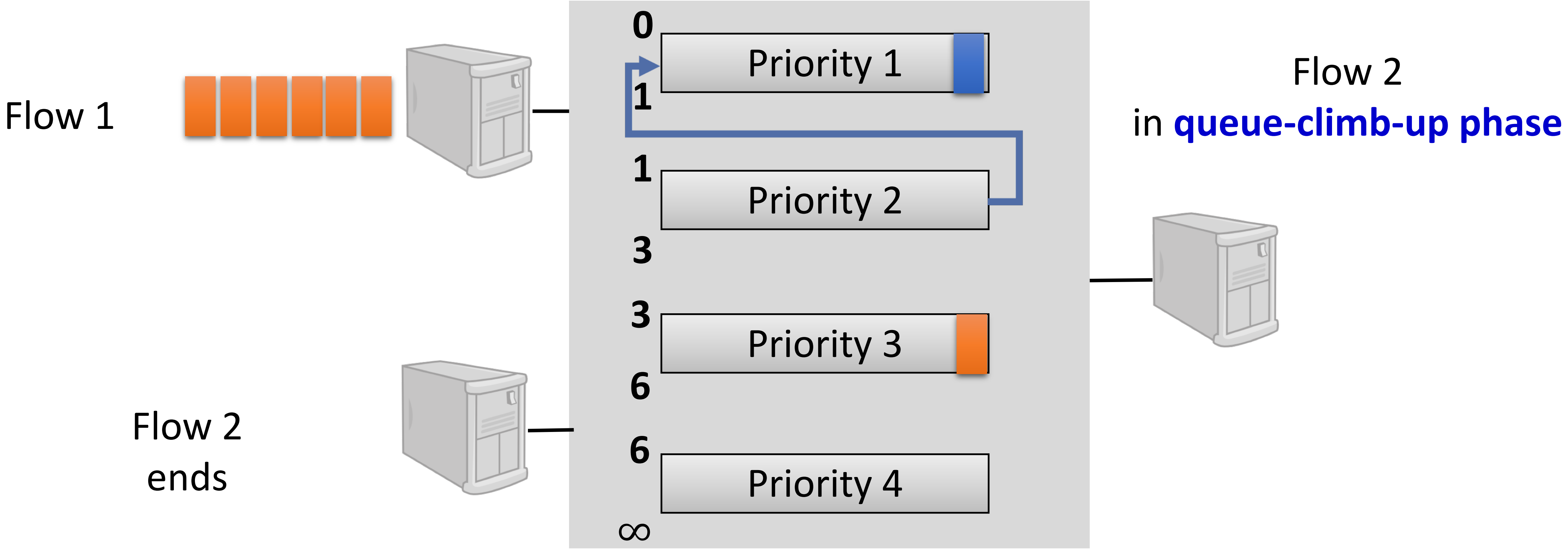
Lower-bound-based Scheduling

Based on lower bounds, flow 1 enters priority queue 3 while flow 2 enters priority queue 2



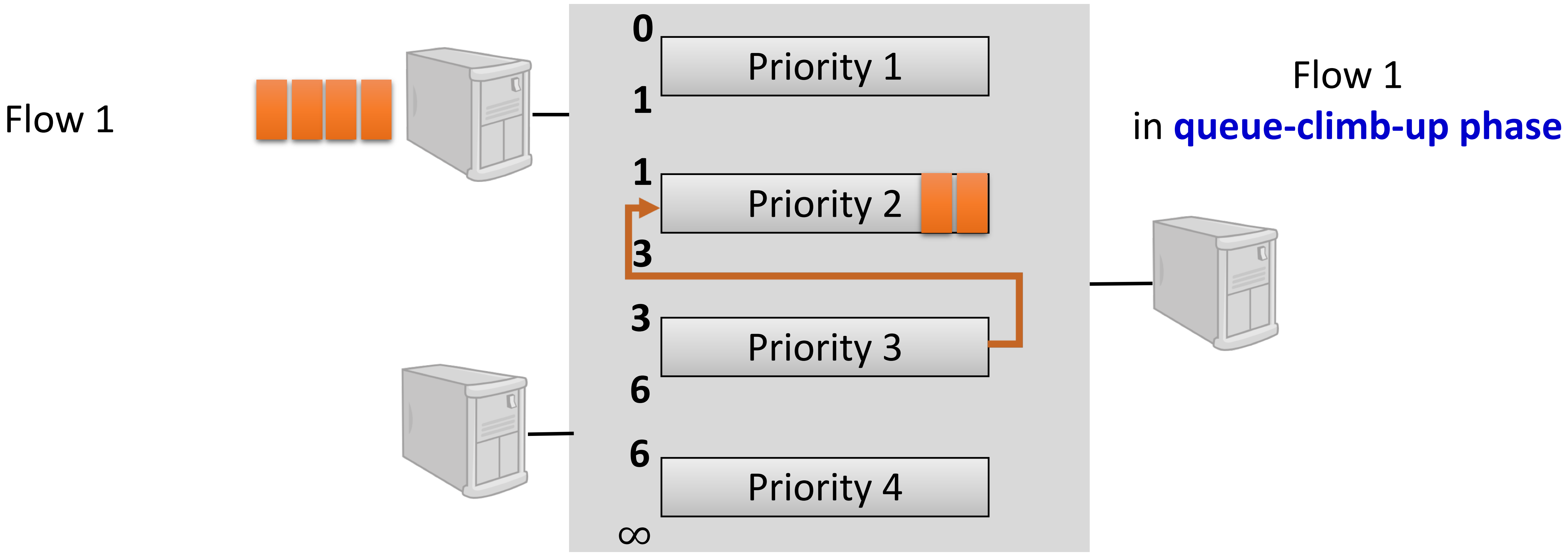
Lower-bound-based Scheduling

flow 2 is promoted to priority 1 (and finishes in this priority) while flow 1 is still in priority 3



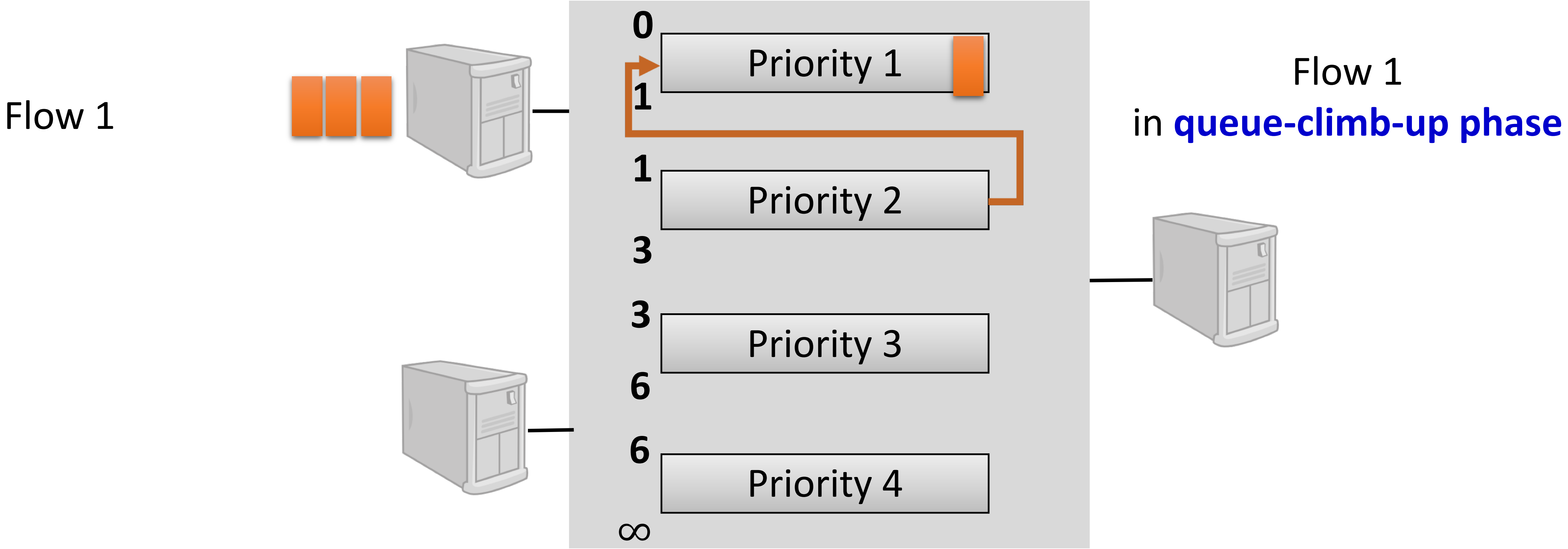
Lower-bound-based Scheduling

flow 1 is promoted to priority 2, where it transmits 2 pkts.



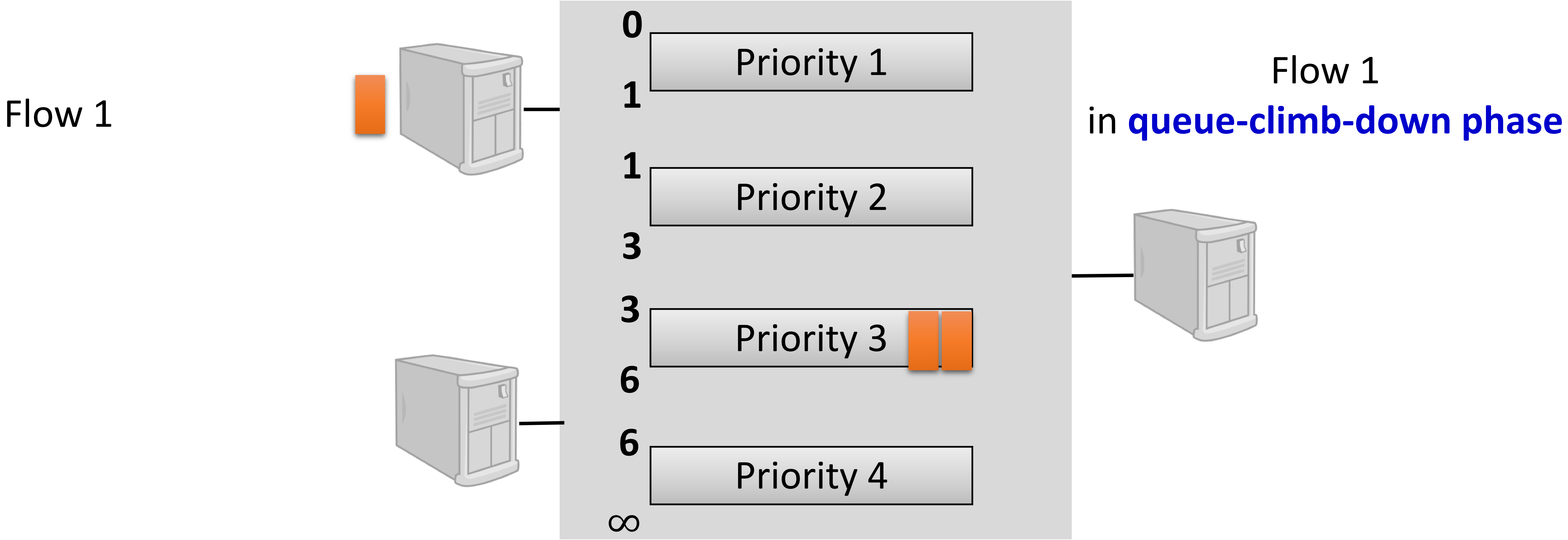
Lower-bound-based Scheduling

flow 1 is promoted to priority 1; lower bound part finishes



Lower-bound-based Scheduling

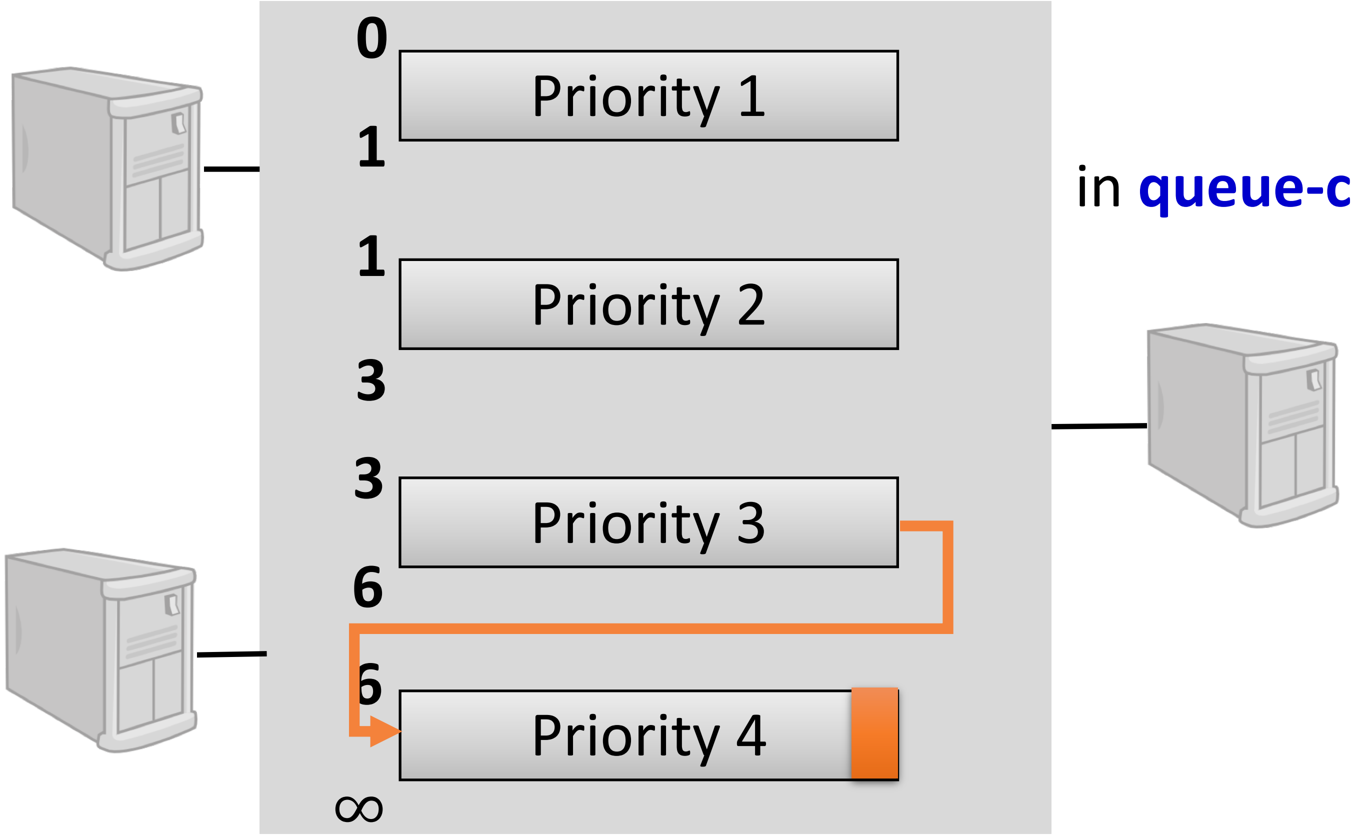
flow 1 is pulled back to its initially entered priority 3



Lower-bound-based Scheduling

flow 1 is demoted to priority 4 and finishes its last pkt. transmission

Flow 1
ends

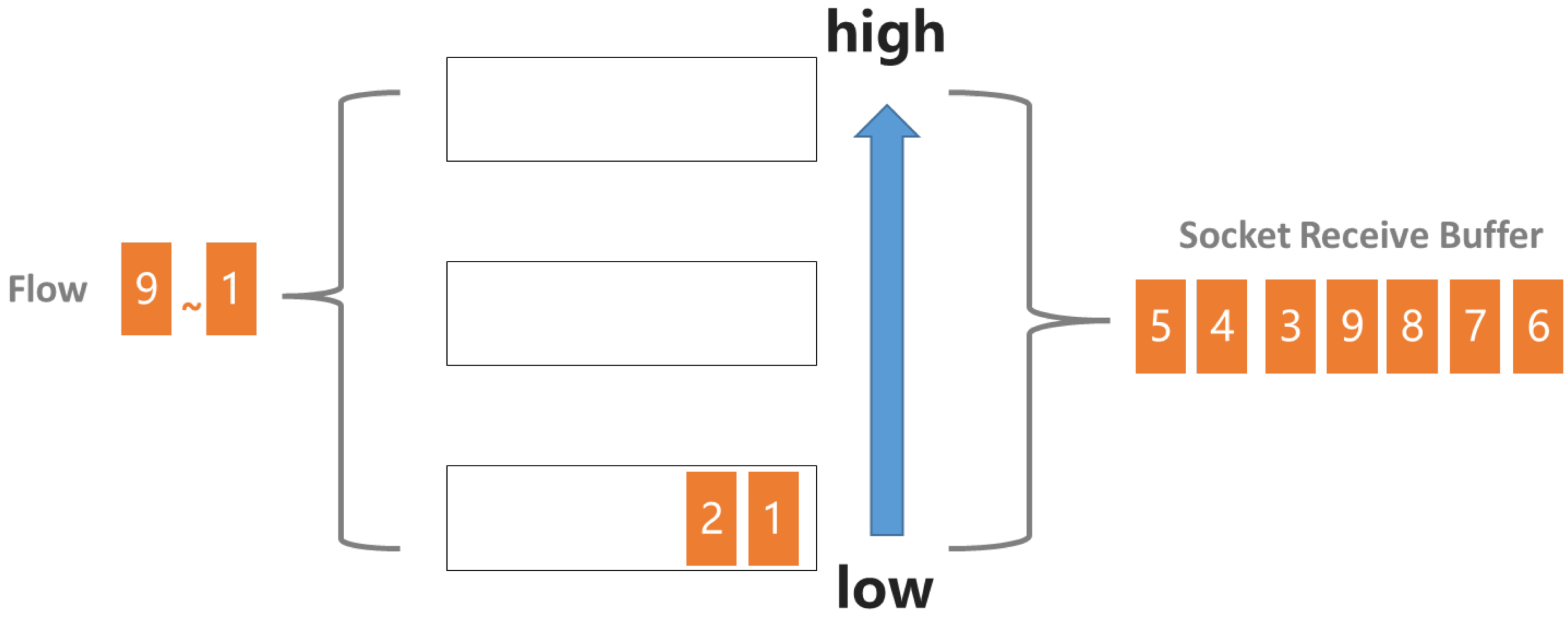


Flow 1
in **queue-climb-down phase**

Out-of-order Handling

Queue-climbing-up allows later pkts. of a flow to enter higher priorities than earlier ones

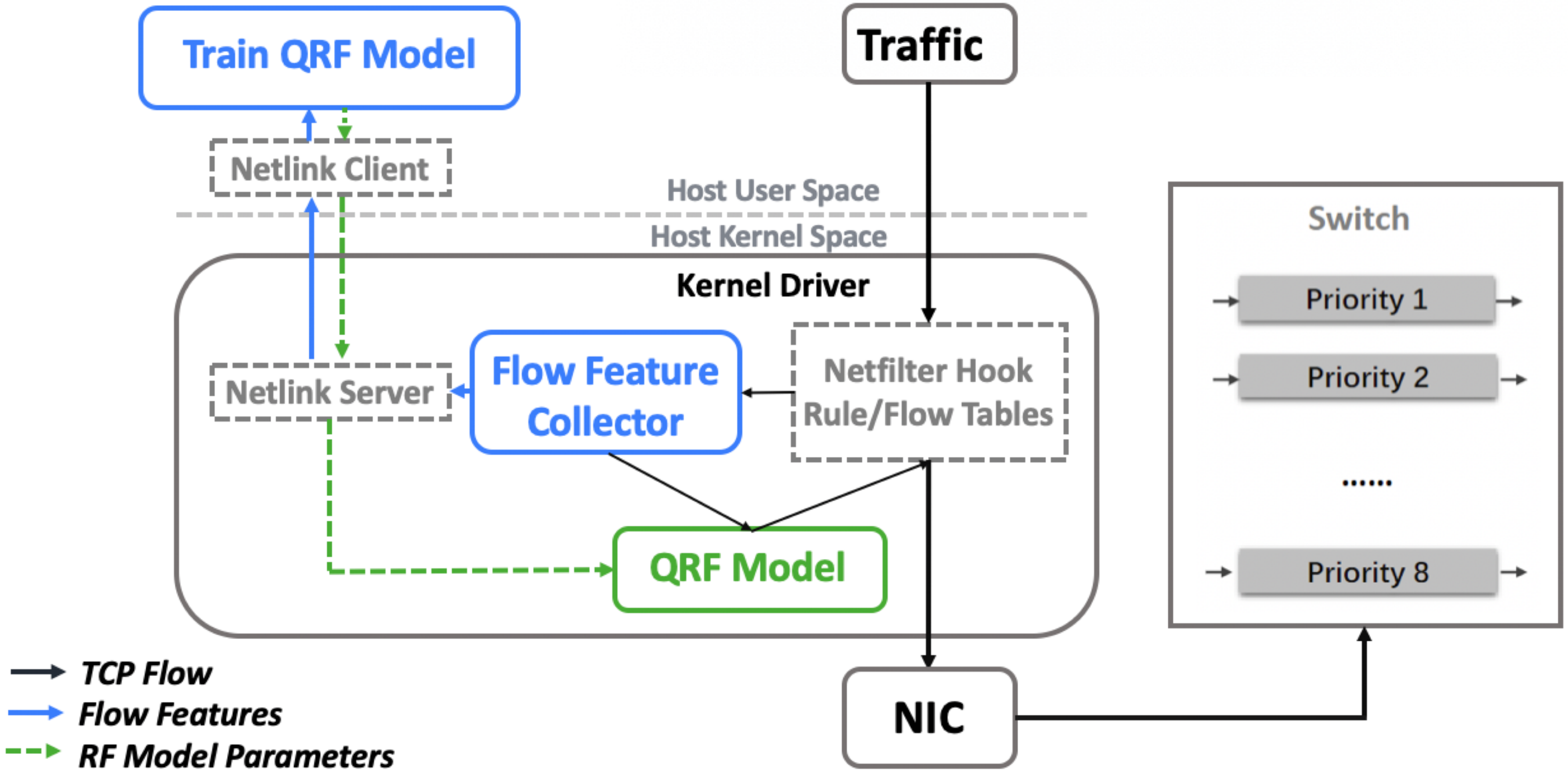
Priority-based loss detection: for a flow, the pkts. carrying the same priority should be in order



Actual packet loss: retransmission is required;
Reorderings: fast & slow path; customized ACKs

Implementation

Only requiring host-side implementation

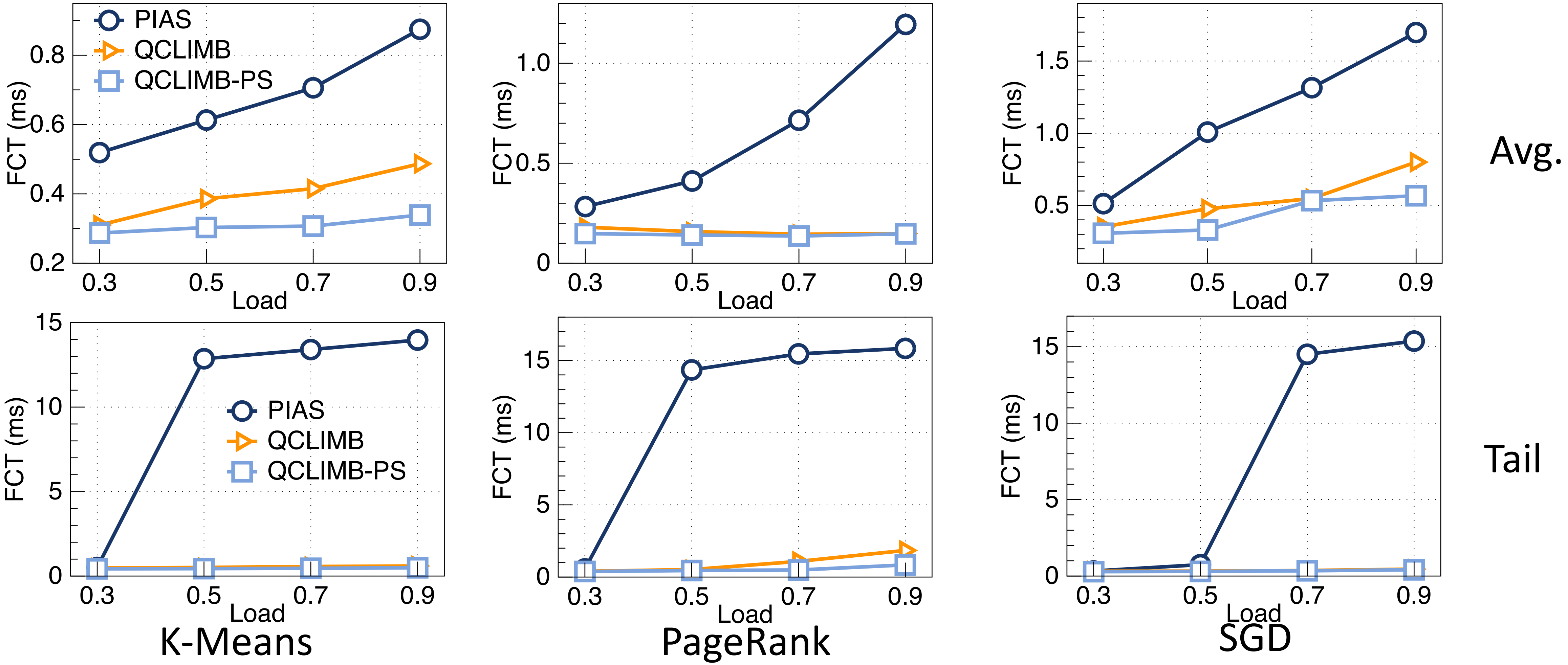


Scheduling, RF model inferencing and OOO handling: all in the kernel

Testbed Experiments

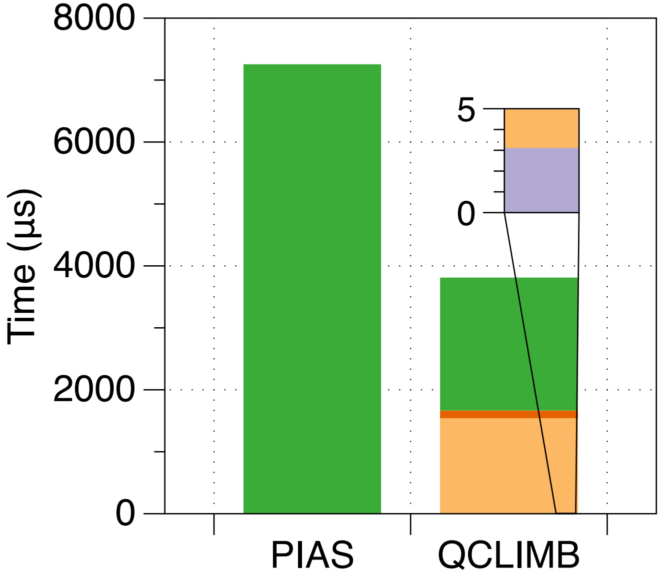
- **Testbed Setup**
  - 1 Barefoot Tofino switch
  - 8 servers with 25G NIC
- **Workloads (from Flux paper)**
  - K-Means
  - PageRank
  - SGD
- **RF model**
  - Maximum tree depth d=10 by default
  - Trained for ~3 minutes over each workload using 80% of the dataset, leaving the remaining for evaluation

Small flows

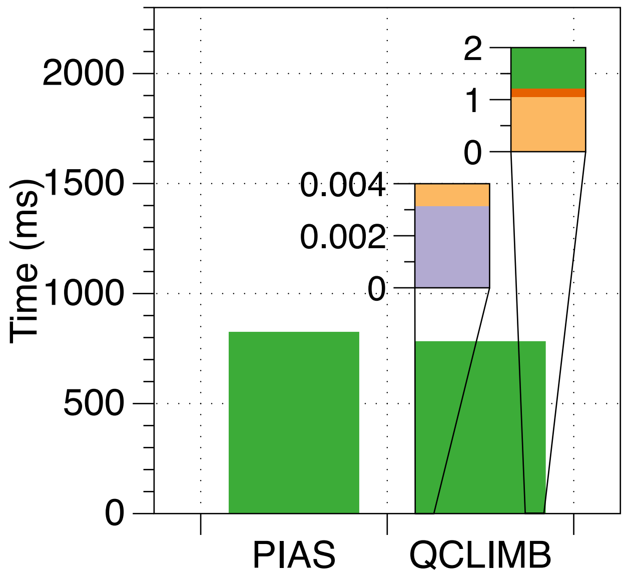


Compared to PIAS, QCLIMB reduces the avg./tail FCT of small flows by up to 88%/97%; Compared to QCLIMB- PS, QCLIMB can deliver a 3% gap for the small average FCT.

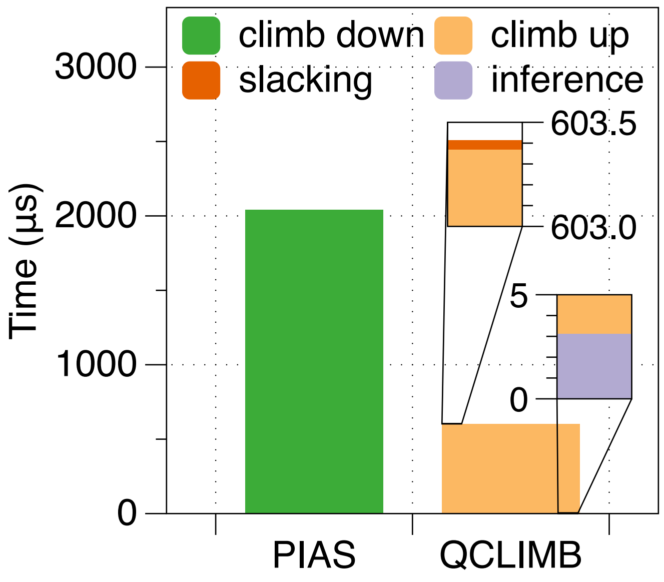
Performance Breakdown



Small flows



Medium flows



Large flows

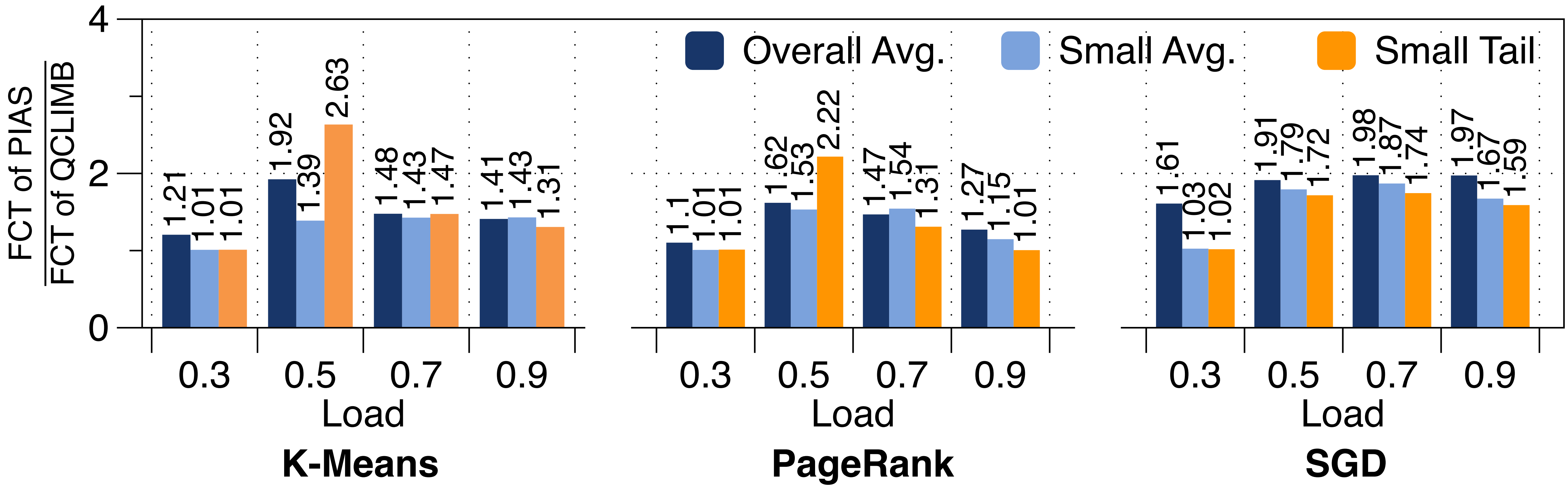
For small flows, the queue-climbing-up phase accounts for 99.47% of the FCT;
RF inference takes ~3 us;

Simulation

- **Topology**
  - 144-host leaf-spine fabric with 40G/100G links

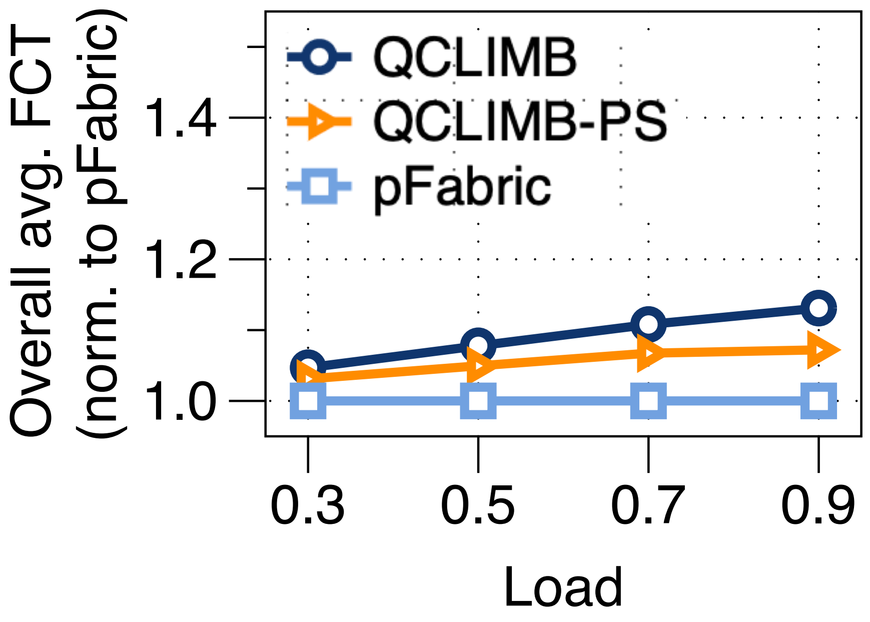
- **Workloads (from Flux paper)**
  - K-Means
  - PageRank
  - SGD

Comparison with PIAS

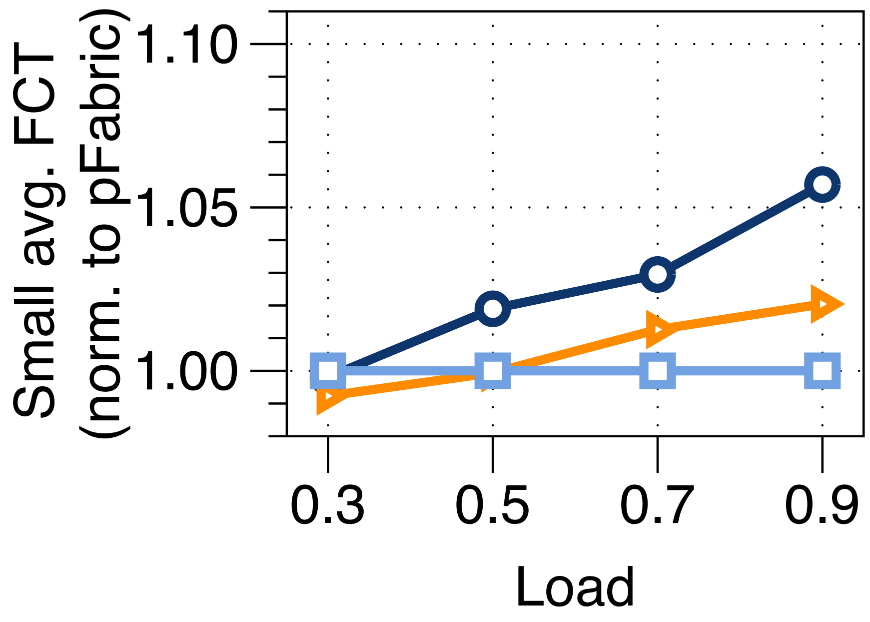


QCLIMB reduces the avg./tail FCT of small flows by up to 46.5%/62%

Comparison with pFabric



Overall flows

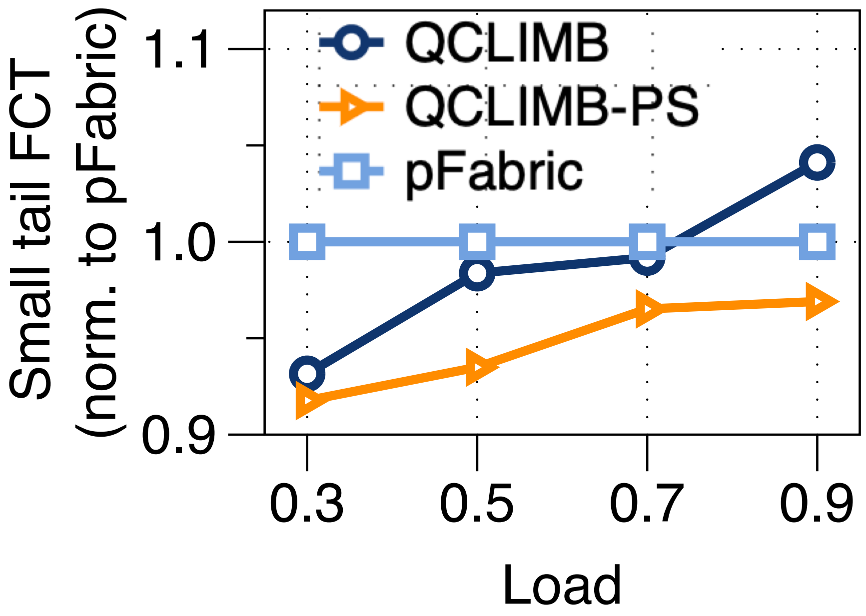


Small flows

Avg. gap to pFabric: 9%

Avg. gap to pFabric: 5.7%

Comparison with pFabric



Flow bins	Small flows	Medium flows	Large flows	All flows
QCLIMB	0	0	124	124 (83 by loss detection)
pFaric	10	25	414	414 (all by timeout)

Loss events comparison

Sometimes lower small tail FCT than pFabric

QCLIMB incurs fewer packet loss events than pFabric

Conclusion

QCLIMB

- **Lower-bound-based Scheduling**
  - Prioritizing small flows over large ones from the start of transmission
- **Out-of-order Handling**
  - Handling reordering issues resulting from the scheduling algorithm

Thanks!

contact: toliwenxin@tju.edu.cn