Enabling Virtual Priority in Data Center Congestion Control

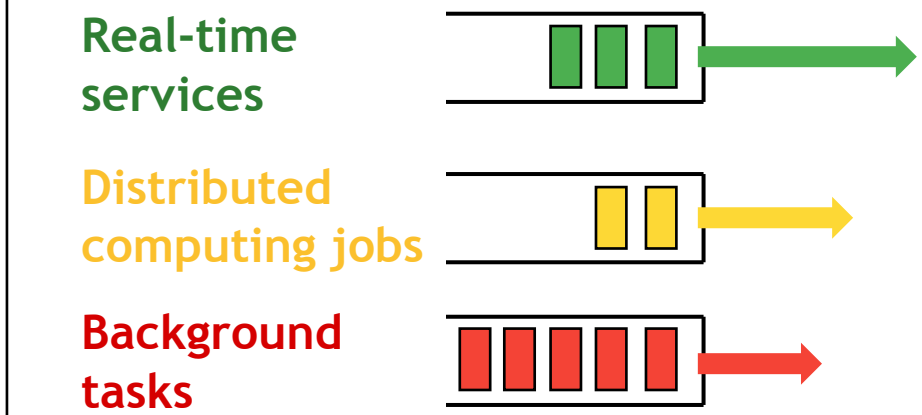
Zhaochen Zhang, Feiyang Xue, Keqiang He, Zhimeng Yin, Gianni Antichi, Jiaqi Gao, Yizhi Wang, Rui Ning, Haixin Nan, Xu Zhang, Peirui Cao, Xiaoliang Wang, Wanchun Dou, Guihai Chen, **Chen Tian**

NANJING UNIVERSITY
SHANGHAI JIAO TONG UNIVERSITY
香港城市大學 City University of Hong Kong
POLITECNICO MILANO 1863
Queen Mary University of London

The Need for a Large Number of Priority Queues

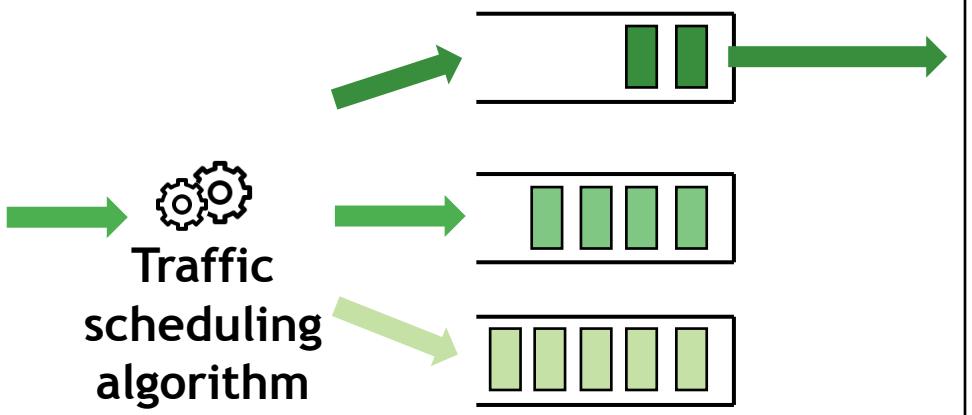
In general, priority queues serve **two** primary purposes:

1 Providing traffic isolation for different traffic classes



Priority queues weighted share the bandwidth

2 Performing traffic scheduling in the same traffic class

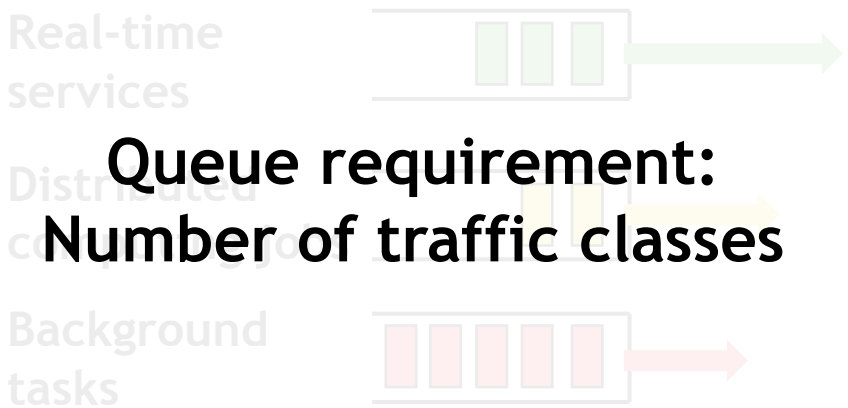


The higher priority queue is strictly prioritized

The Need for a Large Number of Priority Queues

To meet both purposes, a large number of queues are required:

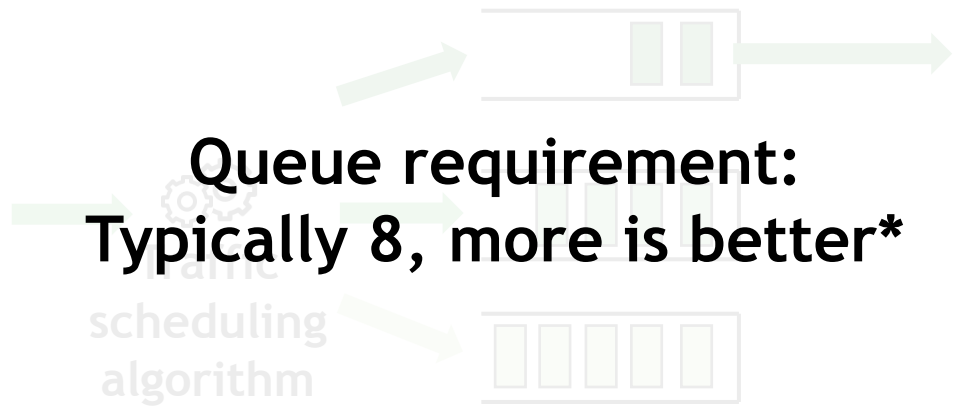
1 **Providing traffic isolation for different traffic classes**



Queue requirement: Number of traffic classes

×

2 **Performing traffic scheduling in the same traffic class**



Queue requirement: Typically 8, more is better*

*PIAS@NSDI'15, QCLIMB@NSDI'24

Total queue demand is the product of # traffic classes and # scheduling priorities

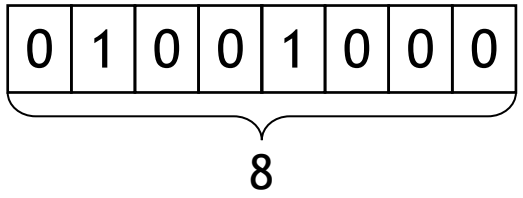
Only a Few Priority Queues Are Offered

Direct reason: Priority-related protocols limit the number of priority queues.

1

PFC only supports 8 priorities in its header format

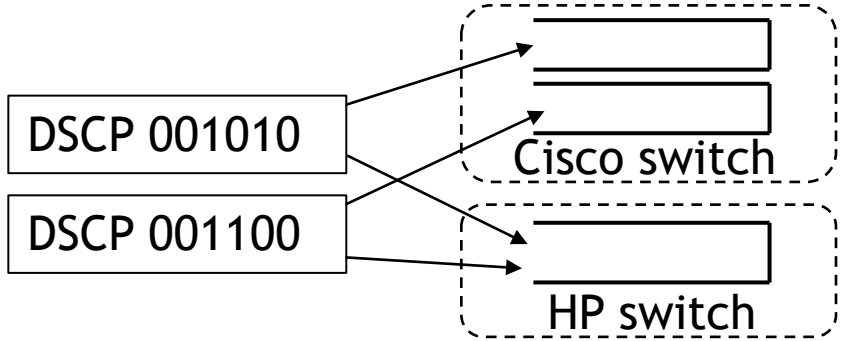
PFC enable vector



PFC frames utilize an 8-bit bitmap to denote the paused or resumed priorities.

2

DSCP only supports 12 priority mapping as per the standard

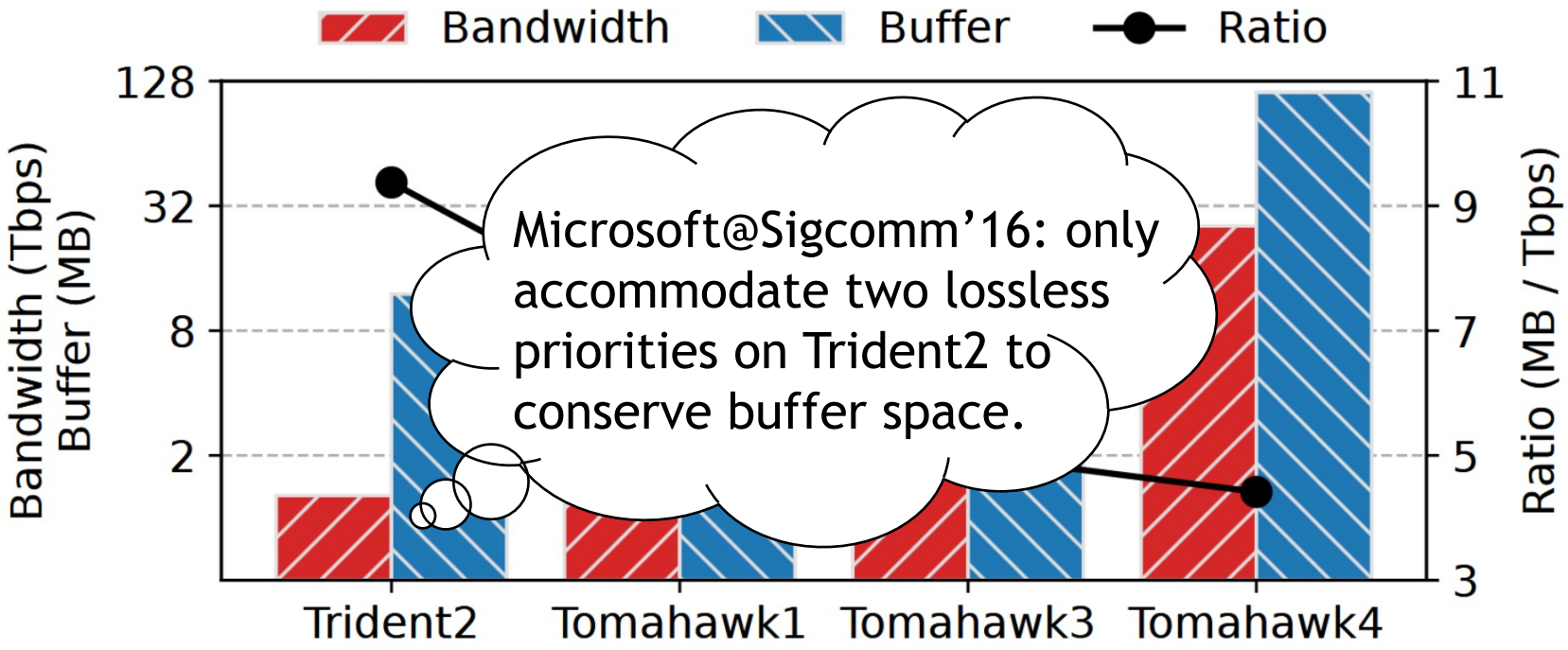


Non-standard DSCP map to different queues across different manufacturers' switches.

Current commercial switches typically support only 8 or 12 priority queues.

Only a Few Priority Queues Are Offered

Trend: The decreasing buffer-to-bandwidth ratio in switches limits the number of priority queues.



Virtual Priority is an Attractive Solution to the Shortage

Virtual priority: the capability of supporting multiple strict priorities within a single physical priority queue.

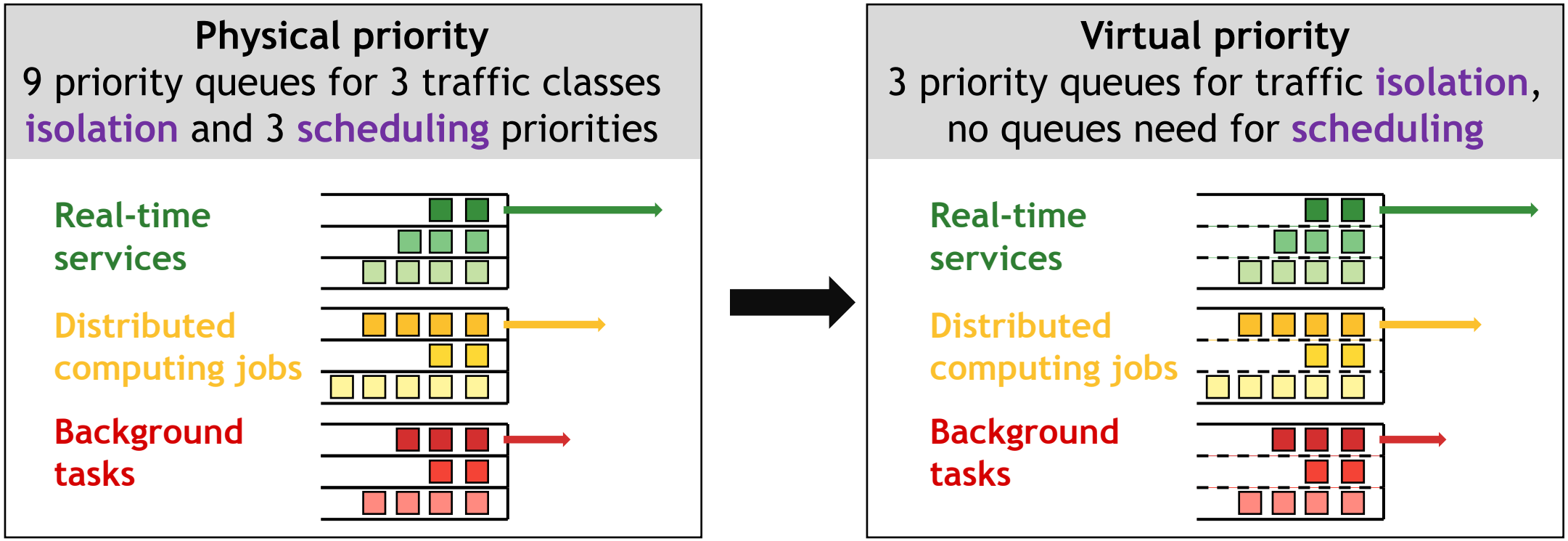
Virtual priority must achieve the following objectives:

- *O1*: **Multi-priority assurance**.
  The virtual priority mechanism must ensure strict prioritization of higher-priority traffic.

- *O2*: **Work conservation.**
  The virtual priority mechanism should aim to fully utilize bandwidth without any waste.

Virtual Priority is an Attractive Solution to the Shortage

With virtual priority, the limited physical priority queues could be reserved for isolation across different traffic classes.



Existing Virtual Priority Solutions are not Readily Deployable

Existing virtual priority technologies primarily involve packet scheduling algorithms that operate on switches. They necessitate:

Programmable switches

N. K. Sharma et al. @NSDI20
AIFO@Sigcomm21, HCSFQ@NSDI21, ……

Figure source: Intel Tofino2

or

Additional specialized hardware

pFabric@Sigcomm13, PIFO@Sigcomm16,
BBQ@NSDI24, vPIFO@Sigcomm24, ……

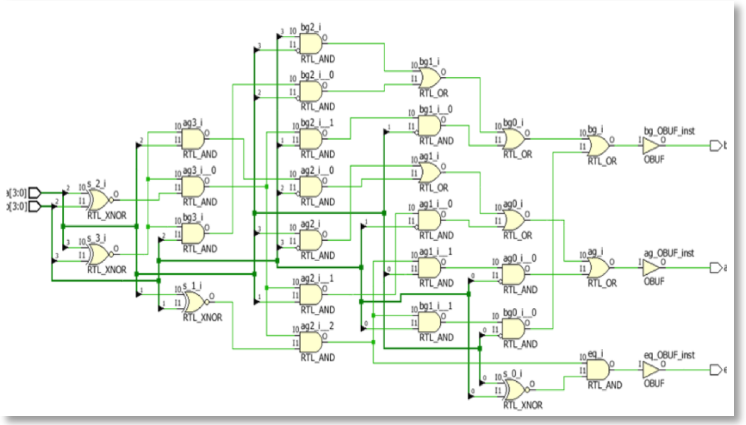


Figure source: BBQ@NSDI24

Existing Virtual Priority Solutions are not Readily Deployable

Existing virtual priority technologies primarily involve packet scheduling algorithms that operate on switches. However, they necessitate:

Can we achieve virtual priority without switch replacement/upgrades?

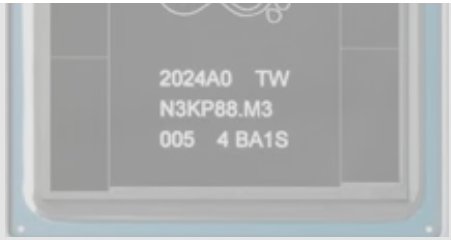


Figure source: Intel Tofino2

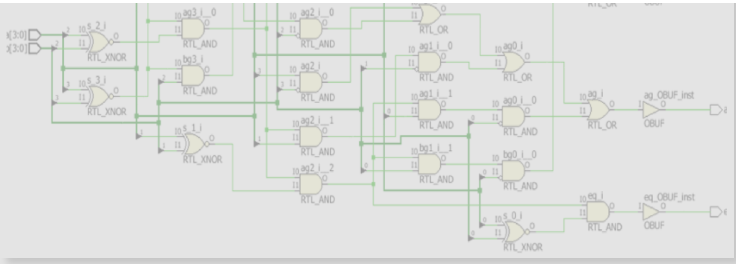


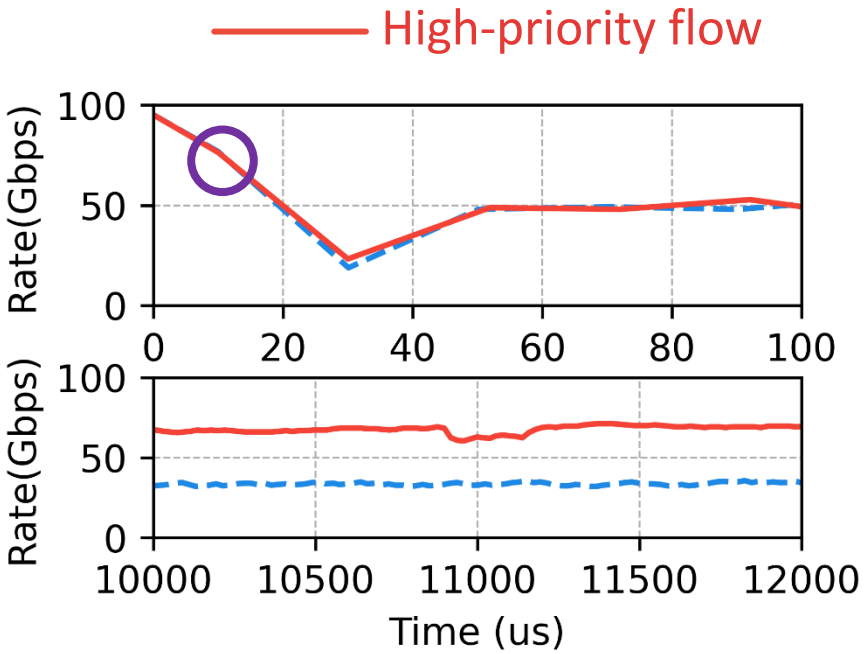
Figure source: BBQ@NSDI24

Our Insight

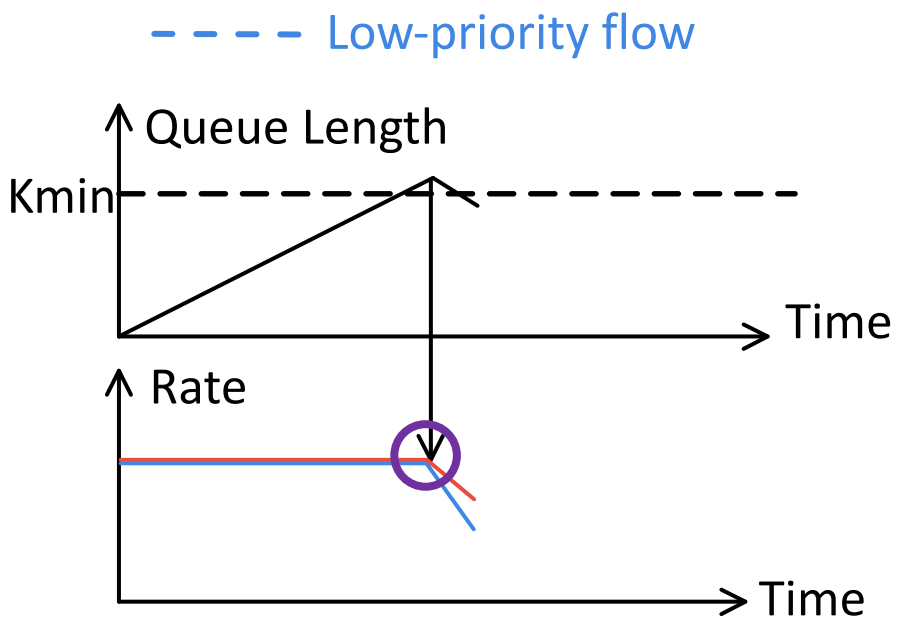
1. Virtual priority can be achieved by carefully **managing bandwidth contention** among flows within a physical priority queue, a task traditionally handled by the **Congestion Control (CC)** at the host.

Existing CC can not Achieve Virtual Priority

D2TCP, the state-of-the-art data center CC that can prioritize a proportion of traffic.



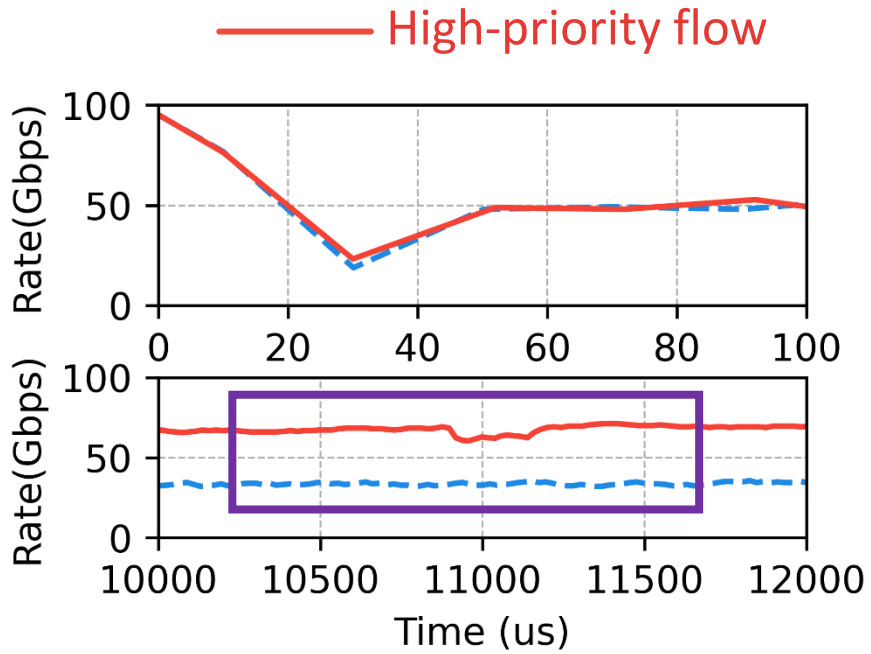
2 D2TCP flows in the simulation



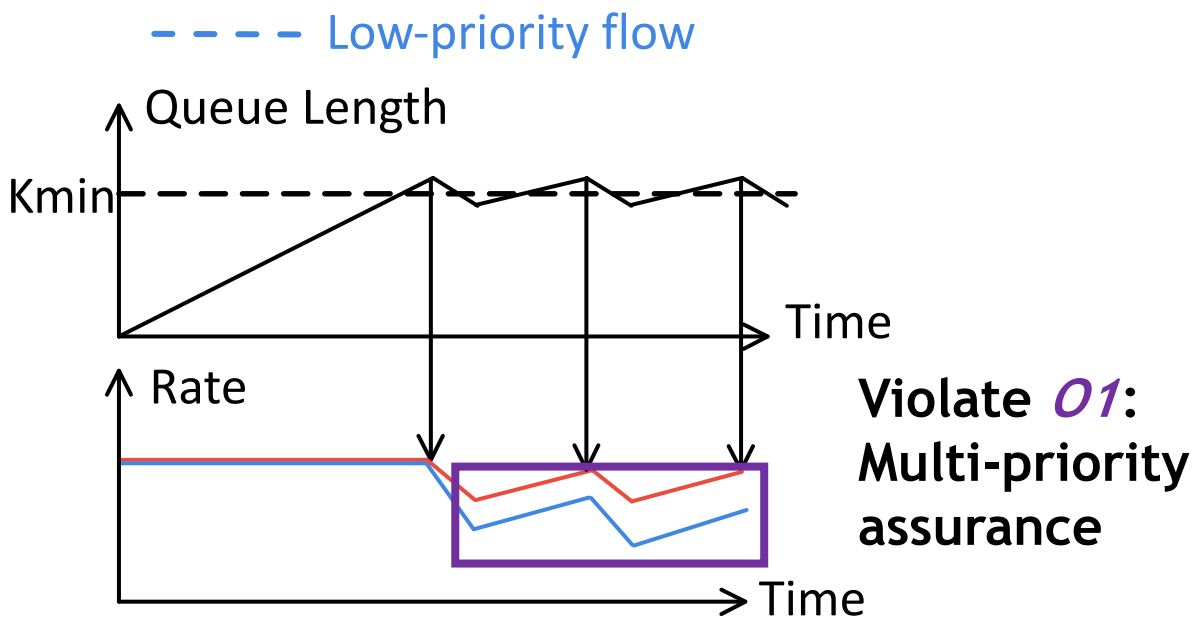
Analysis of D2TCP's behavior

Existing CC can not Achieve Virtual Priority

D2TCP, the state-of-the-art data center CC that can prioritize a proportion of traffic.



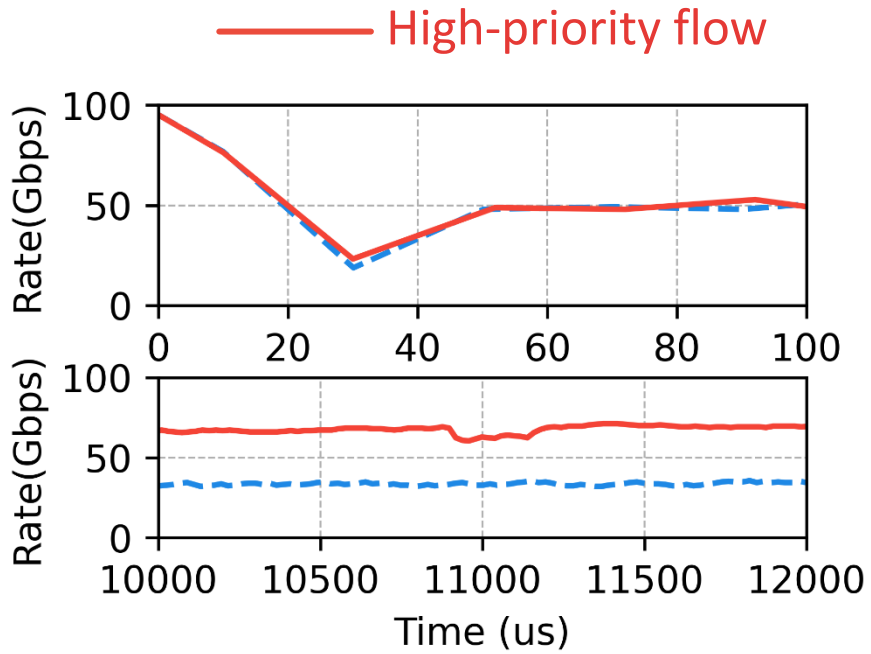
2 D2TCP flows in the simulation



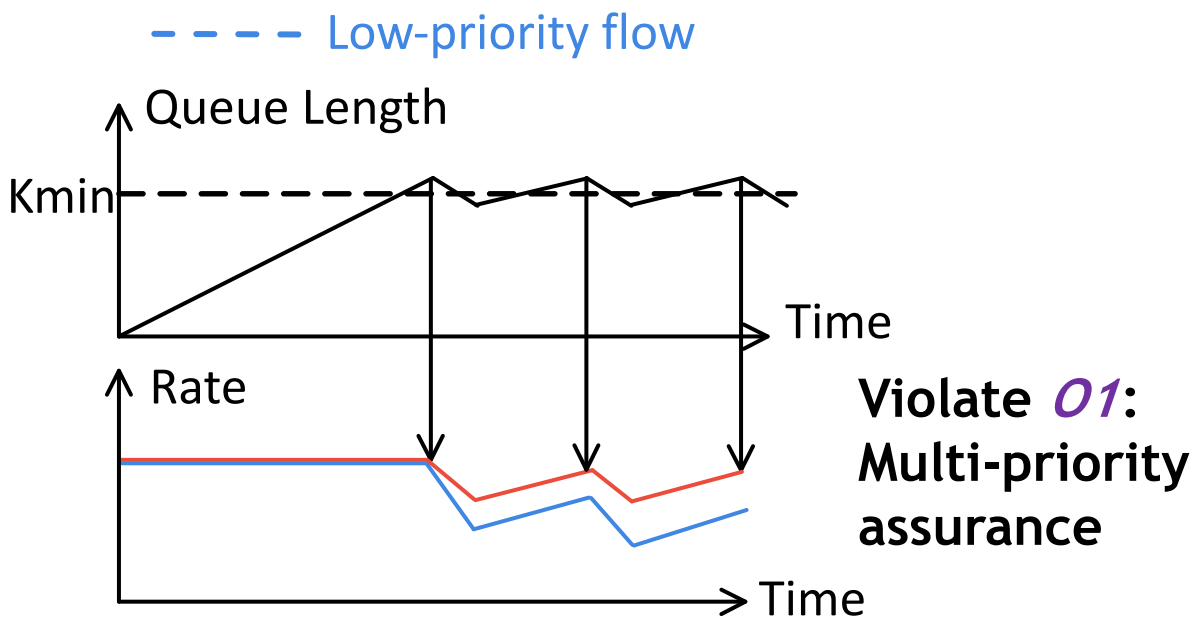
Analysis of D2TCP's behavior

Existing CC can not Achieve Virtual Priority

CC based on the single-bit congestion signal can not achieve virtual priority.



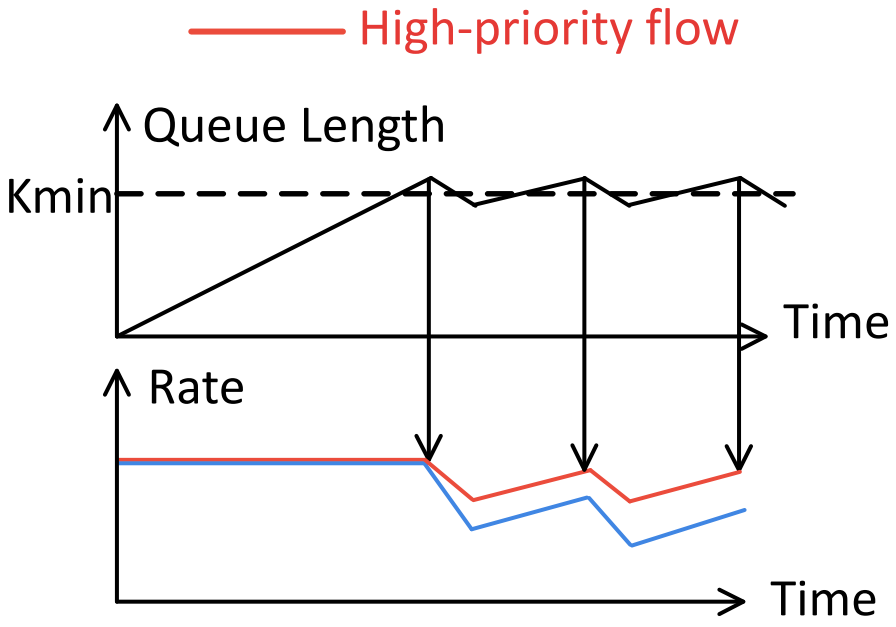
2 D2TCP flows in the simulation



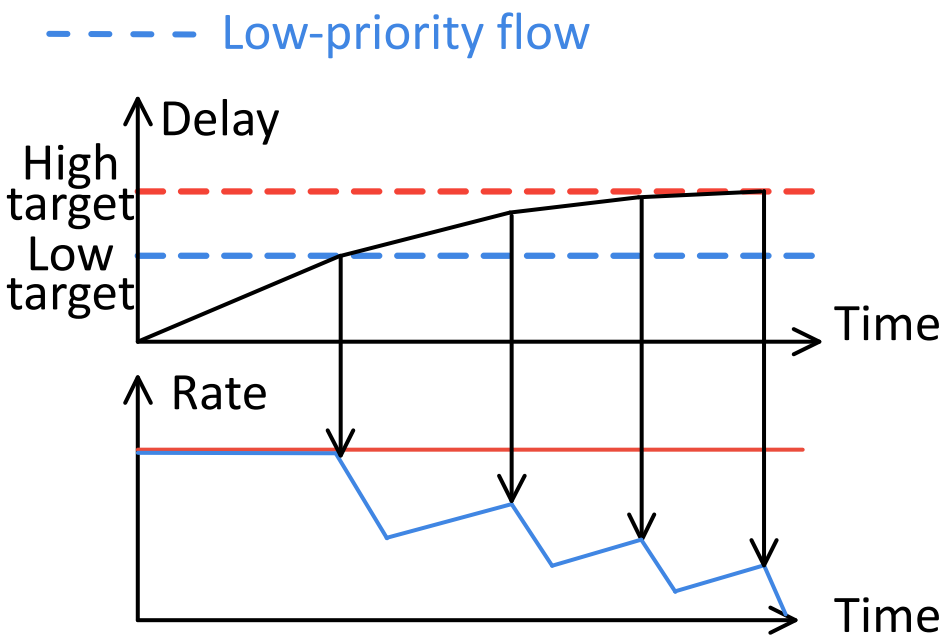
Analysis of D2TCP's behavior

Virtual Priority CC Needs Multi-bit Congestion Signal

Ideally, setting higher congestion thresholds for higher priorities can achieve virtual priority.



Analysis of D2TCP's behavior



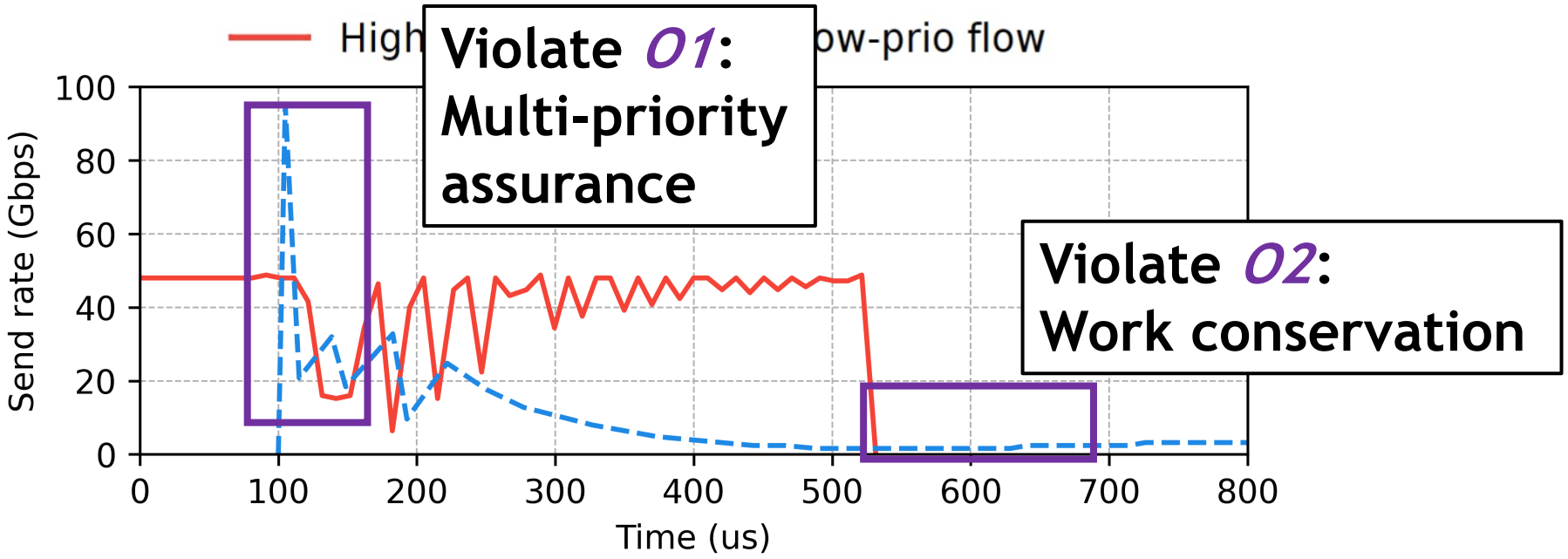
Ideal virtual priority CC behavior

Our Insight

1. Virtual priority can be achieved by carefully **managing bandwidth contention** among flows within a physical priority queue, a task traditionally handled by the **Congestion Control (CC)** at the host.

2. CC that supports virtual priority requires a **multi-bit congestion signal** and **setting distinct congestion thresholds** for flows with different priorities.

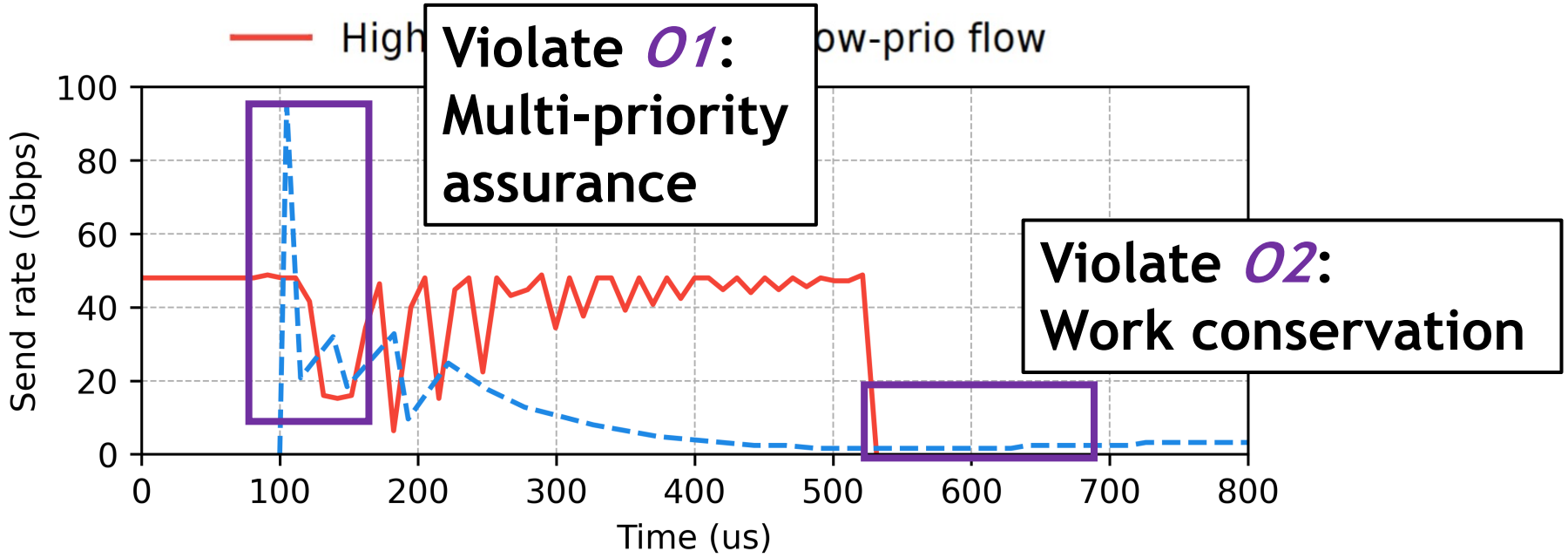
Observation: need careful balance between *O1* and *O2*



2 High-priority Swift flows and 2 Low-priority Swift flows in the simulation

Observation: need careful balance between *O1* and *O2*

Many existing CC designs are not suitable for virtual priority CC, which fail to balance multi-priority assurance (*O1*) with work conservation (*O2*).



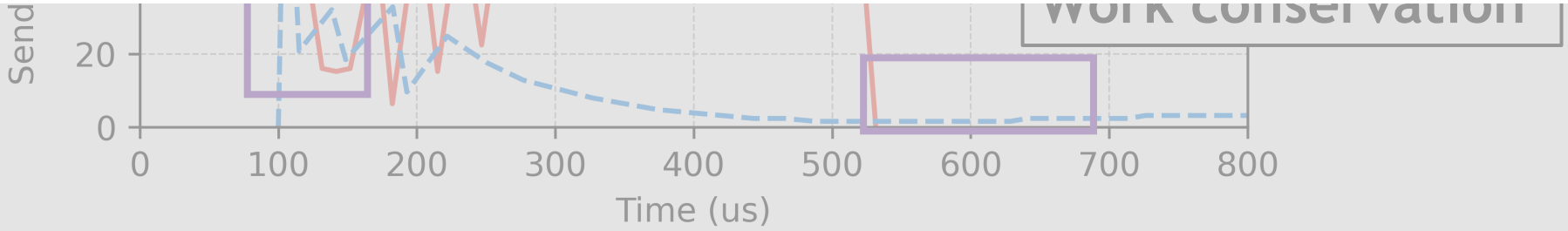
2 High-priority Swift flows and 2 Low-priority Swift flows in the simulation

Observation: need careful balance between *O1* and *O2*

Many existing CC designs are not suitable for virtual priority CC, which fail to balance multi-priority assurance (*O1*) with work conservation (*O2*).

PrioPlus

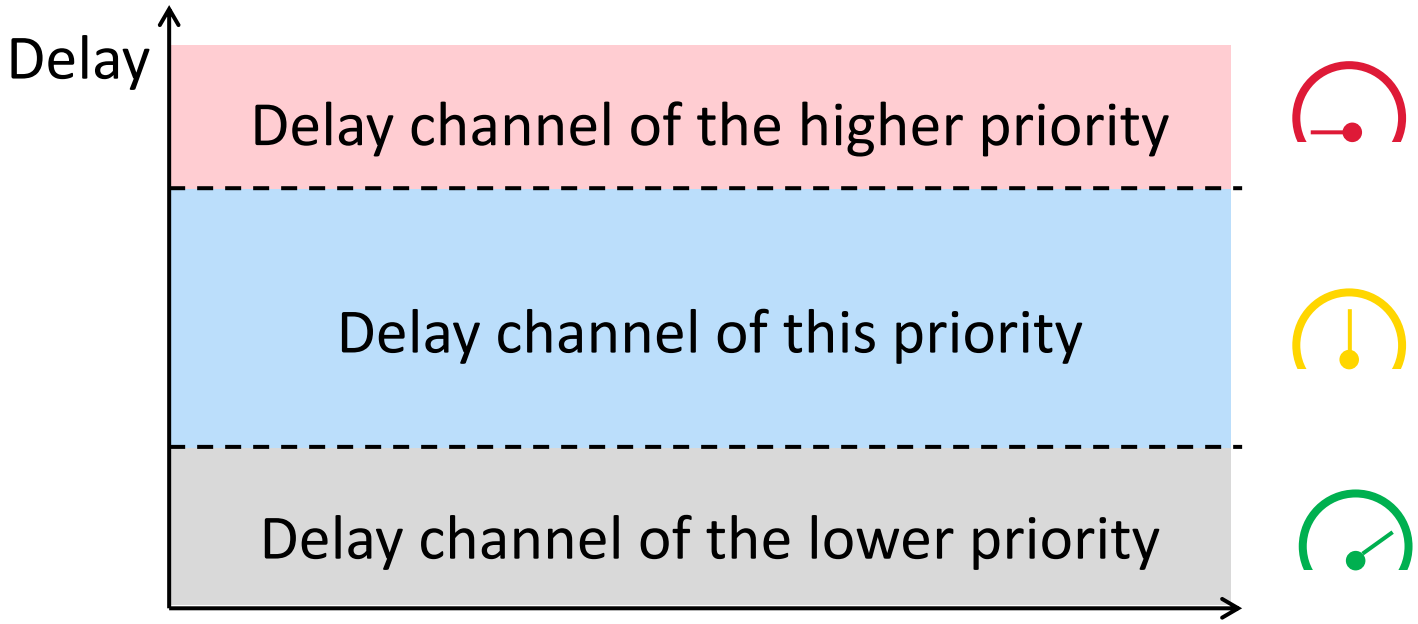
a congestion control enhancement algorithm that can be integrated into existing delay-based CCs to enable virtual priority.



2 High-priority Swift flows and 2 Low-priority Swift flows in the simulation

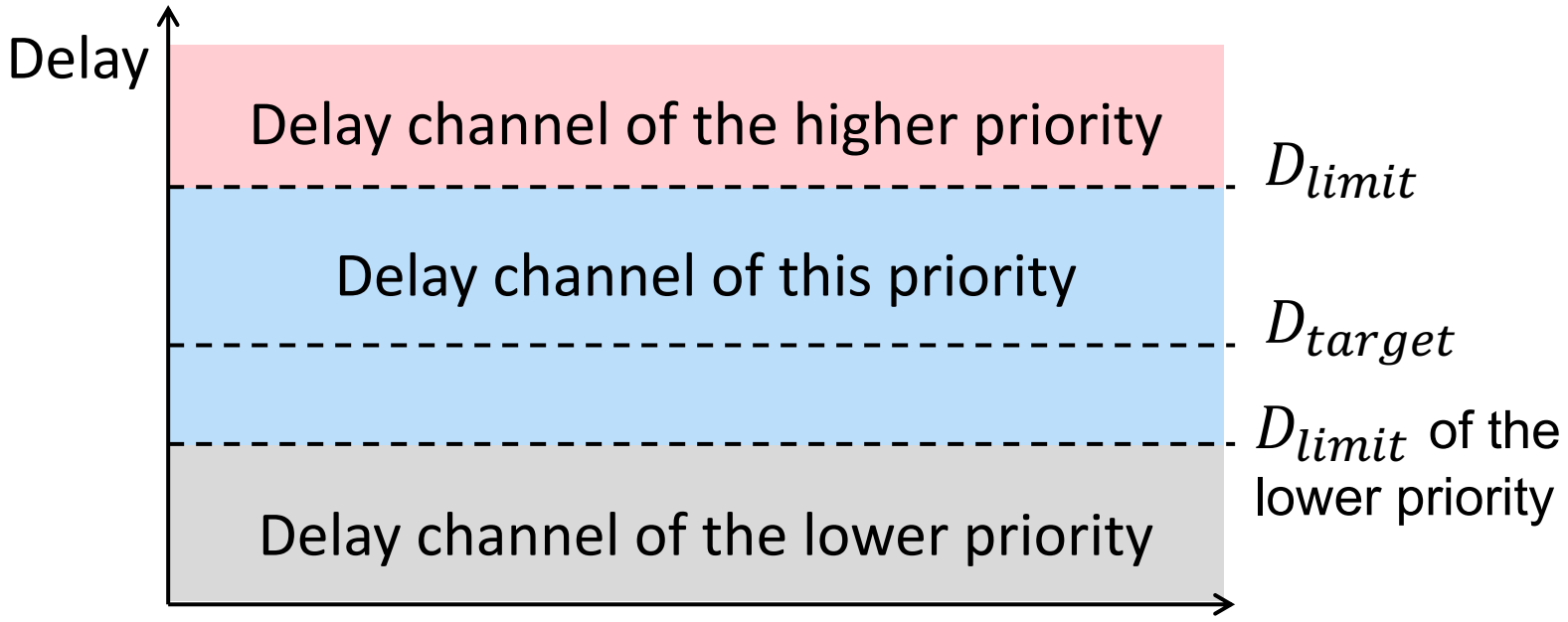
PrioPlus's Key Concept: Delay Channel

PrioPlus assigns specific delay ranges, termed as channels, to each priority.

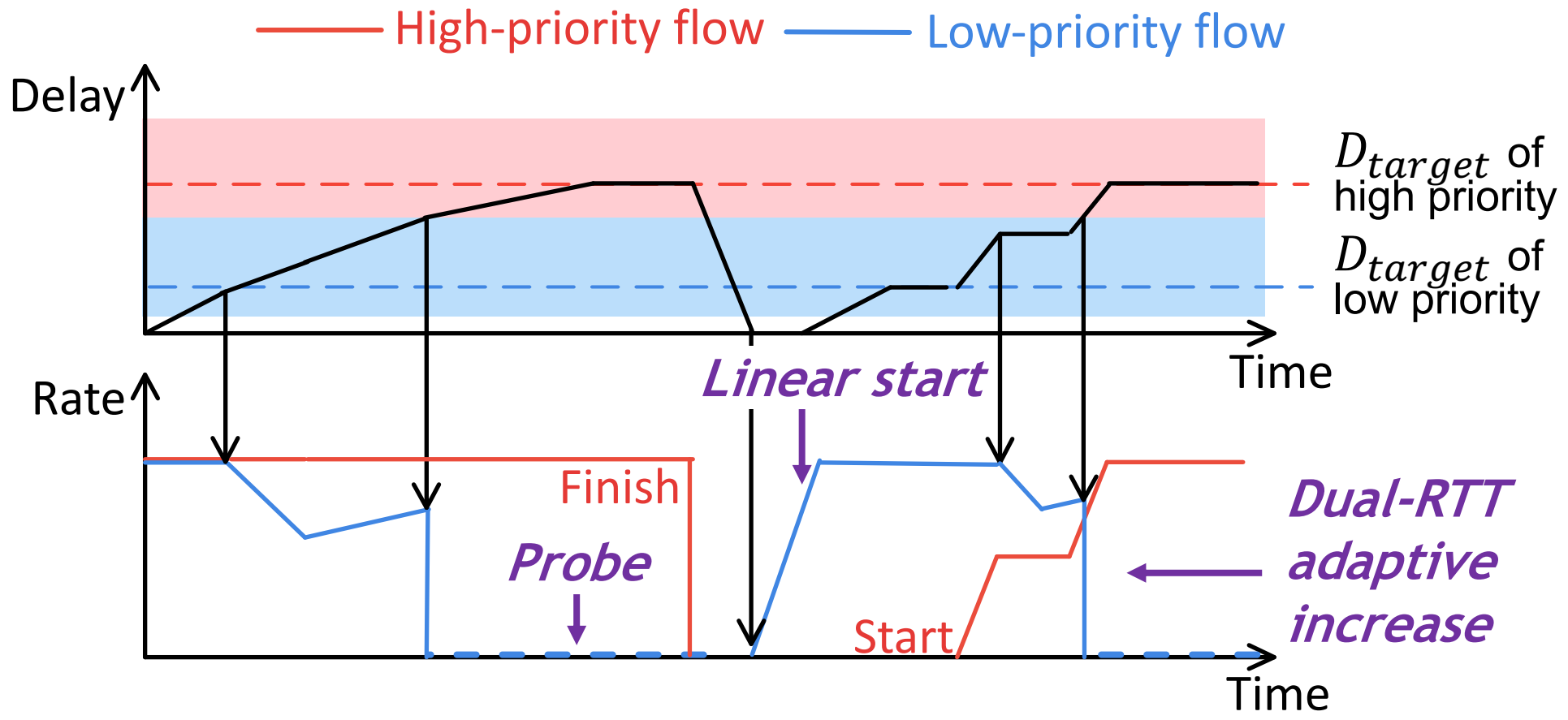


PrioPlus's Key Concept : Delay Channel

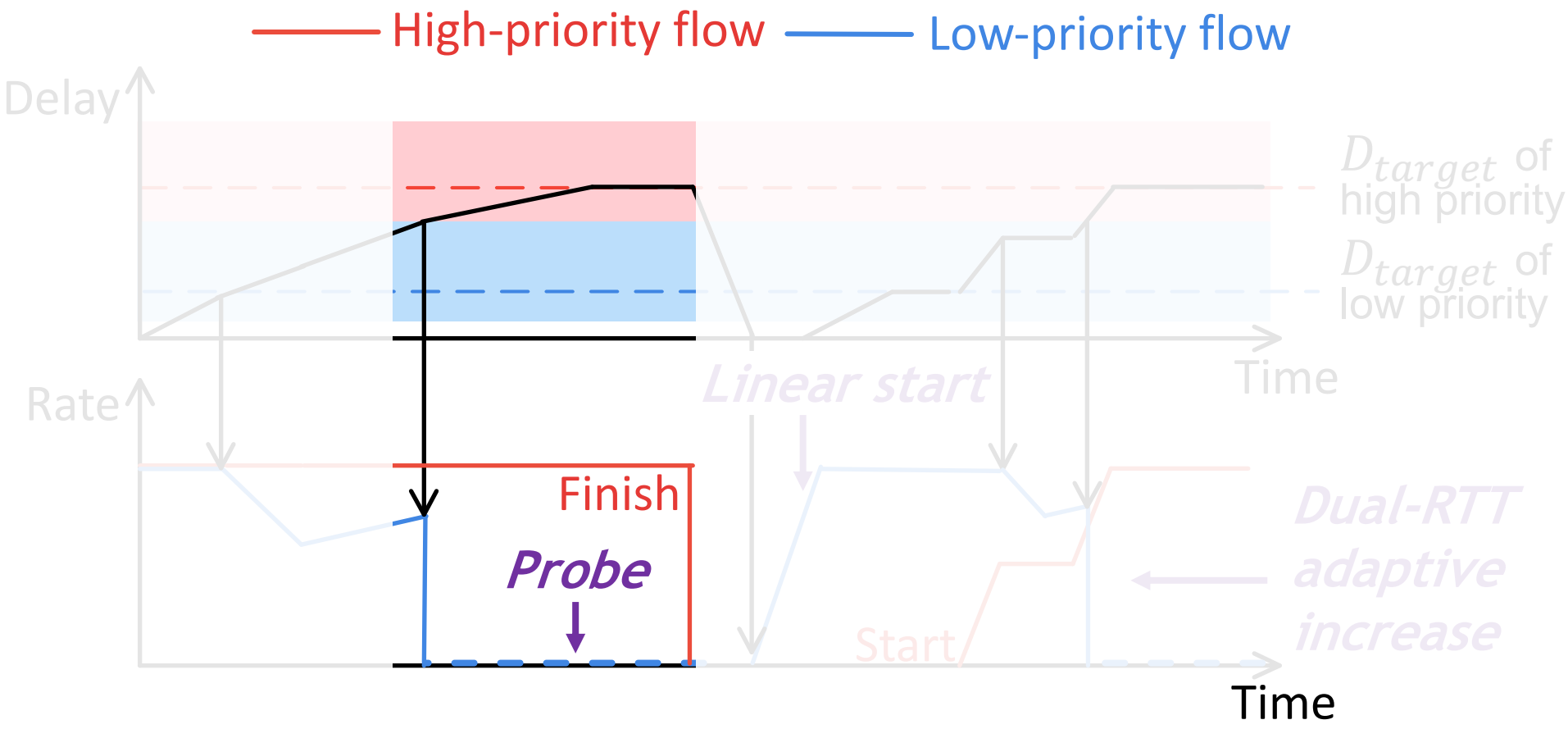
PrioPlus assigns specific delay ranges, termed as channels, to each priority.



PrioPlus's Workflow



PrioPlus's Workflow



PrioPlus Design: Probe with Collision Avoidance

After a lower-priority flow relinquishes the bandwidth, it sends minimal-sized probe packets to probe the network delay.

Strawman solution: send a probe per RTT.

Collisions: flow restart transmission synchronously.

Bandwidth occupancy: typically 42 Mbps per flow.

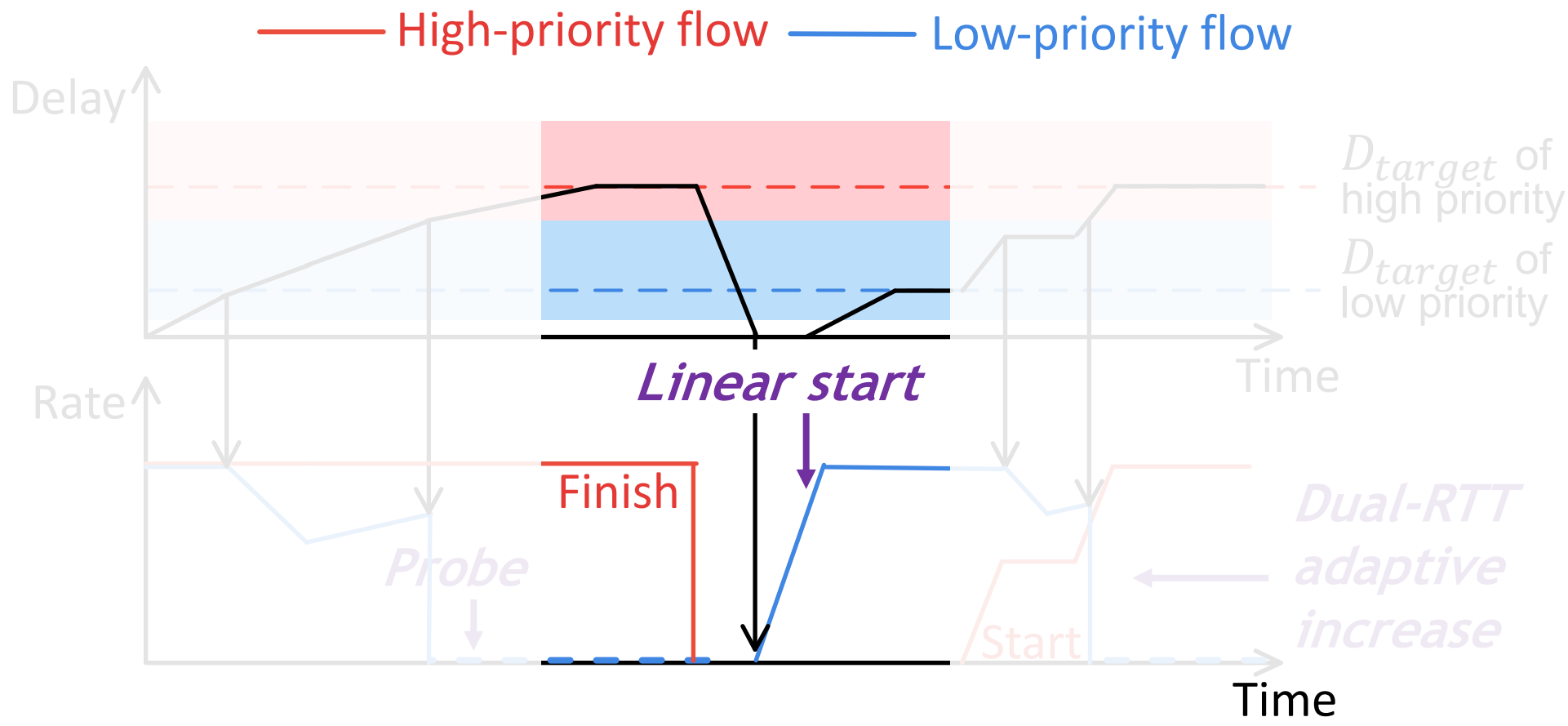
PrioPlus's solution: send a probe after $(delay - D_{target}) + random(BaseRTT)$.

Lower bandwidth occupancy while higher priority has higher probe frequency.

Fewer collisions.

See our paper for PrioPlus's solution on probe loss.

PrioPlus's Workflow



PrioPlus Design: Linear Start

When facing an uncertain environment, a flow should start bold yet meticulous.

- Starting too slow will lead to **delayed data transmission**.
- Starting too fast causes **queue buildup potentially**, resulting in undesirable delay fluctuations.

PrioPlus Design: Linear Start

DCN CCs' solution: start at the line rate.
Internet CCs' solution: start at a low rate and accelerate exponentially.
PrioPlus's solution: start at a low rate and accelerate linearly until the delay is larger than the base RTT.

See our paper for the detailed analysis.

PrioPlus Design: Linear Start

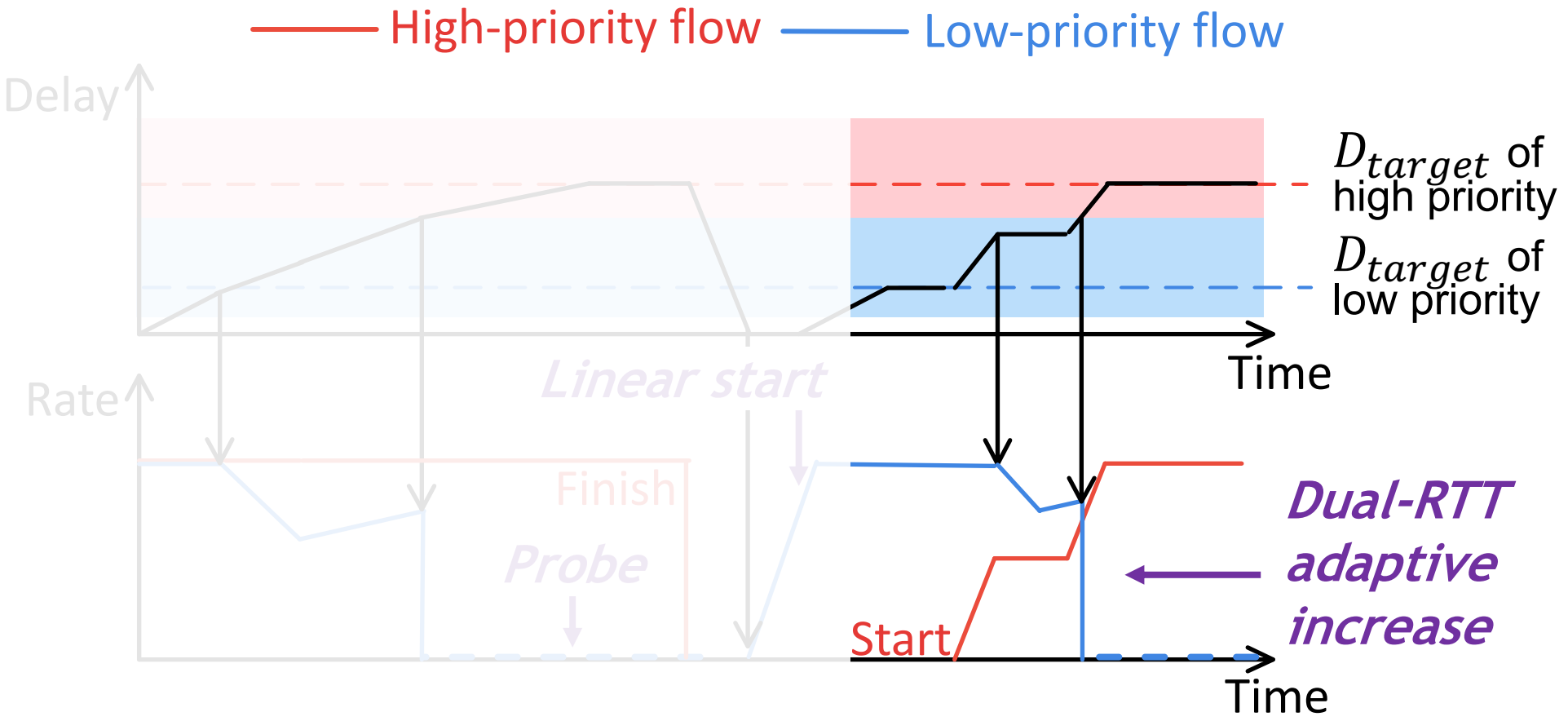
Linear start effectively balances transmission delayed and maximum potential queue buildup.

Flow Start Strategy	Transmission delayed	Potential maximum buildup
Line-rate start	0	1 BDP
Exponential start	$n - 3/2$ BDP	$1/4$ BDP
Linear start	$n/2$ BDP	$1/n$ BDP

Linear start is the start strategy that incurs **the least** potential buffer backlog when increasing the send rate from 0 to the line rate in a given period.

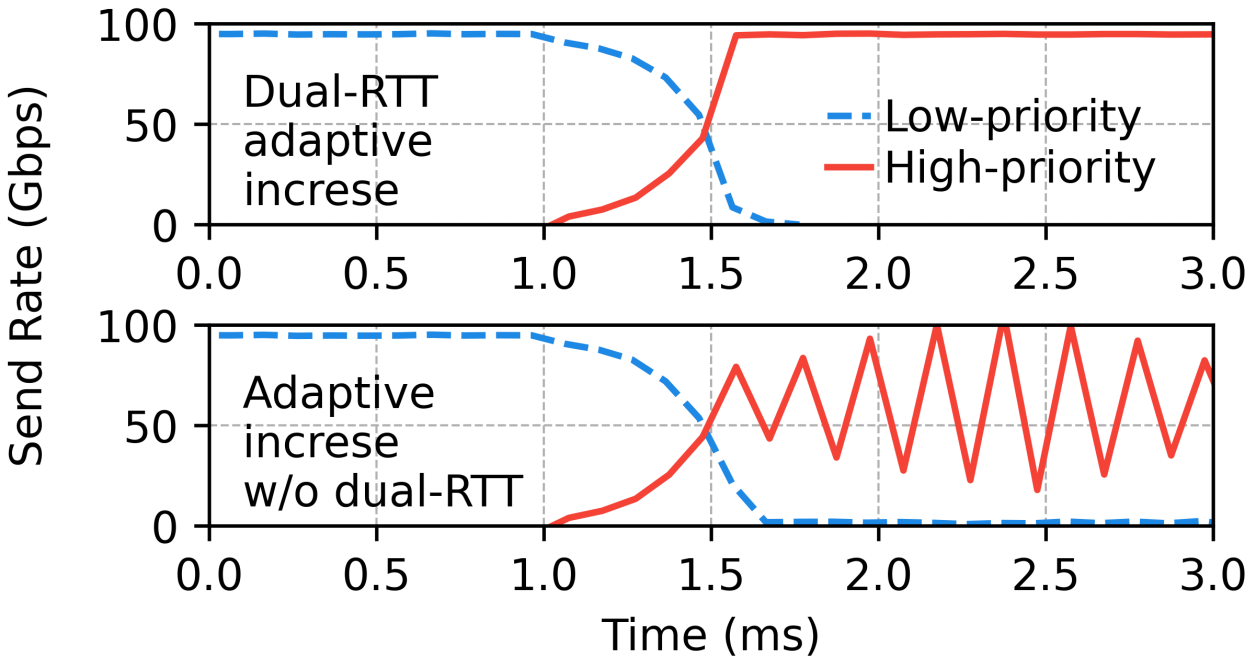
See our paper for the formal proof.

PrioPlus's Workflow



PrioPlus Design: Dual-RTT Adaptive Increase

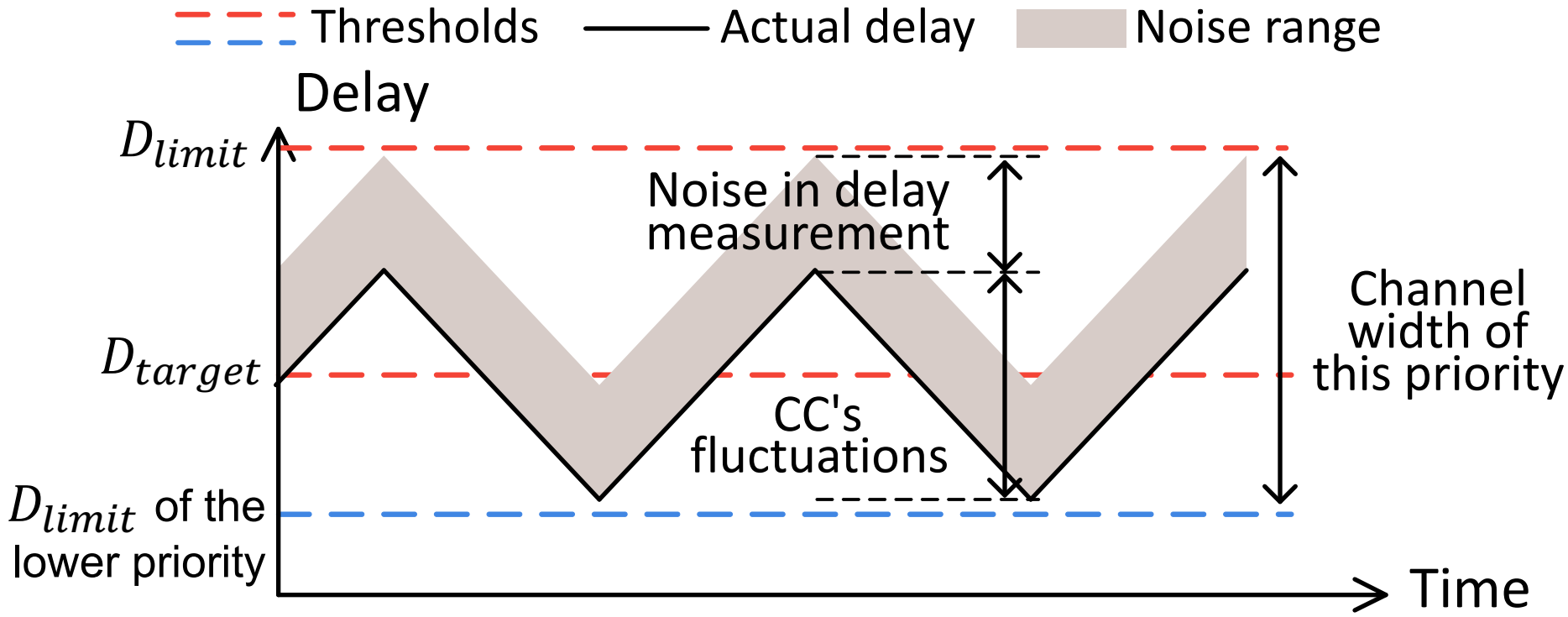
When delay below the given delay channel, a PrioPlus flow will raise the delay to its delay channel precisely using dual-RTT adaptive increase.



See our paper for the detailed design and analysis.

Compress and Configure the PrioPlus Channel Width

The channel width must accommodate CC's normal fluctuations and the noise in the delay measurement.



Compress and Configure the PrioPlus Channel Width

Compress CC's fluctuations.

When the delay exceeds D_{limit}, PrioPlus flow estimates the flow cardinality $\#flow$ as the ratio of inflight size ($delay \cdot LineRate$) and its own window. Then adjust its increase step size inversely proportionally.

Accommodate the delay noise

Considering delay noise in modern data centers exhibits a long-tail distribution, PrioPlus employs a simple *filter mechanism* to filter out infrequent large delay noises: a flow relinquishes bandwidth only when the delay exceeds the D_{limit} twice.

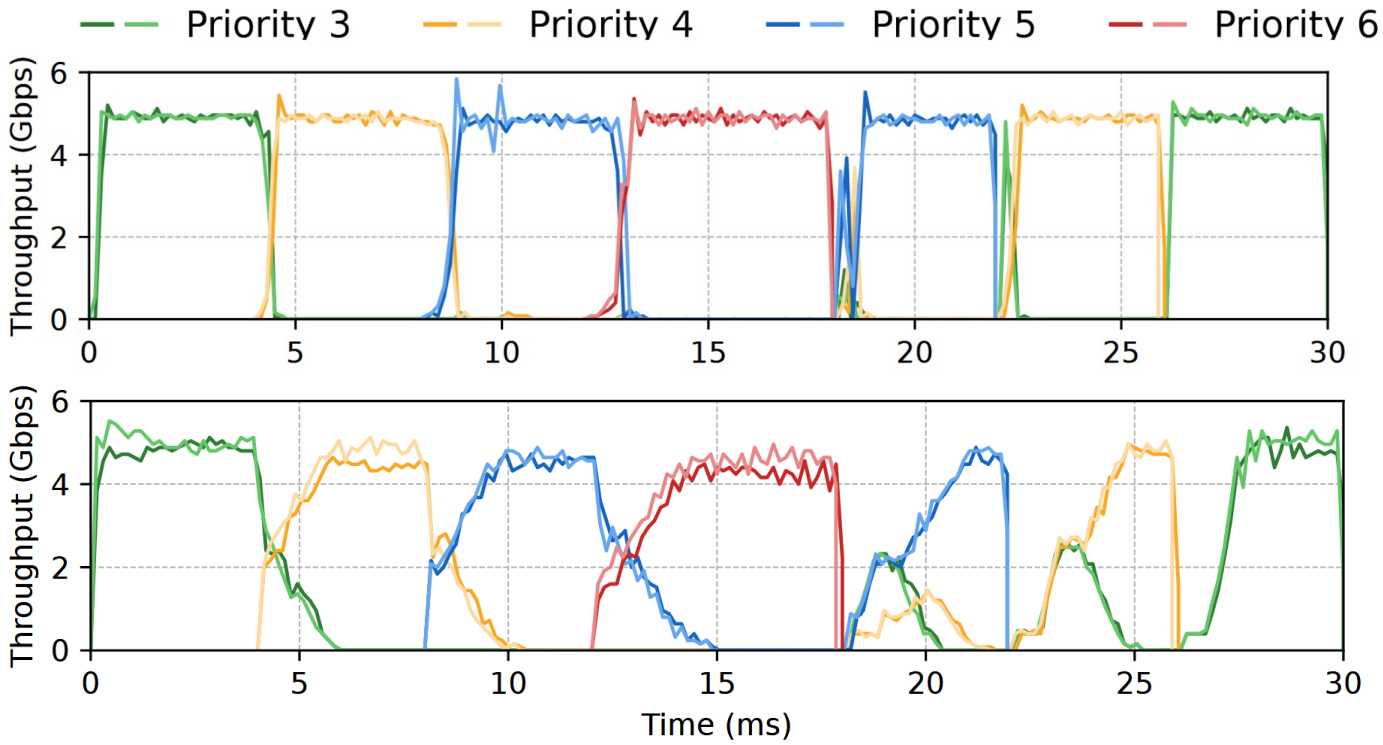
See our paper on how to determine CC's fluctuations and delay noise in different network environments and set channel width accordingly.

Testbed Evaluation

Implement PrioPlus on Swift's DPDK implementation using 79 LoCs.
Topology: a 10 Gbps tree topology with four leaves are senders and the root is the receiver.
Channel width: 4 us.

PrioPlus + Swift:

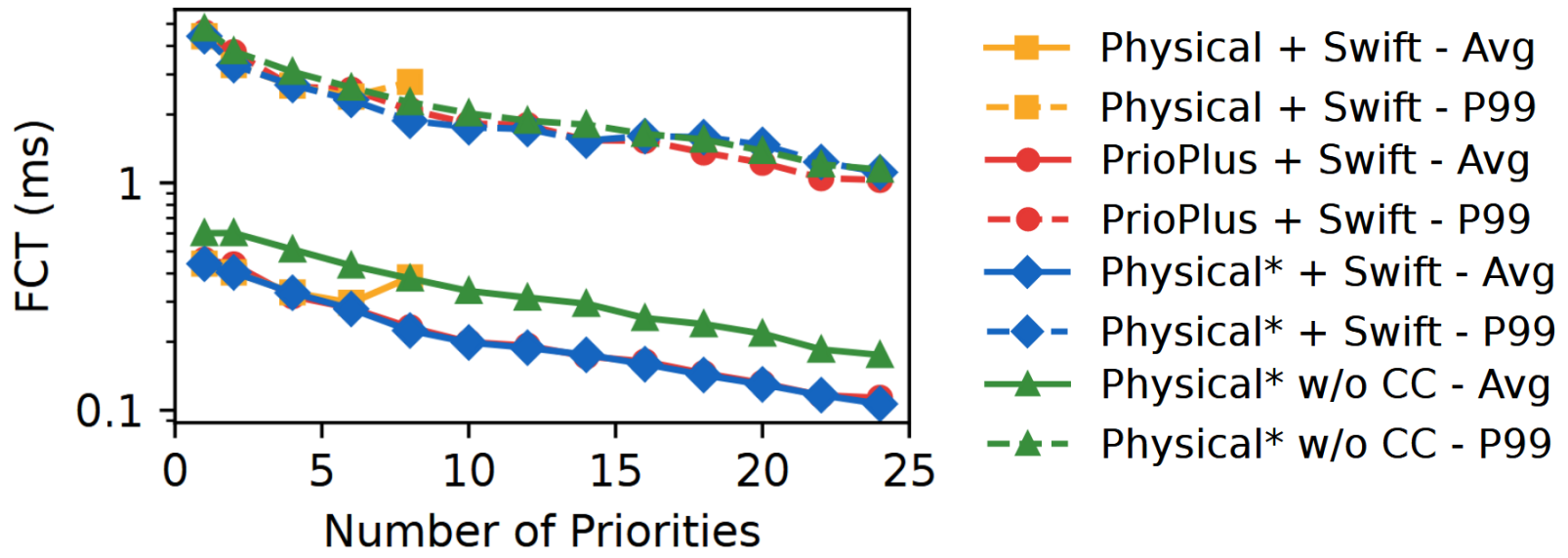
Swift:



Simulation Evaluation

Flow scheduling scenario: The performance of PrioPlus with Swift closely approaches **ideal physical priority** with Swift, with average FCT being **at most 8% worse**.

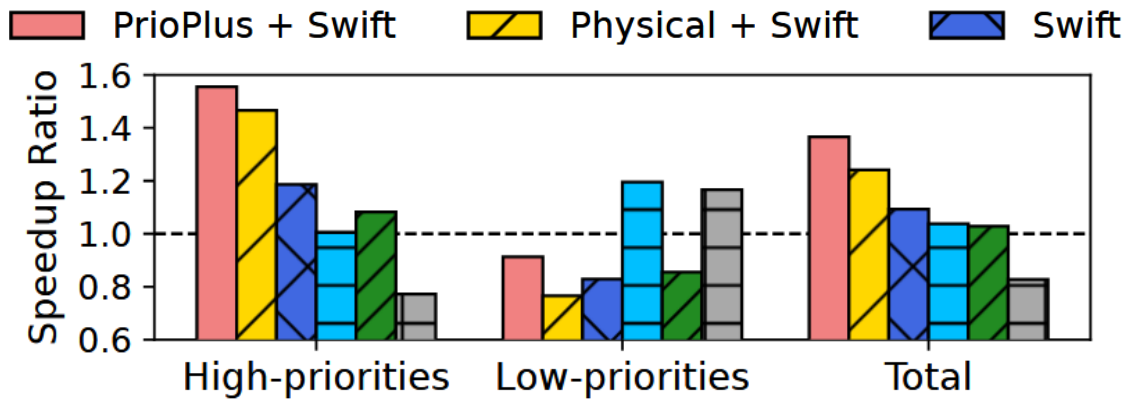
Topology: stranded fat-tree with k=6. Workload: WebSearch. Channel width: 4 us.
Buffer: 4.4 MB/Tbps, aligned with the latest switch.



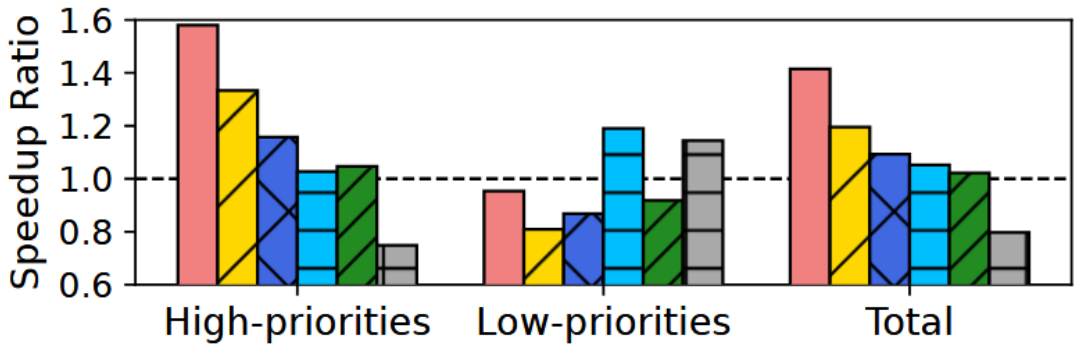
Simulation Evaluation

Coflow scheduling scenario: When the load is at 40%/70%, the average speedup ratio of PrioPlus with Swift is **12%/21% higher** than that of physical priority with Swift.

Topology: fat-tree with 320 hosts. Workload: Facebook Hadoop trace. Channel width: 4 us. Buffer: 32 MB to avoid insufficient buffer impacting the performance of physical priorities.



Coflow scenario at 40% load.



Coflow scenario at 70% load.

See our paper for evaluations on incast, noise robustness,model training and so on…

Conclusion

PrioPlus is a congestion control enhancement algorithm designed to integrate with existing delay-based CC mechanisms, enabling multiple virtual priorities within each physical queue in data centers.

It is lightweight, requiring fewer than 100 lines of code modifications to existing CC implementations.

Find our open-source code with a one-click script to reproduce our results on:
https://github.com/NASA-NJU/PrioPlus