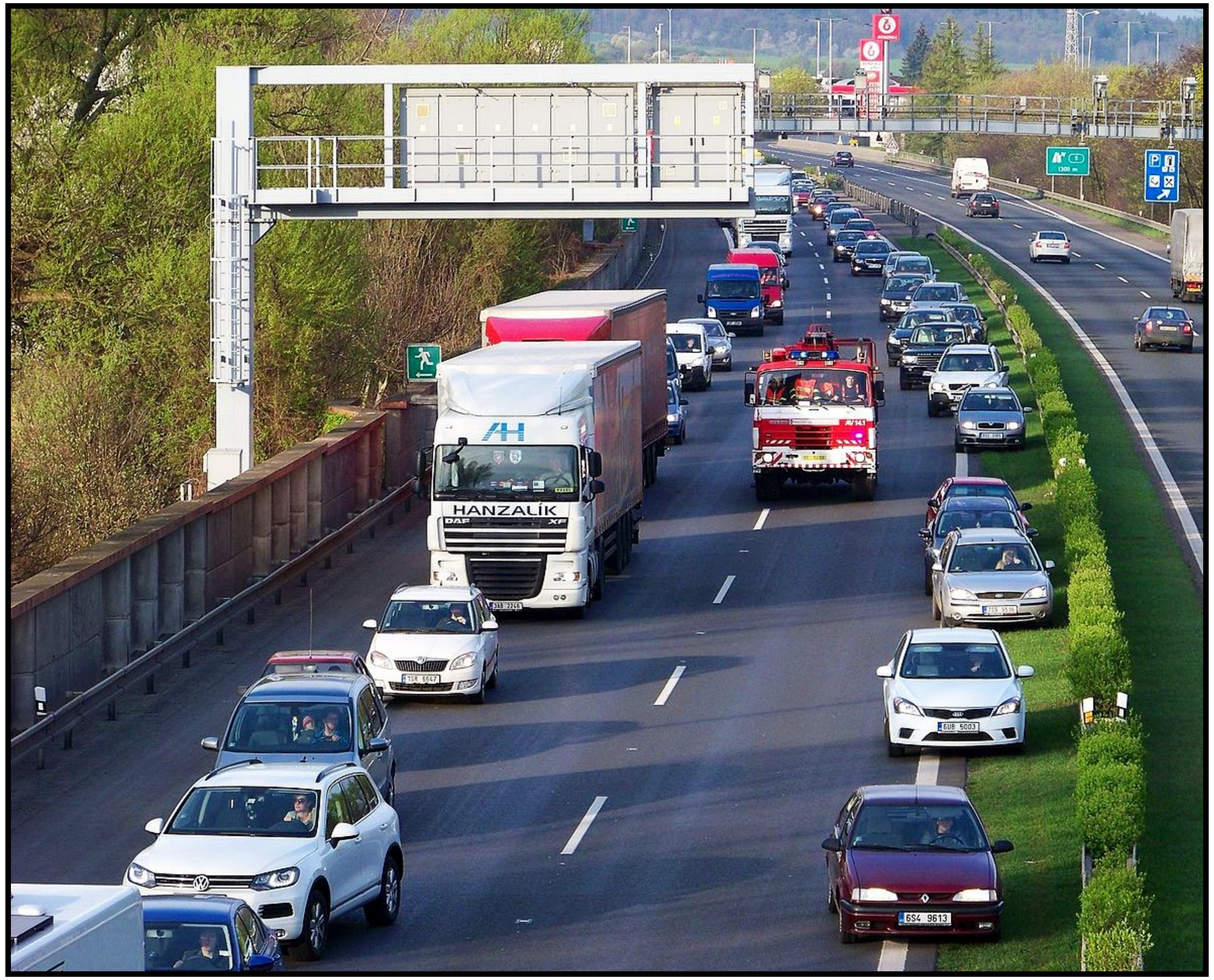
In-Network Congestion Management for Security and Performance

Albert Gran Alcoz

June 18 2024

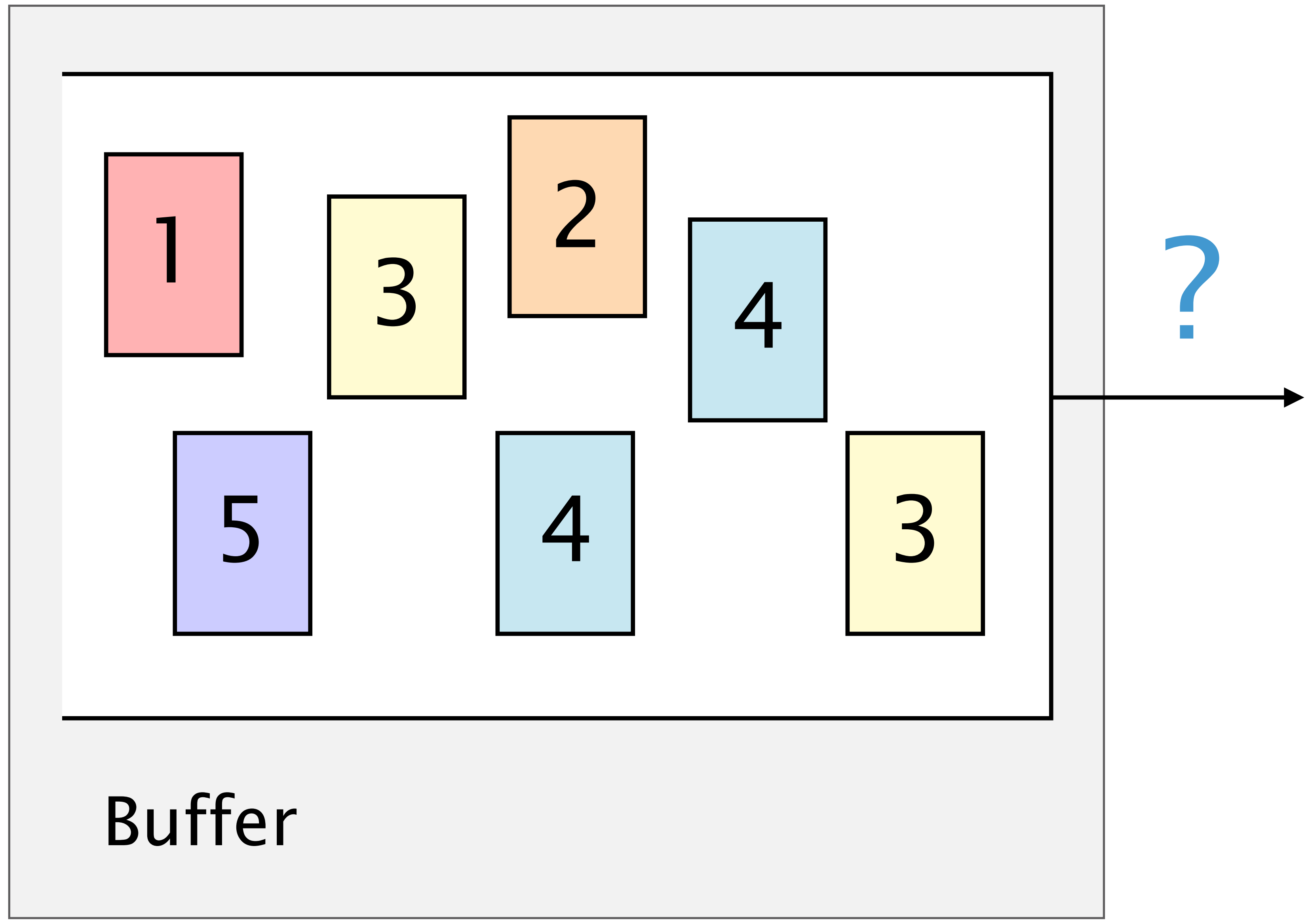

AMBULANCE
ON DUTY


ON DUTY


HANZALIK

Packet scheduling

Packet scheduling defines what packet should we send next and when



Researchers have proposed dozens of scheduling algorithms

Minimize flow completion times		
Prioritize packets from short flows	SRPT, PIAS
Enforce fairness	
Send one packet from each class at a time	RR, WFQ
Minimize tail latency	
Prioritize packets with high slack time	FIFO+, LSTF

A universal scheduling algorithm does not exist

NSDI'16

Universal Packet Scheduling

Radhika Mittal[†] *Rachit Agarwal*[†] *Sylvia Ratnasamy*[†] *Scott Shenker*[†‡]

[†]*UC Berkeley* [‡]*ICSI*

Abstract

In this paper we address a seemingly simple question: *Is there a universal packet scheduling algorithm?* More precisely, we analyze (both theoretically and empirically) whether there is a single packet scheduling algorithm that, at a network-wide level, can perfectly match the results of *any* given scheduling algorithm. We find that in general the answer is "no". However, we show theoretically that the classical Least Slack Time First (LSTF) scheduling algorithm comes closest to being universal and demonstrate empirically that LSTF can closely replay a wide range of scheduling algorithms in realistic network settings. We then evaluate whether LSTF can be used *in practice* to meet various network-wide objectives by looking at popular performance metrics (such as mean FCT, tail packet delays, and fairness); we find that LSTF performs comparable to the state-of-the-art for each of them. We also discuss how LSTF can be used in conjunction with active queue management schemes (such as CoDel) without changing the core of the network.

1 Introduction

There is a large and active research literature on novel packet scheduling algorithms, from simple schemes such as priority scheduling [31], to more complicated mech-

We can define a universal packet scheduling algorithm (hereafter UPS) in two ways, depending on our viewpoint on the problem. From a theoretical perspective, we call a packet scheduling algorithm *universal* if it can replay any *schedule* (the set of times at which packets arrive to and exit from the network) produced by any other scheduling algorithm. This is not of practical interest, since such schedules are not typically known in advance, but it offers a theoretically rigorous definition of universality that (as we shall see) helps illuminate its fundamental limits (i.e., which scheduling algorithms have the flexibility to serve as a UPS, and why).

From a more practical perspective, we say a packet scheduling algorithm is universal if it can achieve different desired performance objectives (such as fairness, reducing tail latency, minimizing flow completion times). In particular, we require that the UPS should match the performance of the best known scheduling algorithm for a given performance objective. [1]

The notion of universality for packet scheduling might seem esoteric, but we think it helps clarify some basic questions. If there exists no UPS then we should *expect* to design new scheduling algorithms as performance objectives evolve. Moreover, this would make a strong argument for switches being equipped with programmable

How to deploy all scheduling algorithms?

✘ Generality

Universal packet scheduler

Flexibility

Customized algorithms

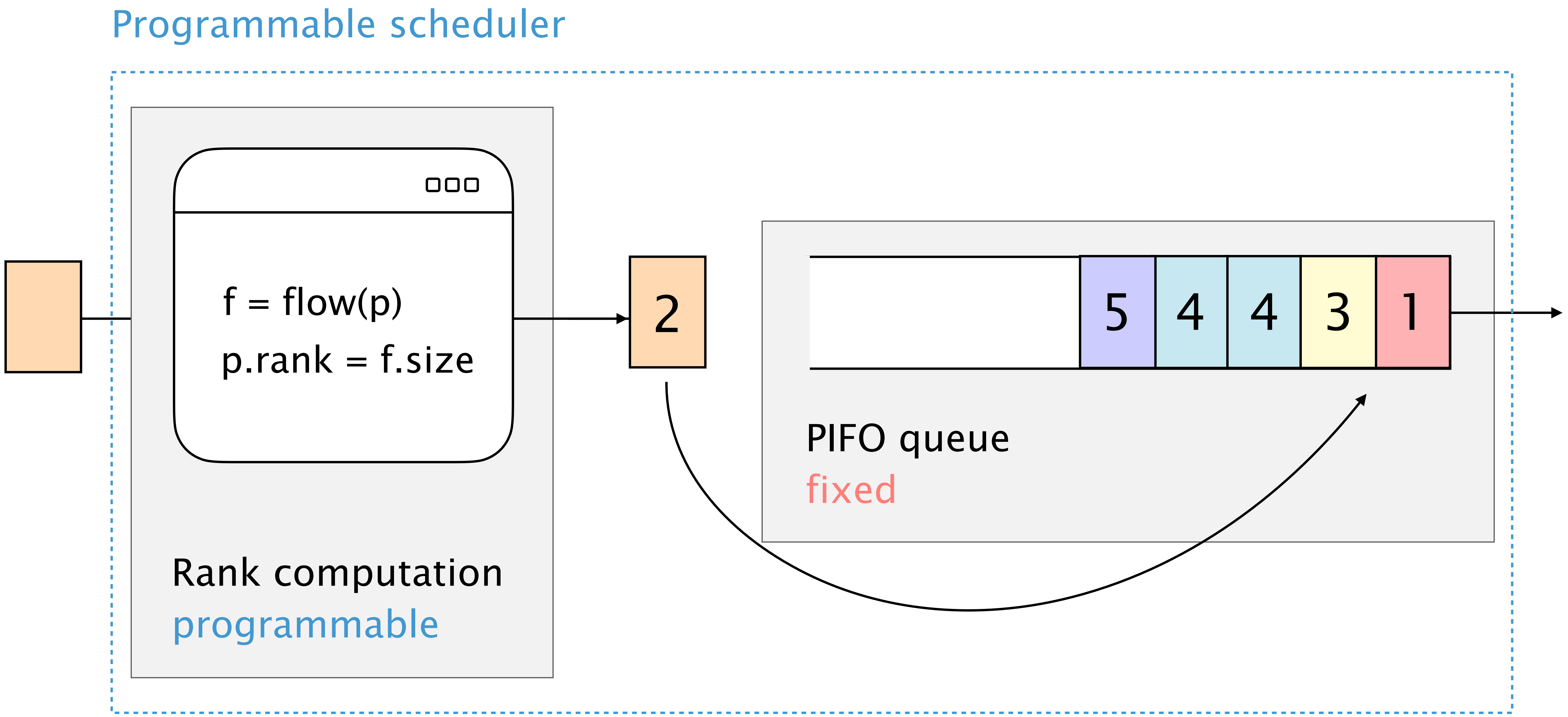
How to deploy all scheduling algorithms?

✘ Generality

Universal packet scheduler

Programmable scheduling

Push-In First-Out (PIFO) queues enable programmable scheduling



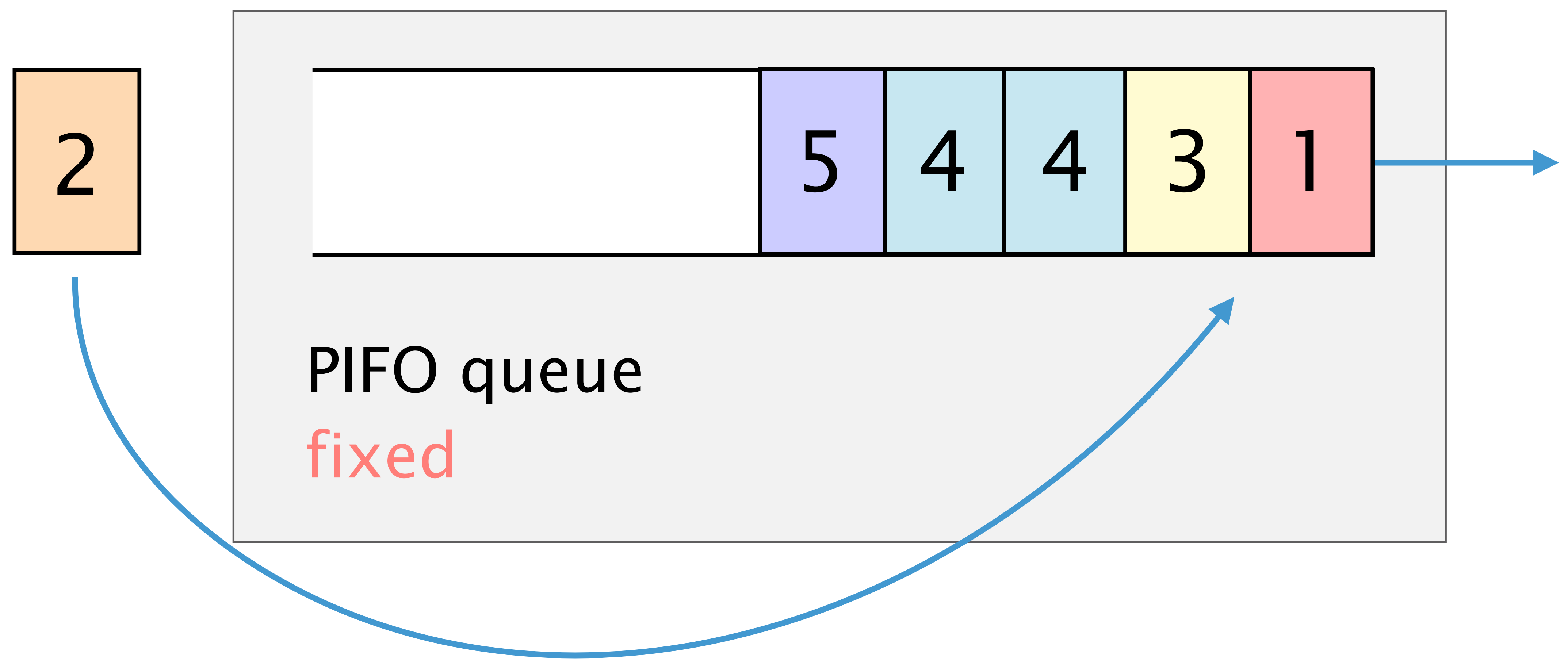
PIFO queues are characterized by two key behaviors

Admission

Enqueue packets with the lowest ranks

Scheduling

Forward packets in rank order



How to implement PIFO queues on hardware?

New ASIC

High performance ✔

~200M $ ✘

Multiple years ✘

How to implement PIFO queues on hardware?

New ASIC		Programmable switches	
High performance	✔		
~200M $	✘	~10K $	✔
Multiple years	✘	Available today	✔

How to implement PIFO queues on hardware?

New ASIC		Programmable switches	
High performance	✔	Enough performance	?
~200M $	✘	~10K $	✔
Multiple years	✘	Available today	✔

Objective

Enable programmable scheduling on existing devices

to improve the Internet's performance and security

How to enable programmable scheduling on existing devices?

SP-PIFO [NSDI'20]	PACKS [NSDI'25]
Approximating PIFO's scheduling	Incorporating PIFO's admission

How to use it to improve the Internet's security?

ACC-Turbo [SIGCOMM'22]

Mitigating DDoS attacks

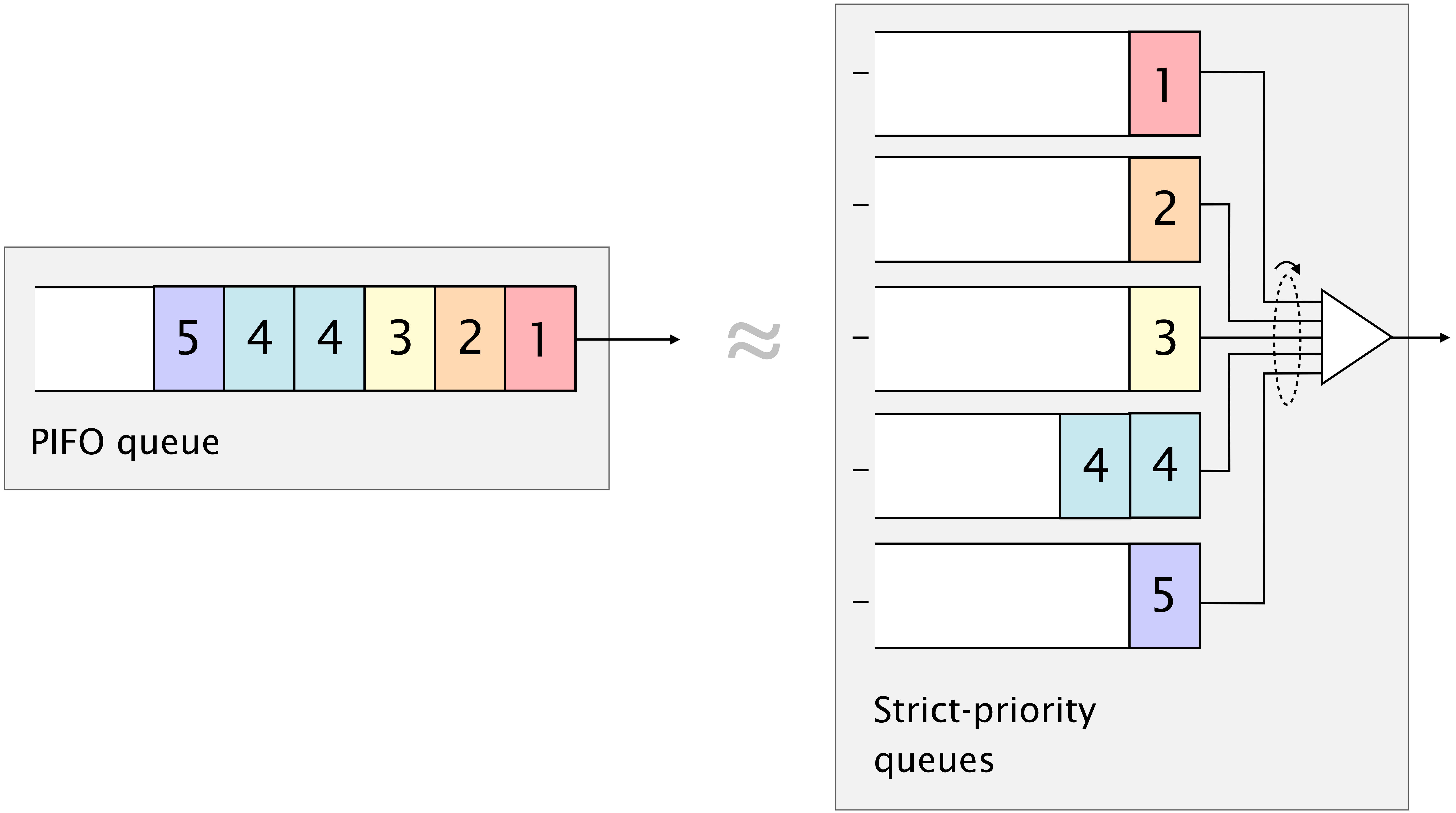
How to enable programmable scheduling on existing devices?

How to use it to improve the Internet's security?

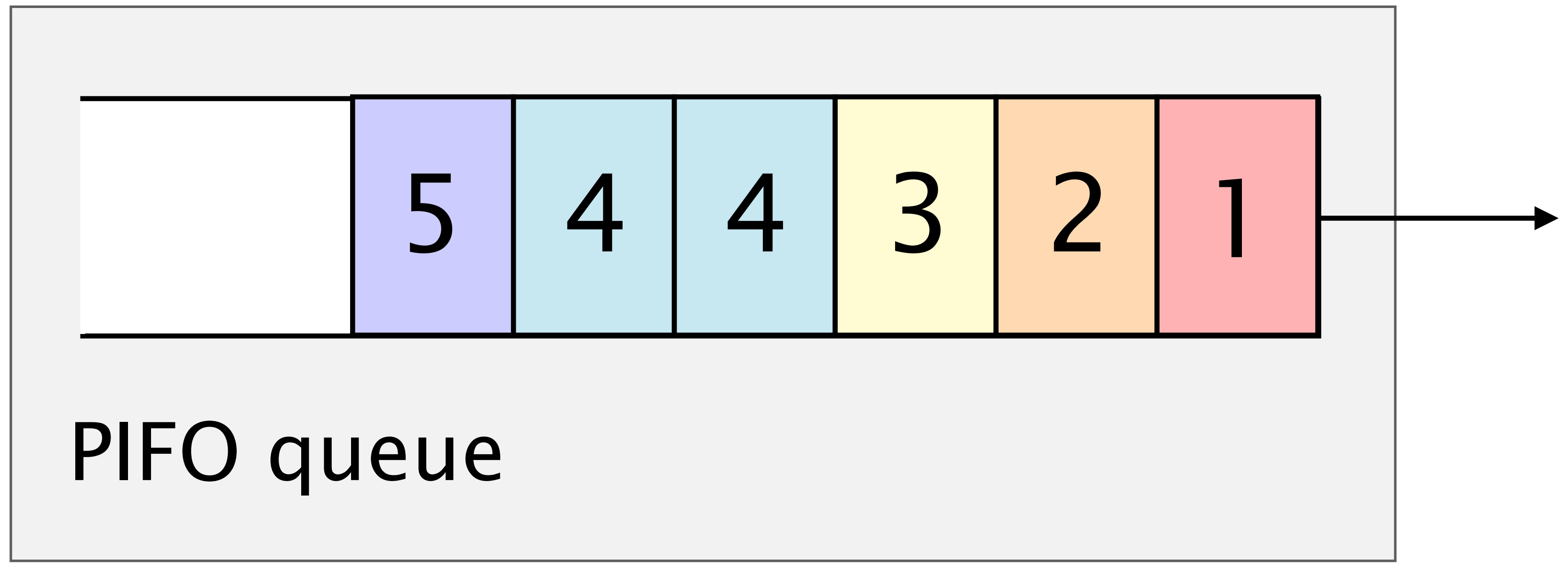
SP-PIFO [NSDI'20]	PACKS [NSDI'25]	ACC-Turbo [SIGCOMM'22]
Approximating PIFO's scheduling	Incorporating PIFO's admission	Mitigating DDoS attacks

We can approximate PIFO queues using strict-priority queues

Ideal case One rank per queue



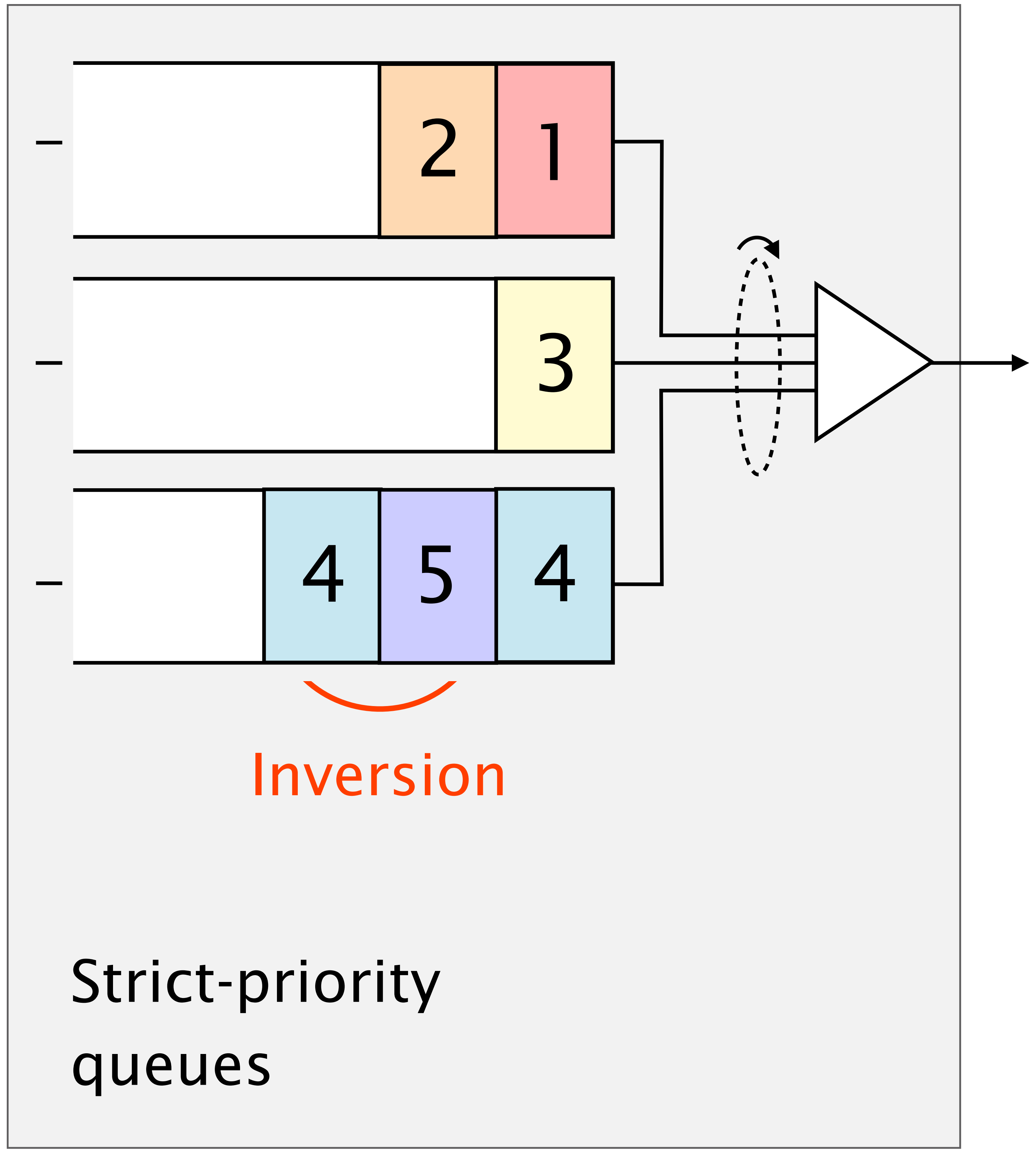
We can approximate PIFO queues using strict-priority queues



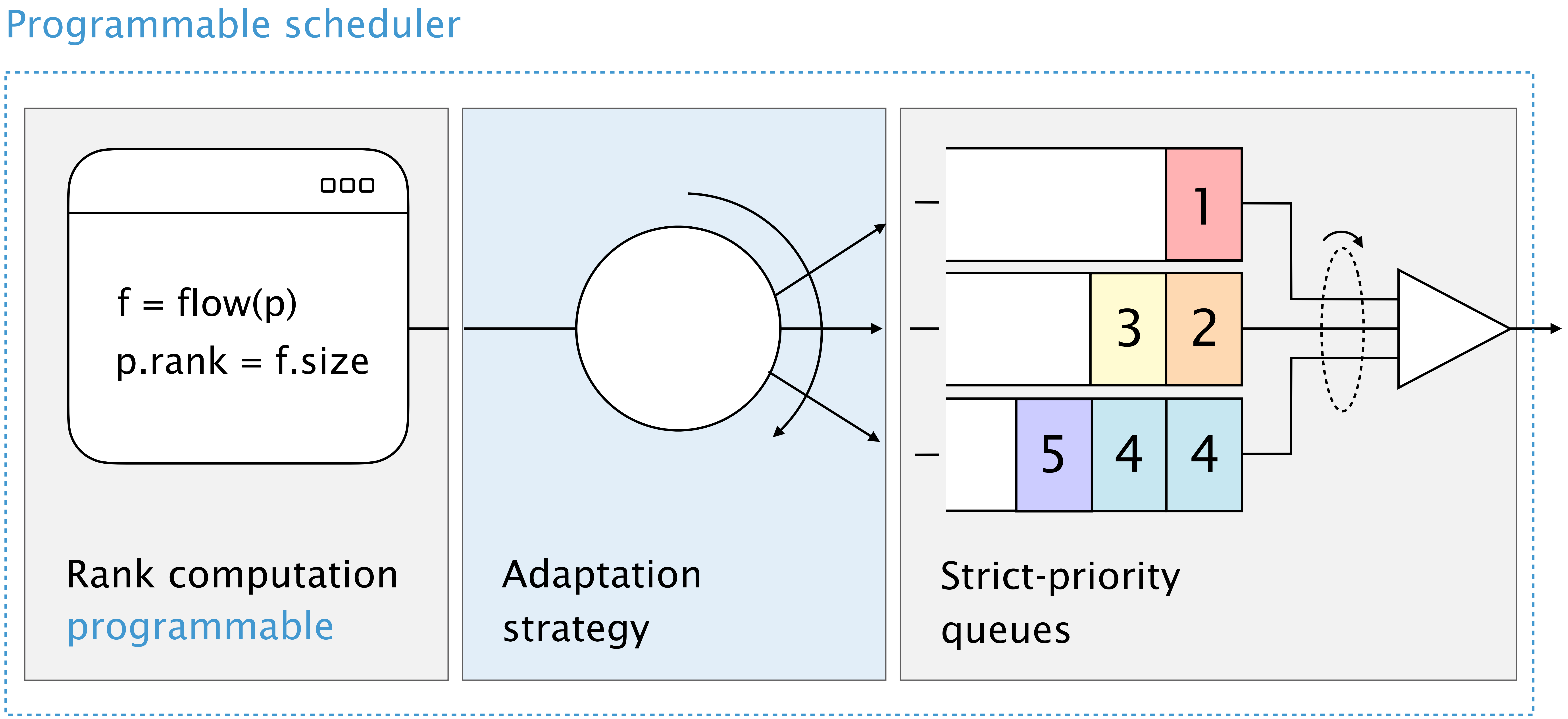
≈

In practice

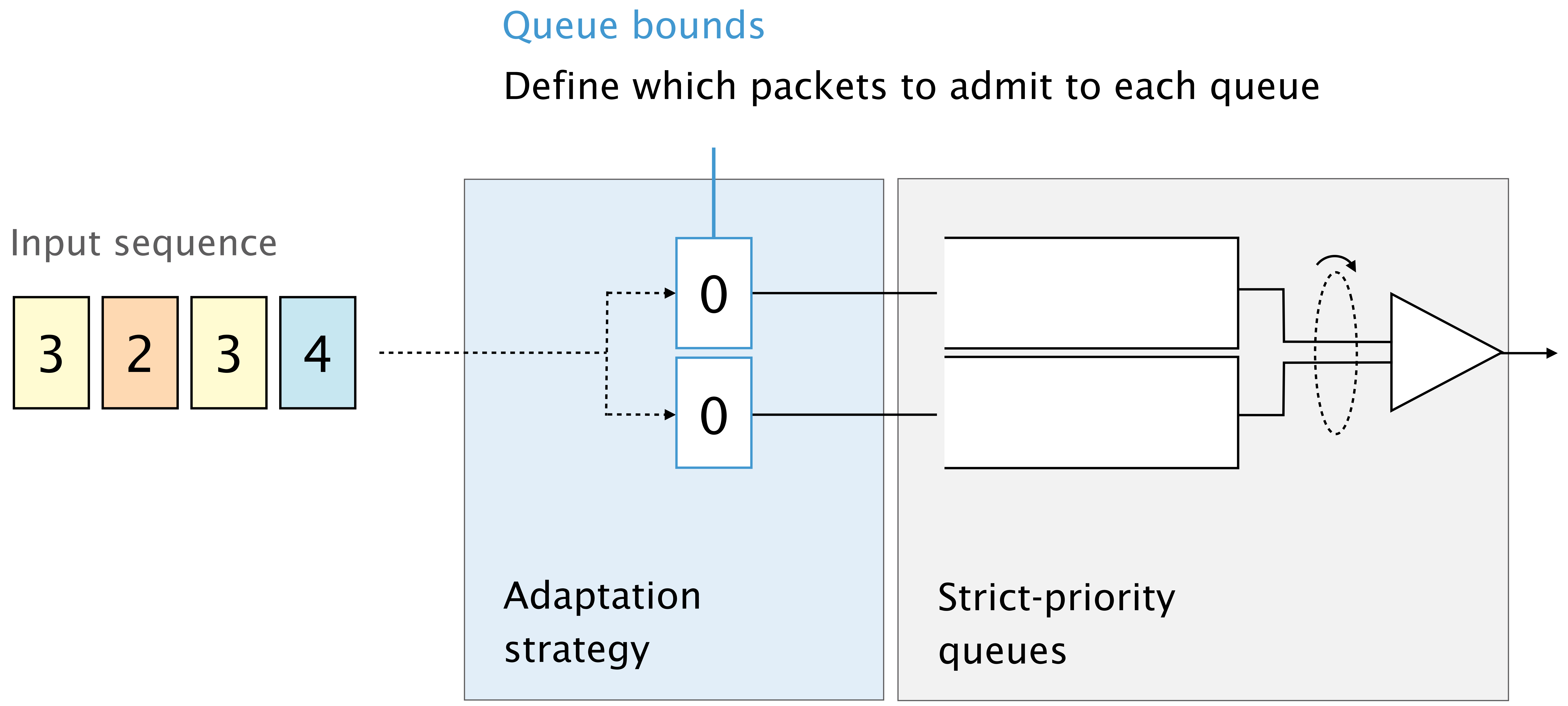
Multiple ranks per queue



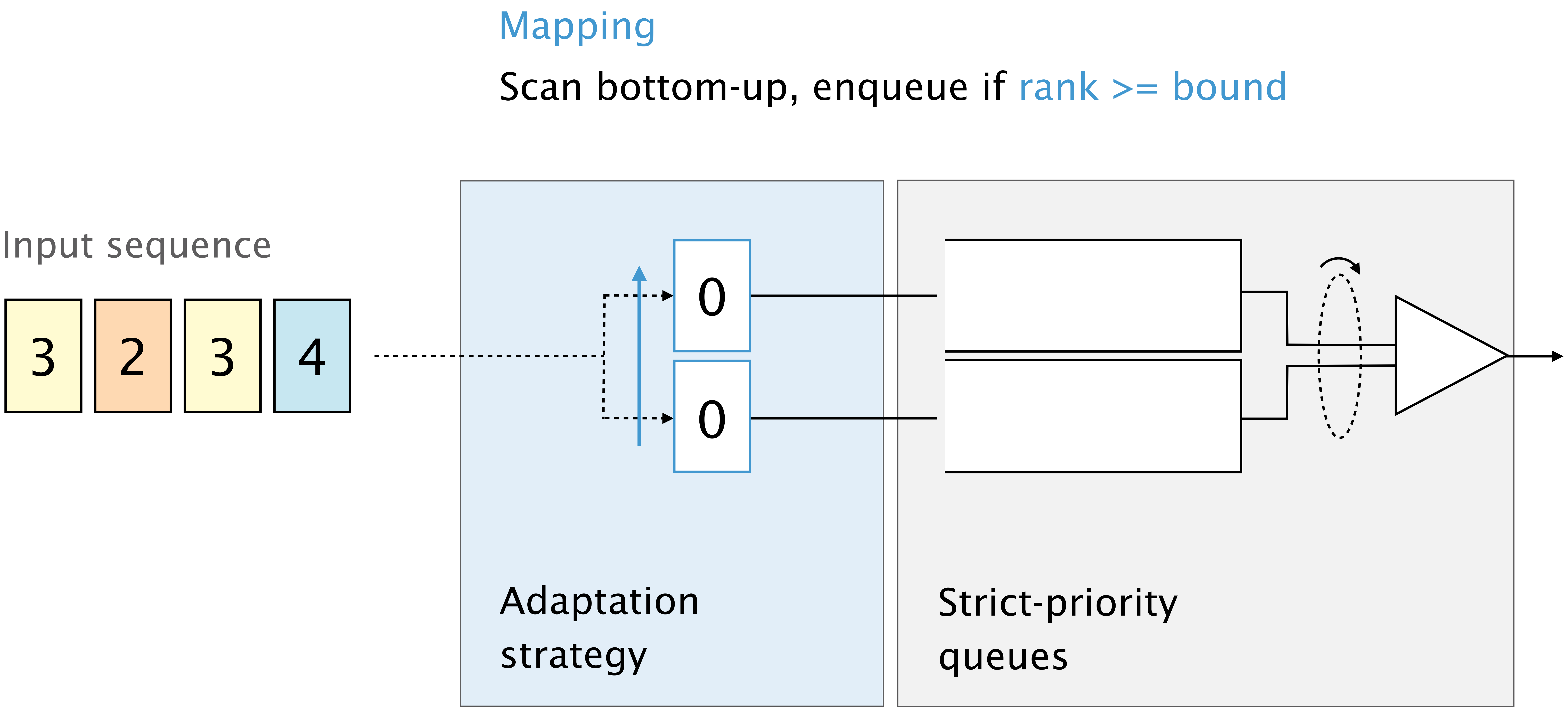
SP-PIFO approximates PIFO queues using strict-priority queues and a dynamic mapping strategy



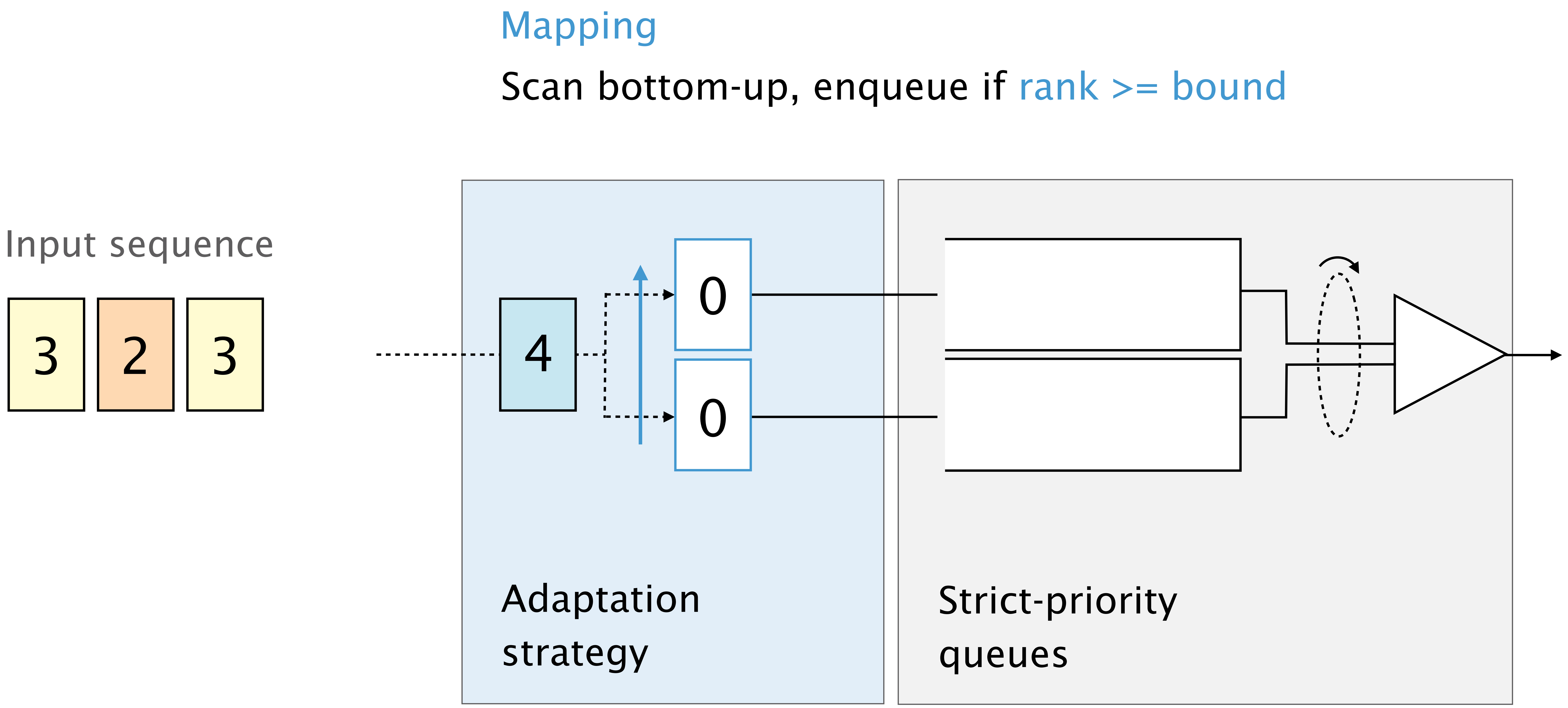
SP-PIFO adapts the mapping of packet ranks to strict-priority queues



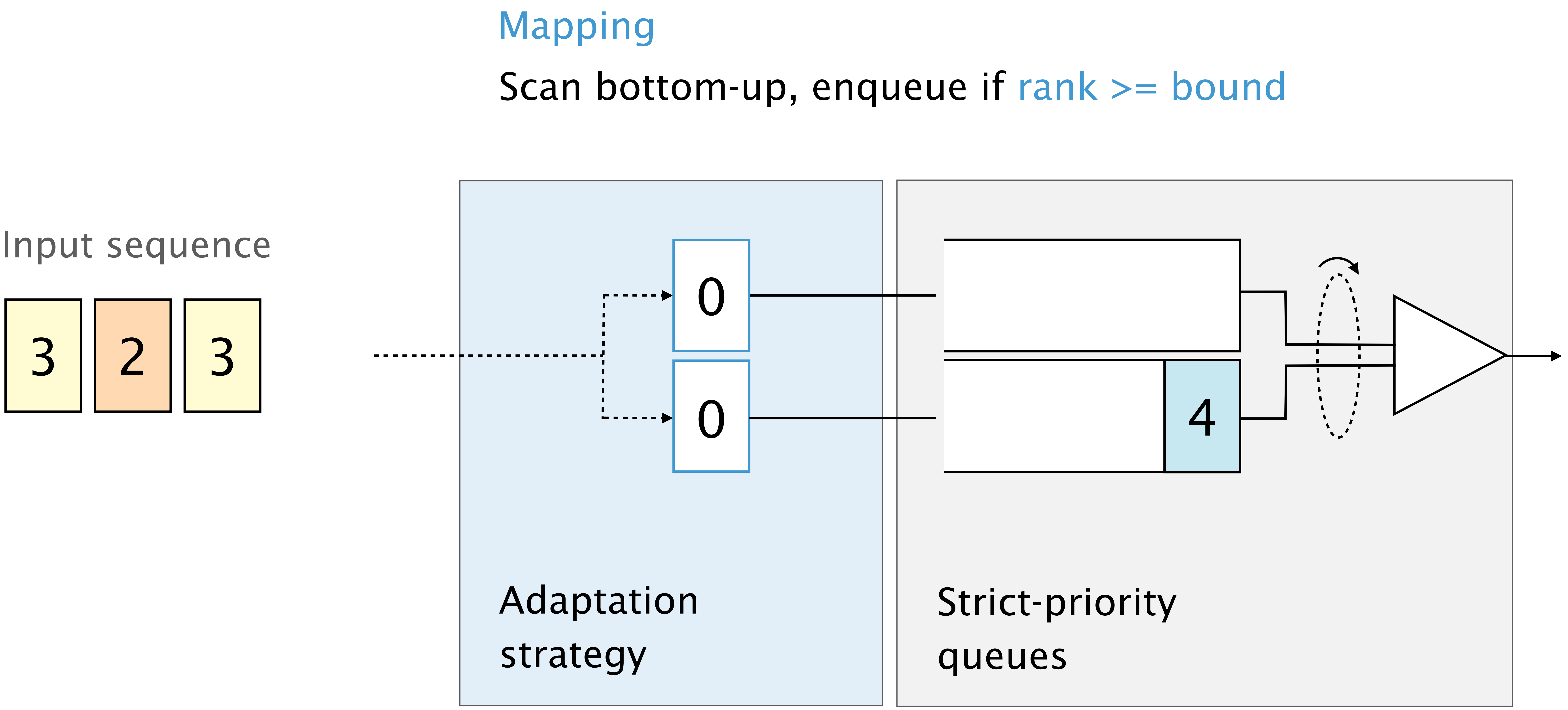
SP-PIFO adapts the mapping of packet ranks to strict-priority queues



SP-PIFO adapts the mapping of packet ranks to strict-priority queues



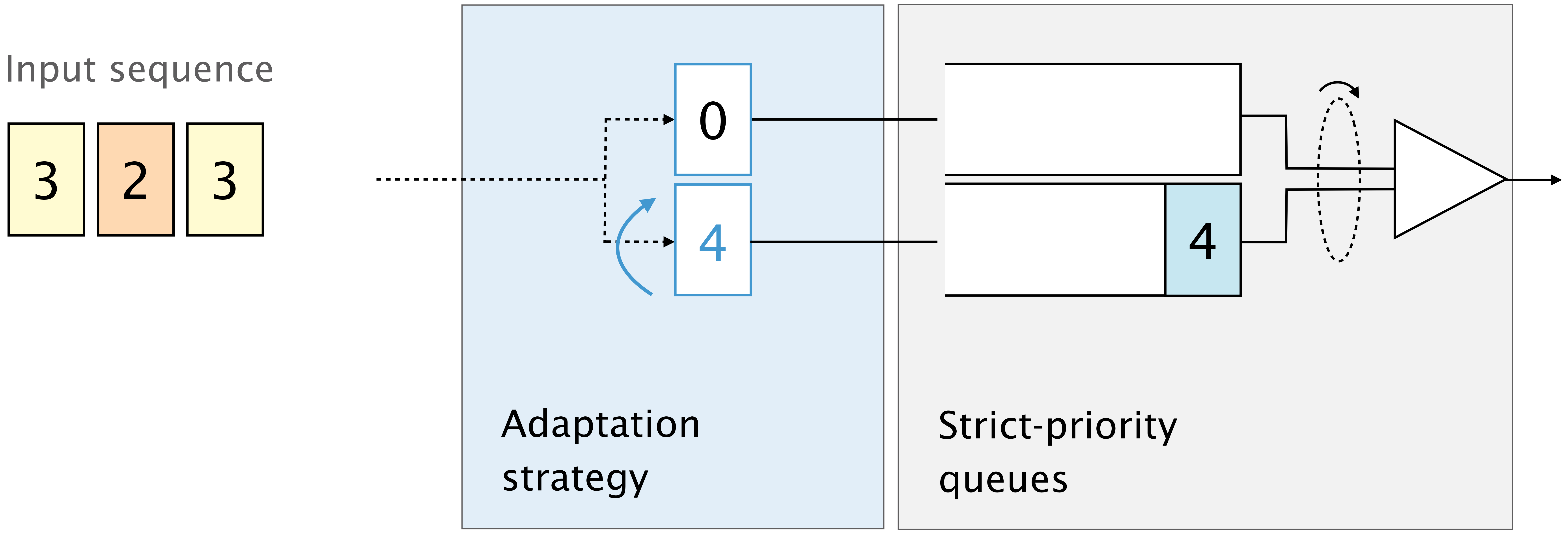
SP-PIFO adapts the mapping of packet ranks to strict-priority queues



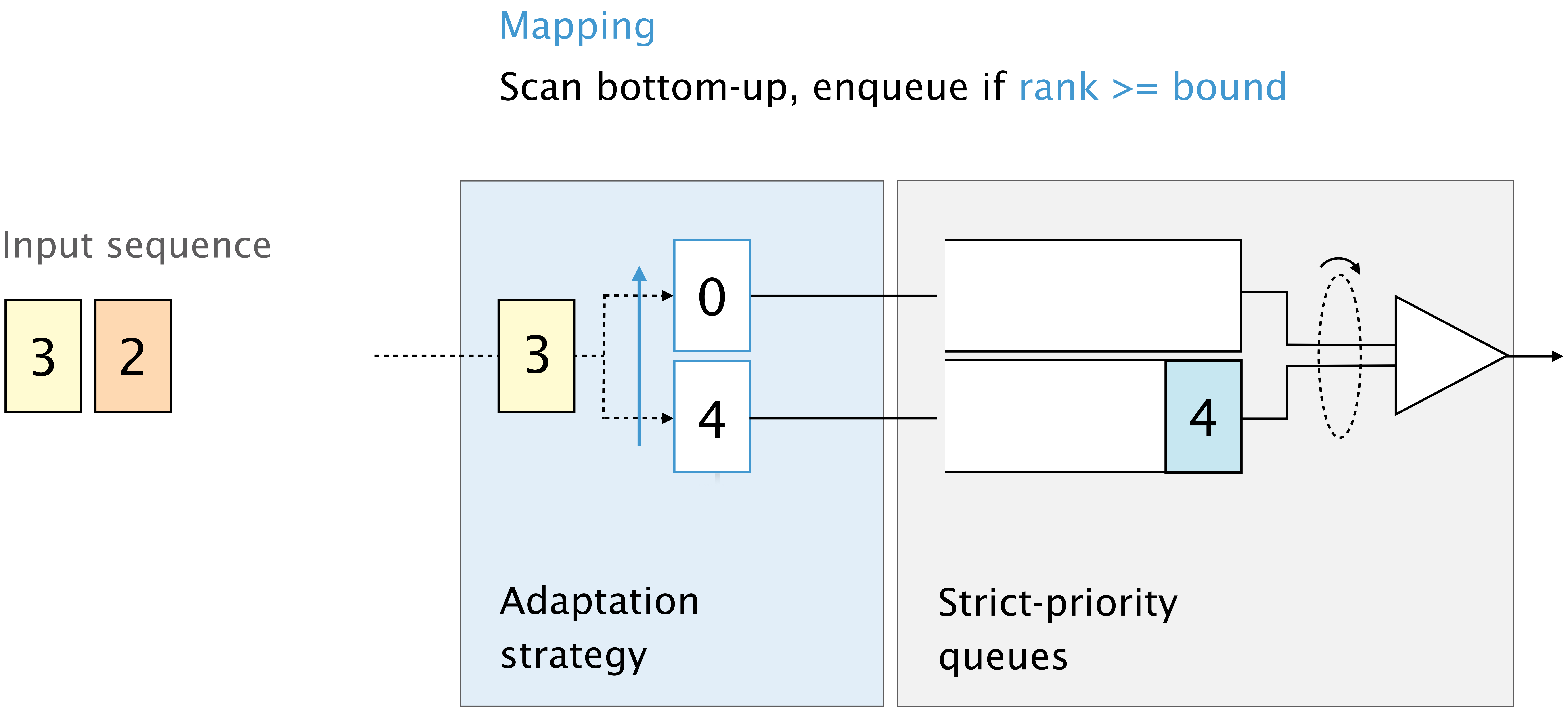
SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Push-up adaptation

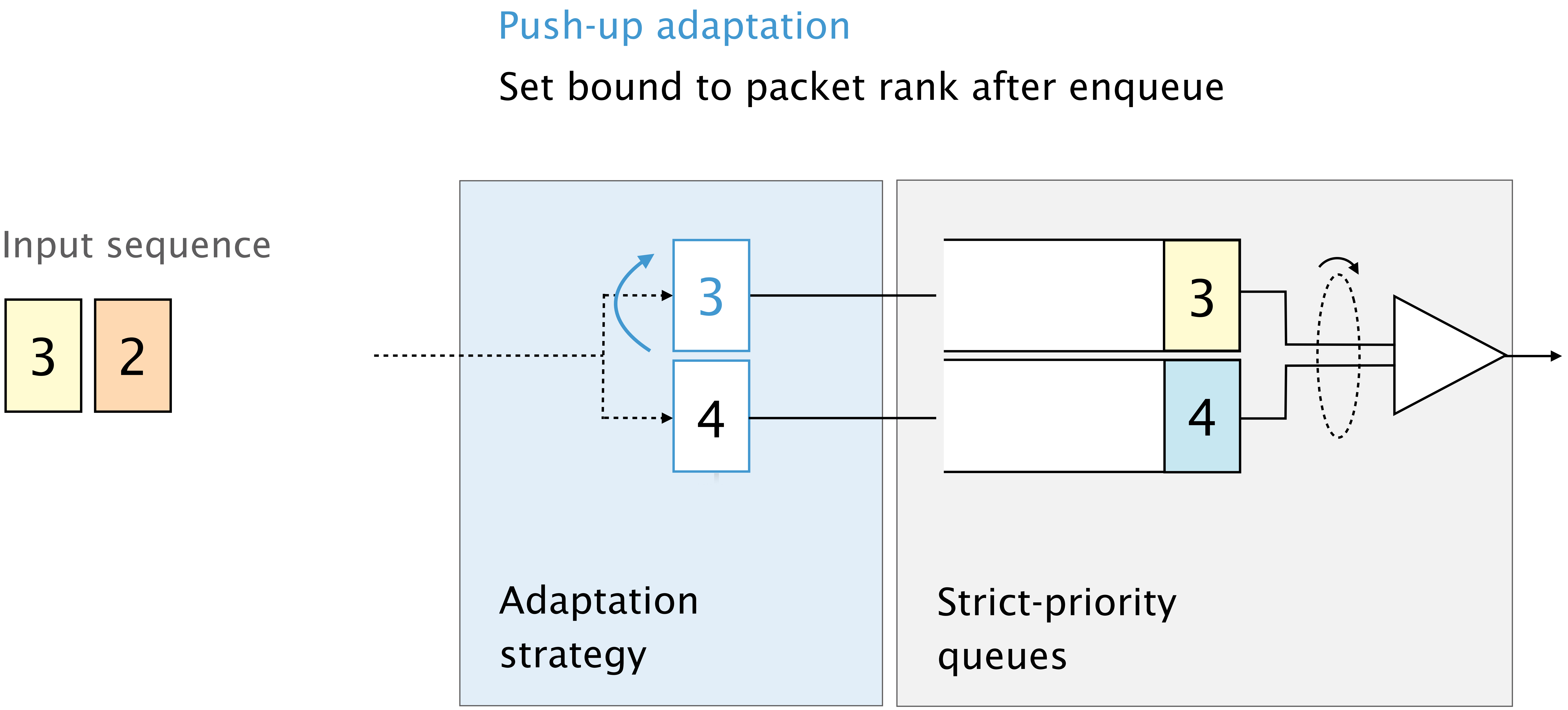
Set bound to packet rank after enqueue



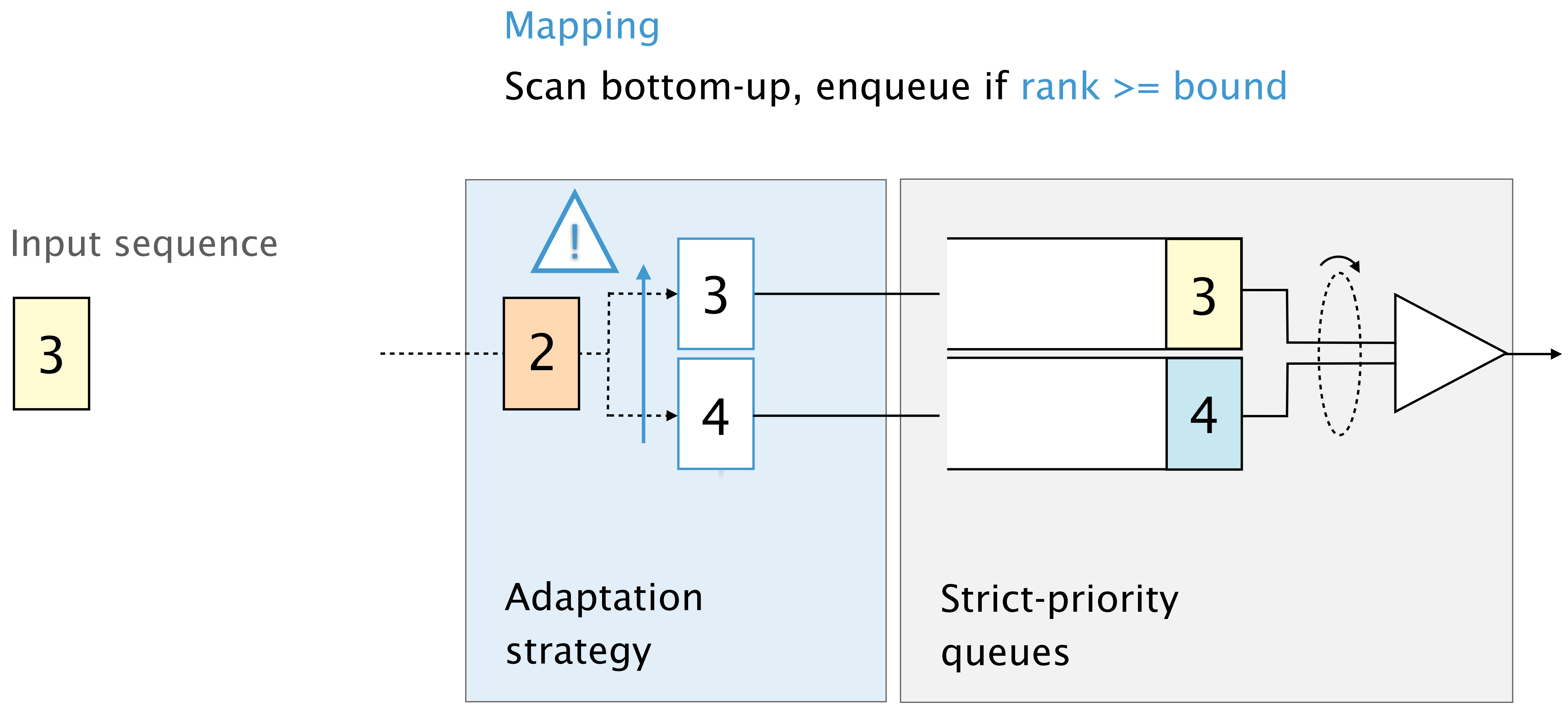
SP-PIFO adapts the mapping of packet ranks to strict-priority queues



SP-PIFO adapts the mapping of packet ranks to strict-priority queues



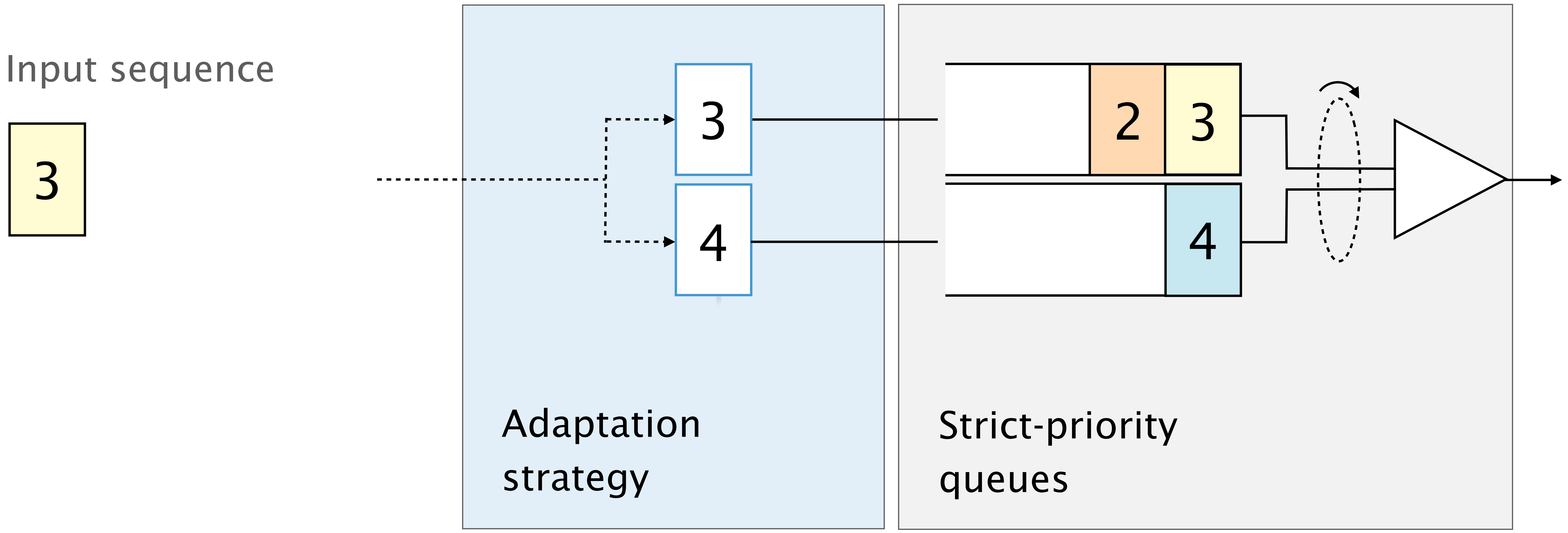
SP-PIFO adapts the mapping of packet ranks to strict-priority queues



SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Push-down adaptation

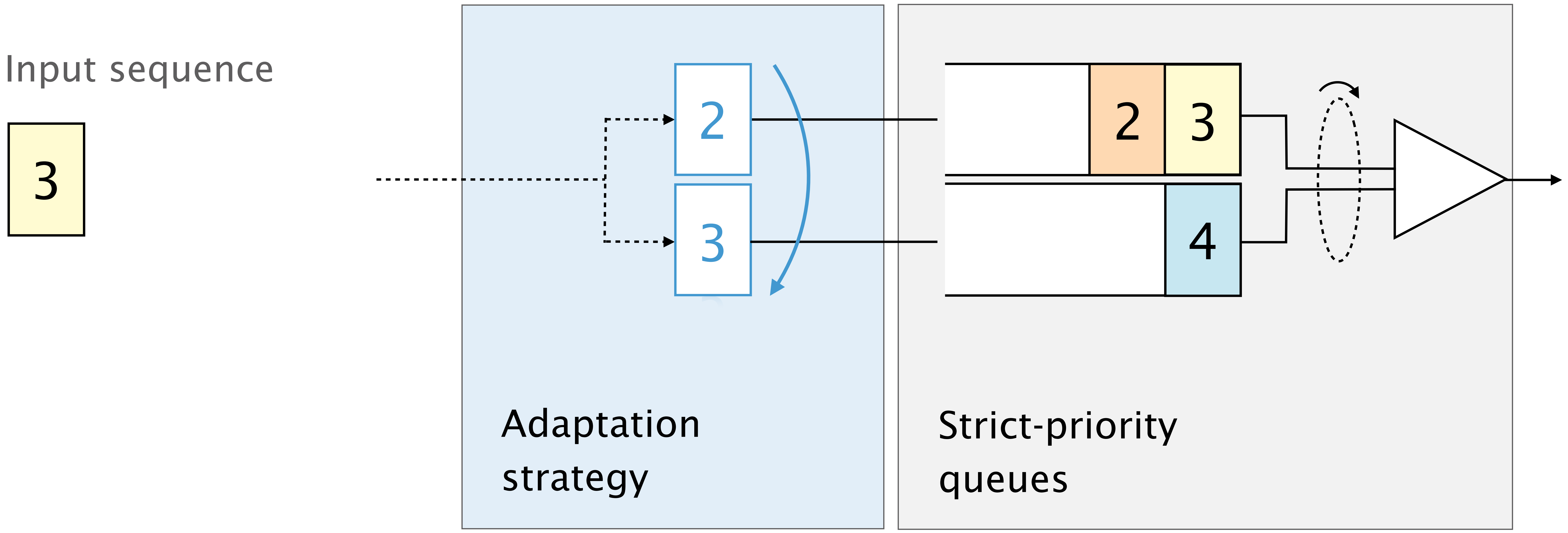
Decrease all bounds after inversion, by inversion cost



SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Push-down adaptation

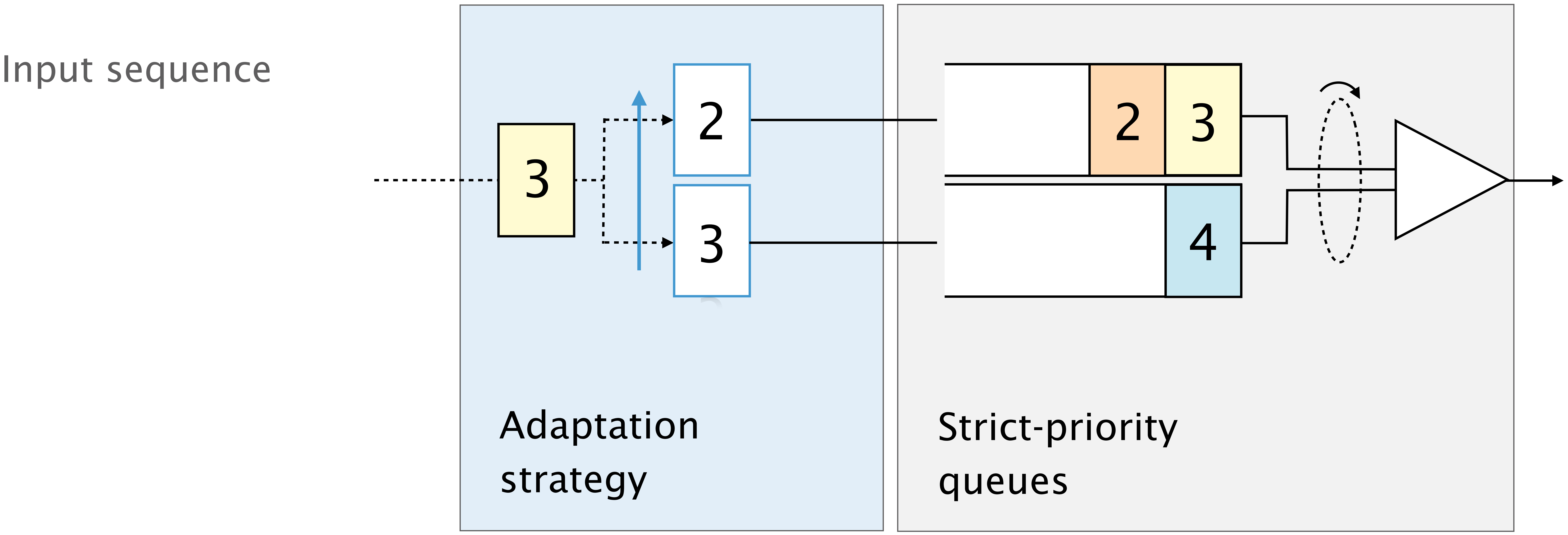
Decrease all bounds after inversion, by inversion cost



SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Mapping

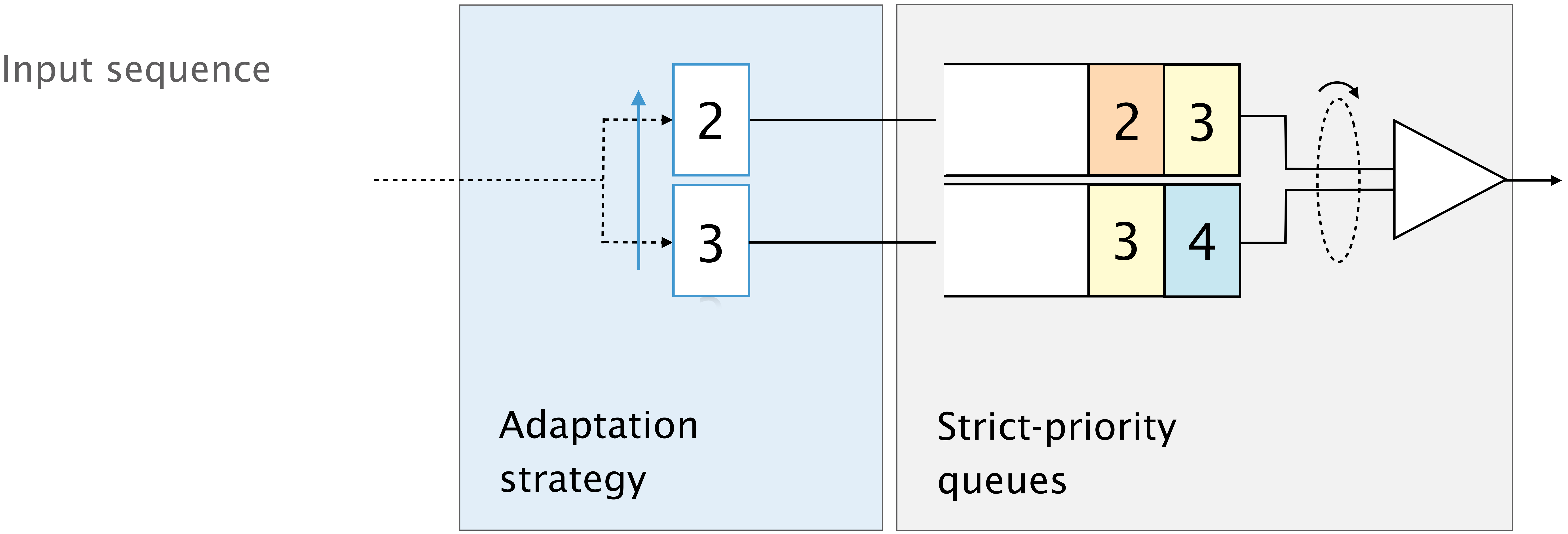
Scan bottom-up, enqueue if rank >= bound



SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Mapping

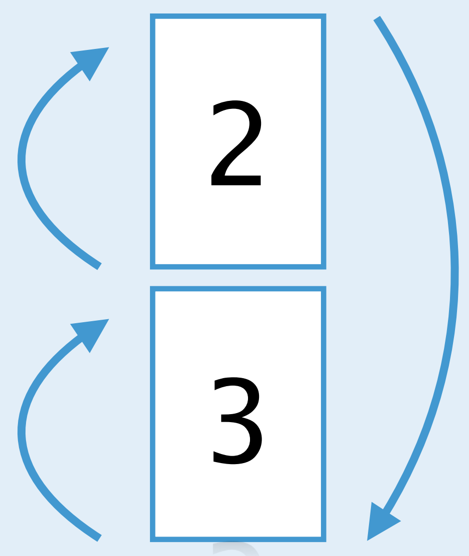
Scan bottom-up, enqueue if rank >= bound



SP-PIFO adapts the mapping of packet ranks to strict-priority queues

Push-up

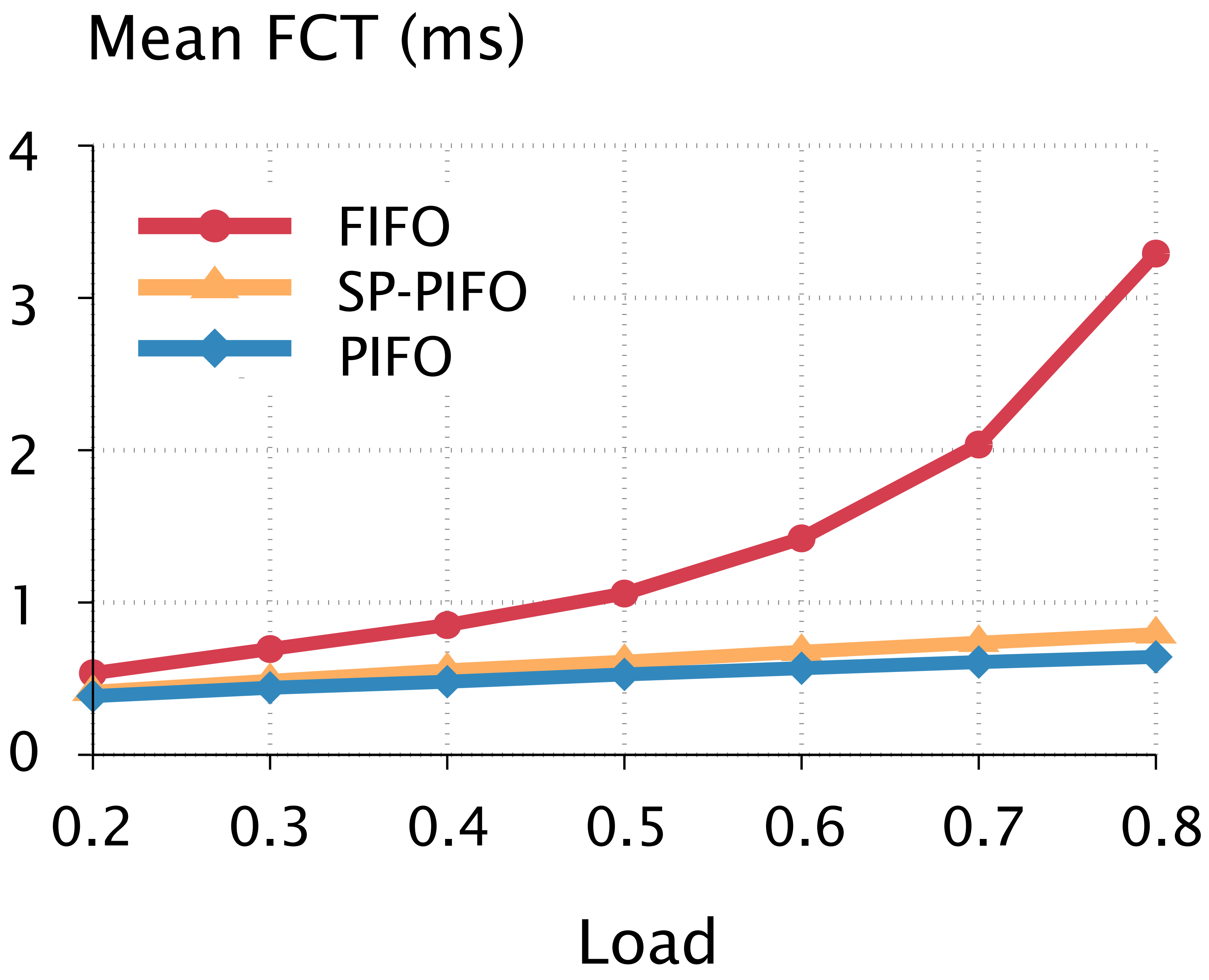
Low-rank packets
to high-priority queues



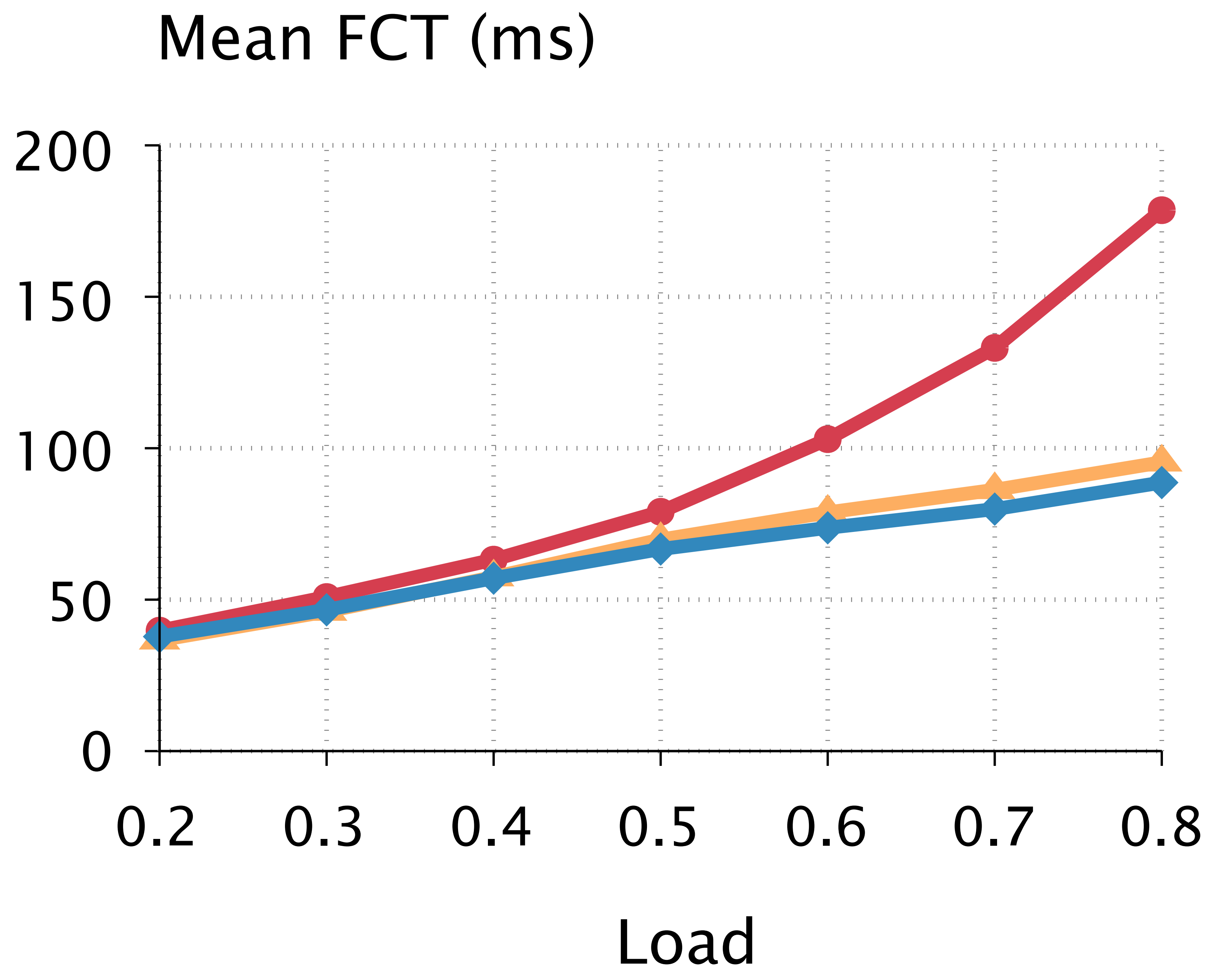
Push-down

High-rank packets
to low-priority queues

SP-PIFO allows us to minimize flow completion times (FCTs)



Small flows <100KB



Big flows ≥1MB

How to enable programmable scheduling on existing devices?

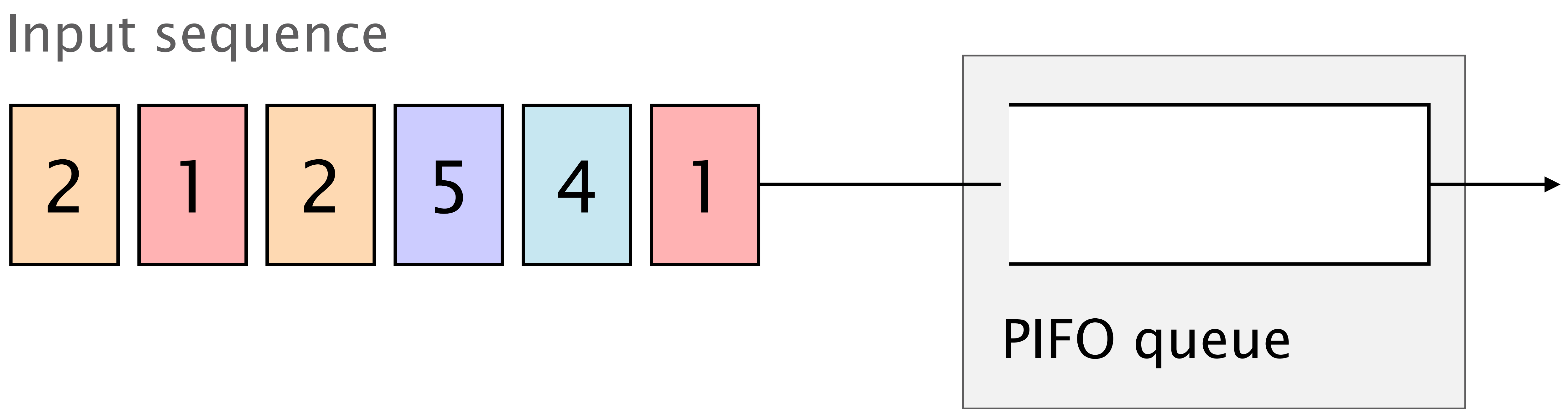
SP-PIFO [NSDI'20]	PACKS [NSDI'25]
Approximating PIFO's scheduling	Incorporating PIFO's admission

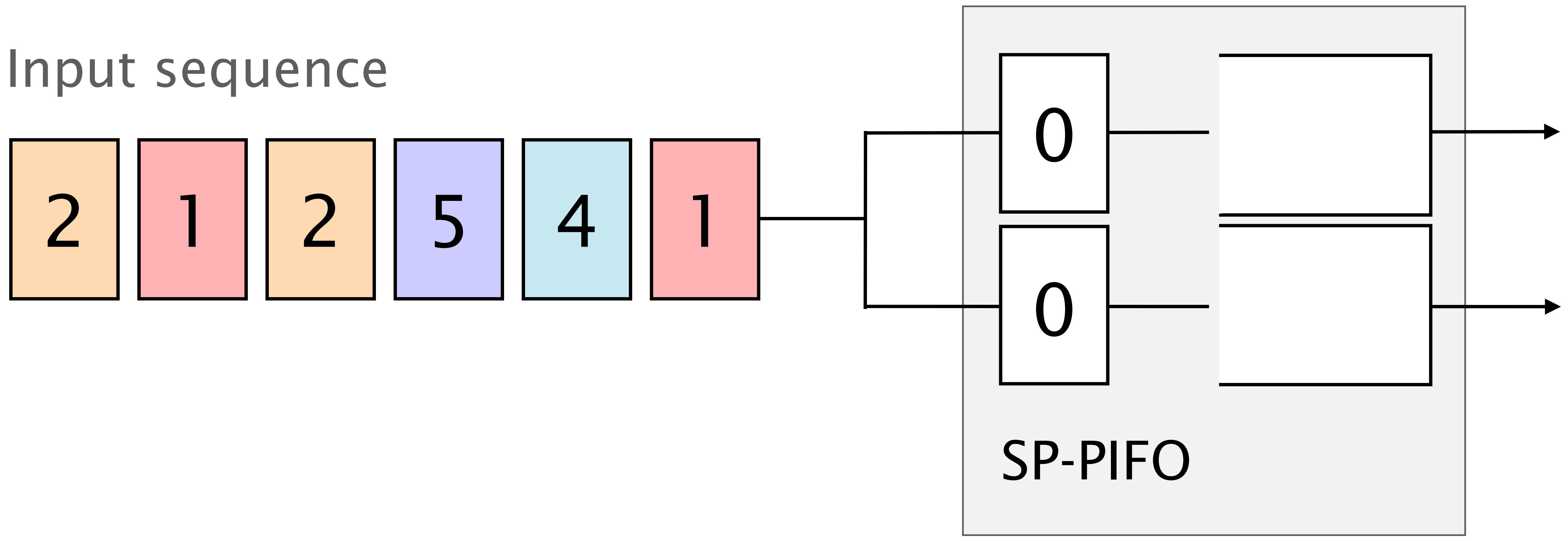
How to use it to improve the Internet's security?

ACC-Turbo [SIGCOMM'22]

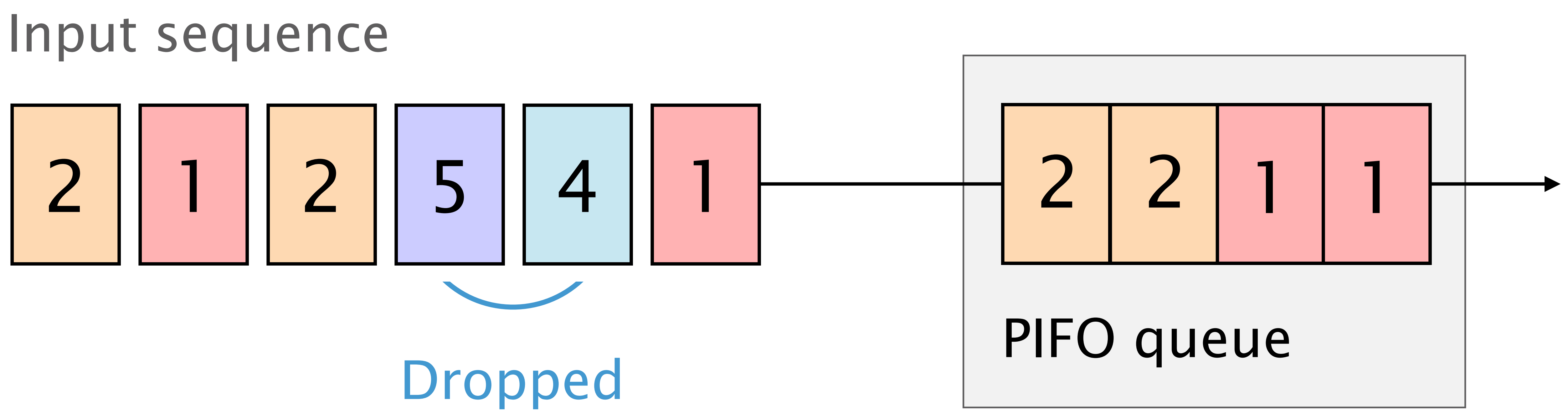
Mitigating DDoS attacks

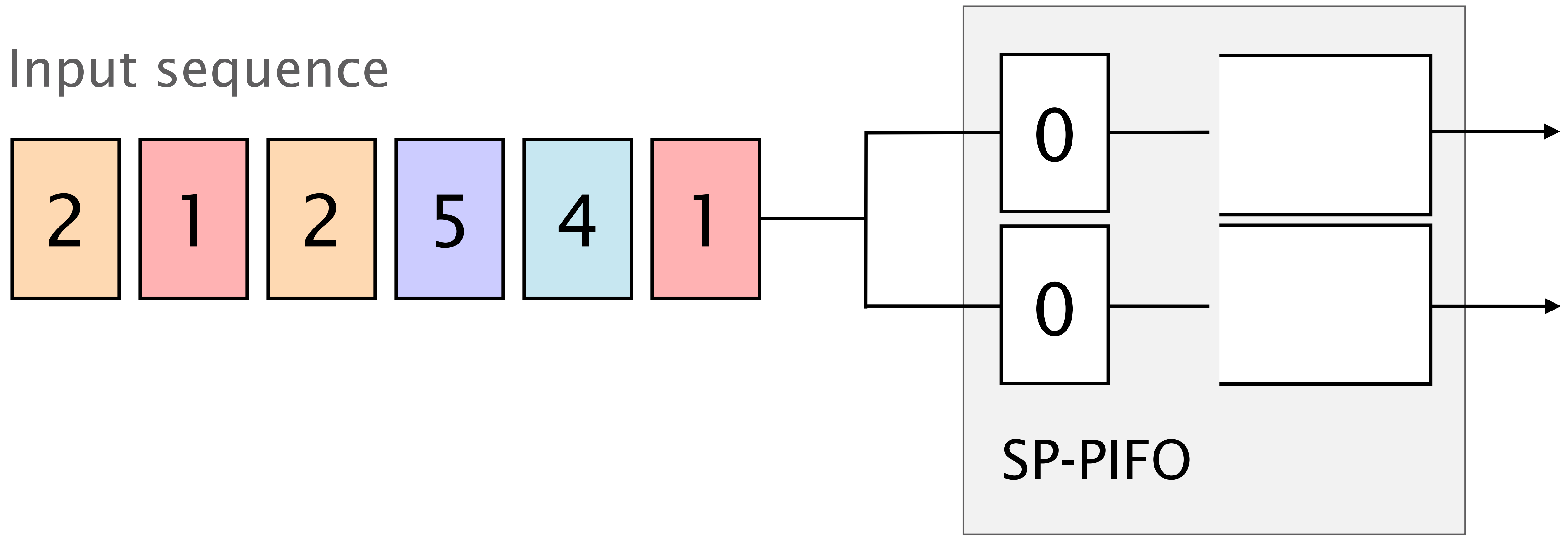
PIFO's admission prevents the dropping of important packets



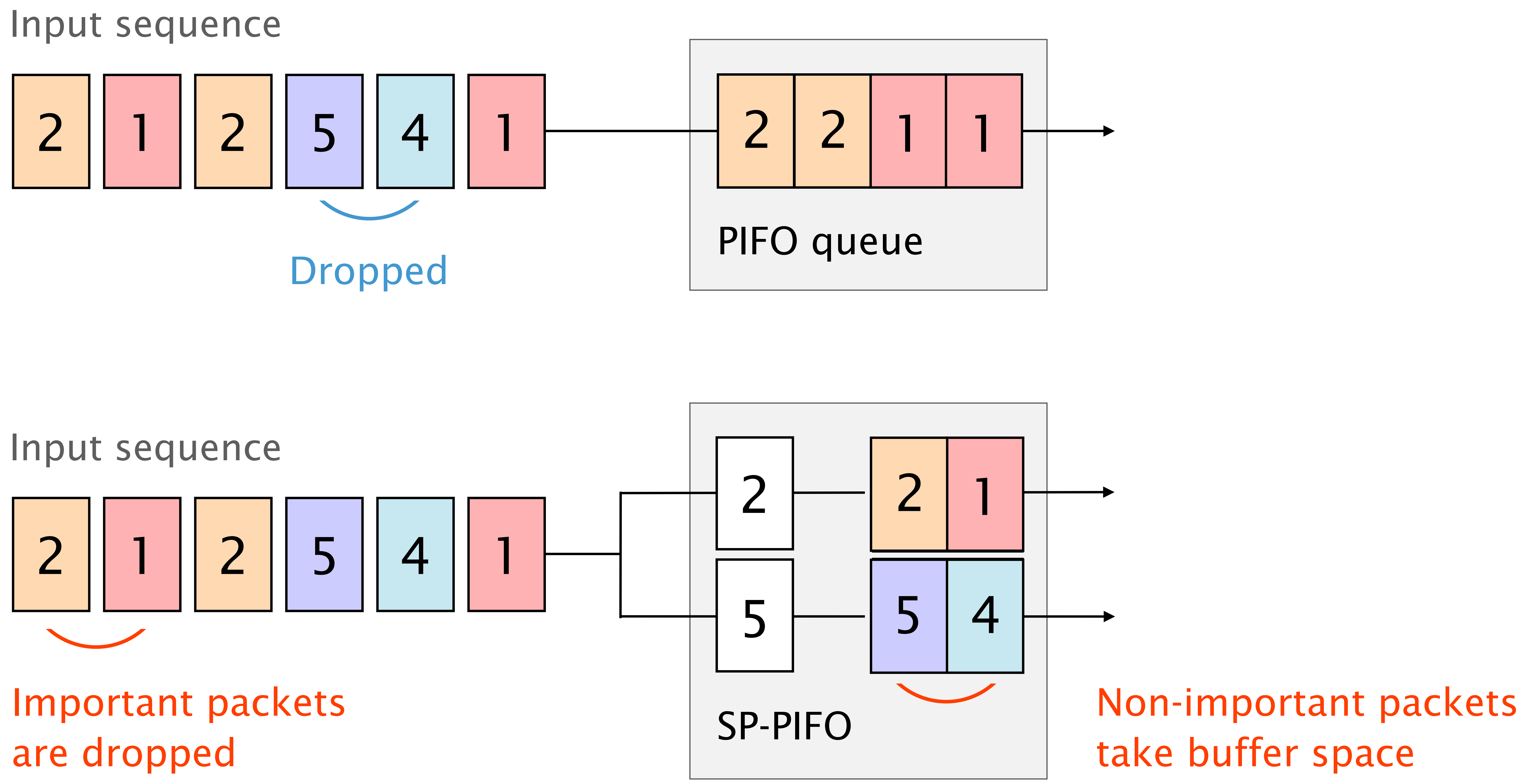


PIFO's admission prevents the dropping of important packets

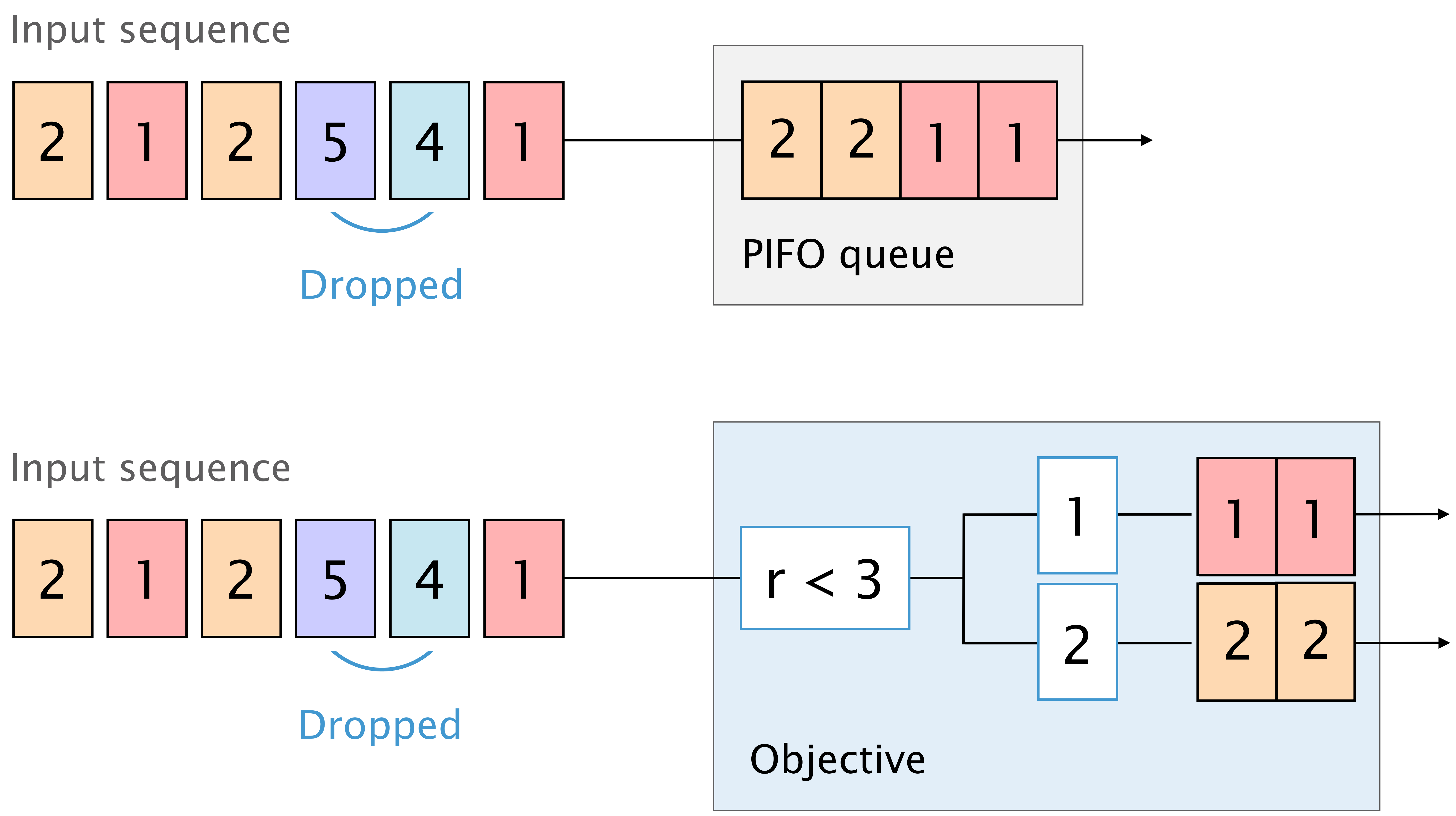




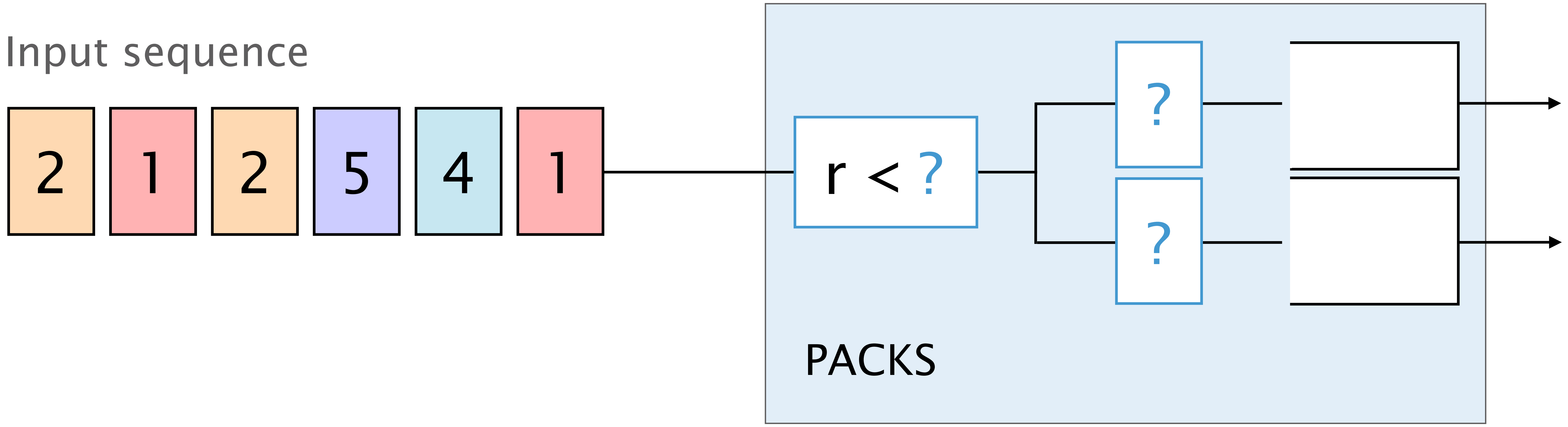
PIFO's admission prevents the dropping of important packets



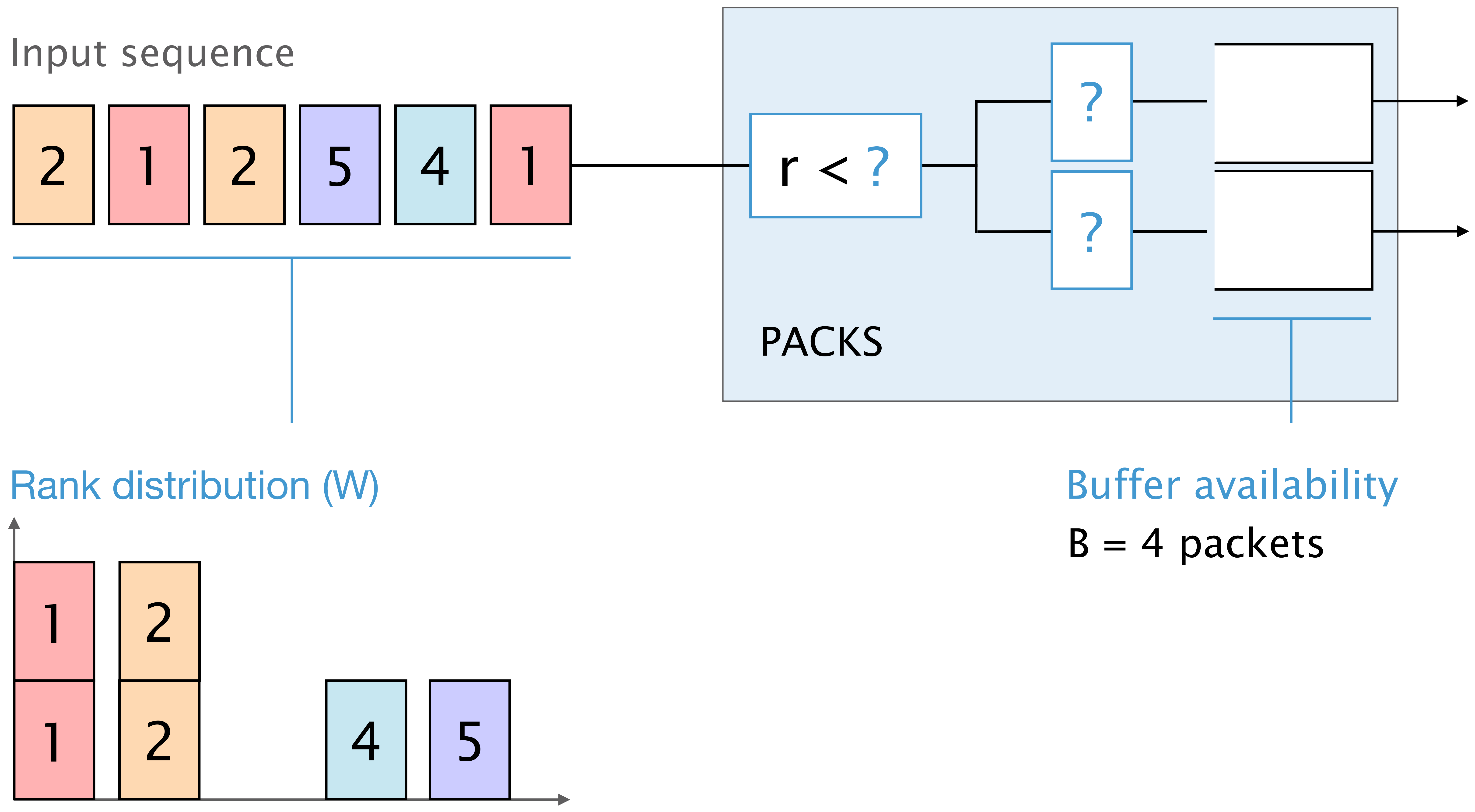
We need to preemptively block non-important packets



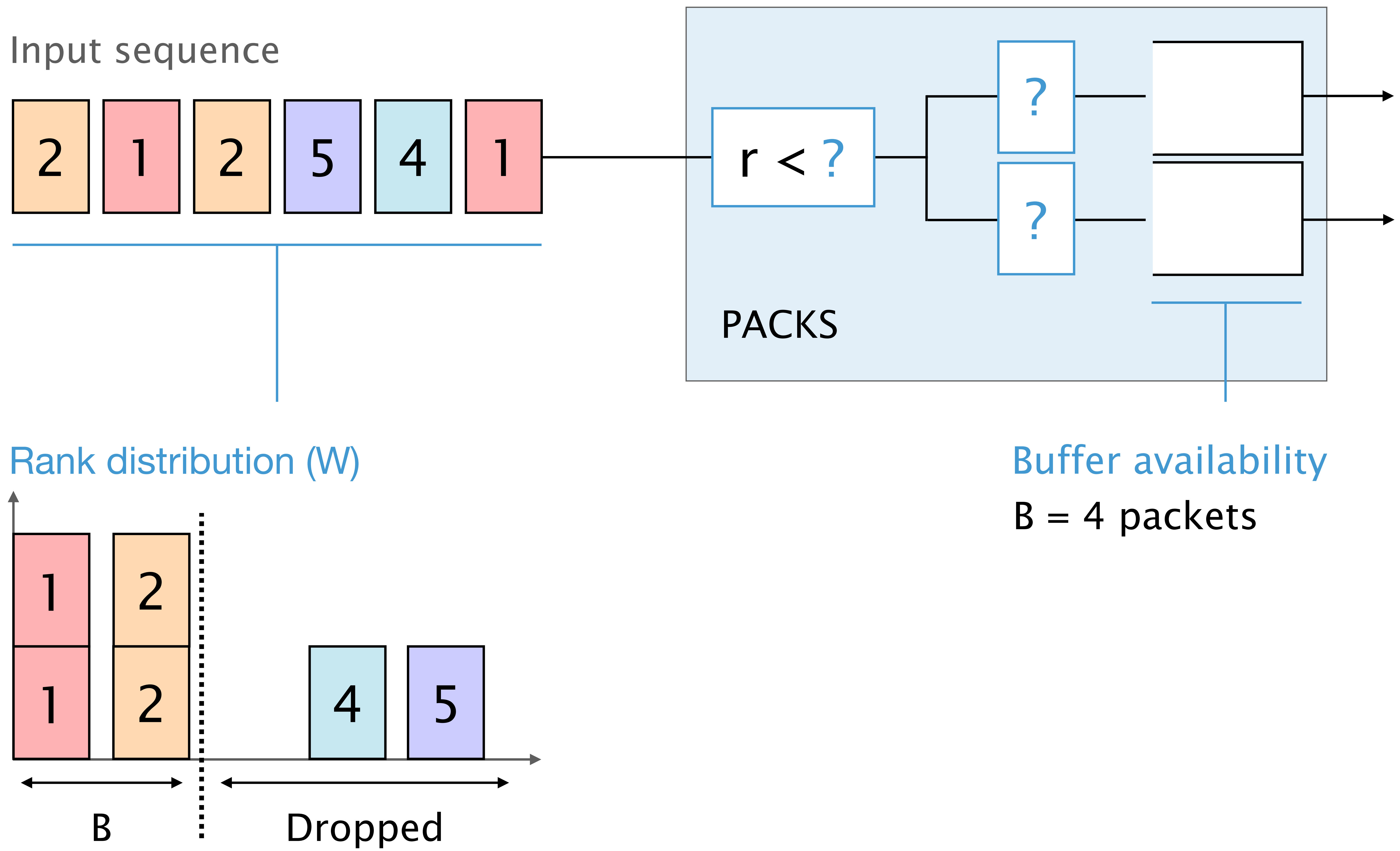
We need to preemptively block non-important packets



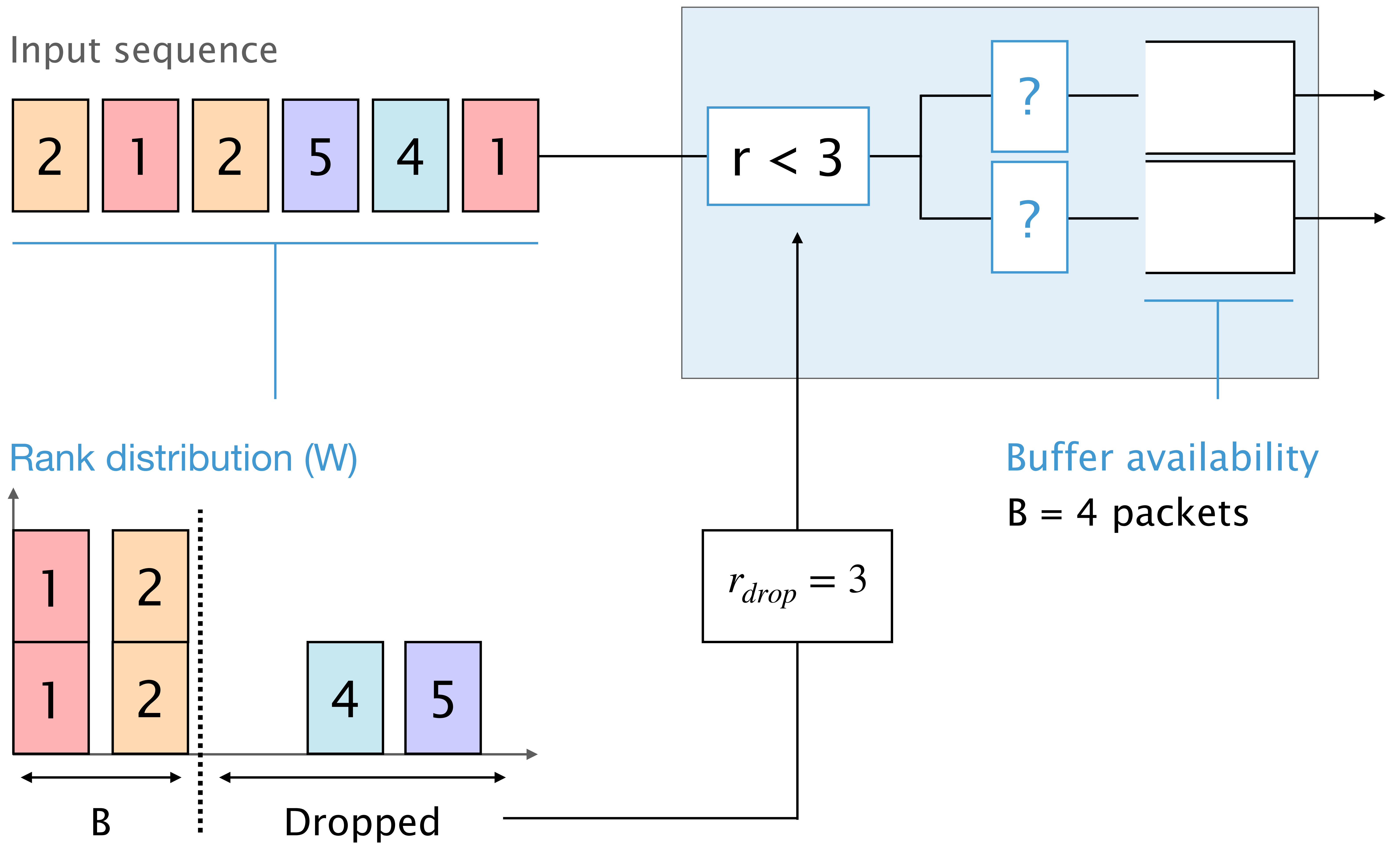
PACKS monitors the rank distribution and the queue occupancy



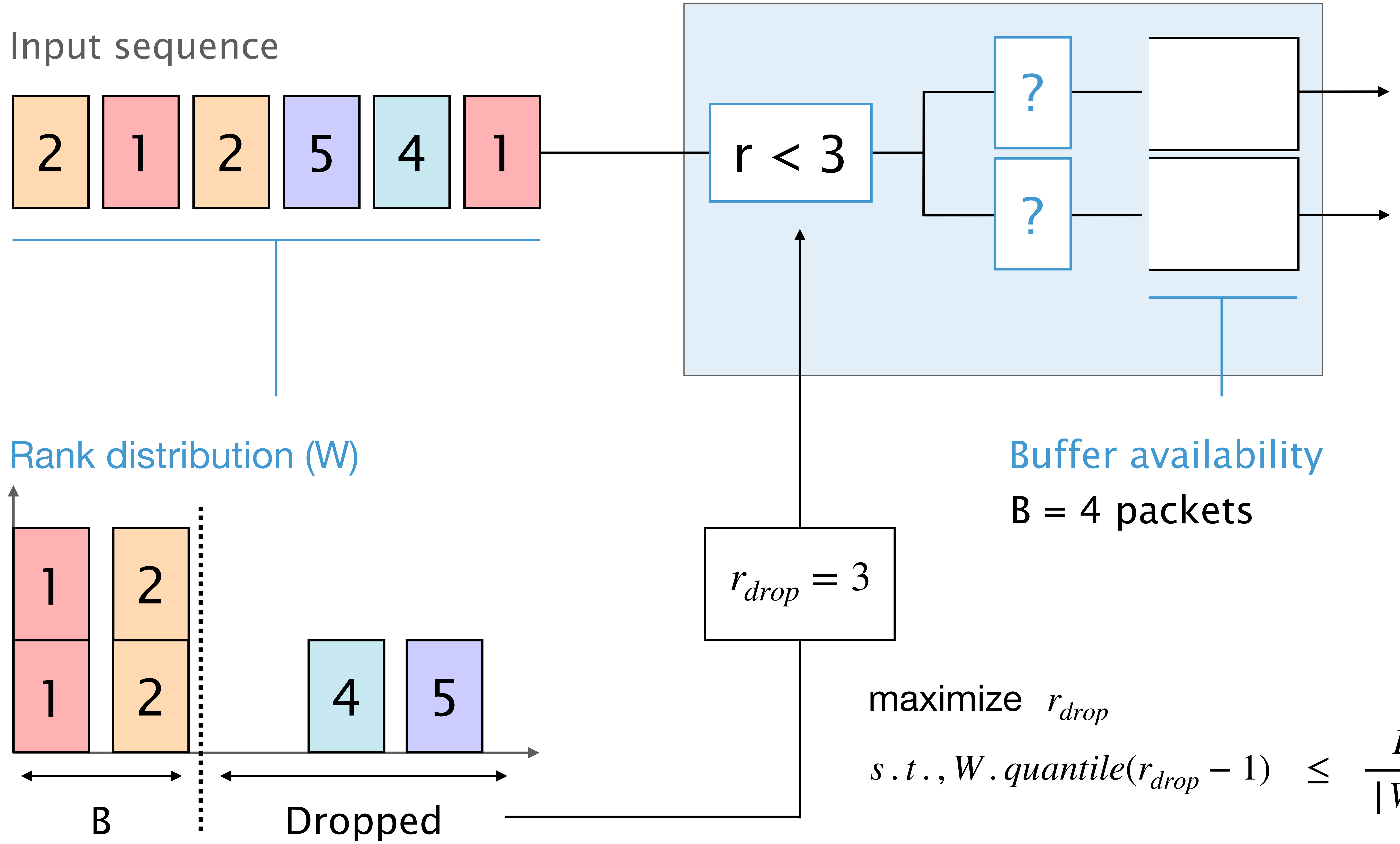
PACKS monitors the rank distribution and the queue occupancy



PACKS monitors the rank distribution and the queue occupancy



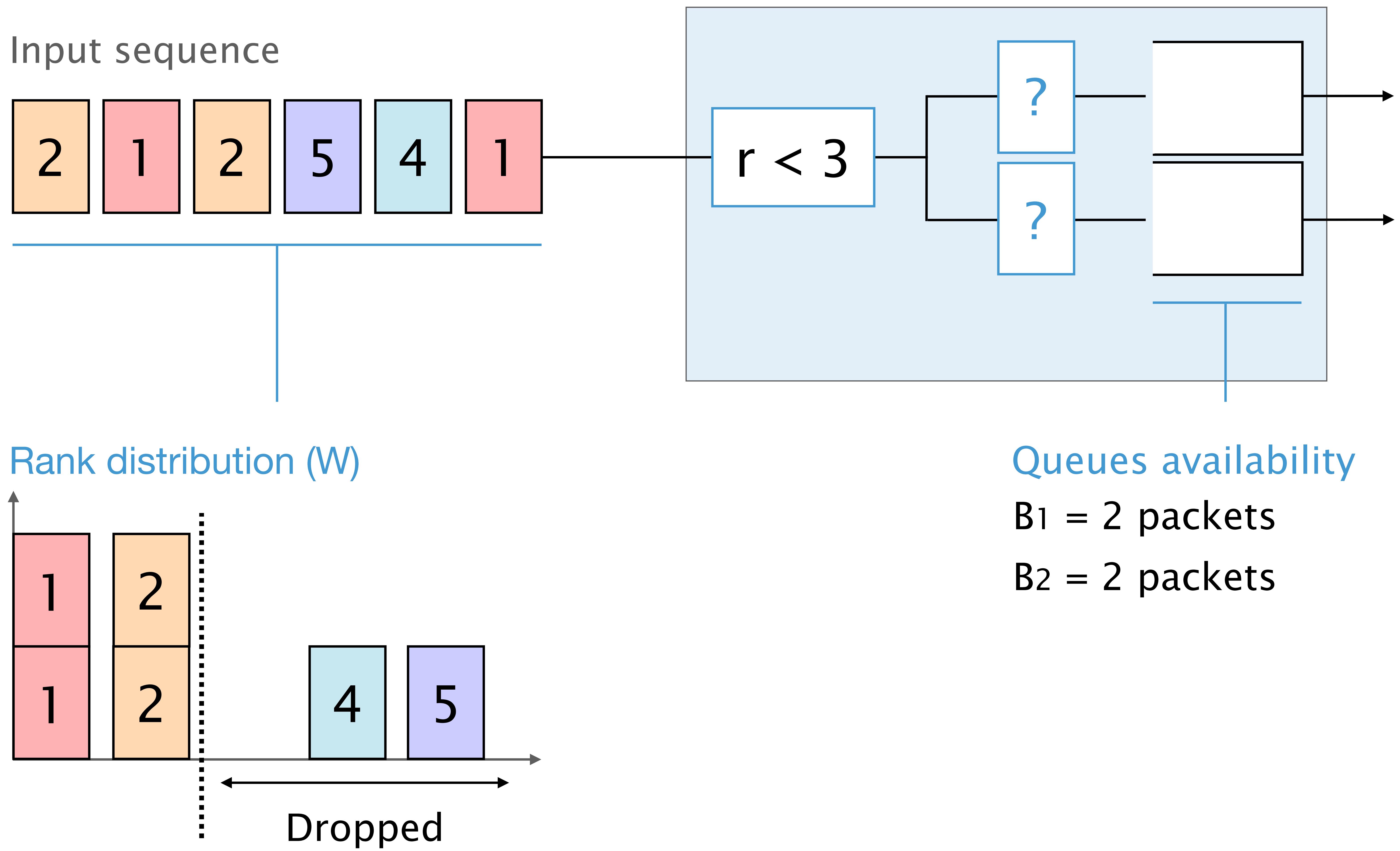
PACKS monitors the rank distribution and the queue occupancy



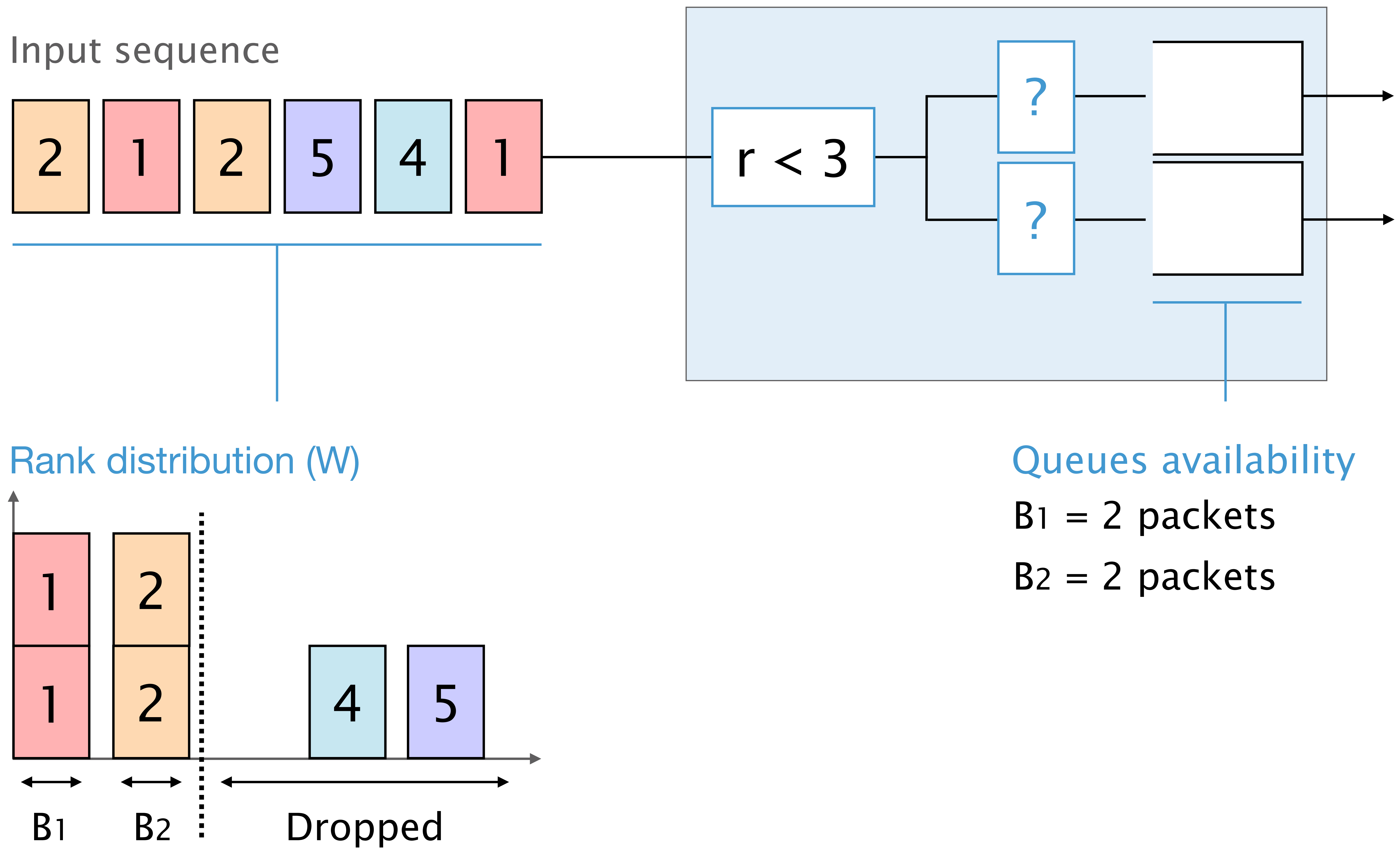
maximize r_{drop}

$$s.t., W.quantile(r_{drop} - 1) \leq \frac{B}{|W|}$$

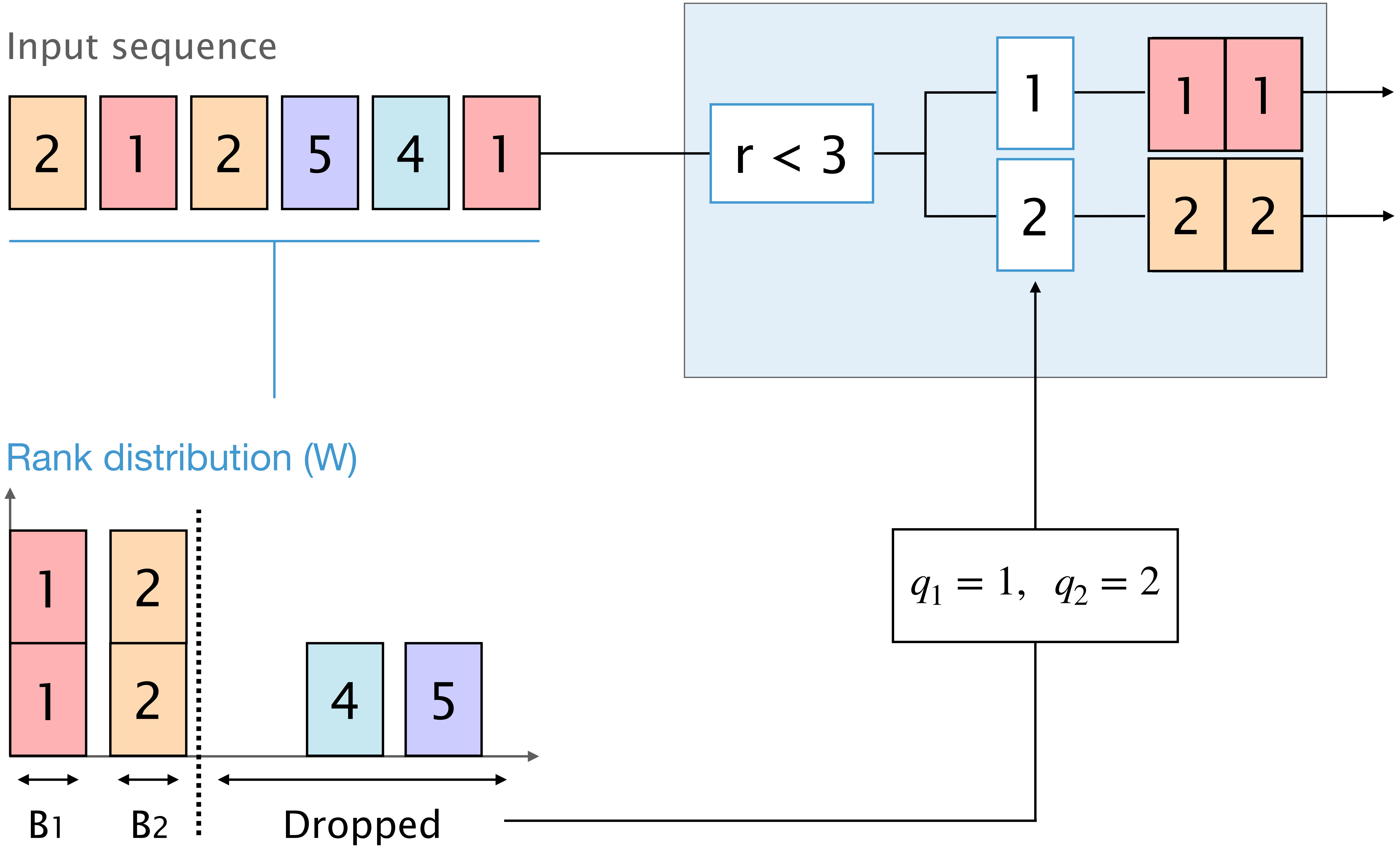
PACKS monitors the rank distribution and the queue occupancy



PACKS monitors the rank distribution and the queue occupancy



PACKS monitors the rank distribution and the queue occupancy



SP-PIFO	PACKS
Per-packet heuristic	Window-based
No traffic knowledge	Rank-distribution aware
No queue information	Queue-occupancy aware

SP-PIFO

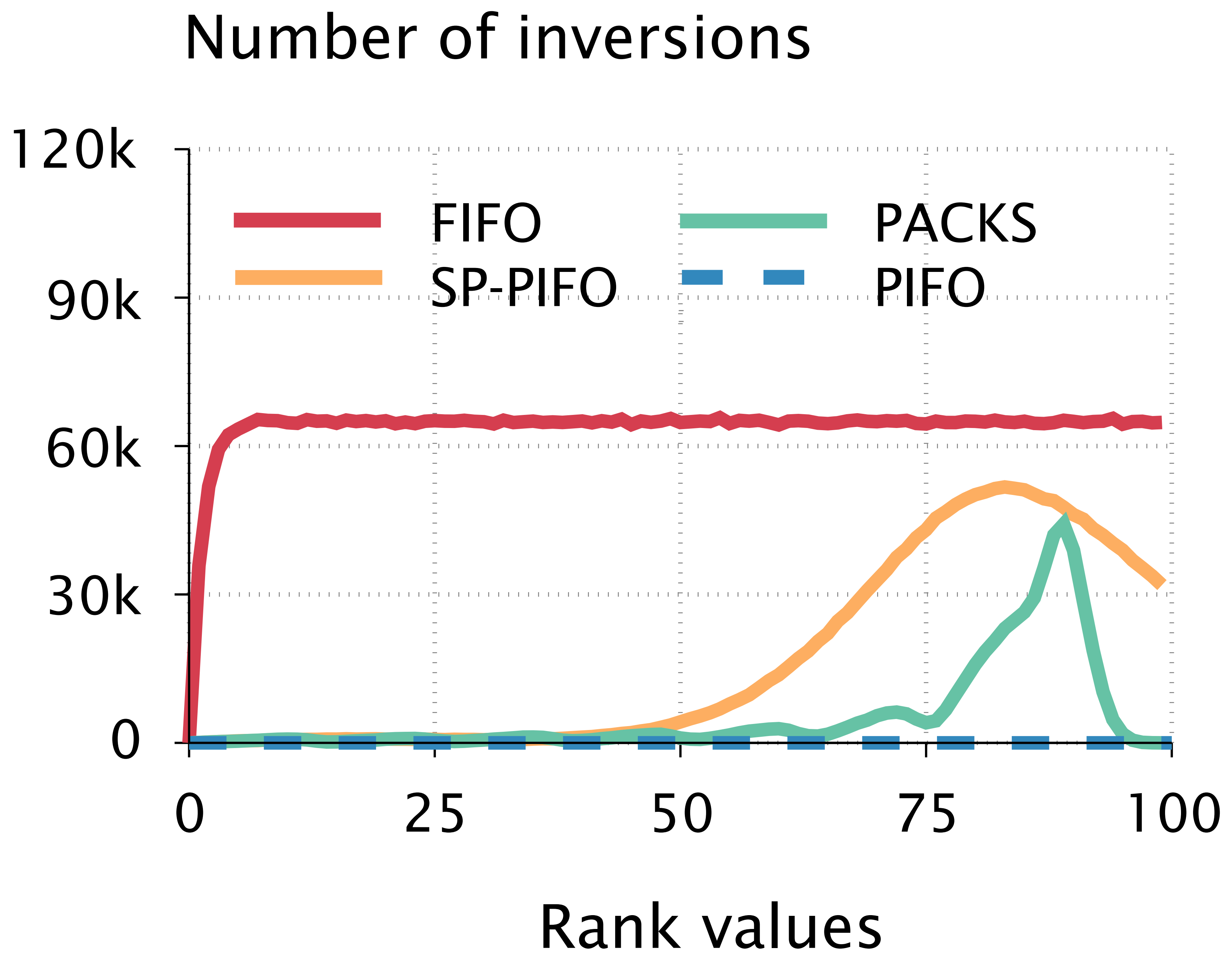
Scheduling ✔

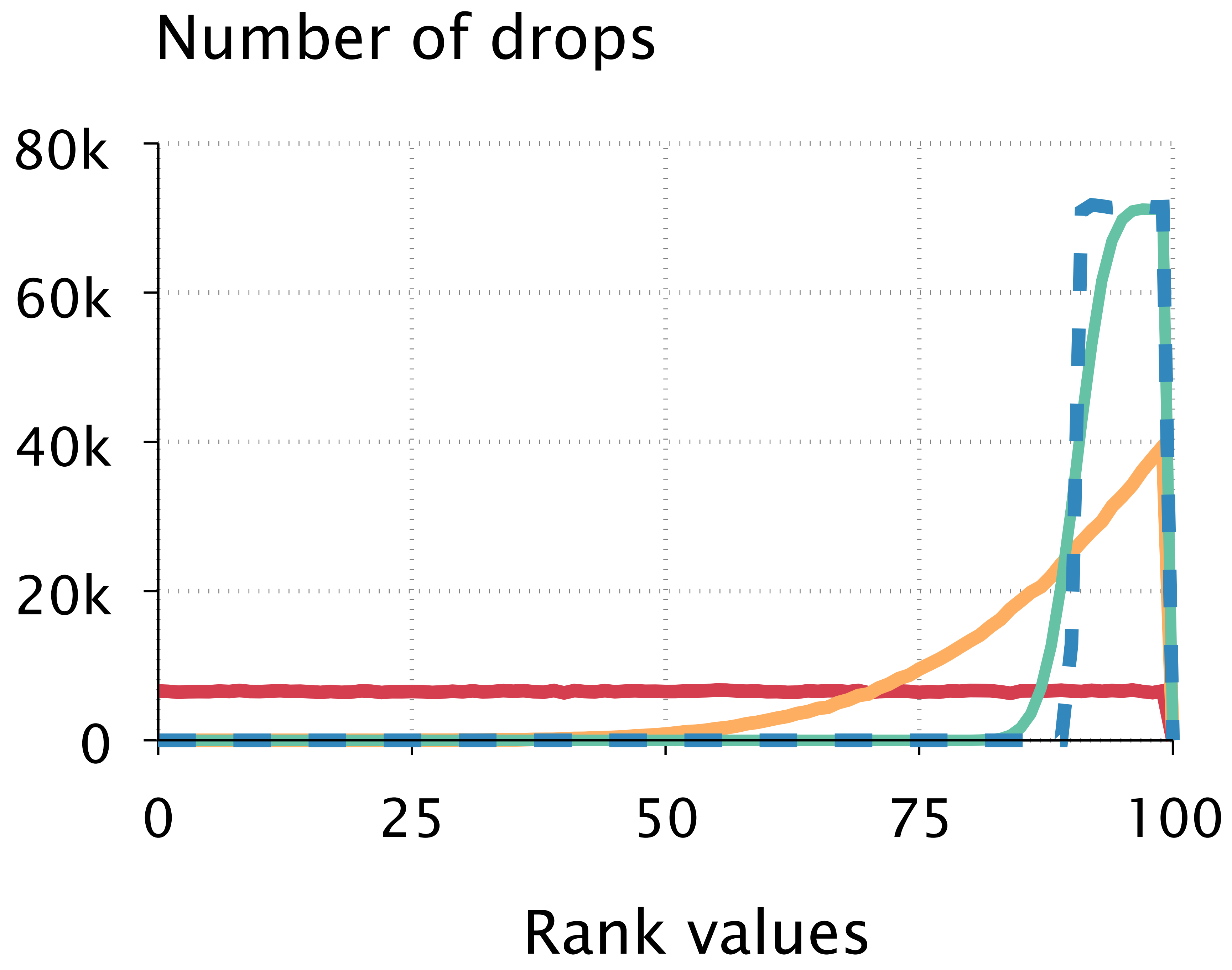
PACKS

Scheduling ✔

Admission ✔

PACKS reduces inversions by up to 7x and drops by up to 60% with respect to SP-PIFO





How to enable programmable scheduling on existing devices?

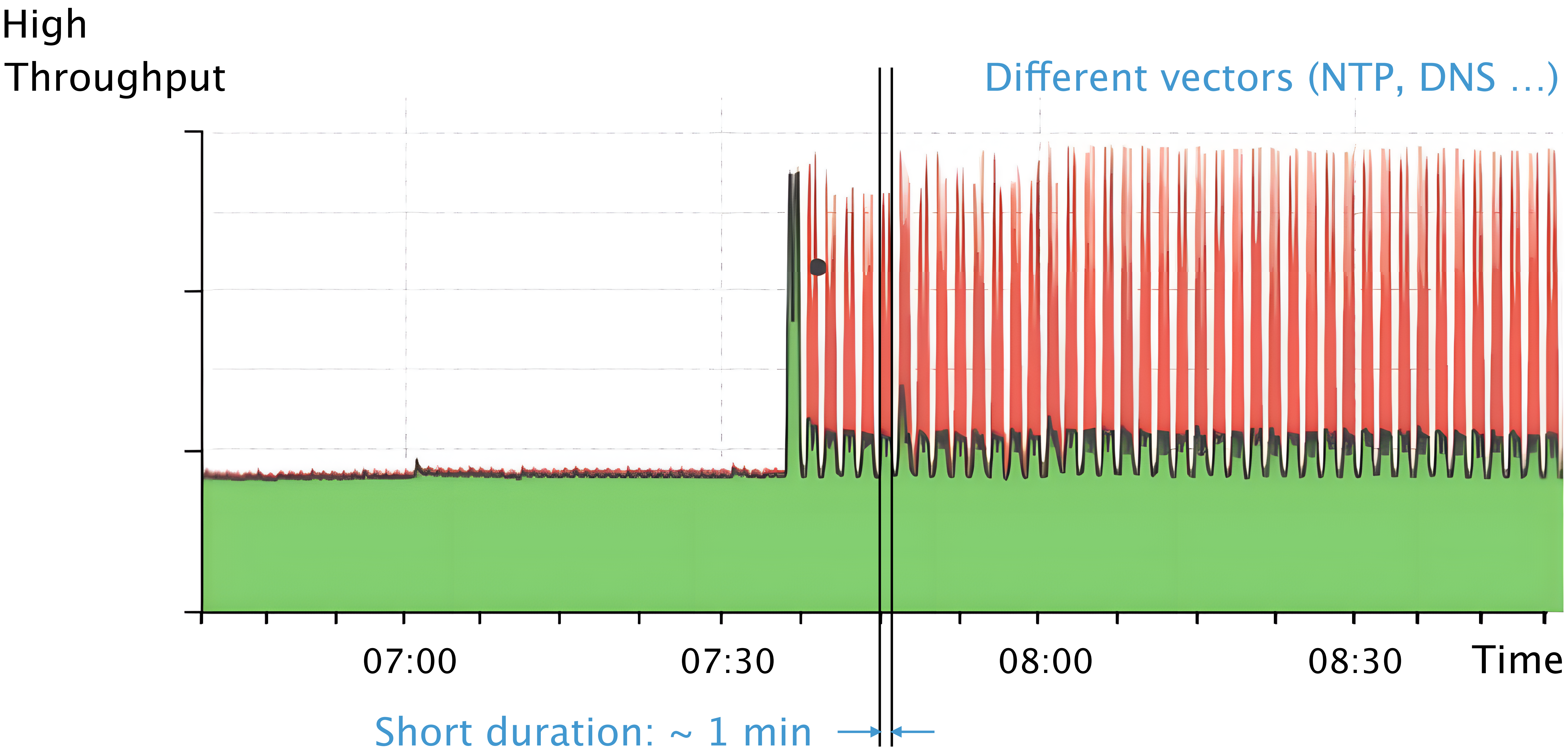
SP-PIFO [NSDI'20]	PACKS [NSDI'25]
Approximating PIFO's scheduling	Incorporating PIFO's admission

How to use it to improve the Internet's security?

ACC-Turbo [SIGCOMM'22]

Mitigating DDoS attacks

Pulse-wave DDoS attacks are an extreme case of congestion



Damian Menscher, Google 2021

A pulse-wave DDoS defense needs to be …

Fast

reaction

Generic

detection

A pulse-wave DDoS defense needs to be …

Fast reaction	In-network, at line rate with limited resources
Generic detection	Unsupervised techniques with uncertainty

A pulse-wave DDoS defense needs to be …

Fast reaction	In–network, at line rate with limited resources	Risk of false positives
Generic detection	Unsupervised techniques with uncertainty	

A pulse-wave DDoS defense needs to be …

Fast reaction	In–network, at line rate with limited resources	Risk of false positives
Generic detection	Unsupervised techniques with uncertainty	
Safe mitigation	Limited impact under misclassification	

A pulse-wave DDoS defense needs to be …

Fast
reaction

In–network, at line rate
with limited resources

Generic
detection

Unsupervised techniques
with uncertainty

Risk of false positives

Safe
mitigation

Limited impact
under misclassification

Filtering ✗

Rerouting ✗

A pulse-wave DDoS defense needs to be …

Fast reaction	In–network, at line rate with limited resources	Risk of false positives
Generic detection	Unsupervised techniques with uncertainty	
Safe mitigation	Limited impact under misclassification	Programmable scheduling

Programmable scheduling is a safe mitigation technique

Leverages the whole uncertainty spectrum

with fine-grained scheduling policies

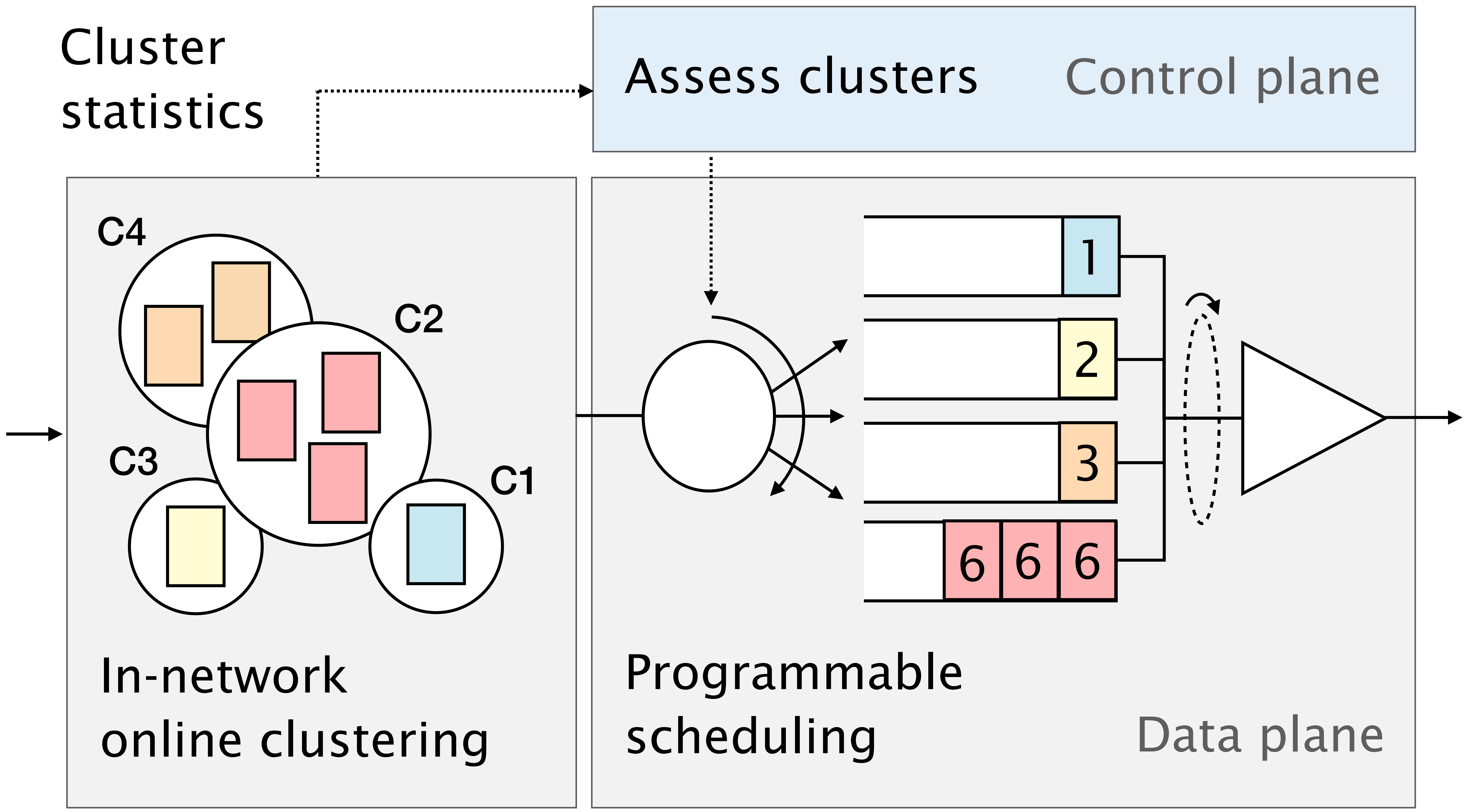
Only drops under congestion

starting by most-malicious packets

Does not require activation

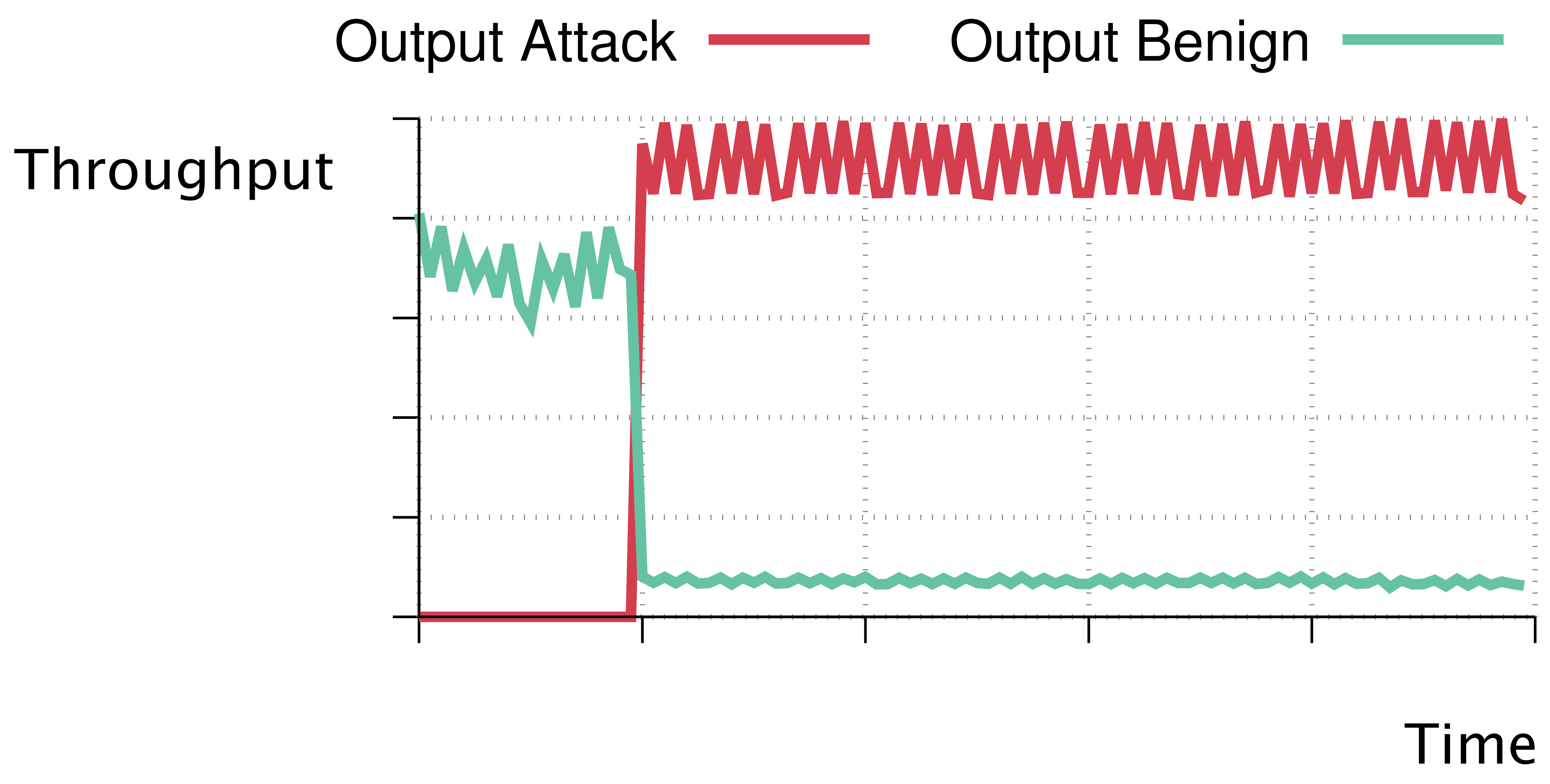
can be always-on

ACC-Turbo utilizes online clustering and programmable scheduling

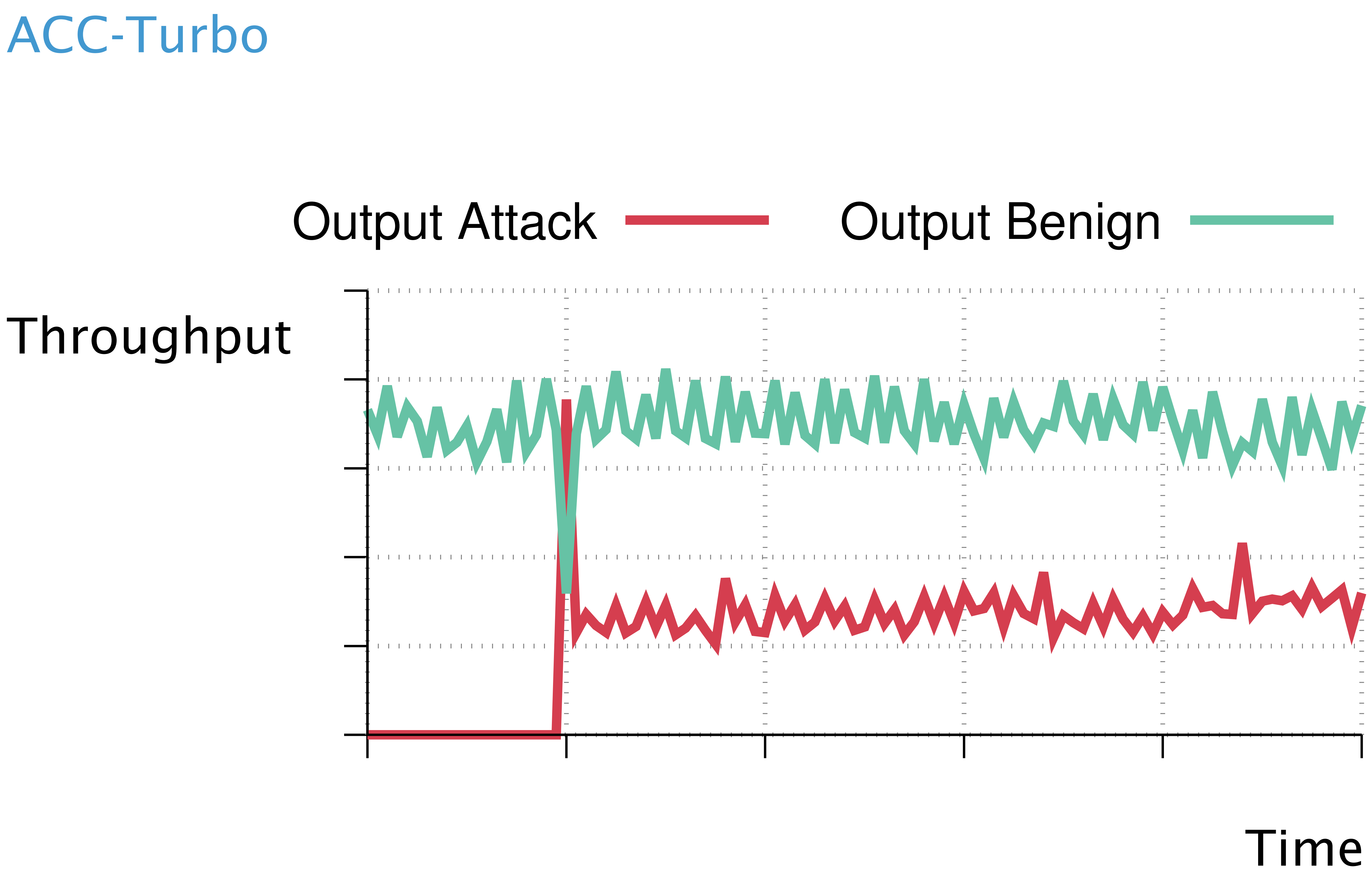


ACC-Turbo outperforms existing defenses and mitigates pulse-wave DDoS attacks

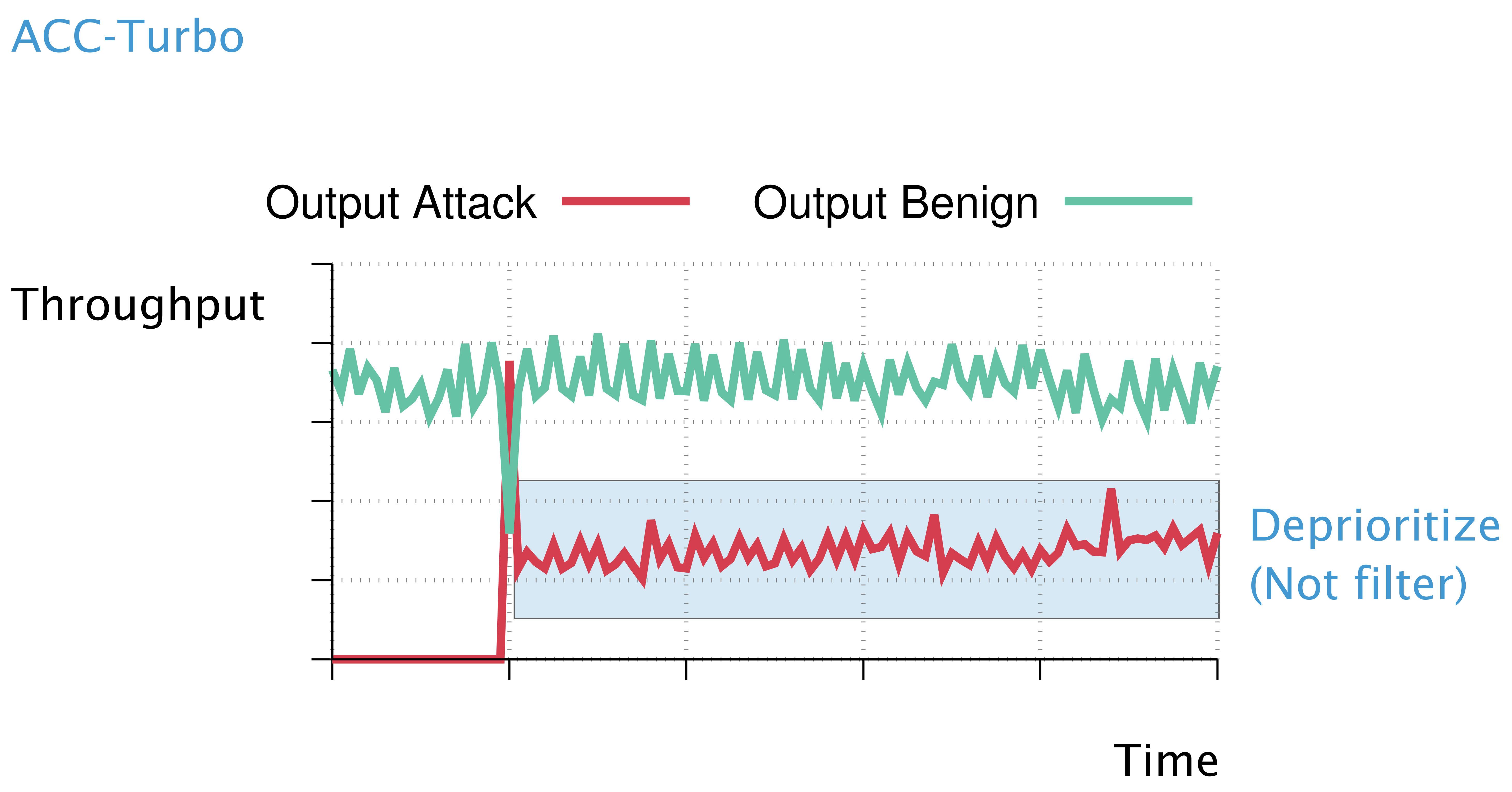
No defense



ACC-Turbo outperforms existing defenses and mitigates pulse-wave DDoS attacks



ACC-Turbo outperforms existing defenses and mitigates pulse-wave DDoS attacks



ACC-Turbo outperforms existing defenses and mitigates pulse-wave DDoS attacks

ACC-Turbo

< 1s reaction time

10x faster than state of the art

How to enable programmable scheduling on existing devices?

SP-PIFO [NSDI'20]	PACKS [NSDI'25]
Approximating PIFO's scheduling	Incorporating PIFO's admission

How to use it to improve the Internet's security?

ACC-Turbo [SIGCOMM'22]

Mitigating DDoS attacks

In-Network Congestion Management for Security and Performance

How to enable programmable scheduling on existing devices?

SP-PIFO [NSDI'20]	PACKS [NSDI'25]
Approximating PIFO's scheduling	Incorporating PIFO's admission

How to use it to improve the Internet's security?

ACC-Turbo [SIGCOMM'22]
Mitigating DDoS attacks

Selected publications

NSDI '20	SP-PIFO: Approximating Push-In First-Out Behaviors using Strict-Priority Queues A. Gran Alcoz, A. Dietmüller, L. Vanbever
SIGCOMM '22	Aggregate-Based Congestion Control for Pulse-Wave DDoS Defense A. Gran Alcoz, M. Strohmeier, V. Lenders, L. Vanbever
HotNets '23	QVISOR: Virtualizing Packet Scheduling Policies A. Gran Alcoz, L. Vanbever
SIGCOMM '24	Principles for Internet Congestion Management L. Brown, A. Gran Alcoz, F. Cangialosi, A. Narayan, M. Alizadeh, H. Balakrishnan, E. Friedman, E. Katz-Bassett, A. Krishnamurthy, M. Schapira, S. Shenker
NSDI '25	Everything Matters in Programmable Packet Scheduling A. Gran Alcoz, B. Vass, P. Namyar, B. Arzani, G. Rétvári, L. Vanbever