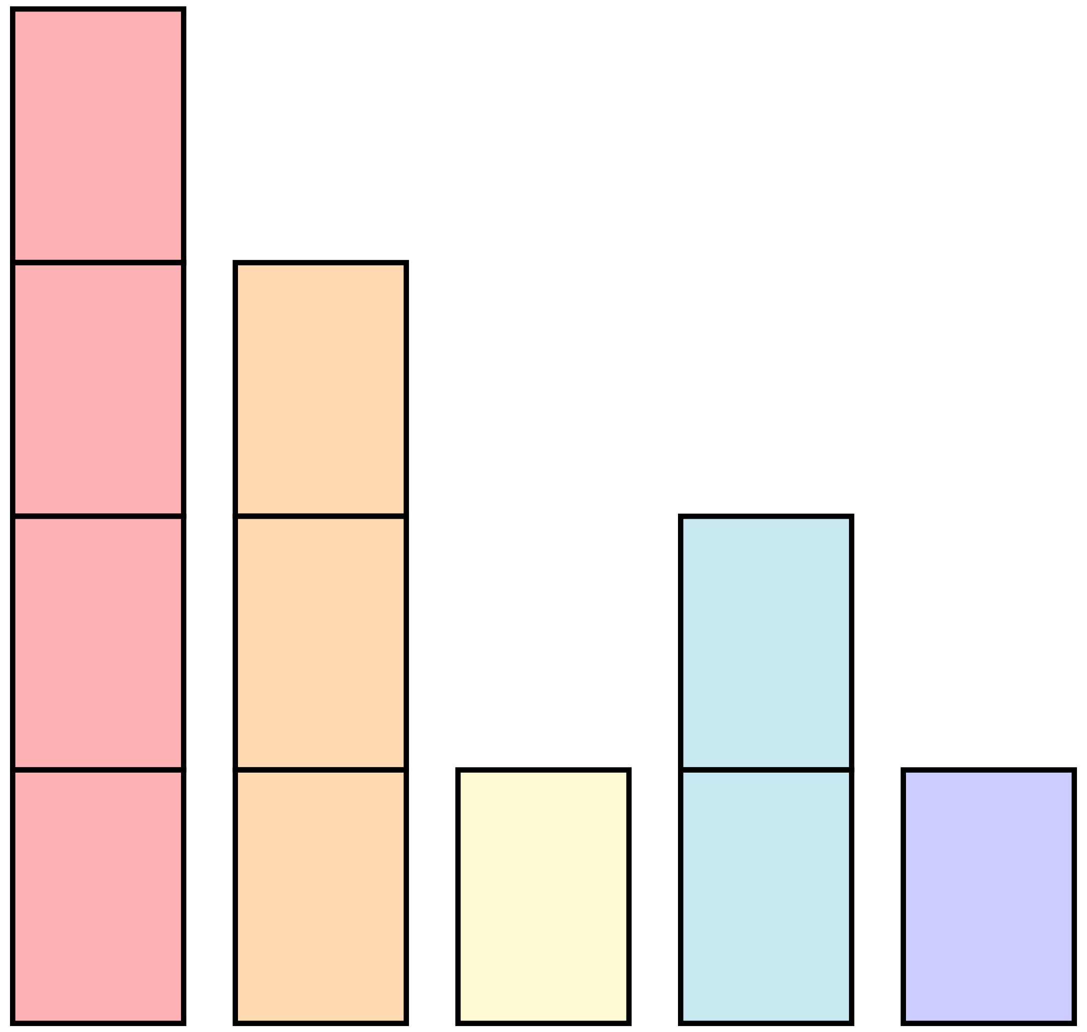
Everything Matters in Programmable Packet Scheduling

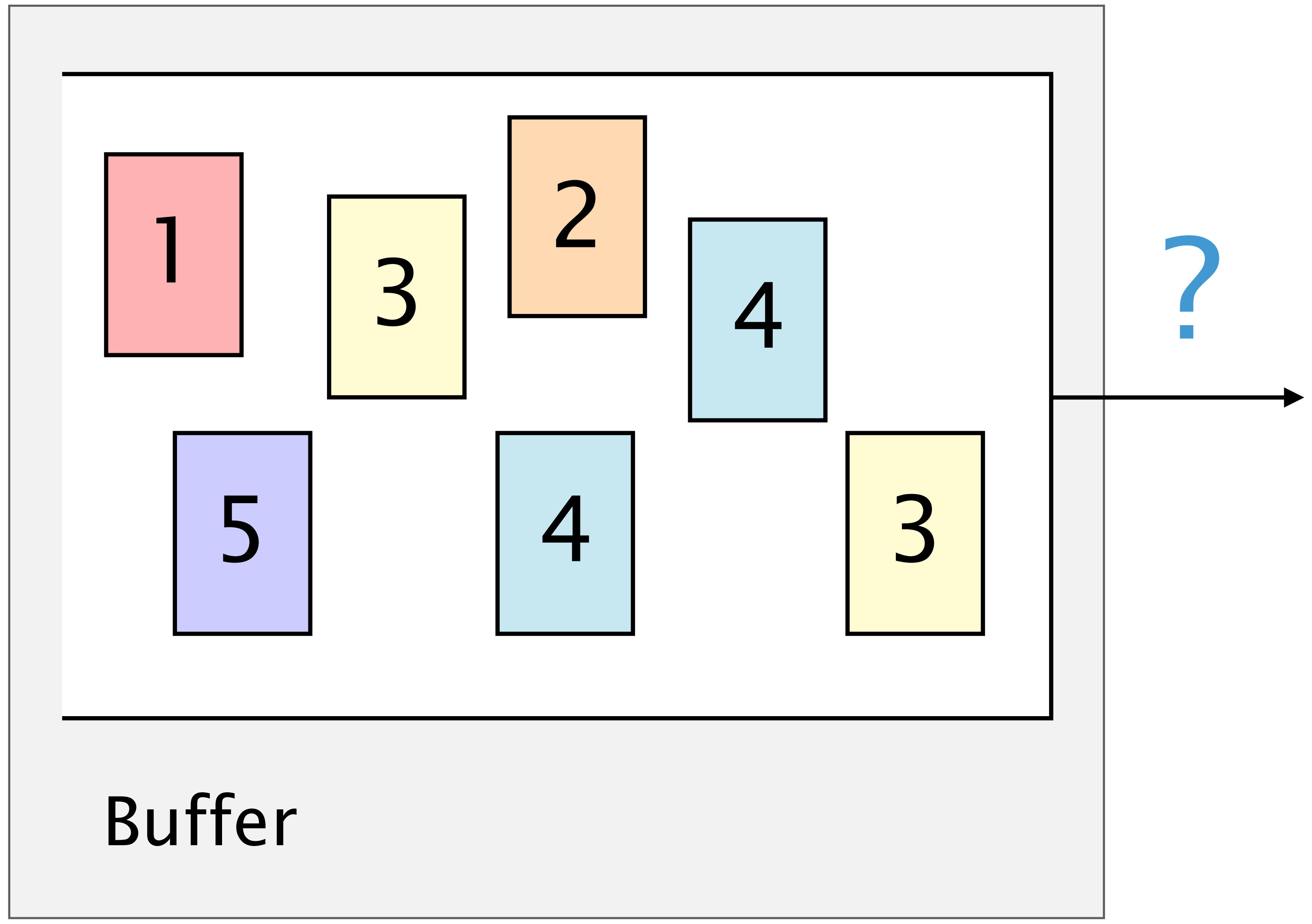


Albert Gran Alcoz
Balázs Vass
Pooria Namyar
Behnaz Arzani
Gábor Rétvári
Laurent Vanbever

NSDI ’25

ETH zürich
MŰEGYETEM 1782
USC
Microsoft

Packet scheduling defines which packet to 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

How can we deploy all scheduling algorithms?

Implement each of them on hardware ✘

ASICs lack sufficient resources

How can we deploy all scheduling algorithms?

Implement each of them on hardware ✗

ASICs lack sufficient resources

Invent a universal packet scheduler ✗

No silver bullet in packet scheduling

How can we deploy all scheduling algorithms?

Implement each of them on hardware ✘

ASICs lack sufficient resources

Invent a universal packet scheduler ✘

No silver bullet in packet scheduling

Design an abstraction to represent all schedulers

How can we deploy all scheduling algorithms?

Implement each of them on hardware ✗

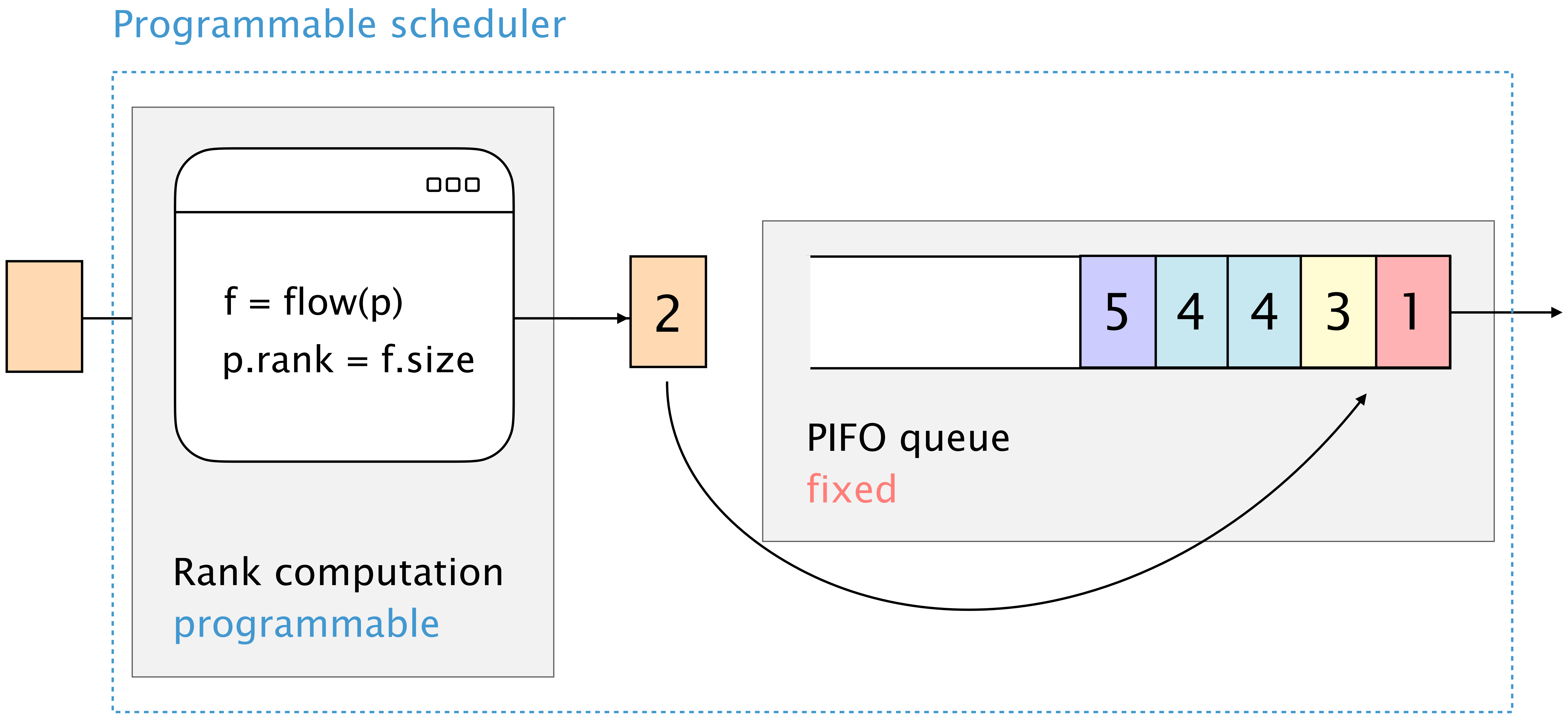
ASICs lack sufficient resources

Invent a universal packet scheduler ✗

No silver bullet in packet scheduling

Programmable scheduling ✔

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



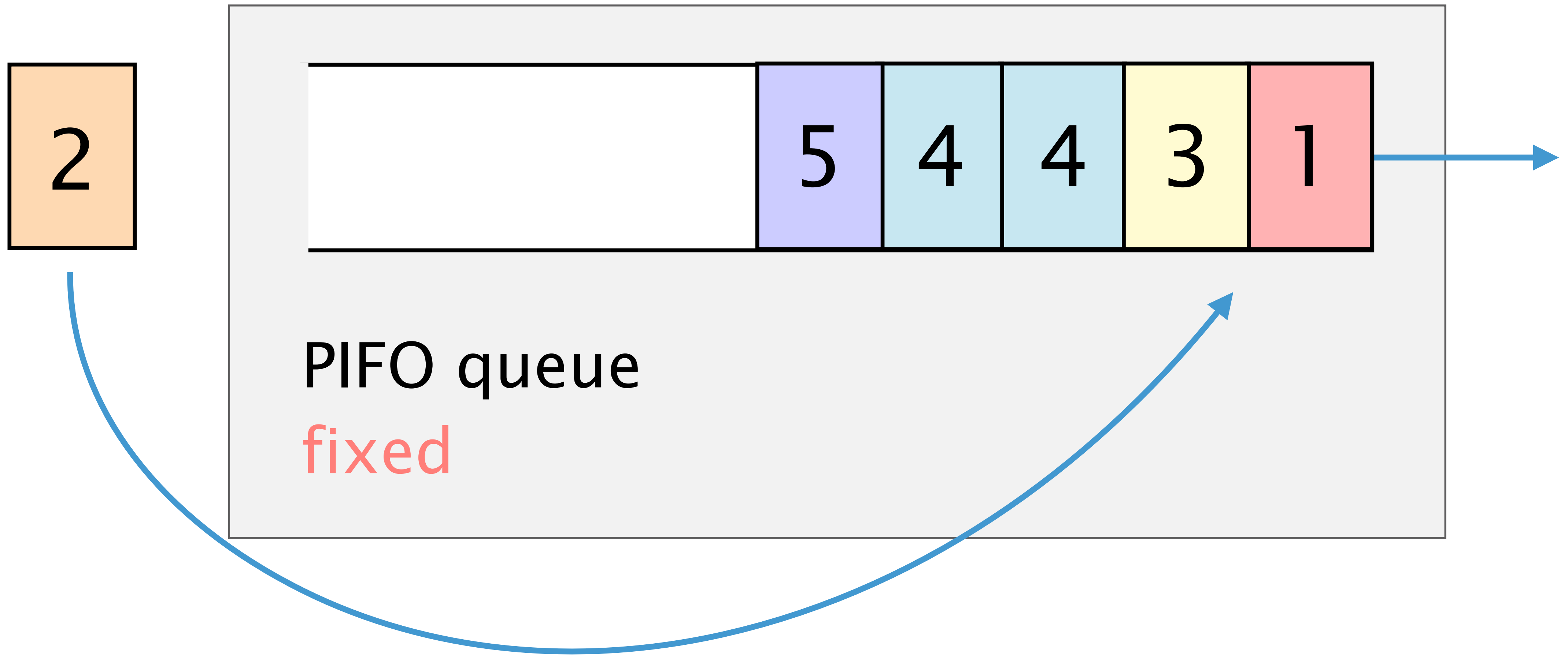
PIFO queues are characterized by
two key behaviors

Admission

Enqueue packets with lowest ranks

Scheduling

Forward packets in rank order



How to implement PIFO queues on hardware?

How to implement PIFO queues on hardware?

New ASIC

High accuracy	✔
~200M $	✘
Multiple years	✘

How to implement PIFO queues on hardware?

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

How to implement PIFO queues on hardware?

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

SP-PIFO approximates PIFO's scheduling using strict-priority queues

NSDI'20

SP-PIFO: Approximating Push-In First-Out Behaviors using Strict-Priority Queues

Albert Gran Alcoz
ETH Zürich

Alexander Dietmüller
ETH Zürich

Laurent Vanbever
ETH Zürich

Abstract

Push-In First-Out (PIFO) queues are hardware primitives which enable programmable packet scheduling by providing the abstraction of a priority queue at line rate. However, implementing them at scale is not easy: just hardware designs (not implementations) exist, which support only about 1k flows.

In this paper, we introduce SP-PIFO, a programmable packet scheduler which closely approximates the behavior of PIFO queues using strict-priority queues—*at line rate, at scale, and on existing devices*. The key insight behind SP-PIFO is to dynamically adapt the mapping between packet ranks and available strict-priority queues to minimize the scheduling errors with respect to an ideal PIFO. We present a mathematical formulation of the problem and derive an adaptation technique which closely approximates the optimal queue mapping without any traffic knowledge.

We fully implement SP-PIFO in P4 and evaluate it on real

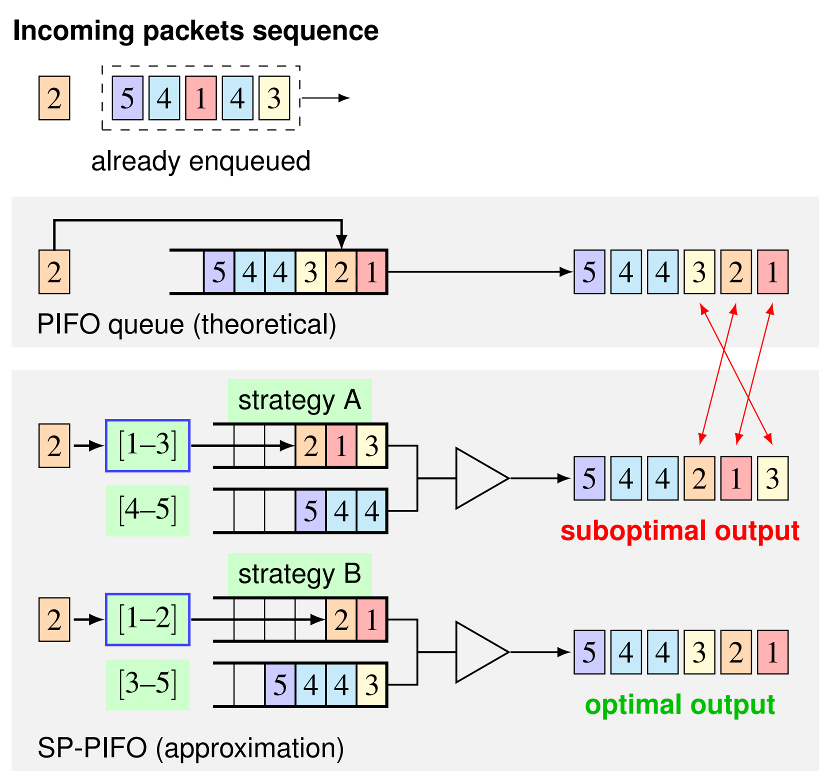
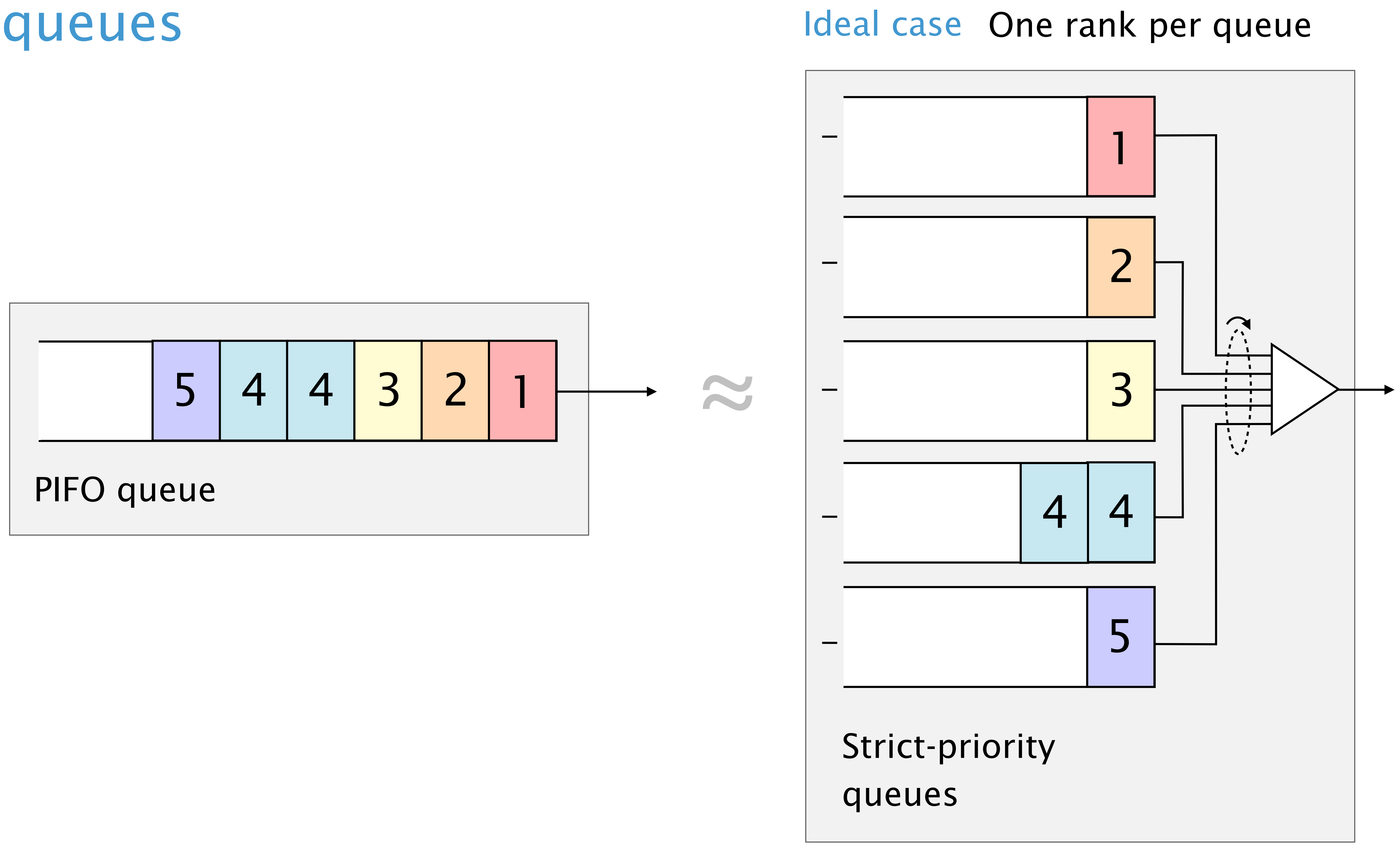


Figure 1: SP-PIFO approximates the behavior of PIFO queues

SP-PIFO approximates PIFO's scheduling using

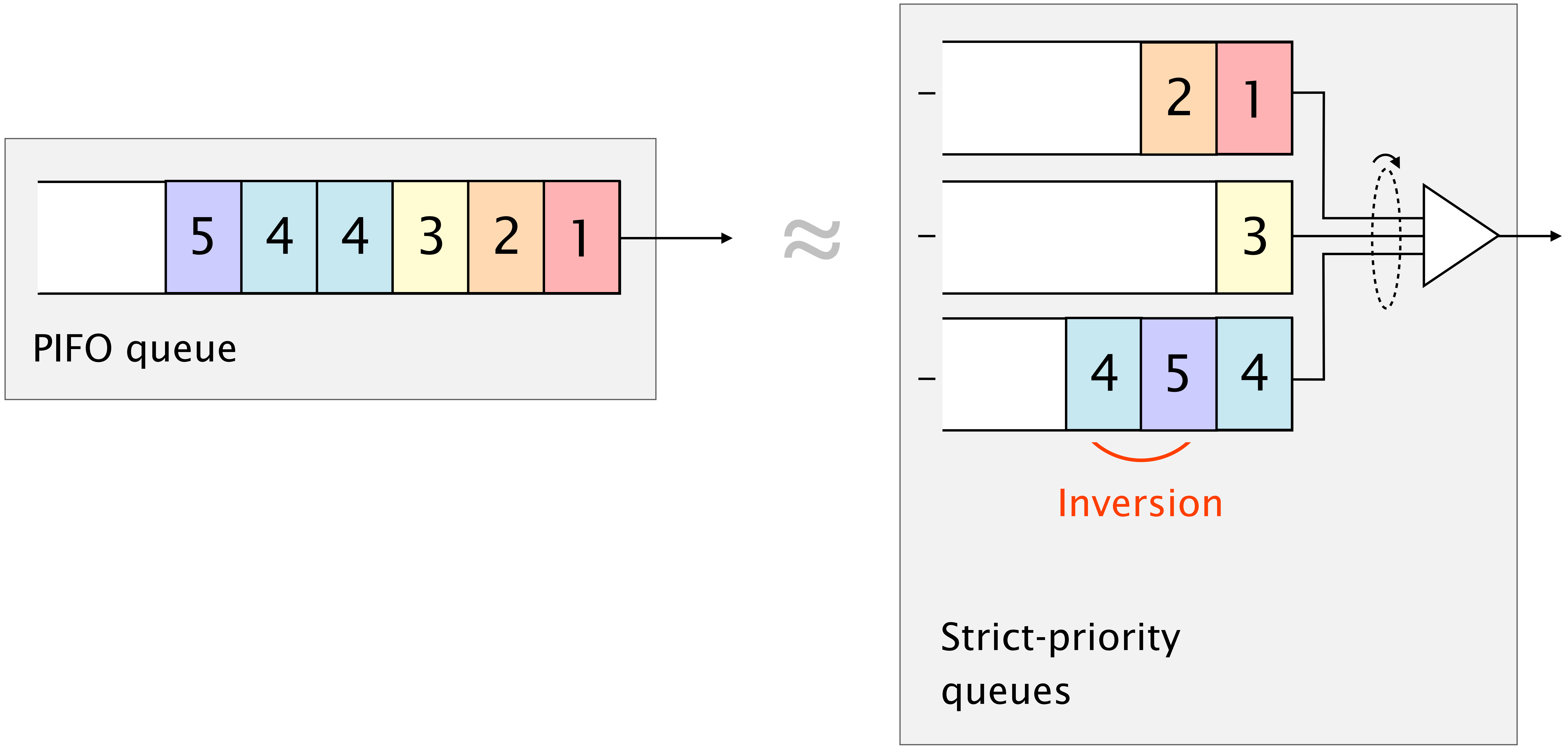
strict-priority queues



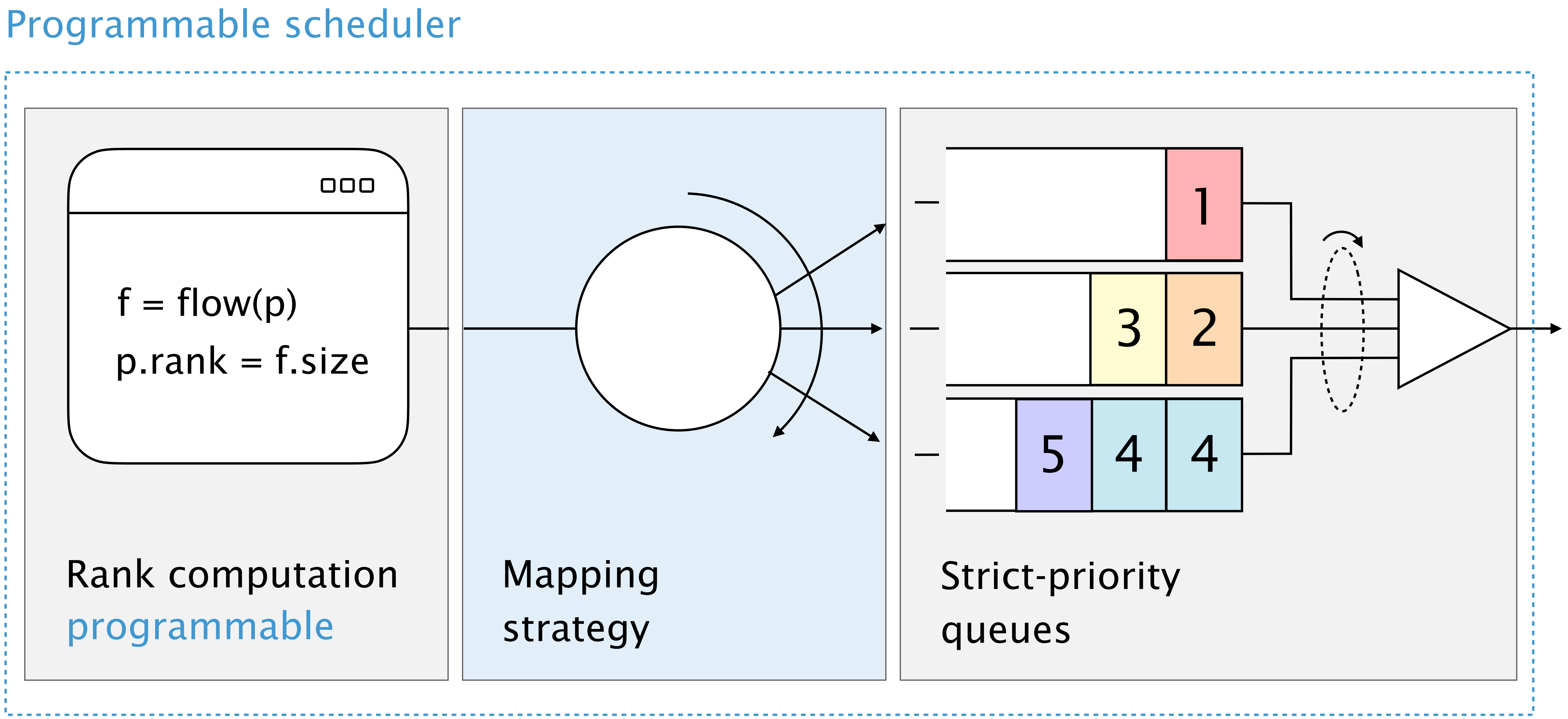
SP-PIFO approximates PIFO's scheduling using strict-priority queues

In practice

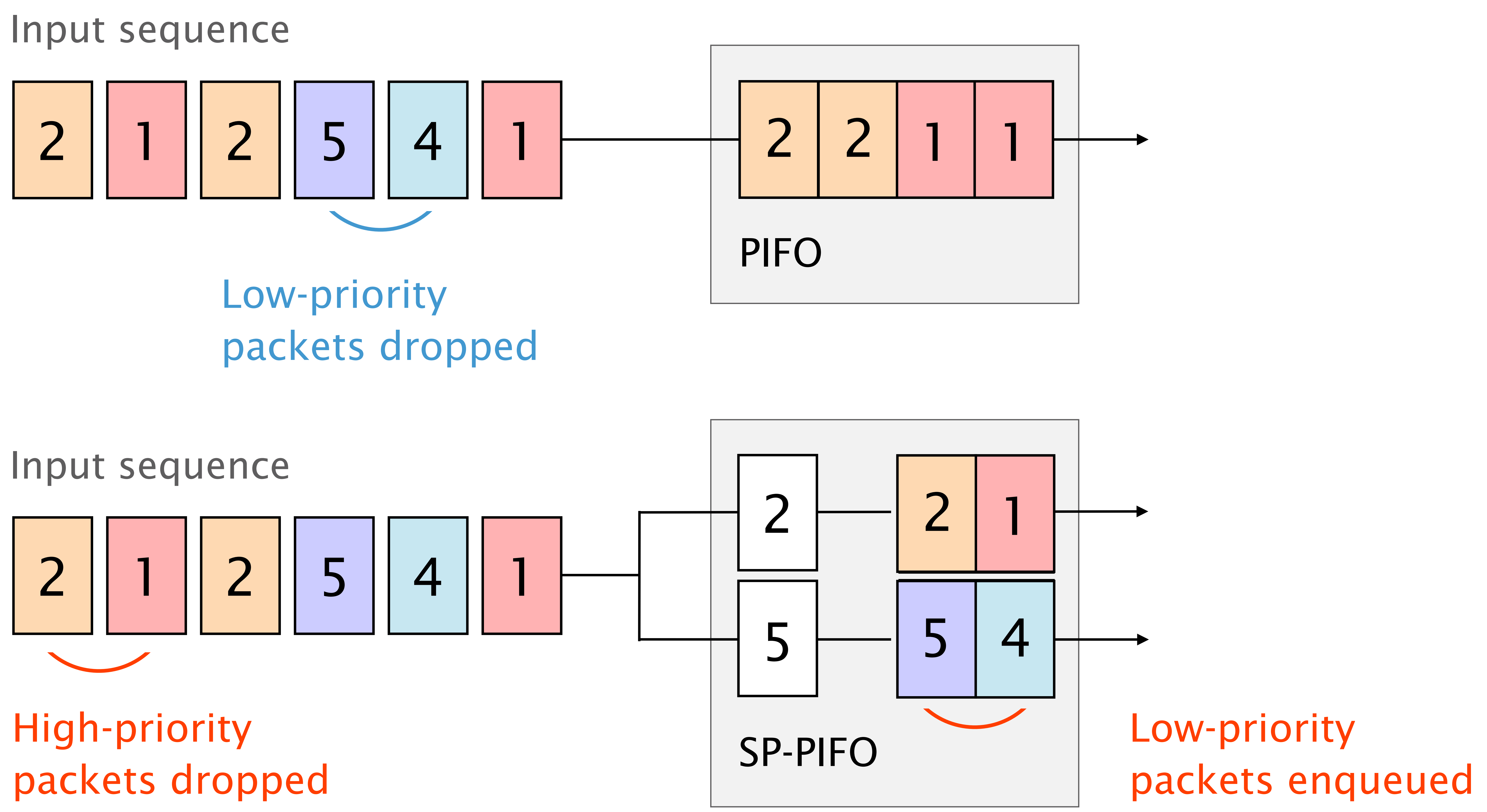
Multiple ranks per queue



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



SP-PIFO approximates PIFO's scheduling, but not admission



AIFO approximates PIFO's admission on a single FIFO queue

SIGCOMM'21

Programmable Packet Scheduling with a Single Queue

Zhuolong Yu
Johns Hopkins University

Chuheng Hu
Johns Hopkins University

Jingfeng Wu
Johns Hopkins University

Xiao Sun
Stony Brook University

Vladimir Braverman
Johns Hopkins University

Mosharaf Chowdhury
University of Michigan

Zhenhua Liu
Stony Brook University

Xin Jin
Peking University

ABSTRACT

Programmable packet scheduling enables scheduling algorithms to be programmed into the data plane without changing the hardware. Existing proposals either have no hardware implementations for switch ASICs or require multiple strict-priority queues.

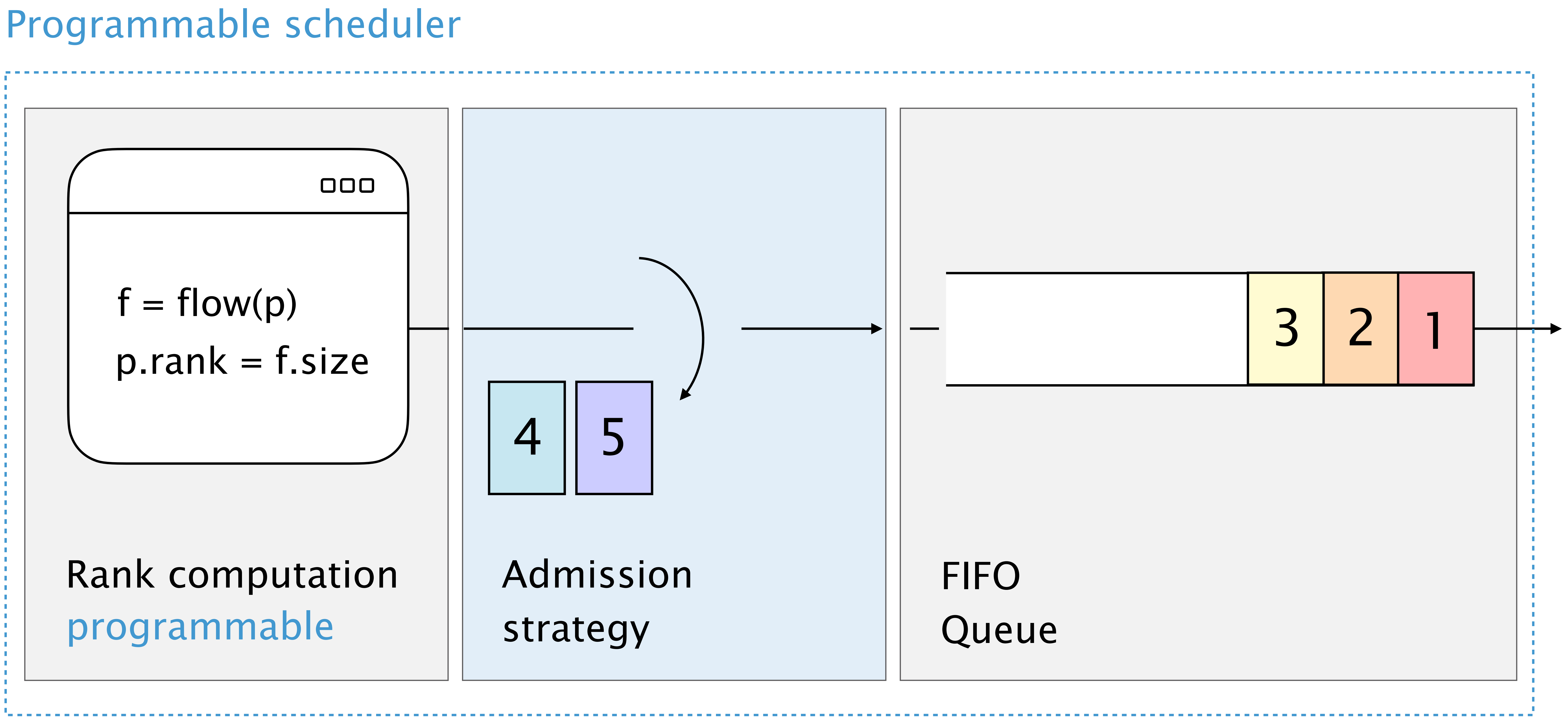
We present Admission-In First-Out (AIFO) queues, a new solution for programmable packet scheduling that uses only a *single* first-in first-out queue. AIFO is motivated by the confluence of two recent trends: *shallow* buffers in switches and *fast-converging* congestion control in end hosts, that together leads to a simple observation: the decisive factor in a flow's completion time (FCT) in modern datacenter networks is often *which* packets are enqueued or dropped, not the *ordering* they leave the switch. The core idea of AIFO is to maintain a sliding window to track the ranks of recent packets and compute the relative rank of an arriving packet in the window for admission control. Theoretically, we prove that AIFO provides bounded performance to Push-In First-Out (PIFO). Empirically, we fully implement AIFO and evaluate AIFO with a range of real workloads, demonstrating AIFO closely approximates PIFO. Importantly, unlike PIFO, AIFO can run at line rate on existing

1 INTRODUCTION

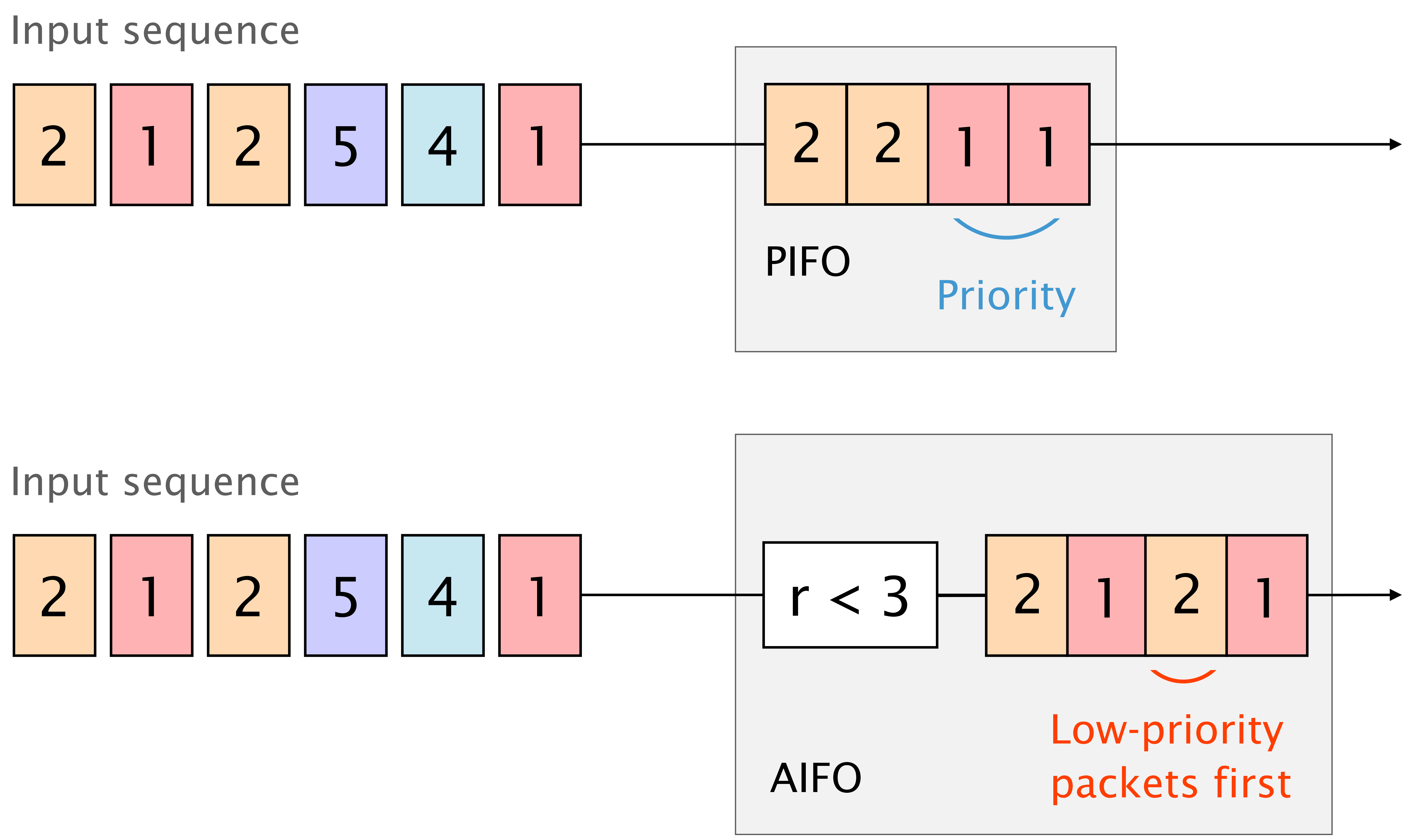
Packet scheduling is a central research topic in computer networking. Over the past several decades, a great many packet scheduling algorithms have been designed to provide different properties and optimize diverse objectives [6, 11, 23, 40, 41]. Unfortunately, most of these algorithms, despite many novel ideas among them, never have found their way to impact the real world. This is largely due to the high cost to design and deploy switch ASICs to implement them, since packet scheduling algorithms must run in the data plane at line rate in order to process every single packet.

Programmable packet scheduling is a holy grail for packet scheduling as it enables scheduling algorithms to be programmed into a switch without changing the hardware design. With programmable packet scheduling, one is able to develop or simply download a packet scheduling algorithm that best matches the operational goals of the network. This enables network operators to highly customize packet scheduling algorithms based on their needs. Particularly, it simplifies the testing and deployment of new scheduling algorithms, and it enables algorithms that are targeted at small niche markets and thus cannot justify the high cost of developing new

AIFO approximates PIFO's admission on a single FIFO queue



AIFO approximates PIFO's admission, but not scheduling



Existing works only approximate one PIFO behavior

Admission

Enqueue packets with lowest ranks

AIFO (SIGCOMM ’21)

HCSFQ (NSDI ’21)

AQ (SIGCOMM ’23)

Scheduling

Forward packets in rank order

SP-PIFO (NSDI ’20)

PCQ (NSDI ’20)

GearBox (NSDI ’22)

QCluster (WWW ’22)

Spring (INFOCOM ’22)

Existing works only approximate one PIFO behavior

Admission

Enqueue packets with lowest ranks

Scheduling

Forward packets in rank order

“Everything matters”

Can we approximate both PIFO behaviors

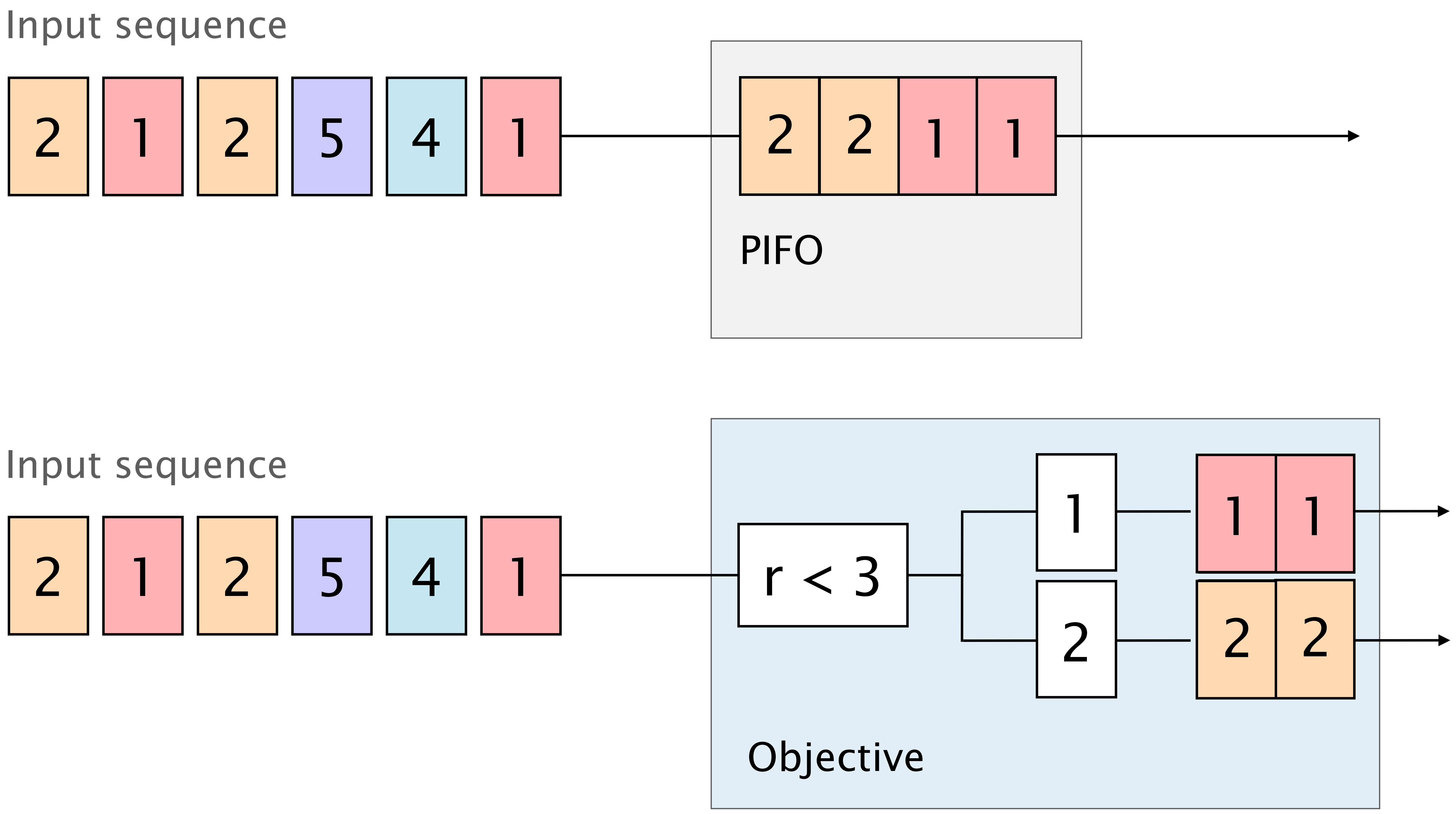
on existing programmable switches?

Introducing…

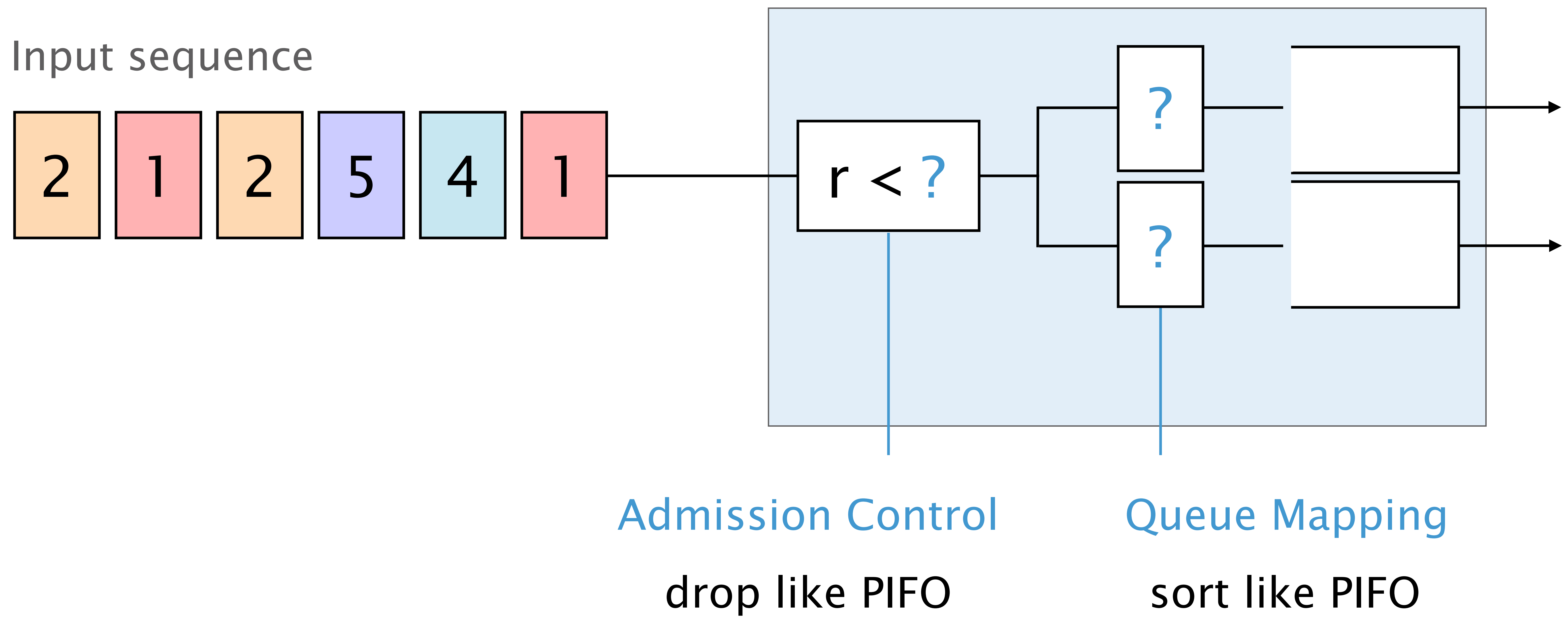
PACKS

A programmable scheduler
approximating both PIFO behaviors

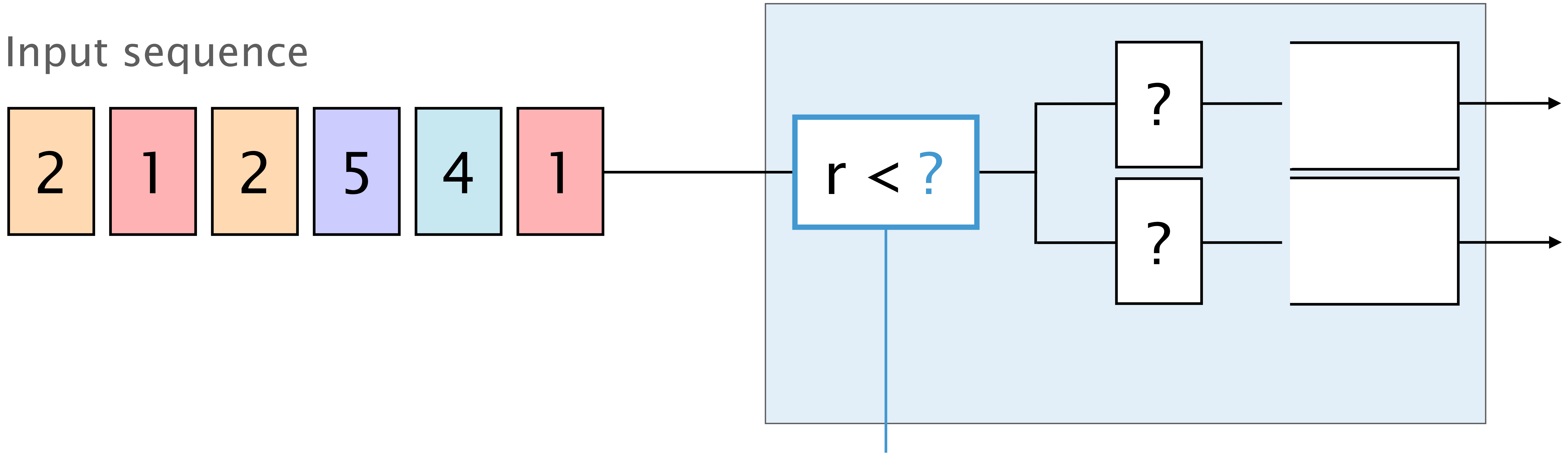
PACKS combines an admission- and a queue mapping-strategy



PACKS combines an admission- and a queue mapping-strategy



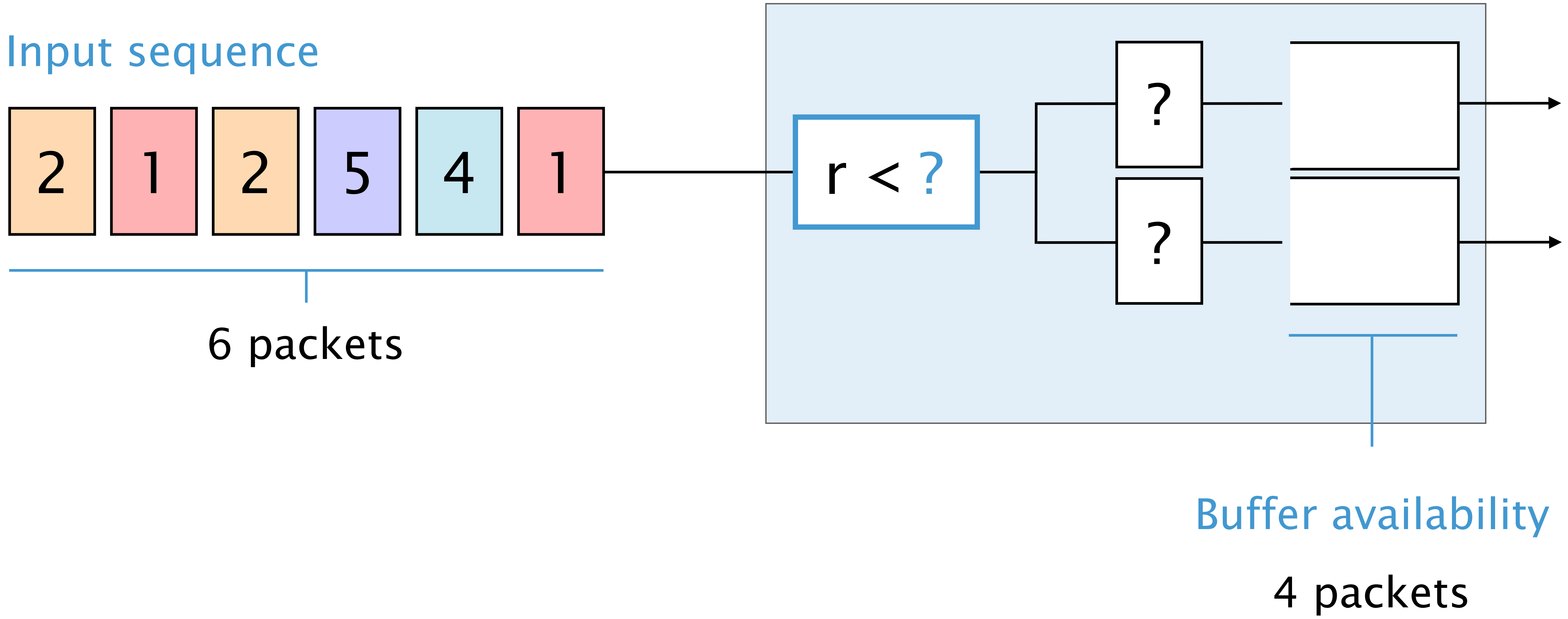
PACKS combines an admission- and a queue mapping-strategy



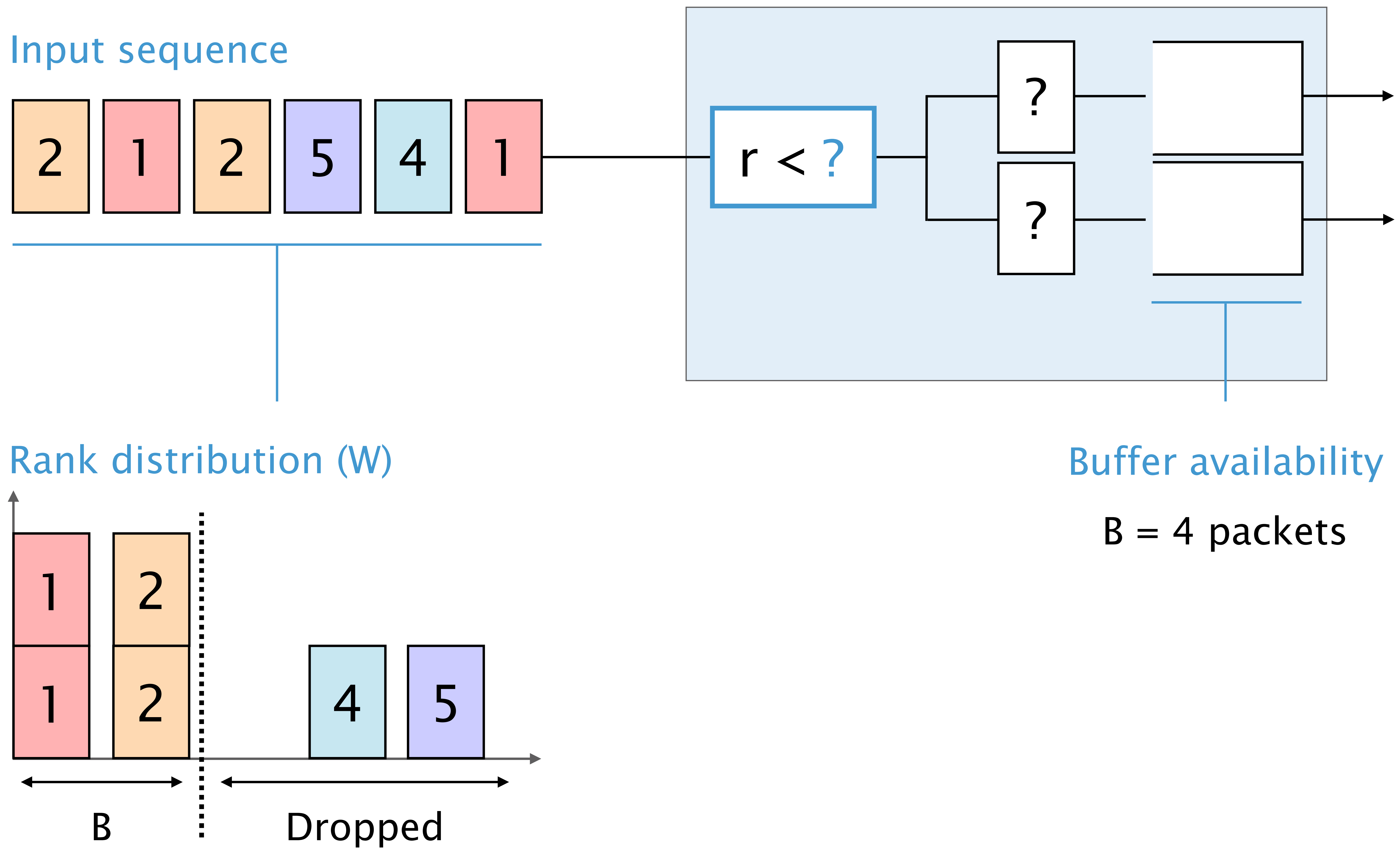
Admission Control

drop like PIFO

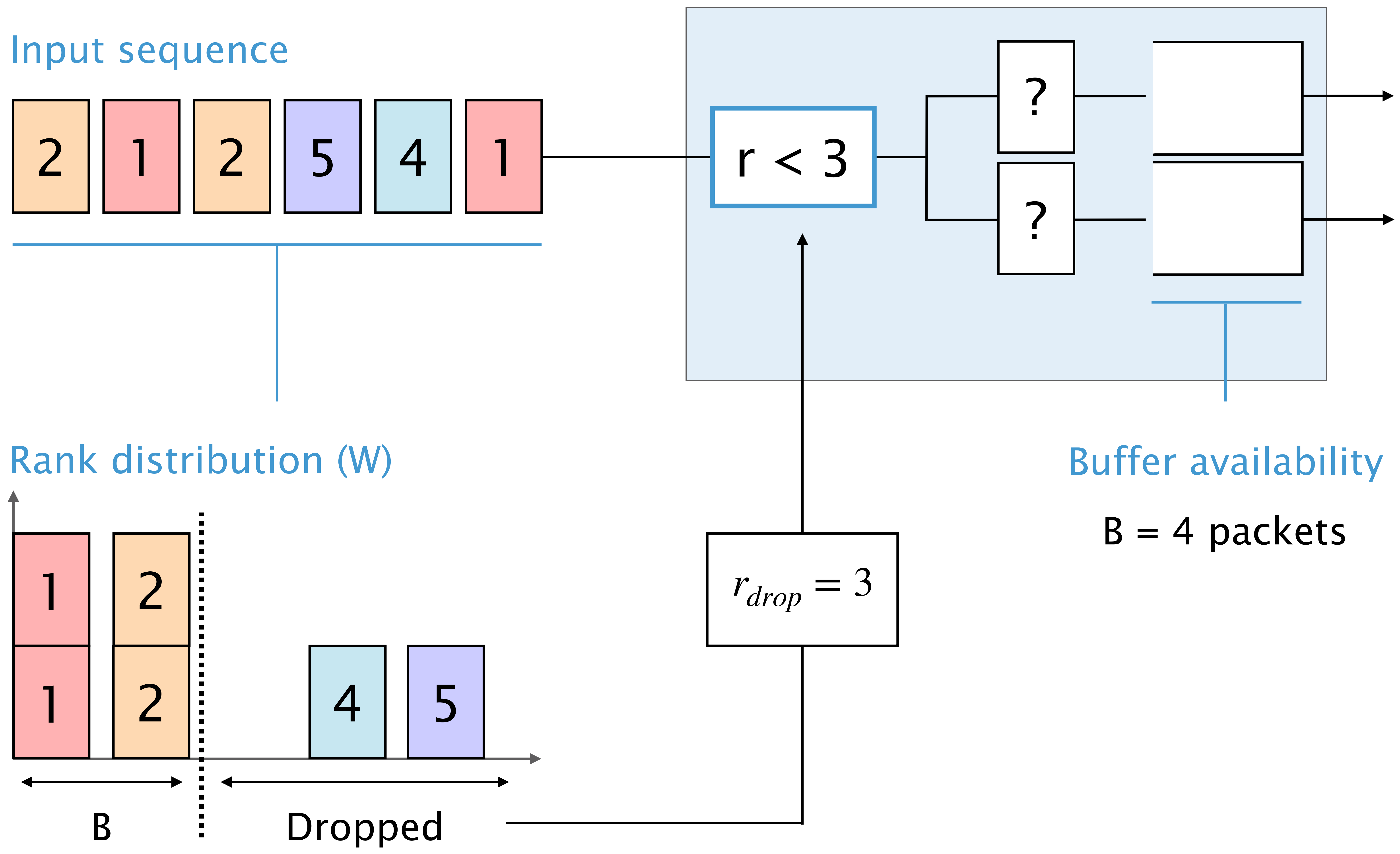
PACKS combines an admission- and a queue mapping-strategy



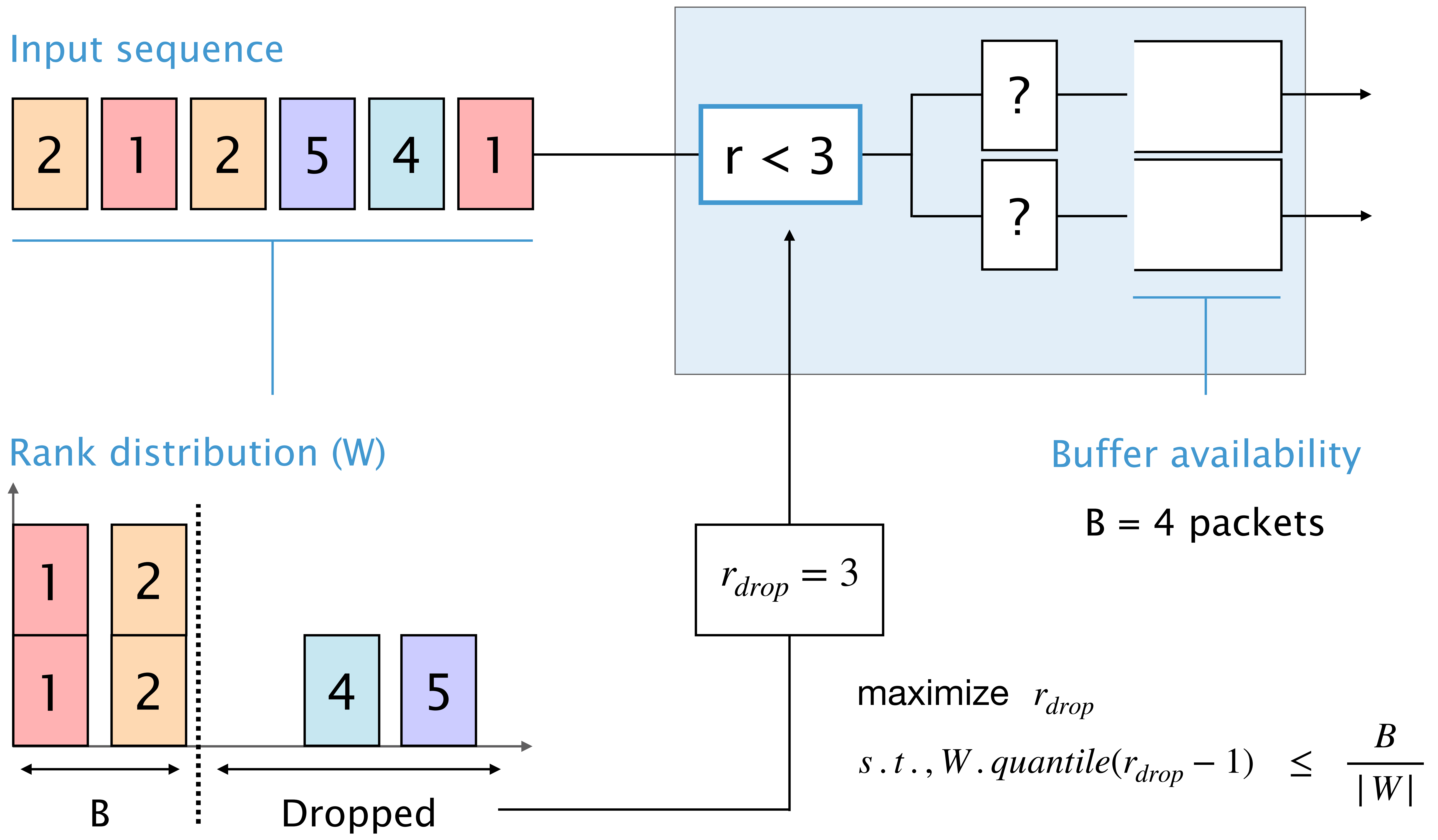
PACKS combines an admission- and a queue mapping-strategy



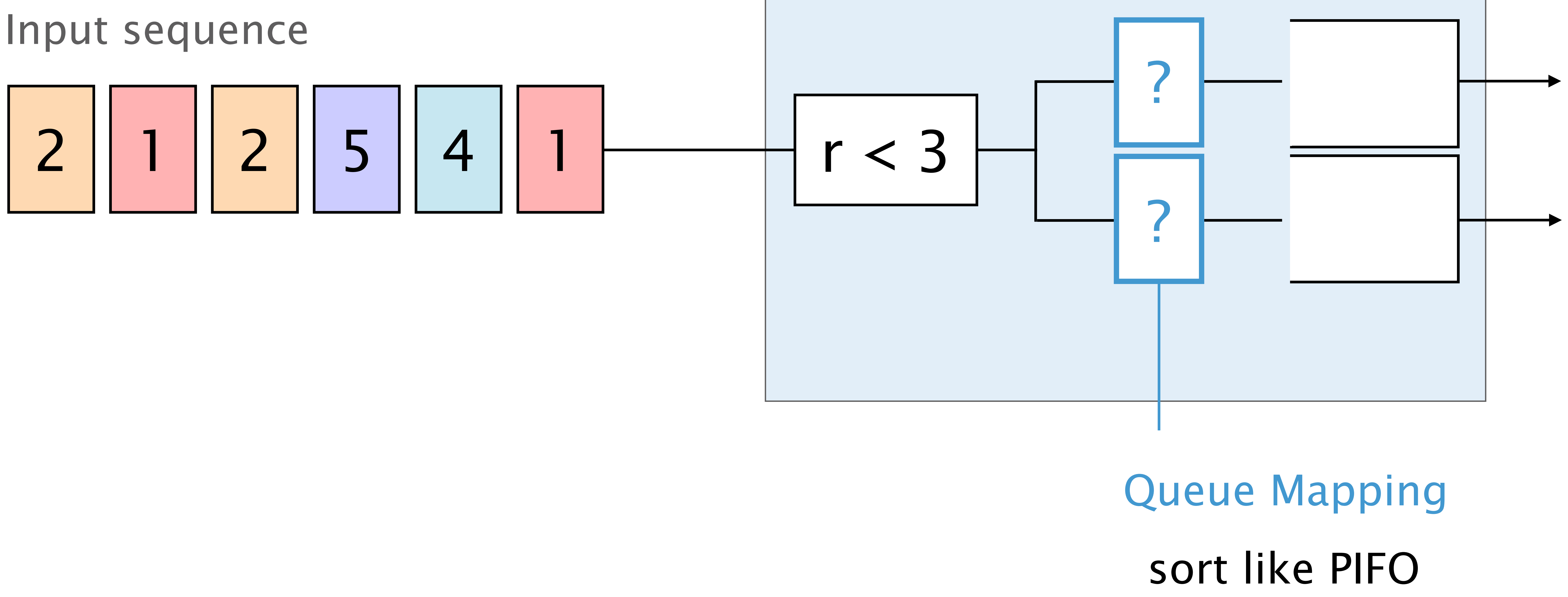
PACKS combines an admission- and a queue mapping-strategy



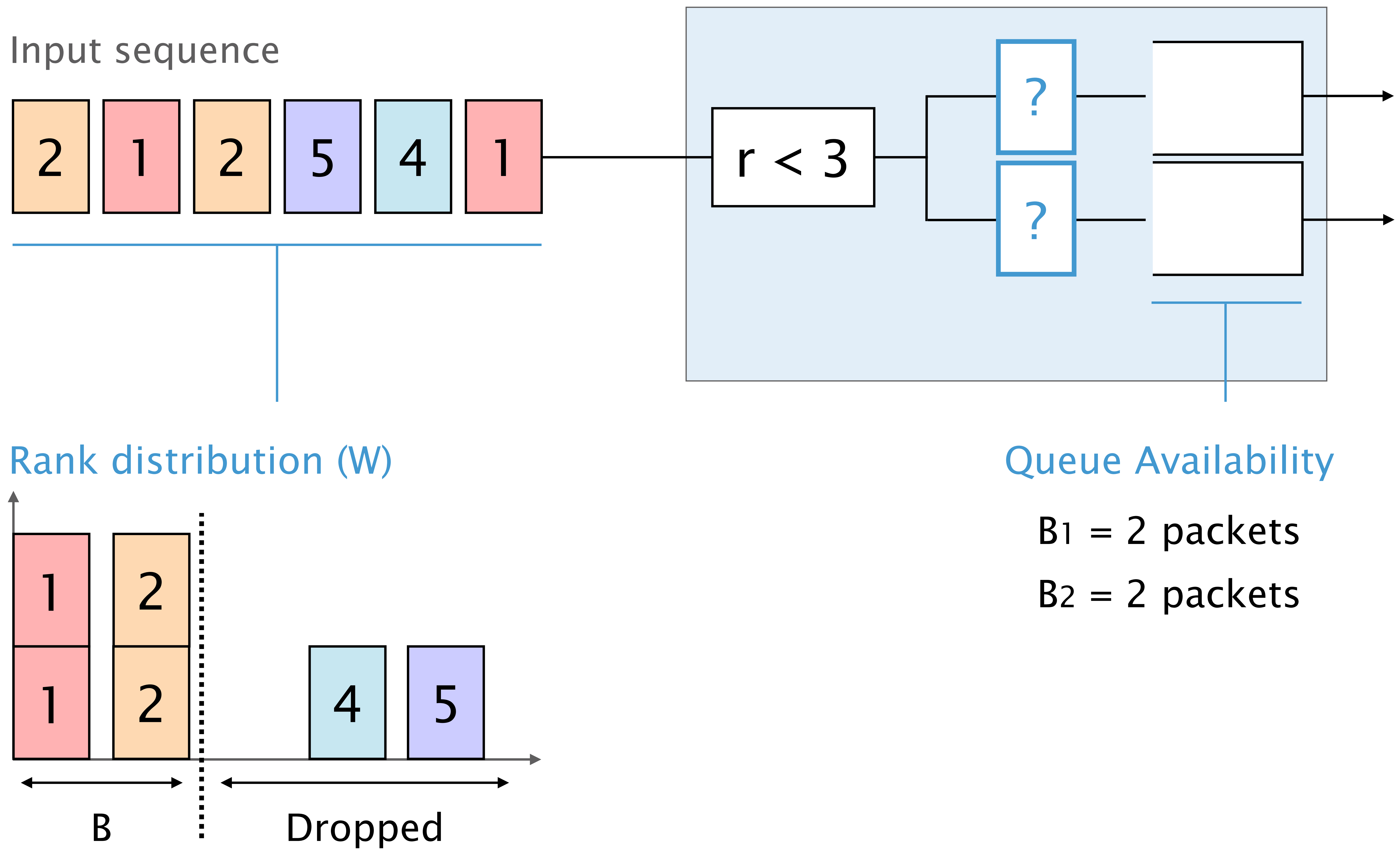
PACKS combines an admission- and a queue mapping-strategy



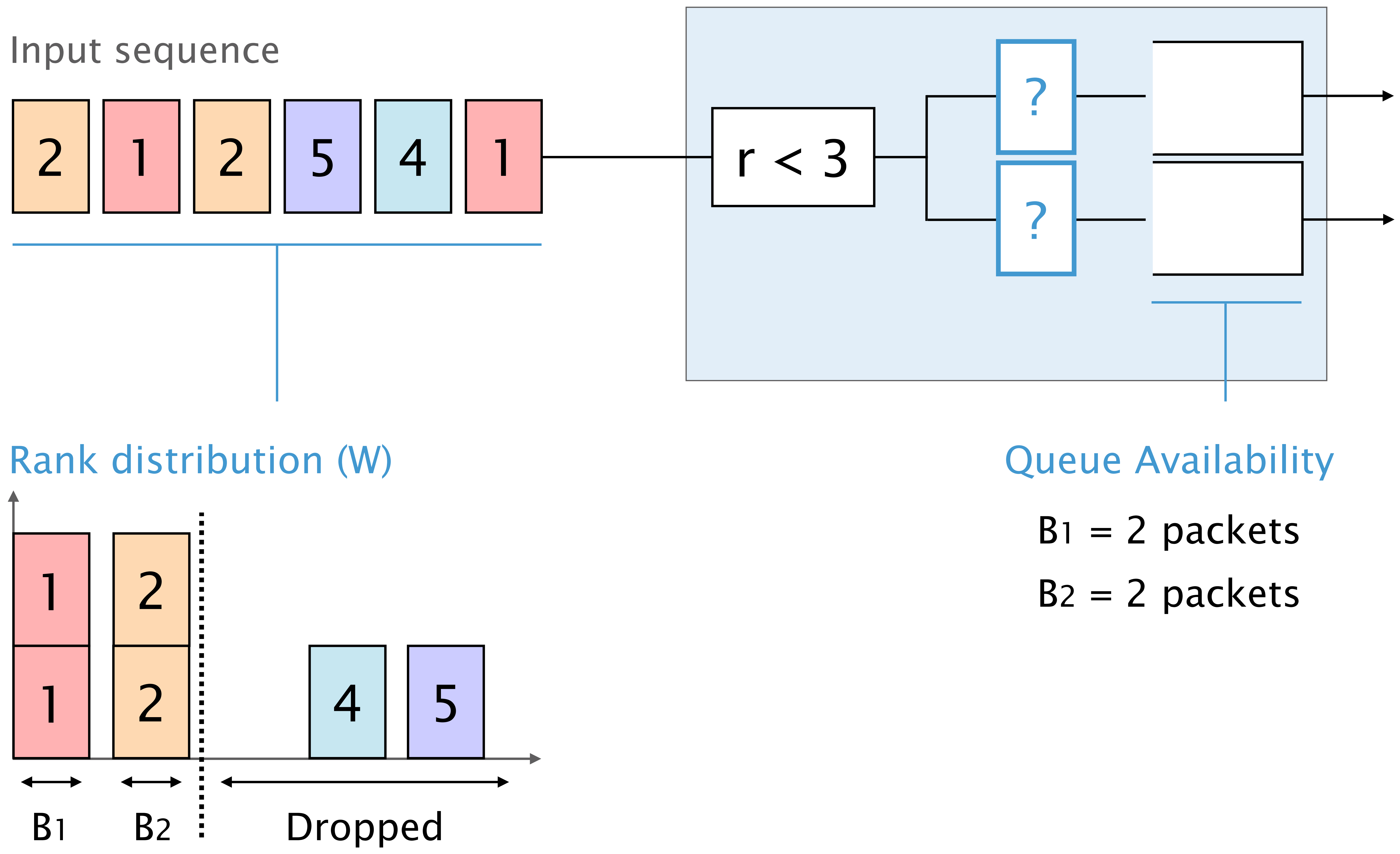
PACKS combines an admission- and a queue mapping-strategy



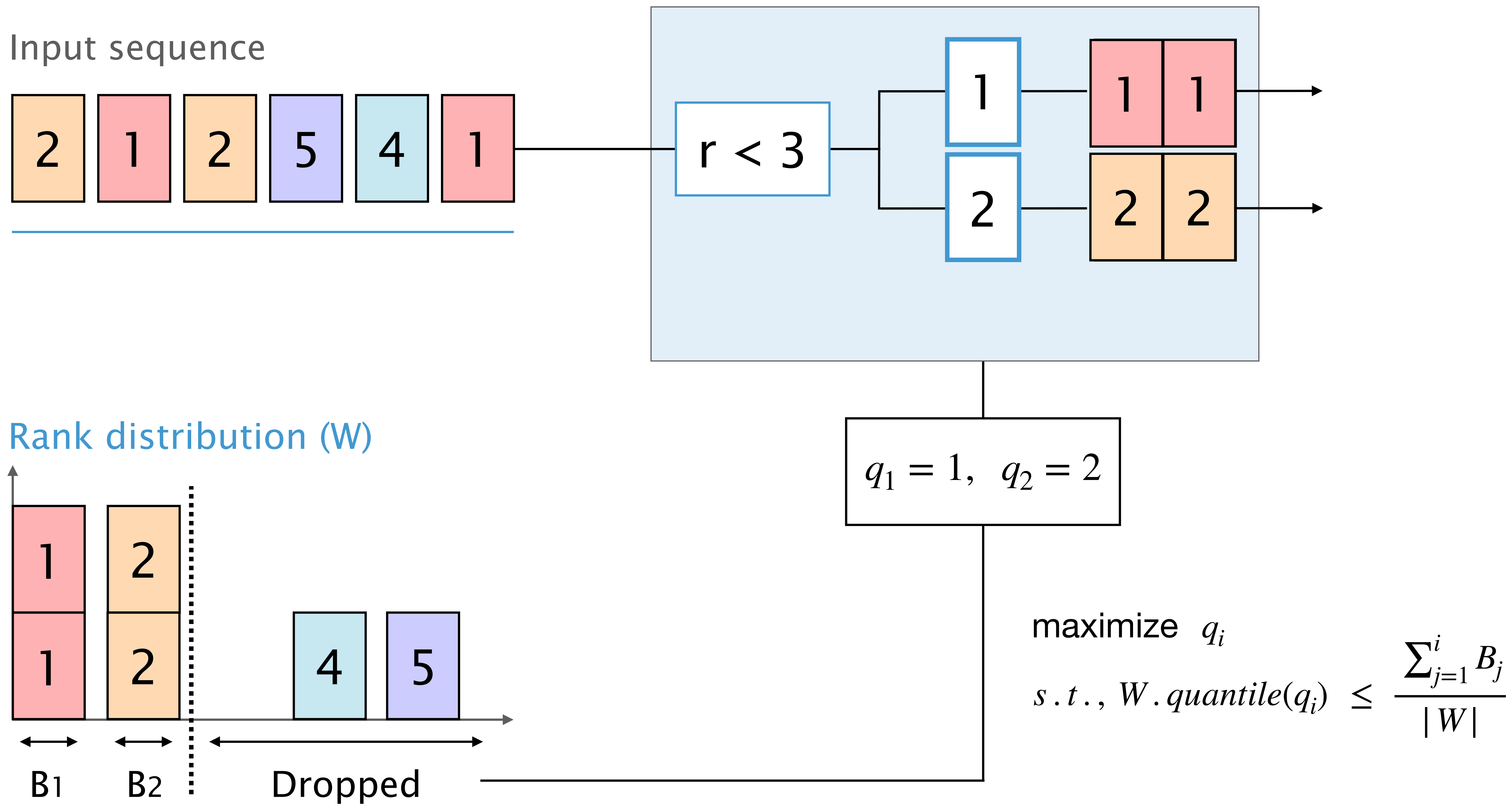
PACKS combines an admission- and a queue mapping-strategy



PACKS combines an admission- and a queue mapping-strategy



PACKS combines an admission- and a queue mapping-strategy



How to translate the algorithm to the online case?

How to monitor the rank distribution?

How to adapt to buffer dynamism?

How to account for workload shifts?

How to translate the algorithm to the online case?

How to monitor the rank distribution?

Use a sliding window of latest ranks

How to adapt to buffer dynamism?

How to account for workload shifts?

How to translate the algorithm to the online case?

How to monitor the rank distribution?

Use a sliding window of latest ranks

How to adapt to buffer dynamism?

Measure per-packet queue occupancy

$$W.quantile(r) \leq \frac{\sum_{j=1}^{i} (B_j - bj)}{B}$$

Per-packet queue occupancy

How to account for workload shifts?

How to translate the algorithm to the online case?

How to monitor the rank distribution?

Use a sliding window of latest ranks

How to adapt to buffer dynamism?

Measure per-packet queue occupancy

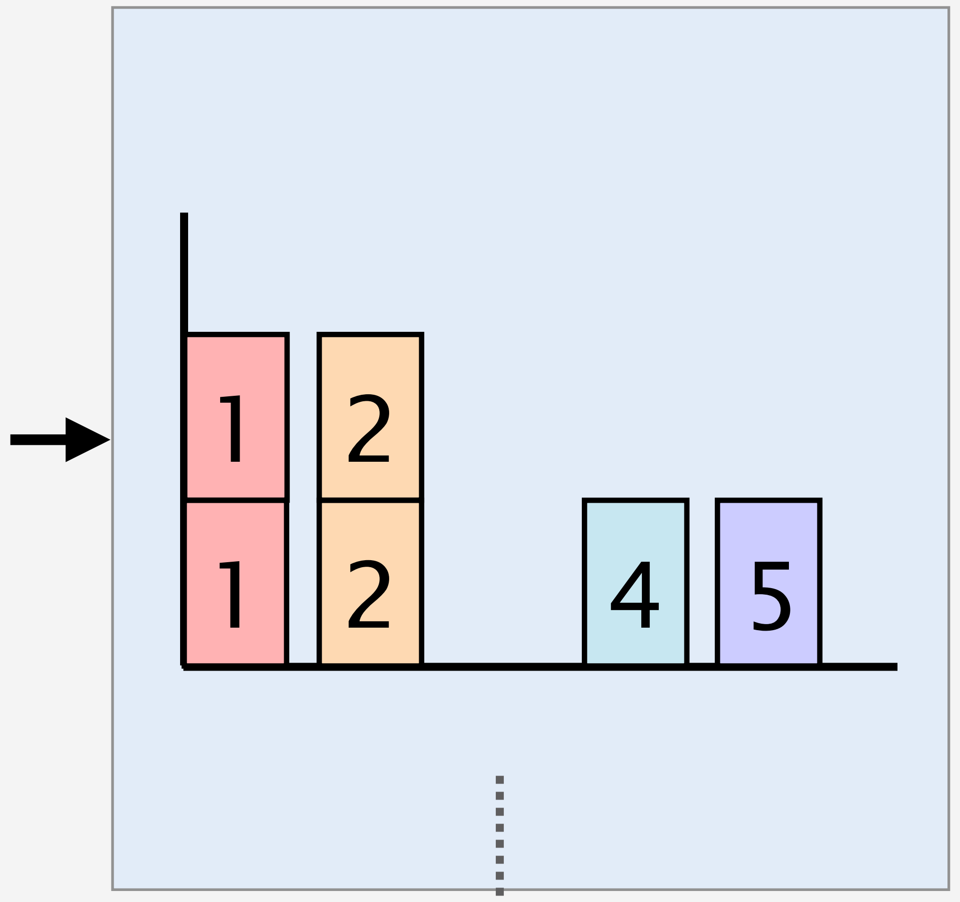
How to account for workload shifts?

Allow a certain amount of bursts

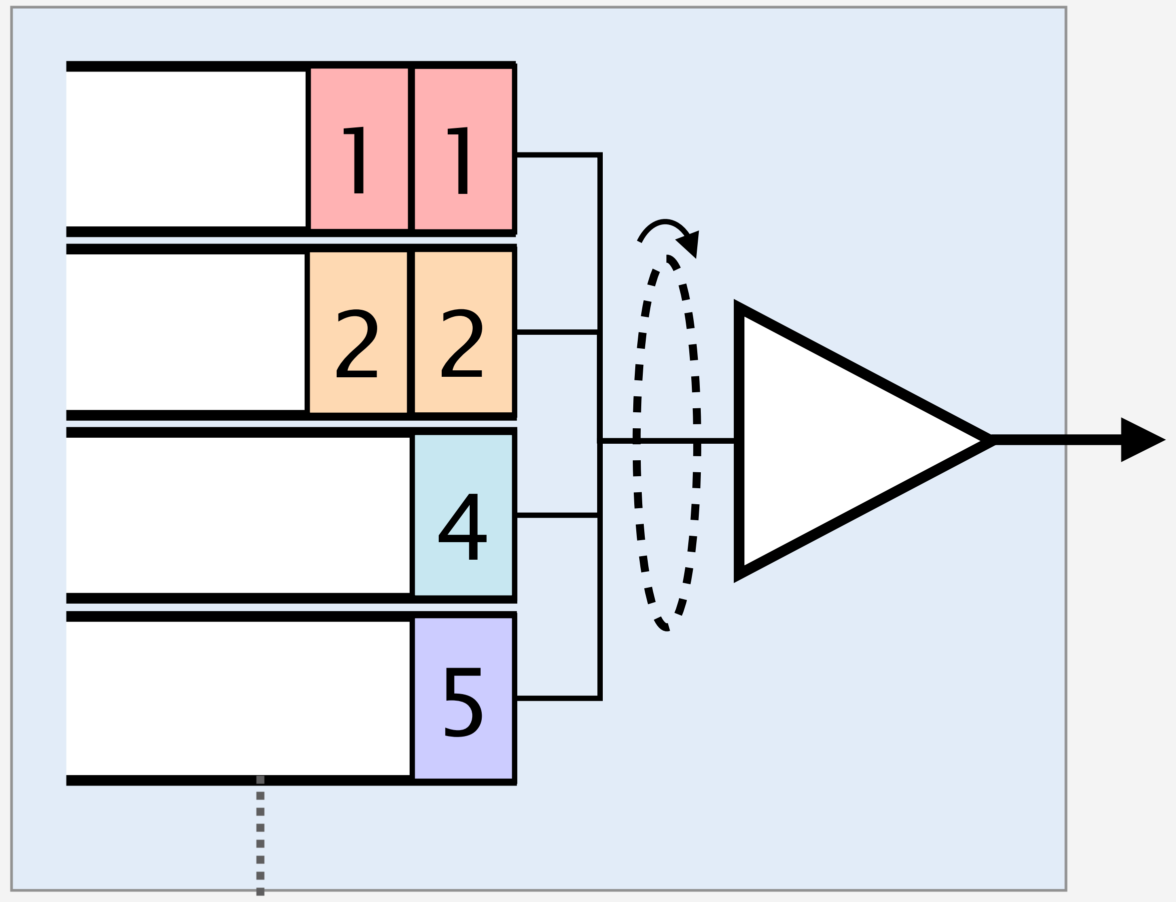
$$W.quantile(r) \leq \underbrace{\alpha}_{\text{Burst allowance}} \cdot \frac{\sum_{j=1}^{i} (B_j - bj)}{B}$$

PACKS

Sliding window tracking



Buffer occupancy monitoring

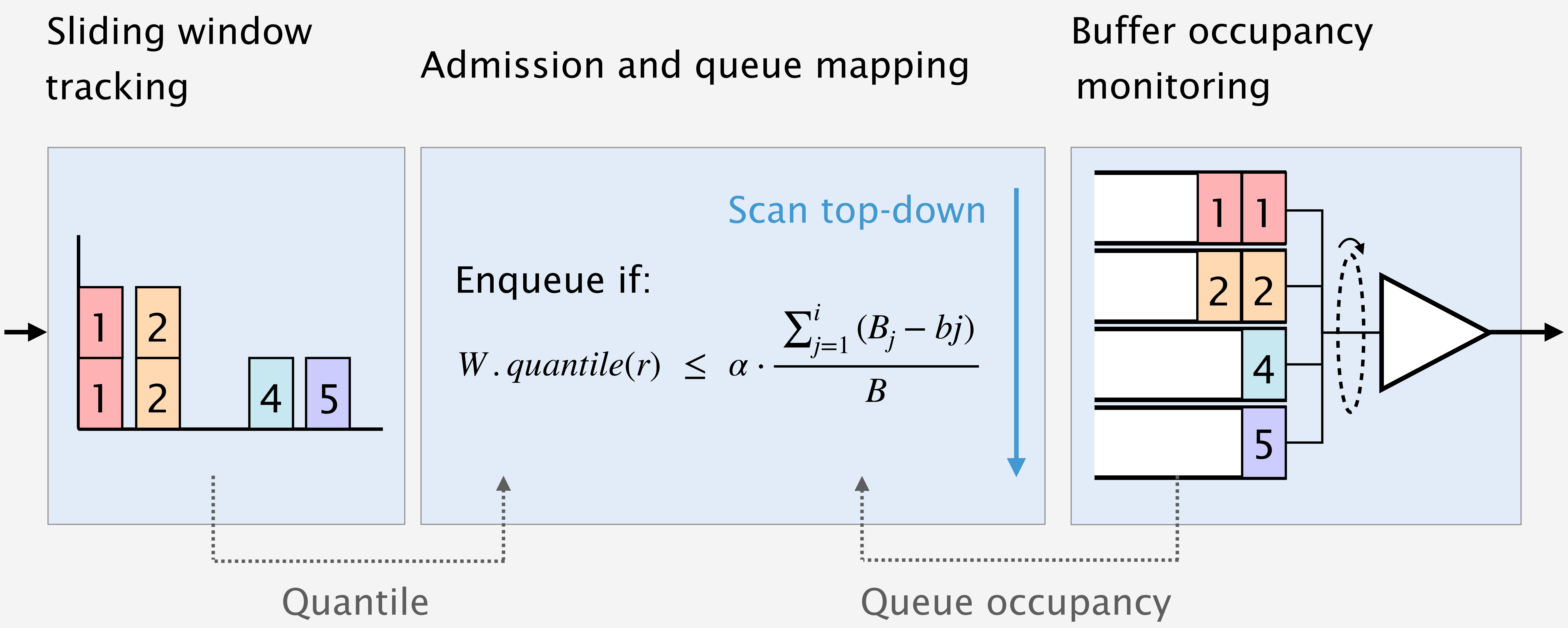


$$W.quantile(r) \leq \alpha \cdot \frac{\sum_{j=1}^{i} (B_j - bj)}{B}$$

Quantile

Queue occupancy

PACKS



Sliding window tracking
Admission and queue mapping
Buffer occupancy monitoring
1
1
2
2
4
5
Scan top-down
Enqueue if:
$W.quantile(r) \leq \alpha \cdot \frac{\sum_{j=1}^{i} (B_j - bj)}{B}$
1 1
2 2
4
5
Quantile
Queue occupancy

We evaluated PACKS on hardware and simulations

Packet-level simulation (NetBench)

Performance in approximating PIFO

Sensitivity to configuration parameters

Practicality under pFabric and FQ scenarios

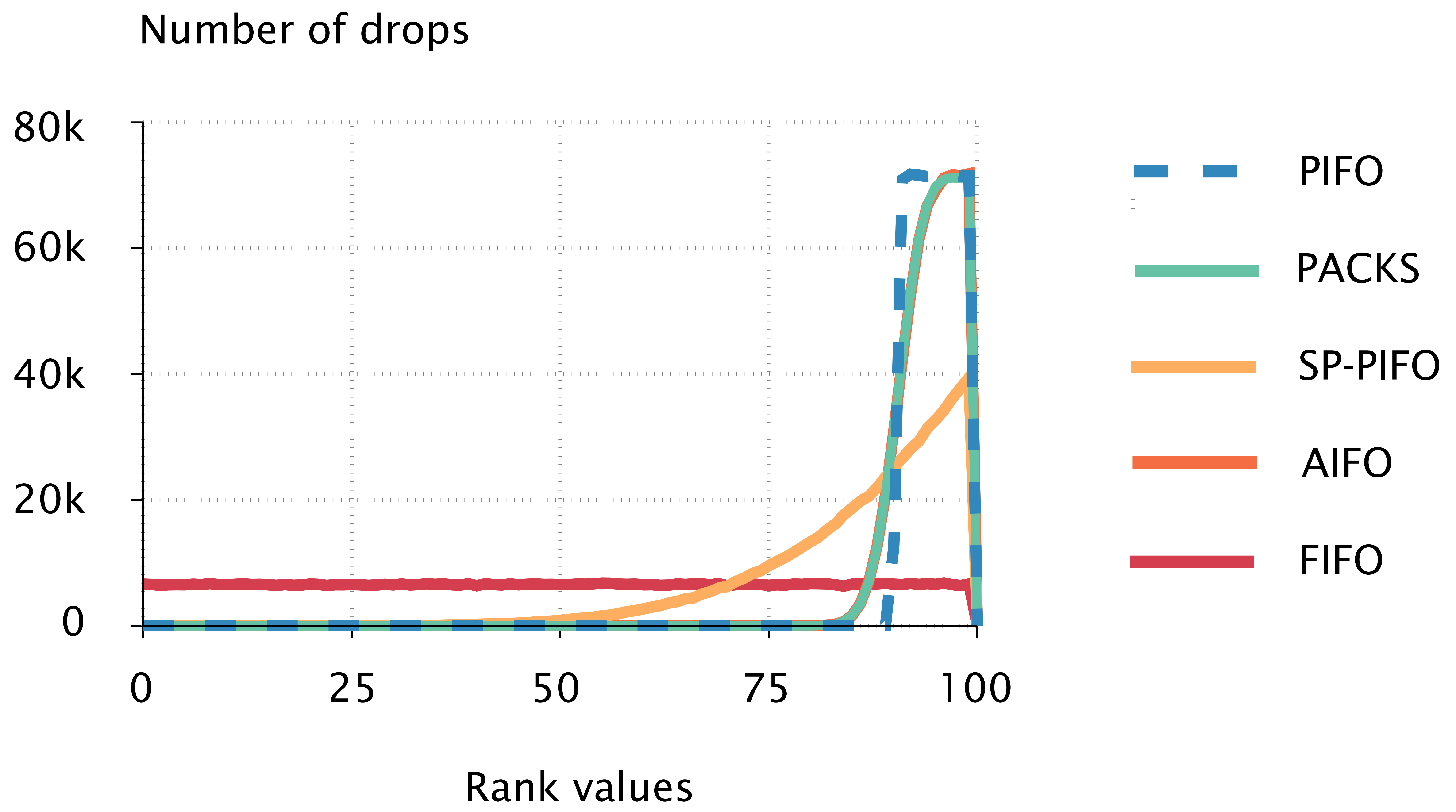
Hardware evaluation (Intel Tofino2)

Bandwidth allocation across priorities

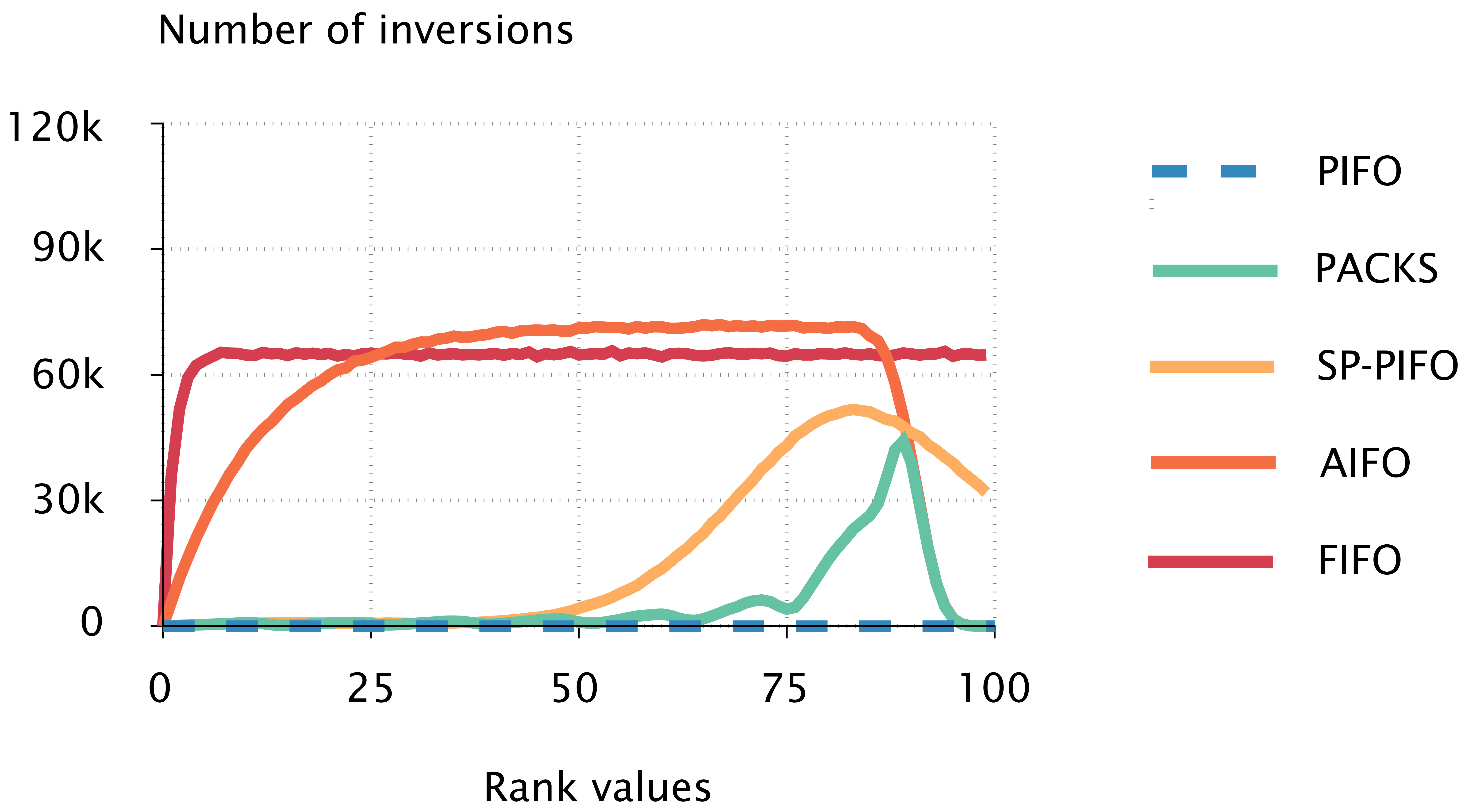
Heuristic analysis (MetaOpt)

Adversarial workload analysis

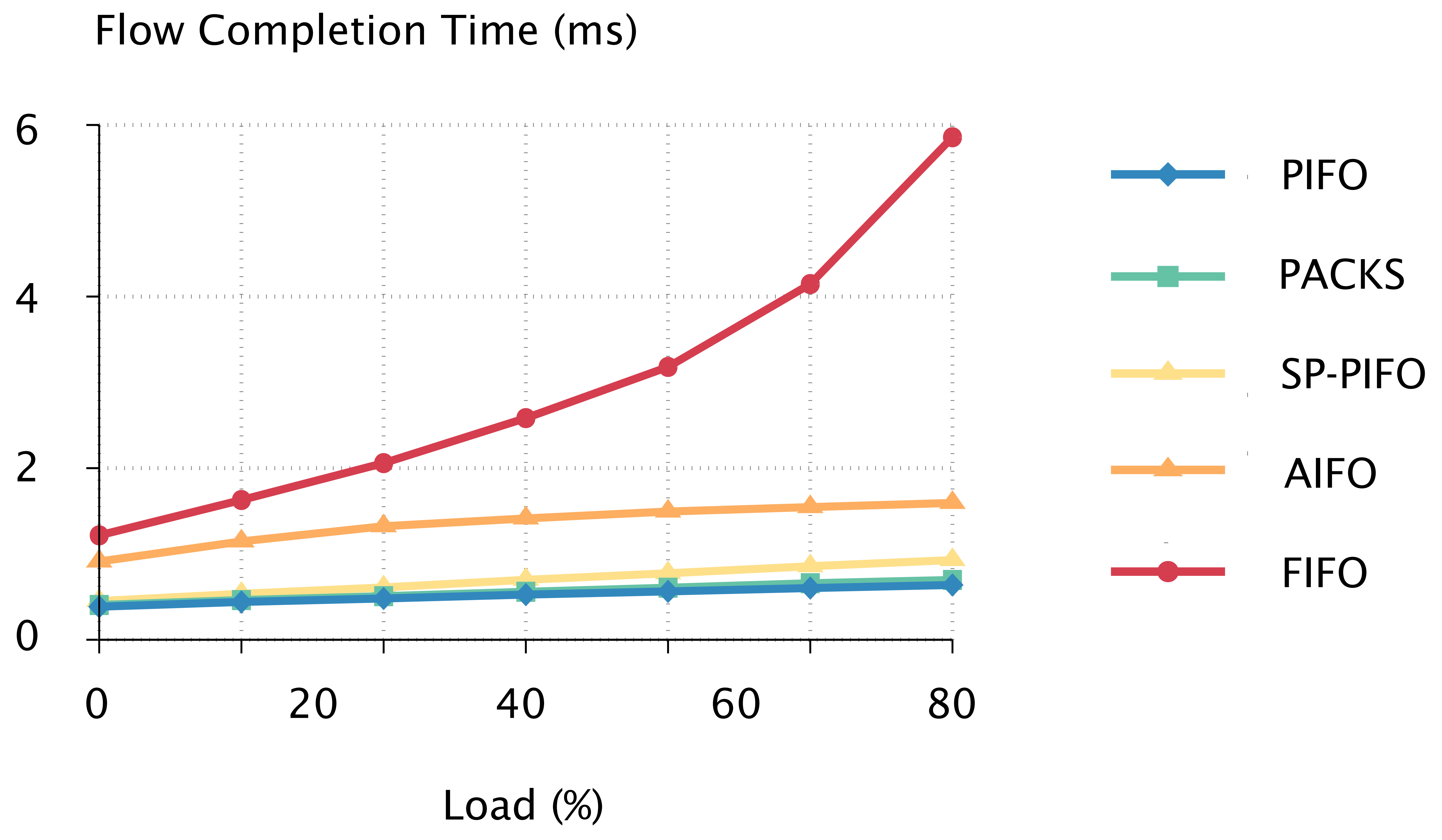
PACKS reduces packet drops by up to 60% compared to SP-PIFO



PACKS reduces inversions by up to 7x and 15x compared to SP-PIFO and AIFO



PACKS reduces mean FCTs by up to 33% and 2.6x compared to SP-PIFO and AIFO



Everything Matters in Programmable Packet Scheduling

PACKS approximates PIFO's admission and scheduling behaviors

at line rate, on existing programmable switches

PACKS adapts to traffic workloads in real time

using a sliding window and queue-aware policies

PACKS outperforms existing approaches

reducing drops, inversions, and flow completion times

github.com/nsg-ethz/packs