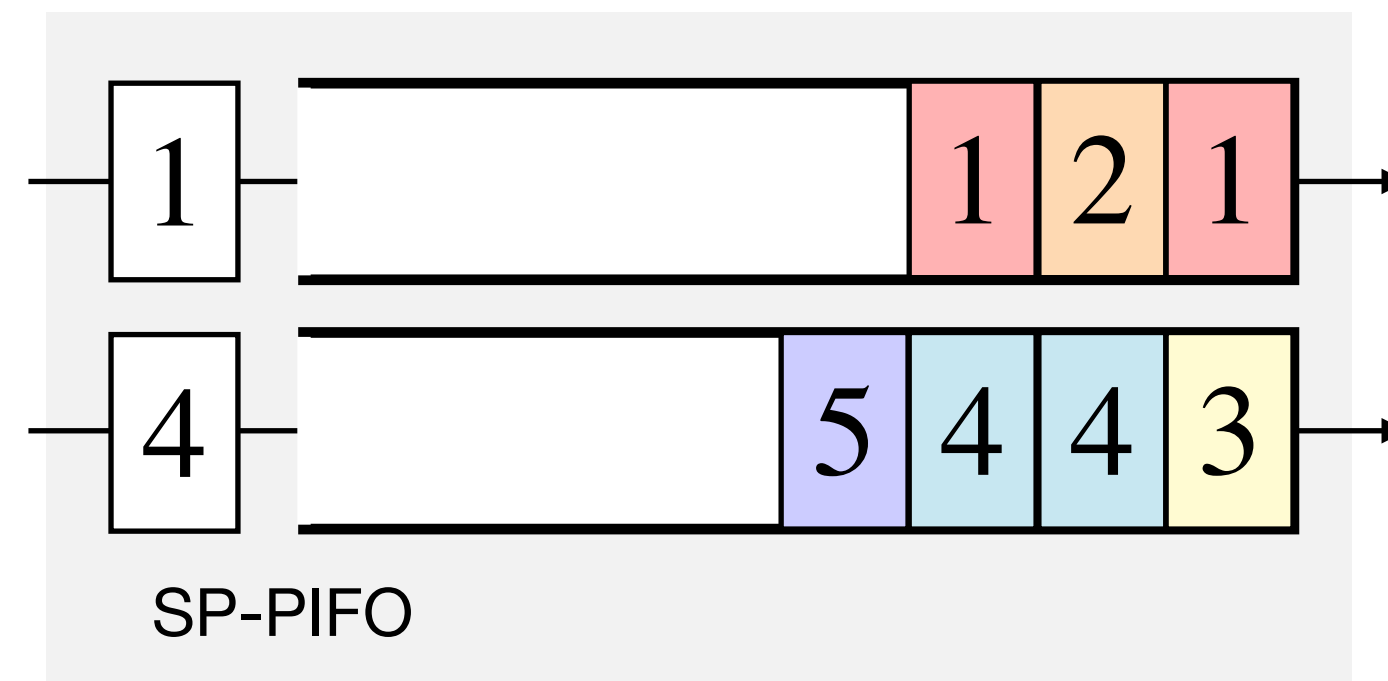


# SP-PIFO: Approximating Push-In First-Out Behaviors Using Strict-Priority Queues



**Albert Gran Alcoz,**  
Alexander Dietmüller,  
Laurent Vanbever

[sp-pifo.ethz.ch](http://sp-pifo.ethz.ch)

Barefoot Networks  
February, 28 2020

**ETH** zürich

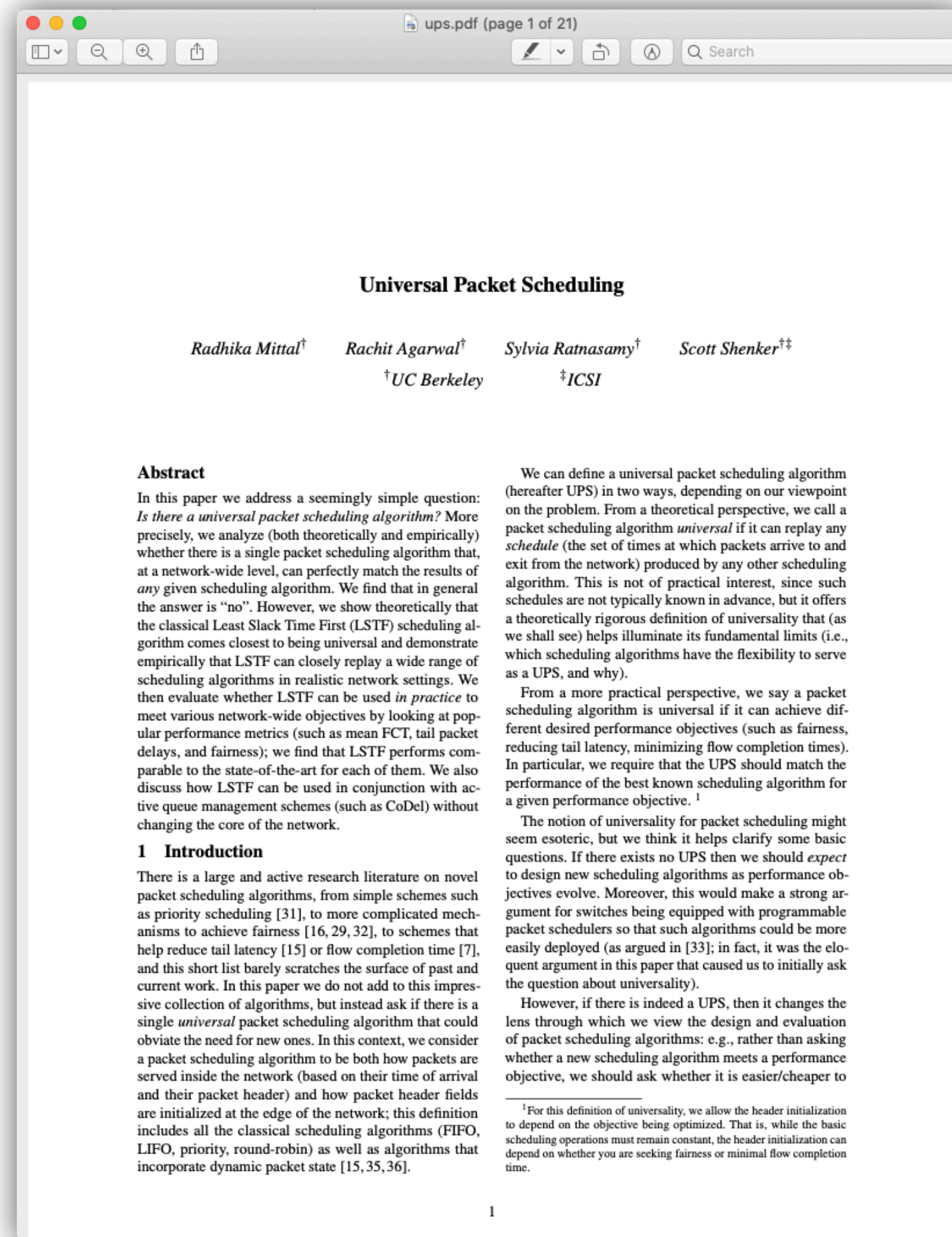








Is there a universal packet scheduler?  
[NSDI '16]



“You can’t have *everything* you want,  
but you can have *anything* you want”

Generality  
Universal packet scheduler

Flexibility  
Customized algorithms

Is there a universal packet scheduler?  
[NSDI '16]





“You can’t have *everything* you want,  
but you can have *anything* you want”

Generality  
Universal packet scheduler

Programmable  
scheduling

Is there a universal packet scheduler?  
[NSDI '16]

# Implementing PIFO queues in hardware is difficult

## Deployability

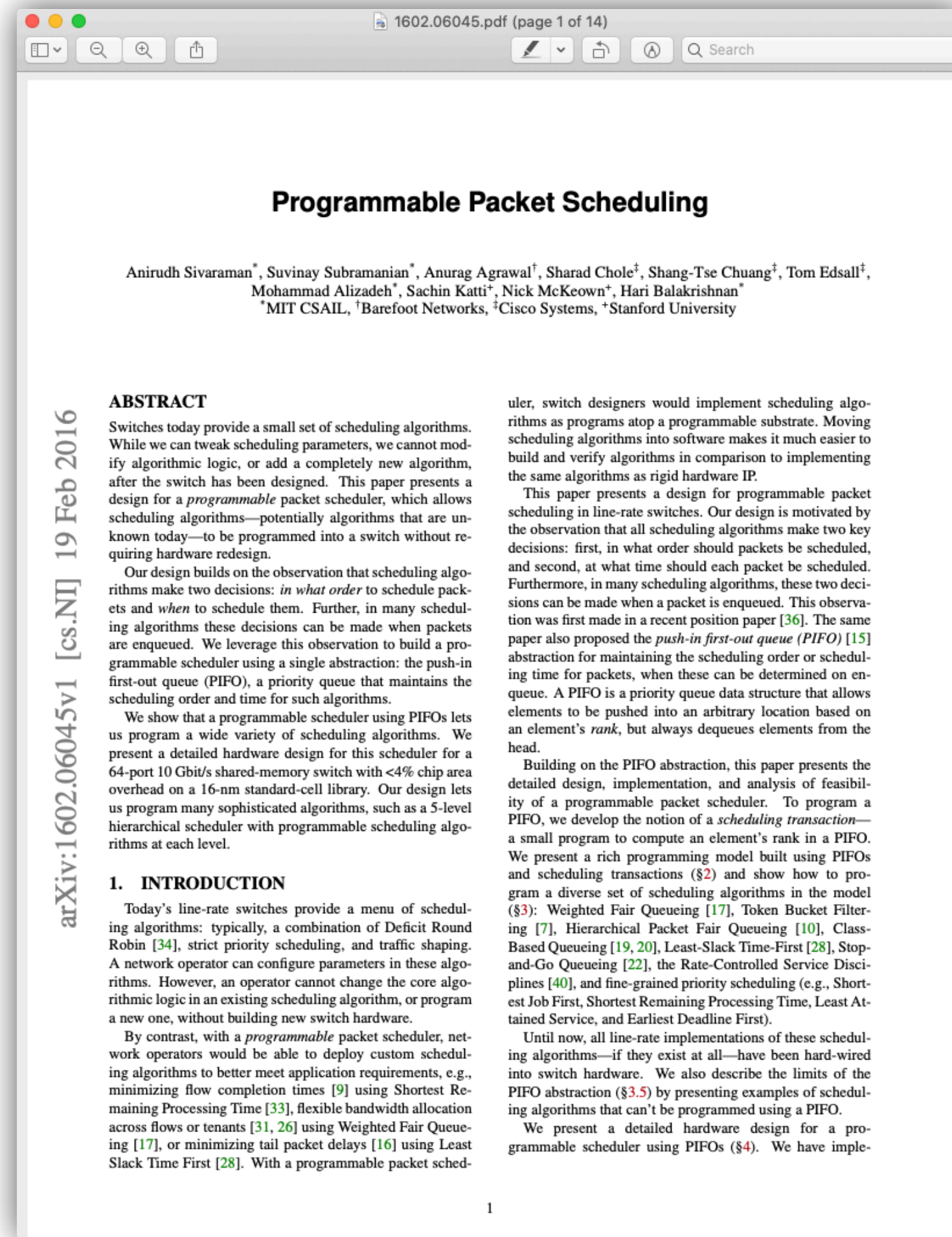
Requires new ASIC implementation, which might take years

## Scalability

Supports ~1k flows and ~10 Gbps

## Flexibility

Assumes monotonic increase of ranks within flows



PIFO abstraction for programmable scheduling

[SIGCOMM '16]

Can we approximate PIFO queues...

at line rate,

at scale, and

on existing devices?

Introducing...

# SP-PIFO

*A deployable, scalable and flexible*

PIFO approximation



# PIFO queues can be used as an abstraction to make scheduling programmable

## The PIFO queue

Allows packets to be pushed into arbitrary locations

Only drains packets from the head

Packet ranks



# PIFO queues can be used as an abstraction to make scheduling programmable

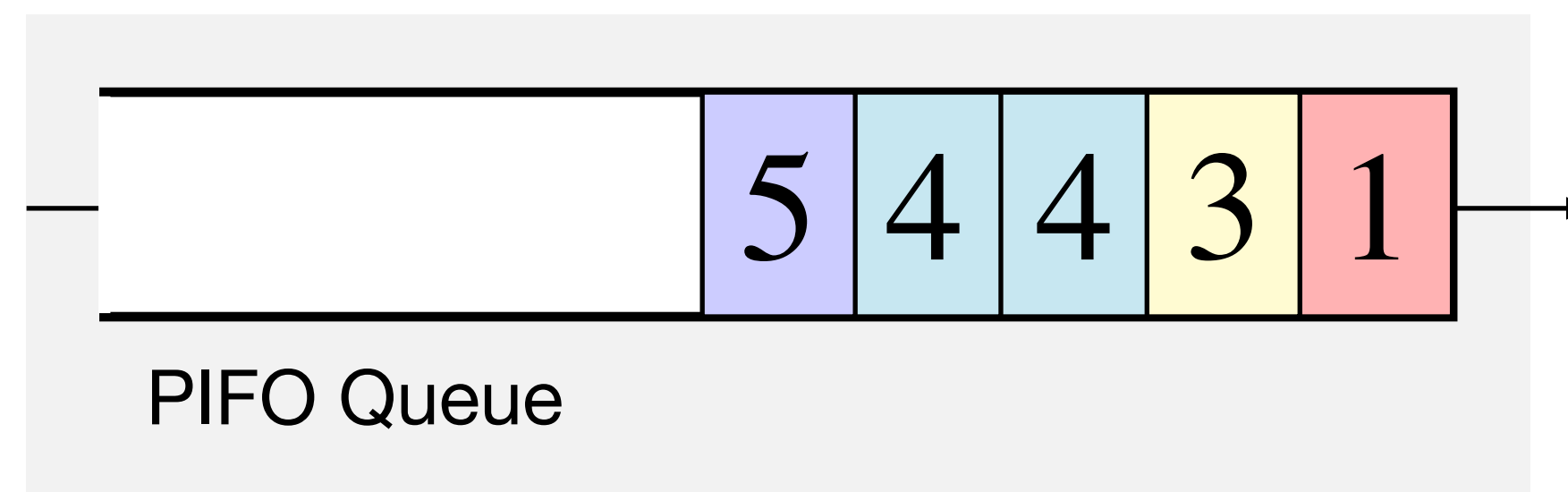
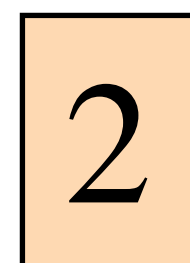
## The PIFO queue

Allows packets to be pushed into arbitrary locations

Only drains packets from the head

Packet ranks

Input sequence



Output sequence

# PIFO queues can be used as an abstraction to make scheduling programmable

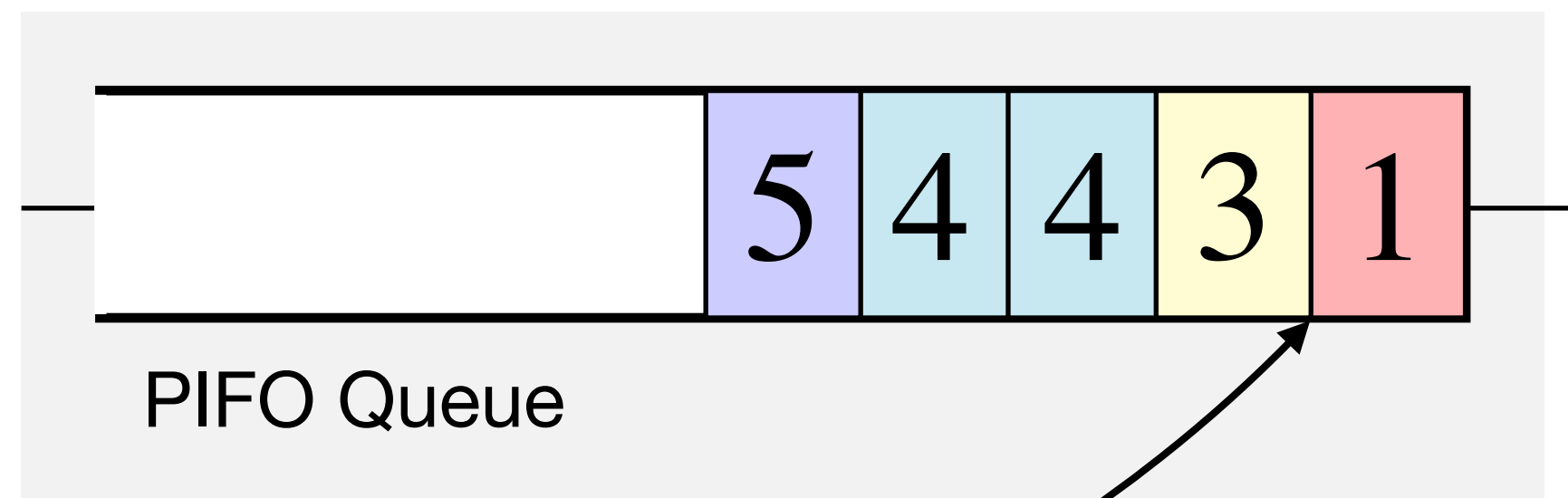
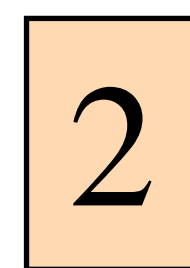
## The PIFO queue

Allows packets to be pushed into arbitrary locations

Only drains packets from the head

Packet ranks

Input sequence



Output sequence



# PIFO queues can be used as an abstraction to make scheduling programmable

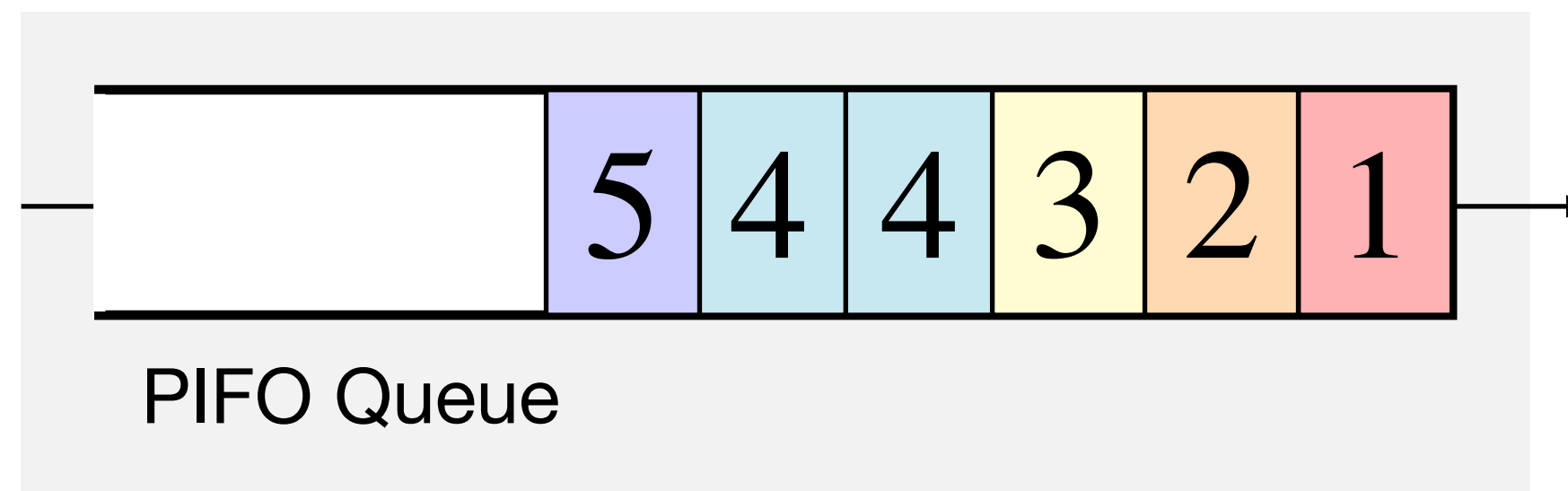
## The PIFO queue

Allows packets to be pushed into arbitrary locations

Only drains packets from the head

Packet ranks

Input sequence



Output sequence

# PIFO queues can be used as an abstraction to make scheduling programmable

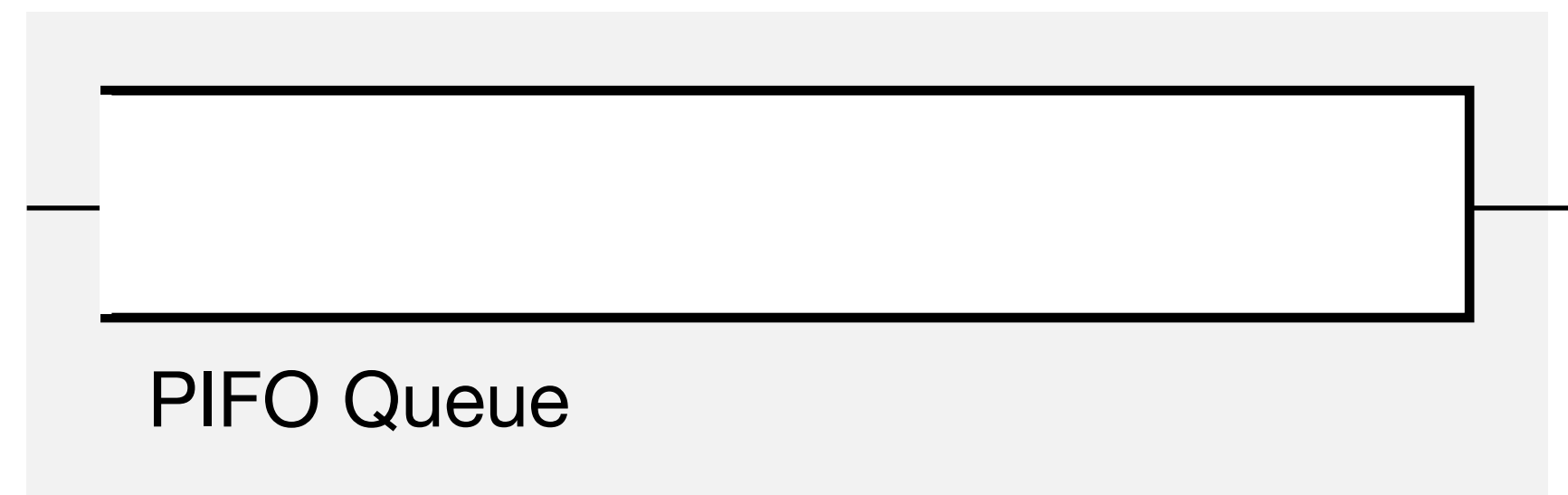
## The PIFO queue

Allows packets to be pushed into arbitrary locations

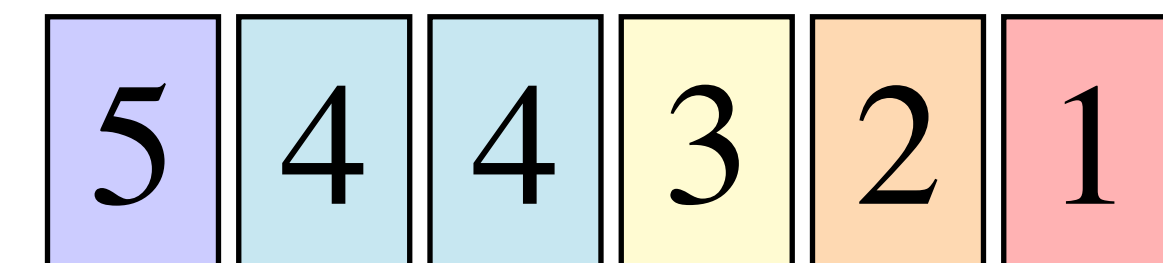
Only drains packets from the head

Packet ranks

Input sequence



Output sequence

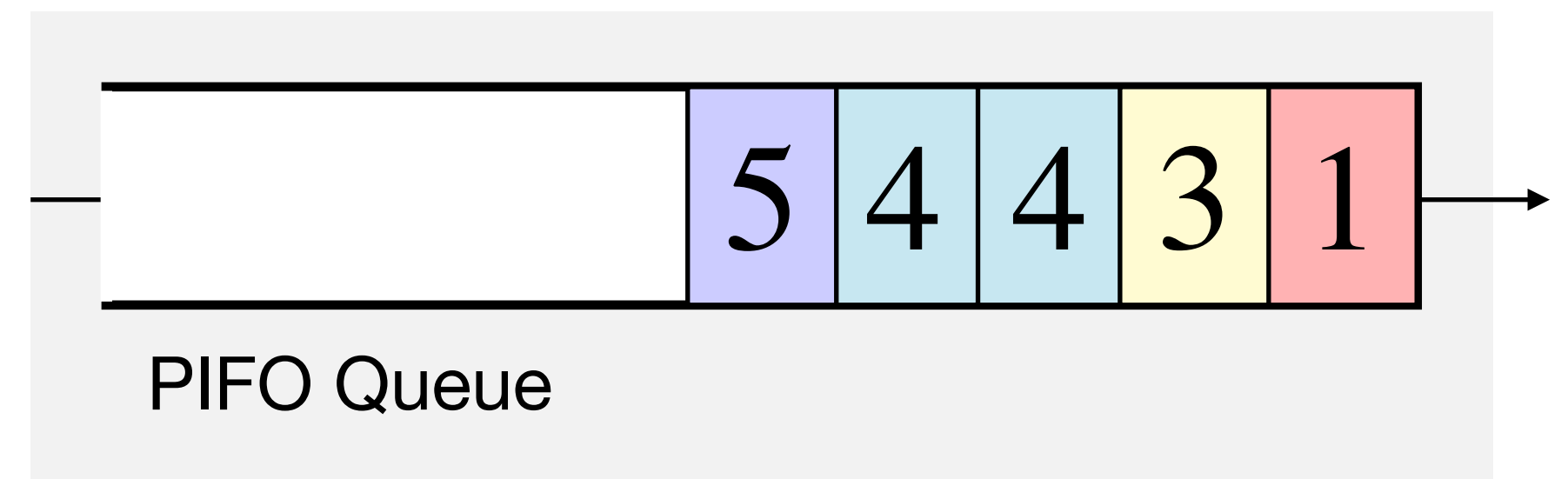
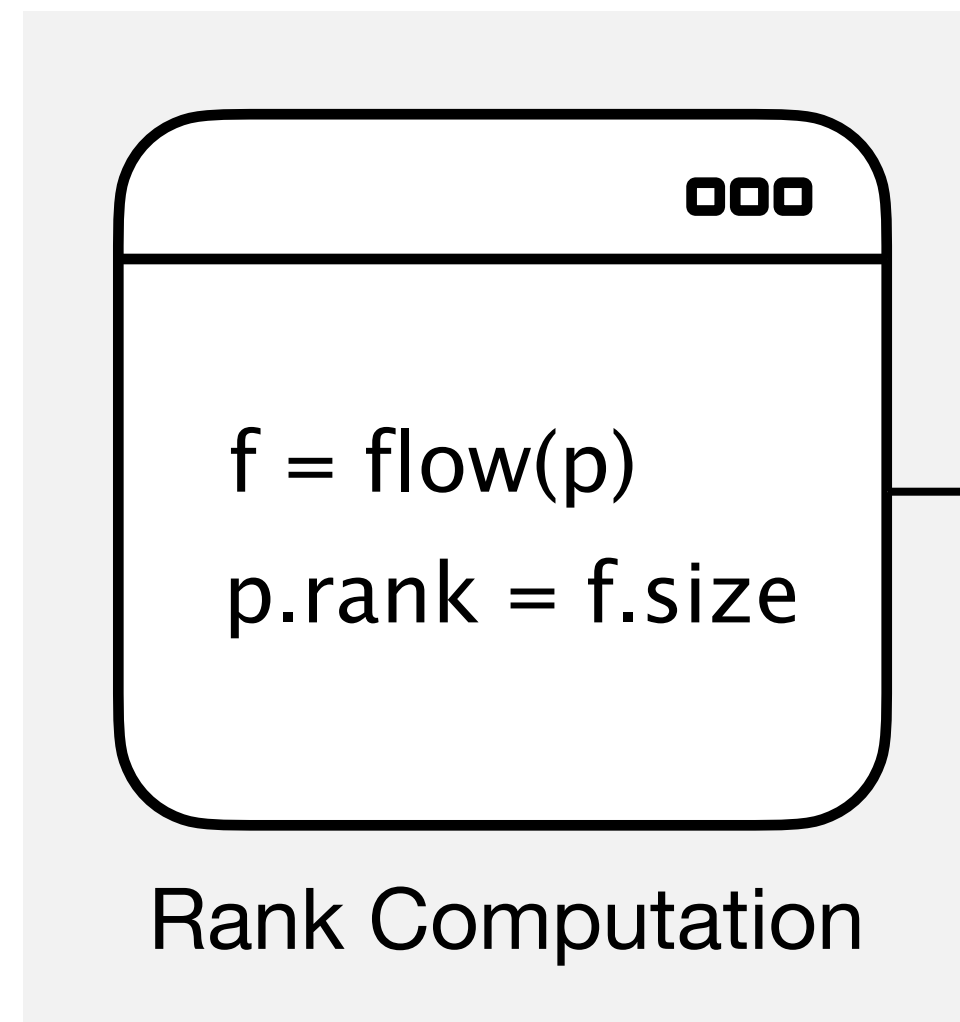
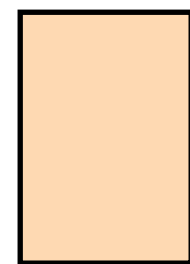


# PIFO queues can be used as an abstraction to make scheduling programmable

## Programmable scheduler

Rank computation (programmable)  
PIFO queue (fixed logic)

Input sequence



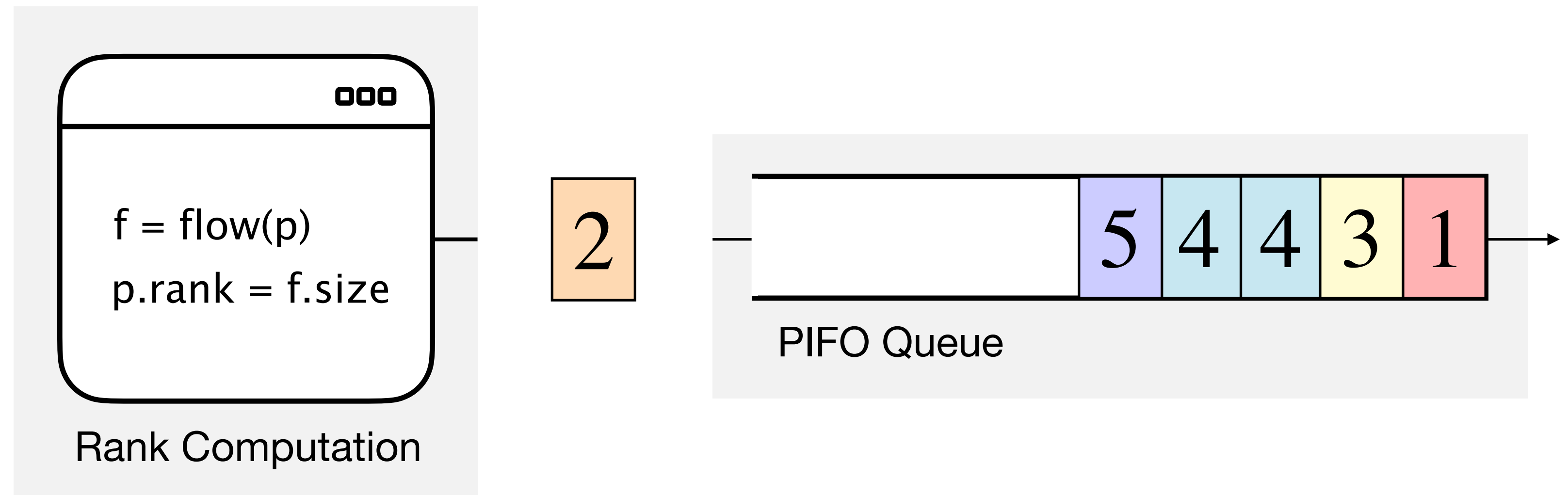


# PIFO queues can be used as an abstraction to make scheduling programmable

## Programmable scheduler

Rank computation (programmable)  
PIFO queue (fixed logic)

Input sequence

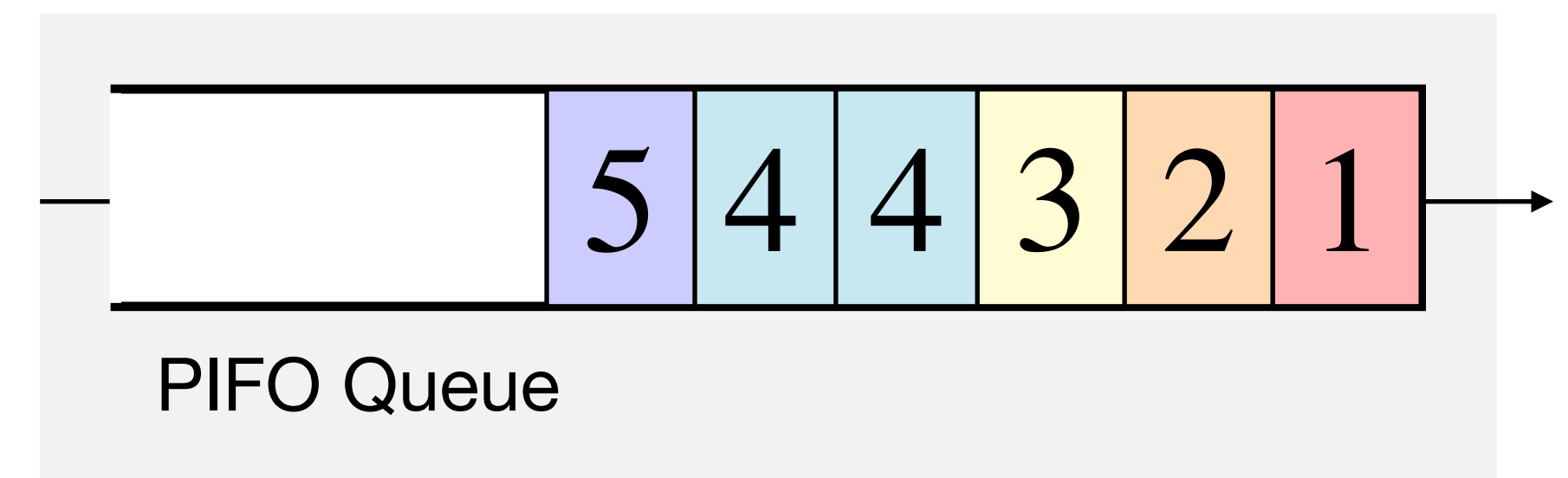
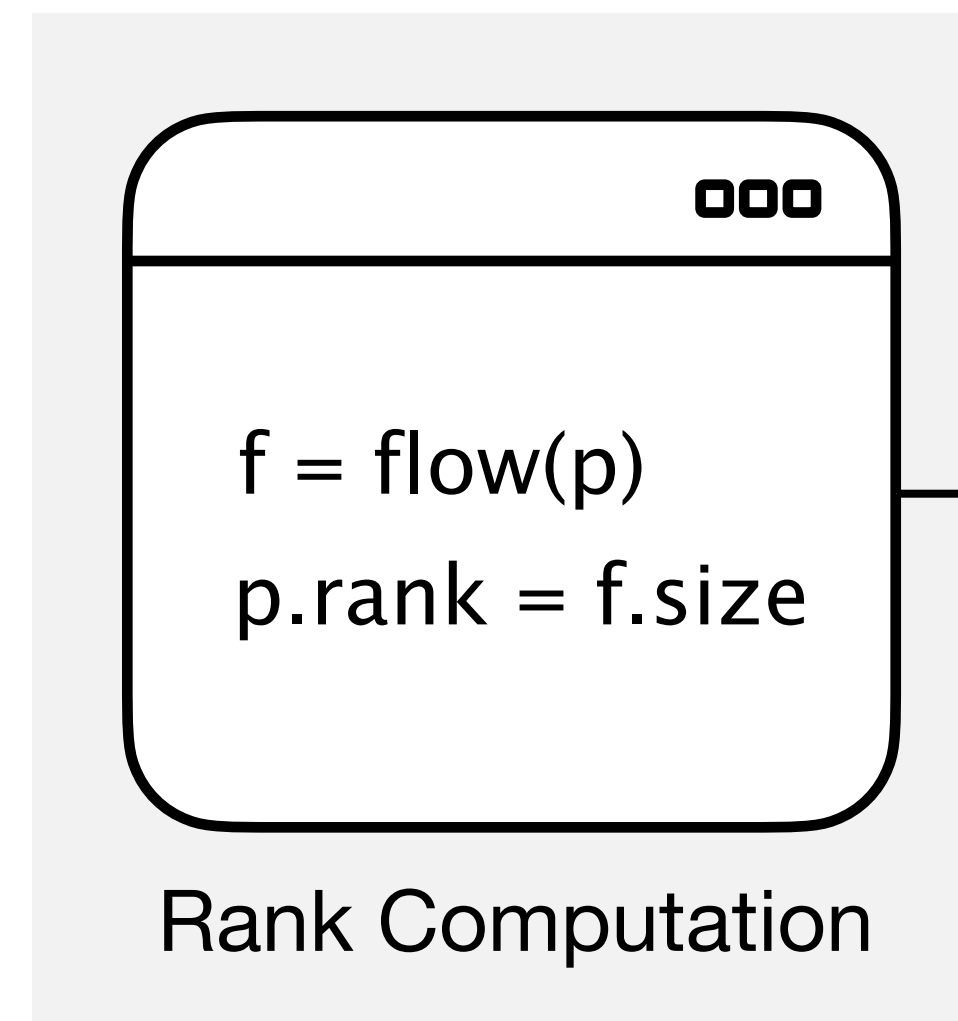


# PIFO queues can be used as an abstraction to make scheduling programmable

## Programmable scheduler

Rank computation (programmable)  
PIFO queue (fixed logic)

Input sequence

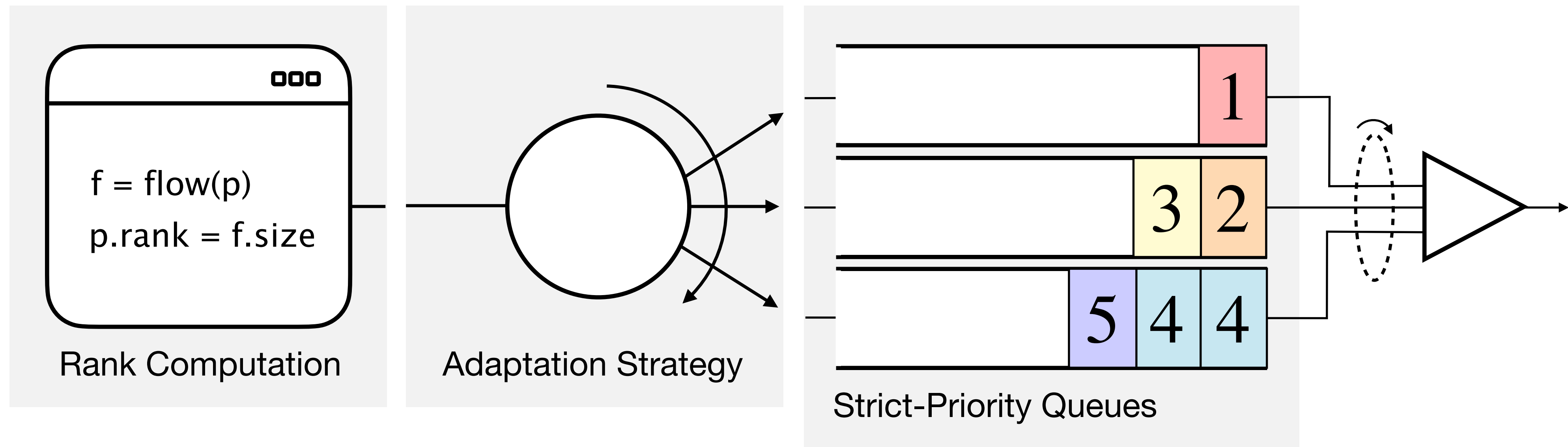


# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**Programmable scheduler**

Rank computation (programmable)

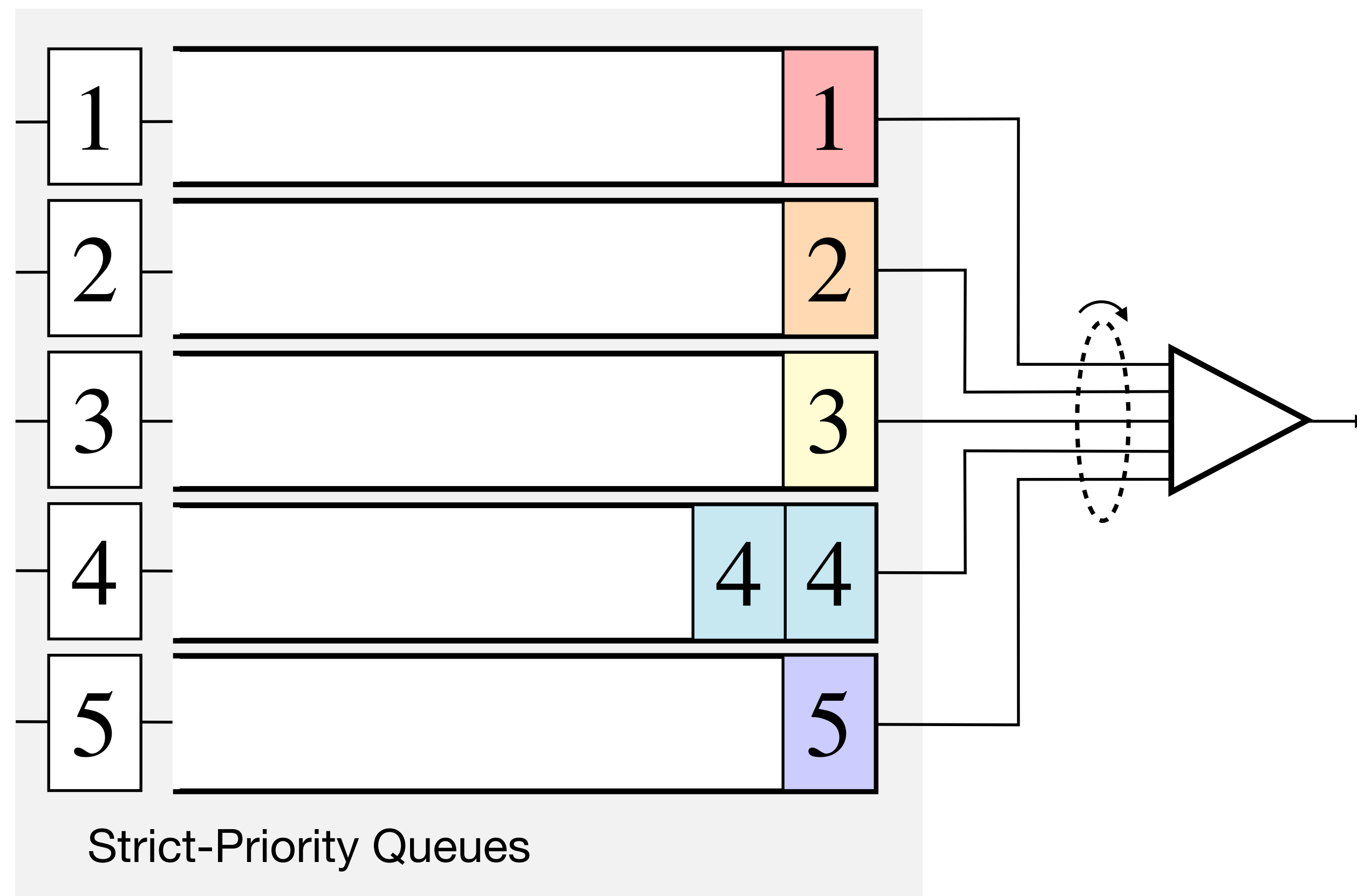
**Adaptation strategy + strict-priority queues** (fixed logic)





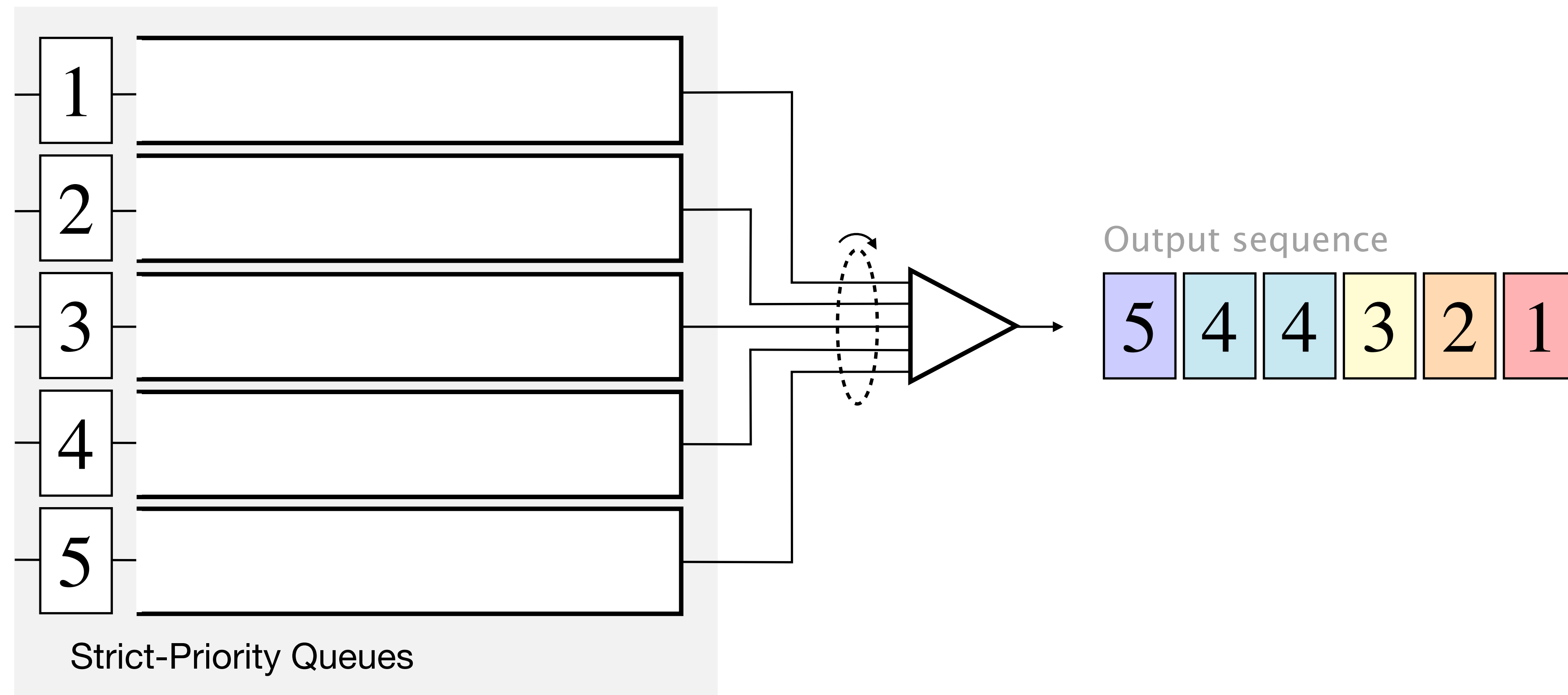
# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**Ideal case** Perfect PIFO if number of queues  $\geq$  number of ranks



# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**Ideal case** Perfect PIFO if number of queues  $\geq$  number of ranks



SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**In practice**      Number of queues < number of ranks

SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

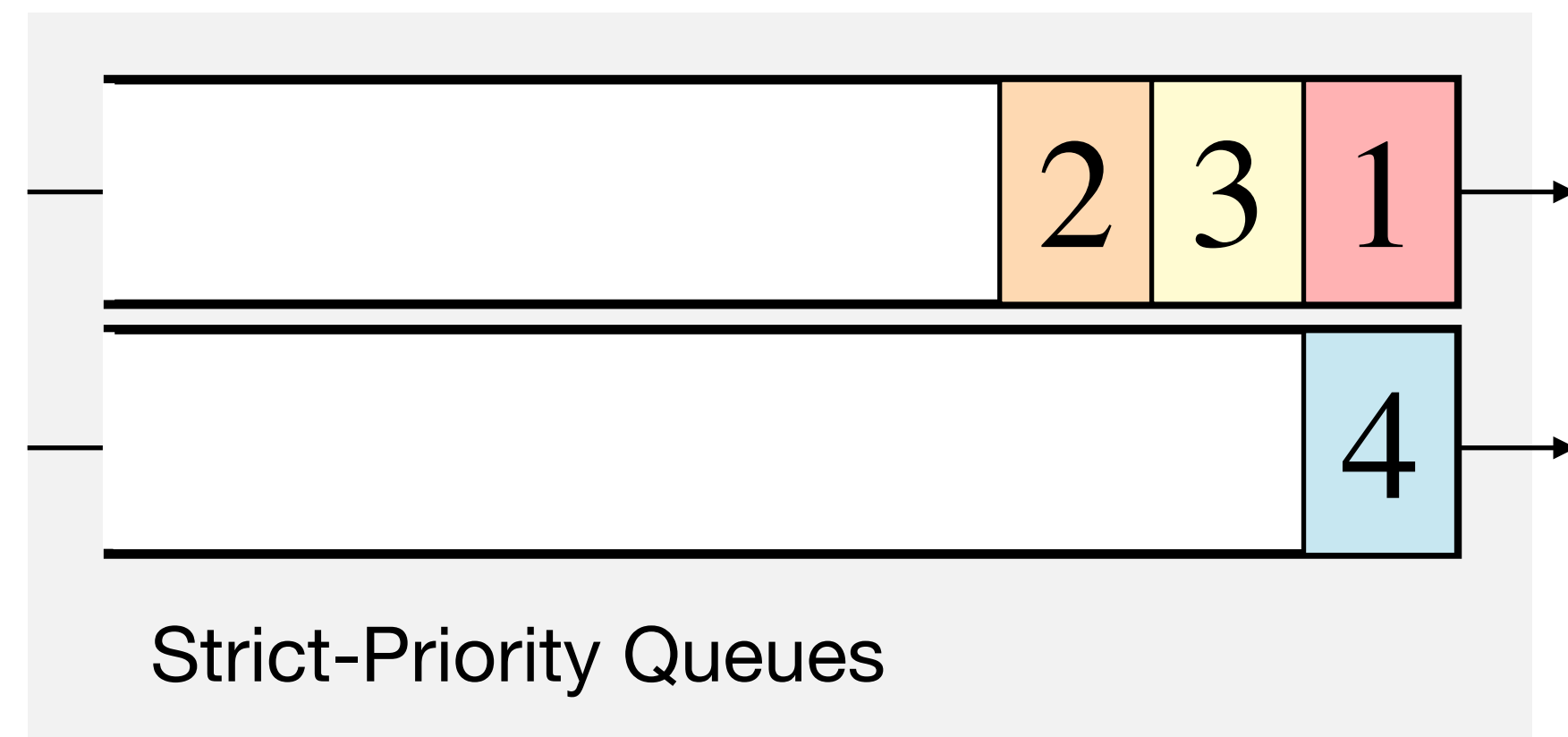
**In practice**      Number of queues < number of ranks

**Problem**      Output sequence can have **scheduling errors**

# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**In practice**      Number of queues < number of ranks

**Problem**      Output sequence can have **scheduling errors**

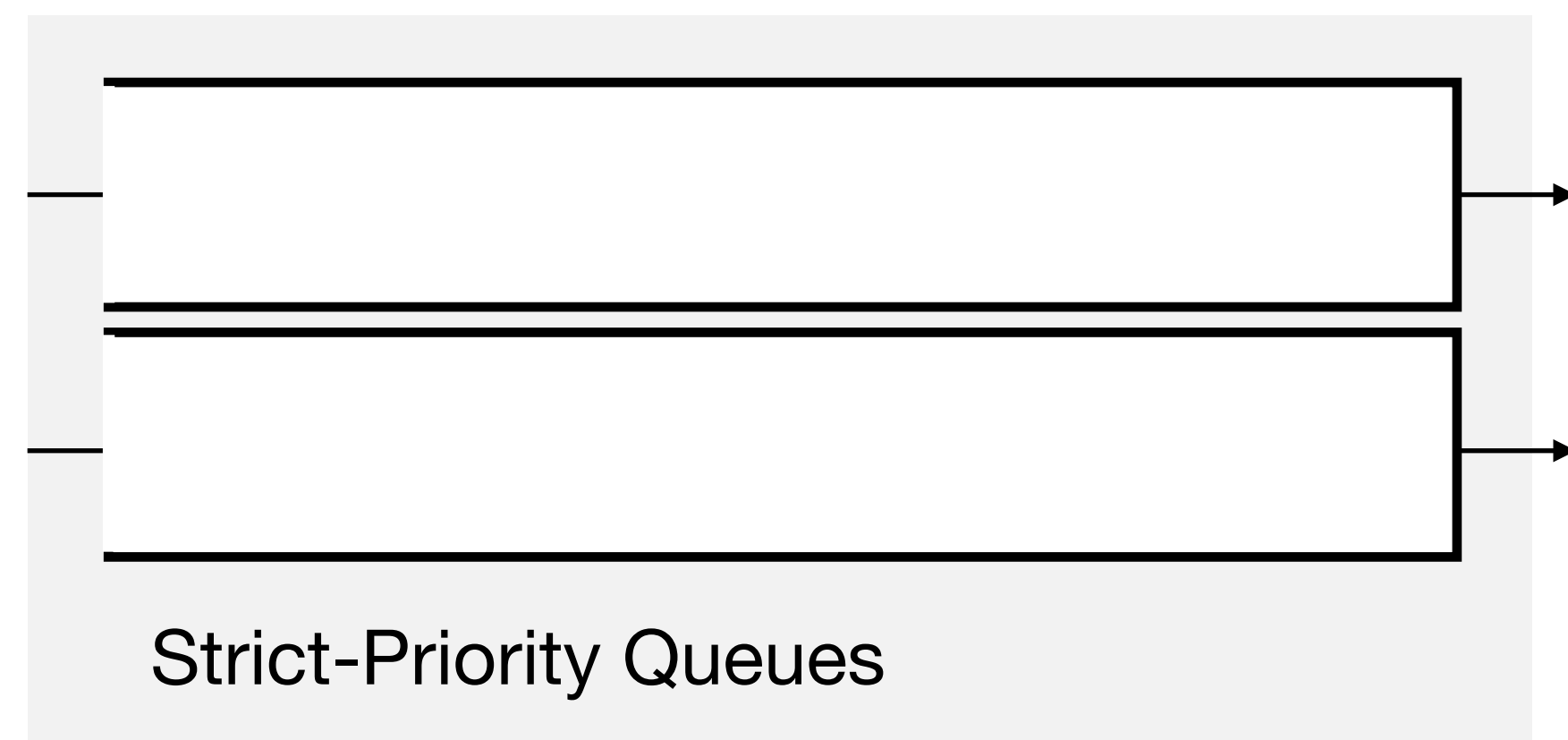




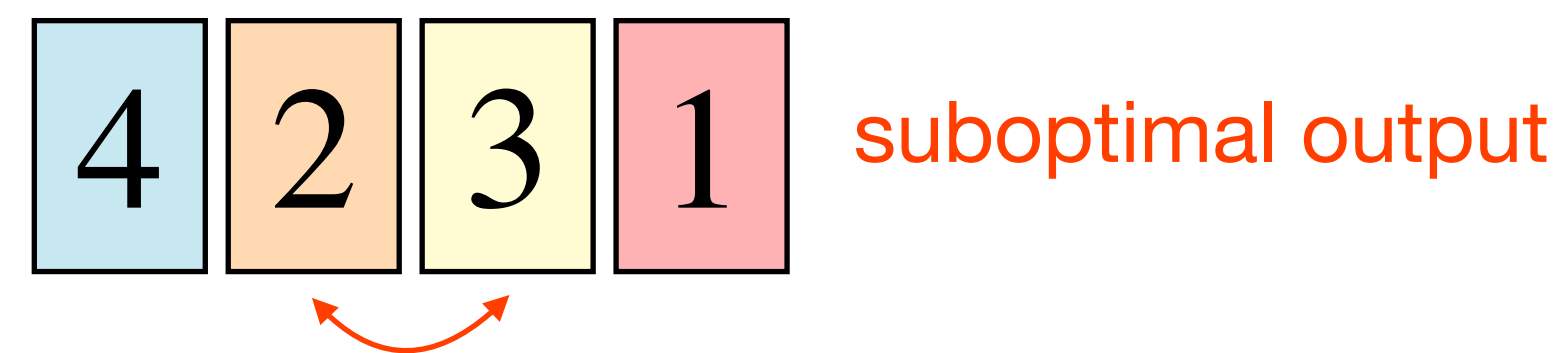
# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**In practice** Number of queues < number of ranks

**Problem** Output sequence can have **scheduling errors**



Low-ranked packets drained **after** high-ranked packets



# SP-PIFO approximates PIFO behaviors using Strict-Priority queues and a dynamic mapping strategy

**In practice**      Number of queues < number of ranks

**Problem**            Output sequence can have scheduling errors

**Opportunity**        Design mapping strategies that minimize scheduling errors

# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

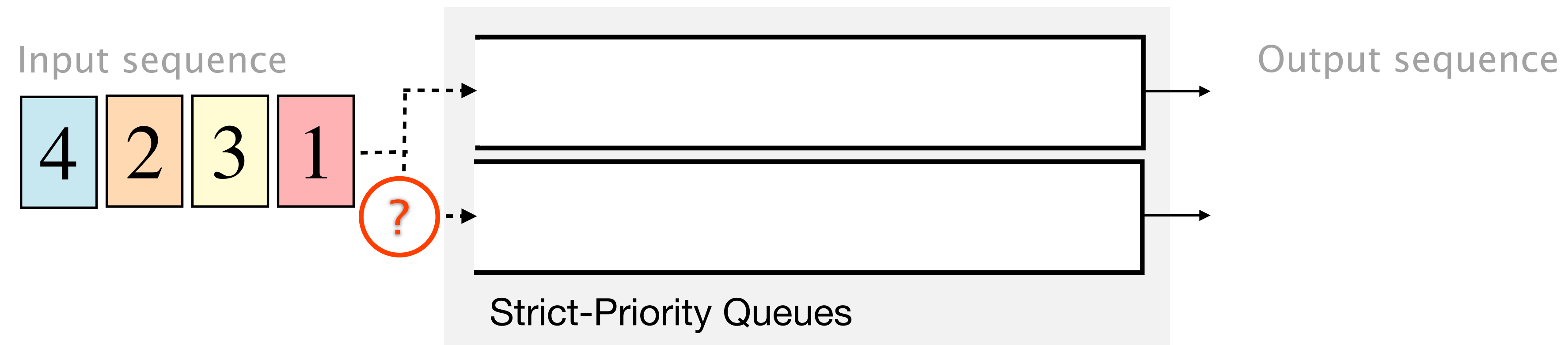
Packet enqueued if  $\text{rank} \geq \text{queue bound}$

# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if  $\text{rank} \geq \text{queue bound}$

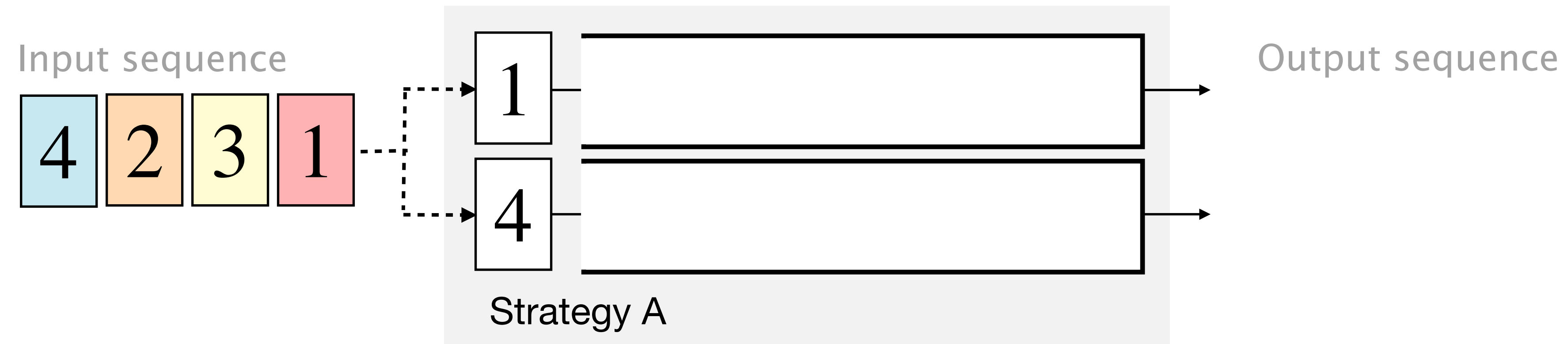


# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if **rank**  $\geq$  **queue bound**



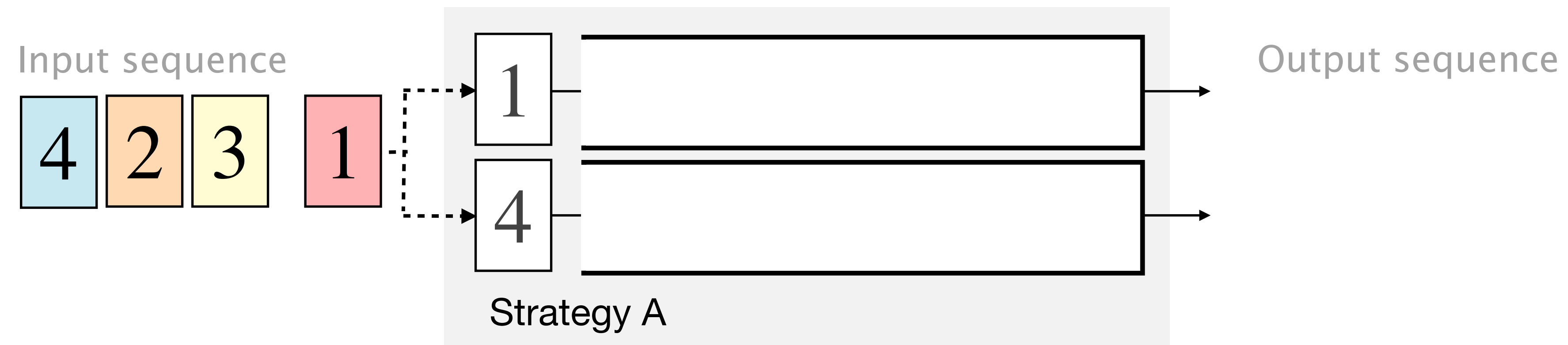


# SP-PIFO defines mapping through 'queue bounds'

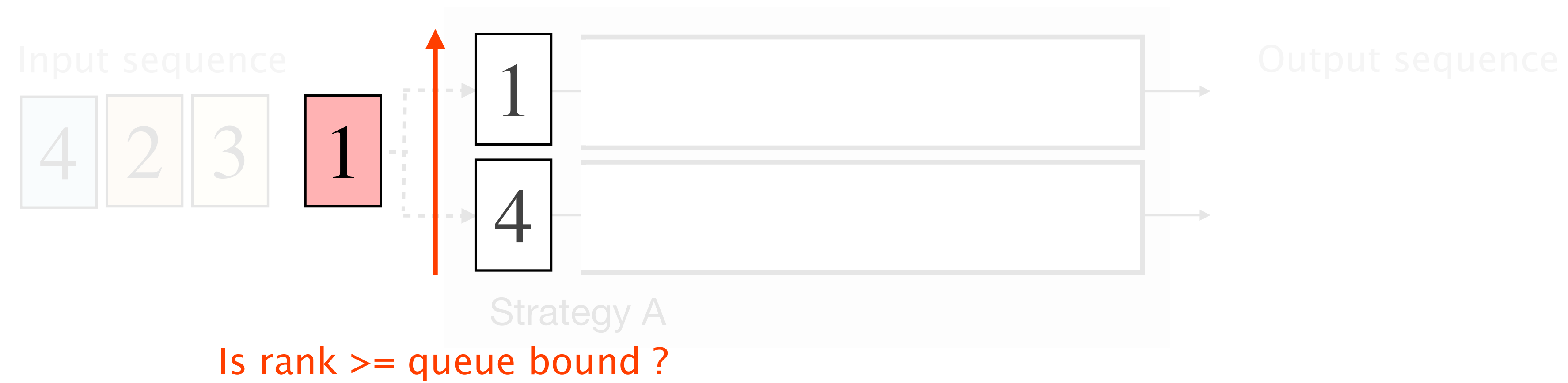
## Mapping

Queue bounds scanned bottom-up

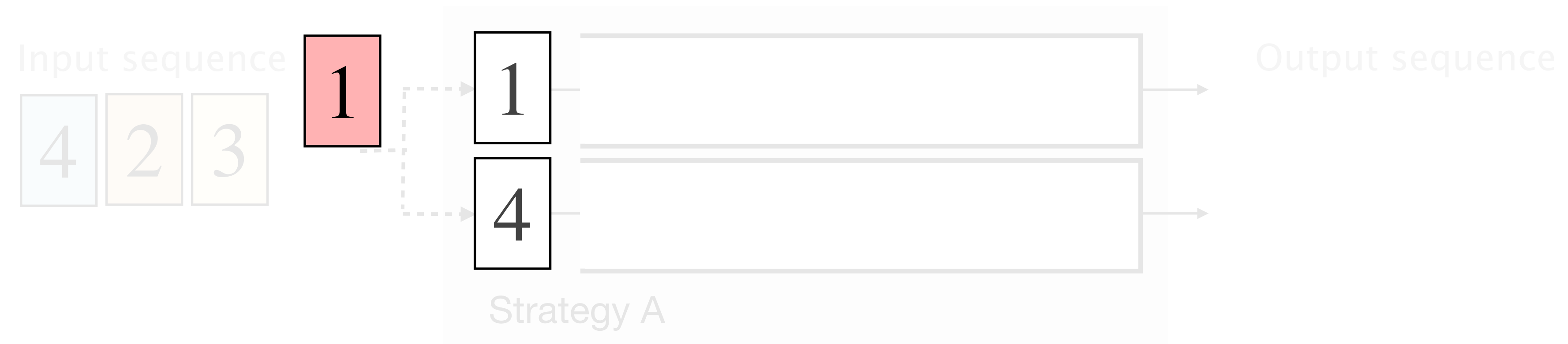
Packet enqueued if  $\text{rank} \geq \text{queue bound}$



# SP-PIFO defines mapping through 'queue bounds'



# SP-PIFO defines mapping through 'queue bounds'

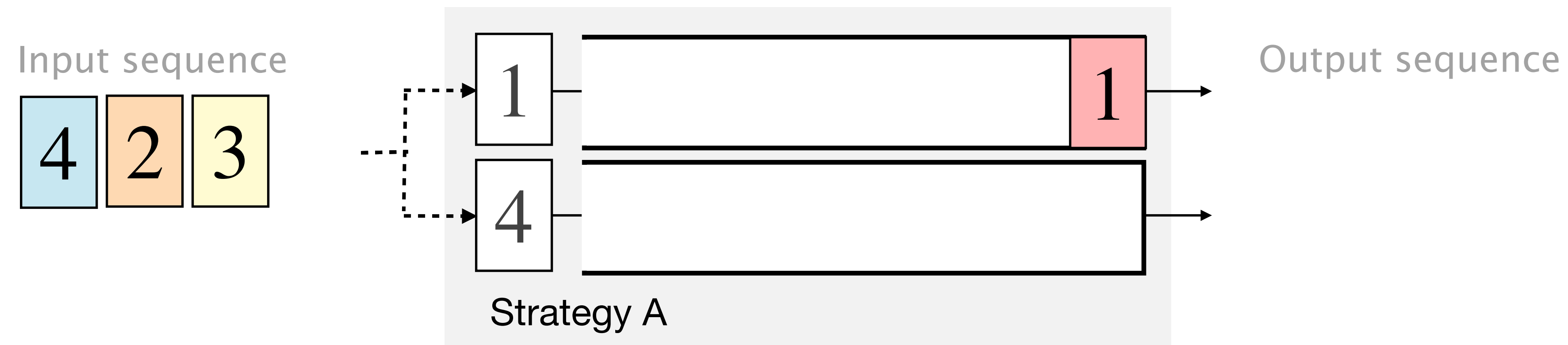


# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if **rank**  $\geq$  **queue bound**

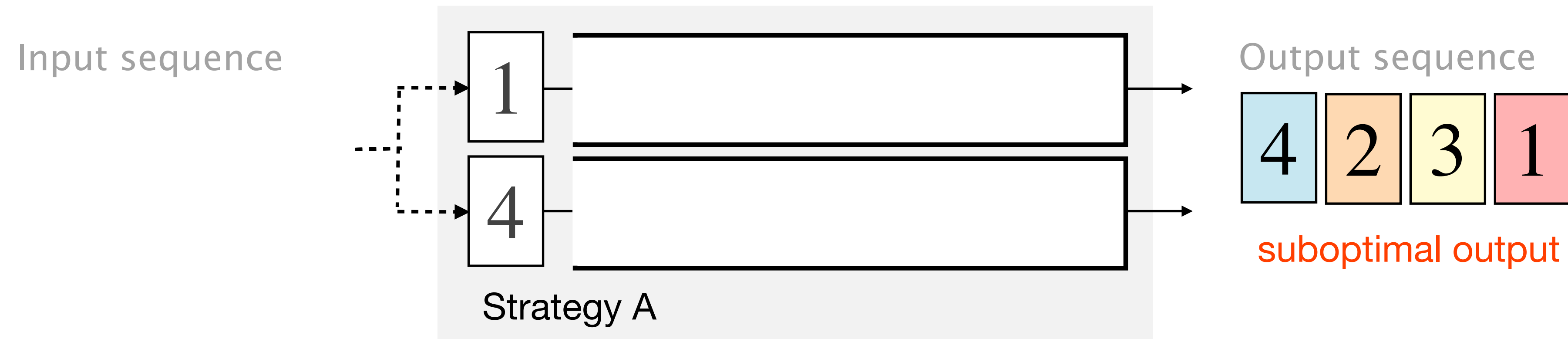


# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if  $\text{rank} \geq \text{queue bound}$



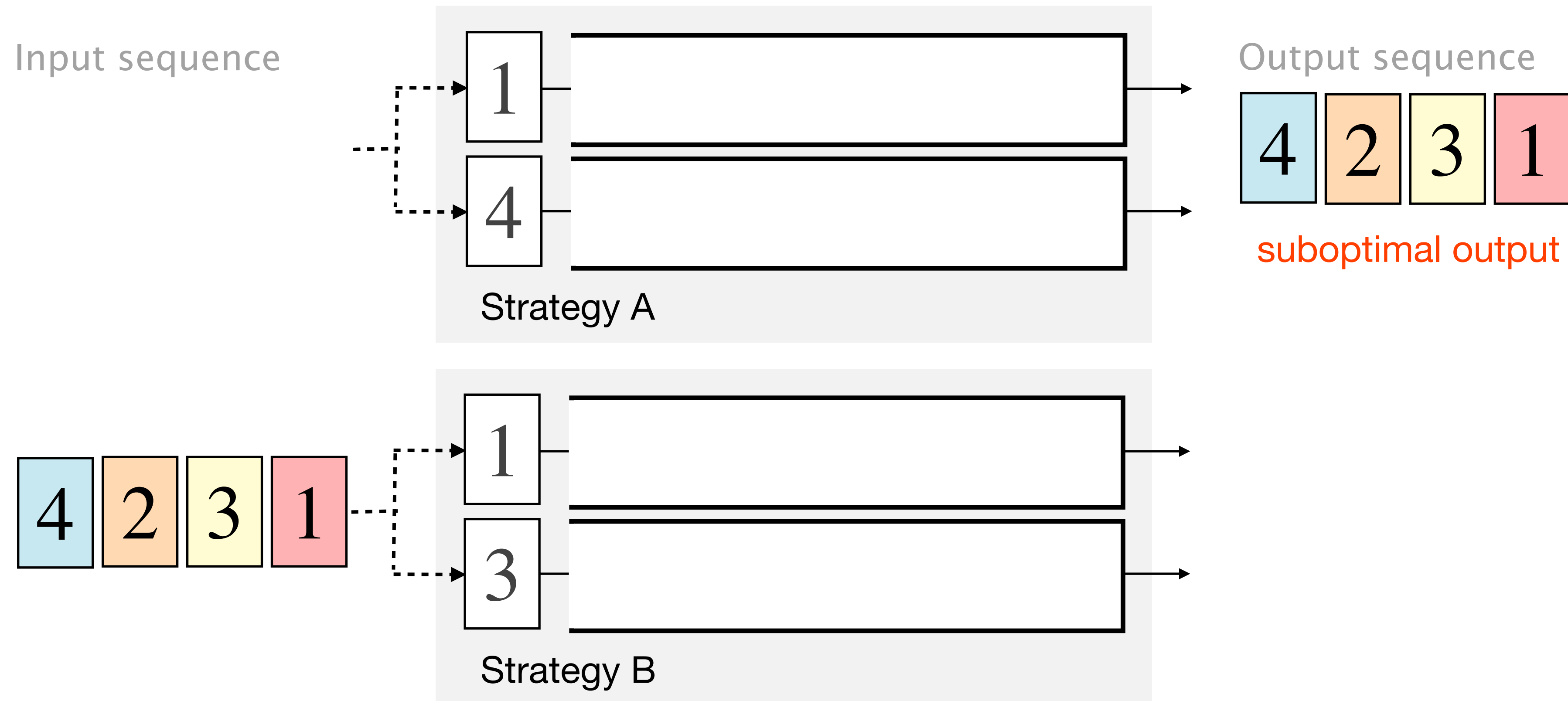


# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if  $\text{rank} \geq \text{queue bound}$

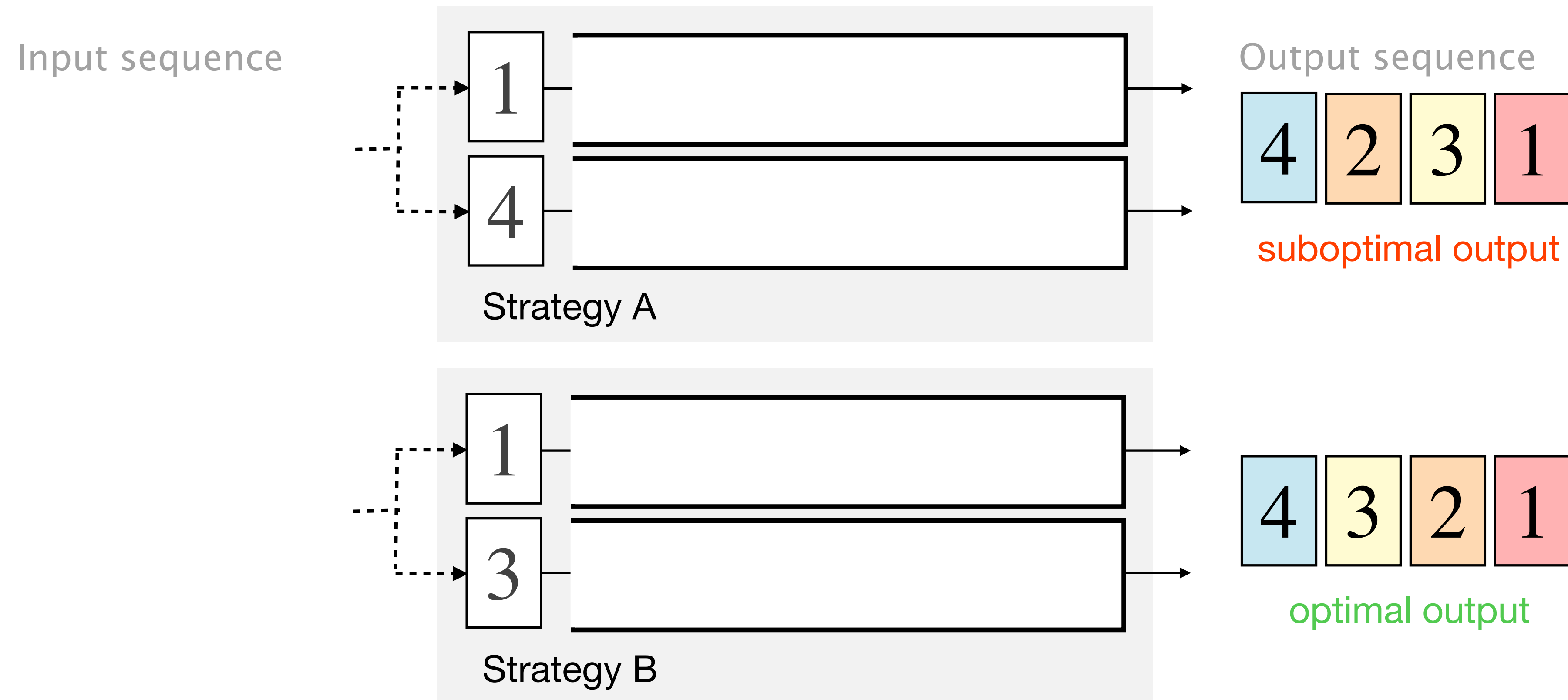


# SP-PIFO defines mapping through 'queue bounds'

## Mapping

Queue bounds scanned bottom-up

Packet enqueued if  $\text{rank} \geq \text{queue bound}$



How can we design a mapping strategy that minimizes scheduling errors?

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

- 1      **Adaptation design**  
How does it work
- 2      **Implementation**  
How can it be deployed
- 3      **Evaluation**  
How well does it perform

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

- 1 **Adaptation design**  
How does it work
- 2 **Implementation**  
How can it be deployed
- 3 **Evaluation**  
How well does it perform



# Problem formulation

**Objective** Find optimal queue bounds  $q^*$   
That minimize the expected loss  $U$  for all ranks

$$q^* = \underset{q \in Q}{\operatorname{argmin}} E_{r \sim R} [ U(q, r) ]$$

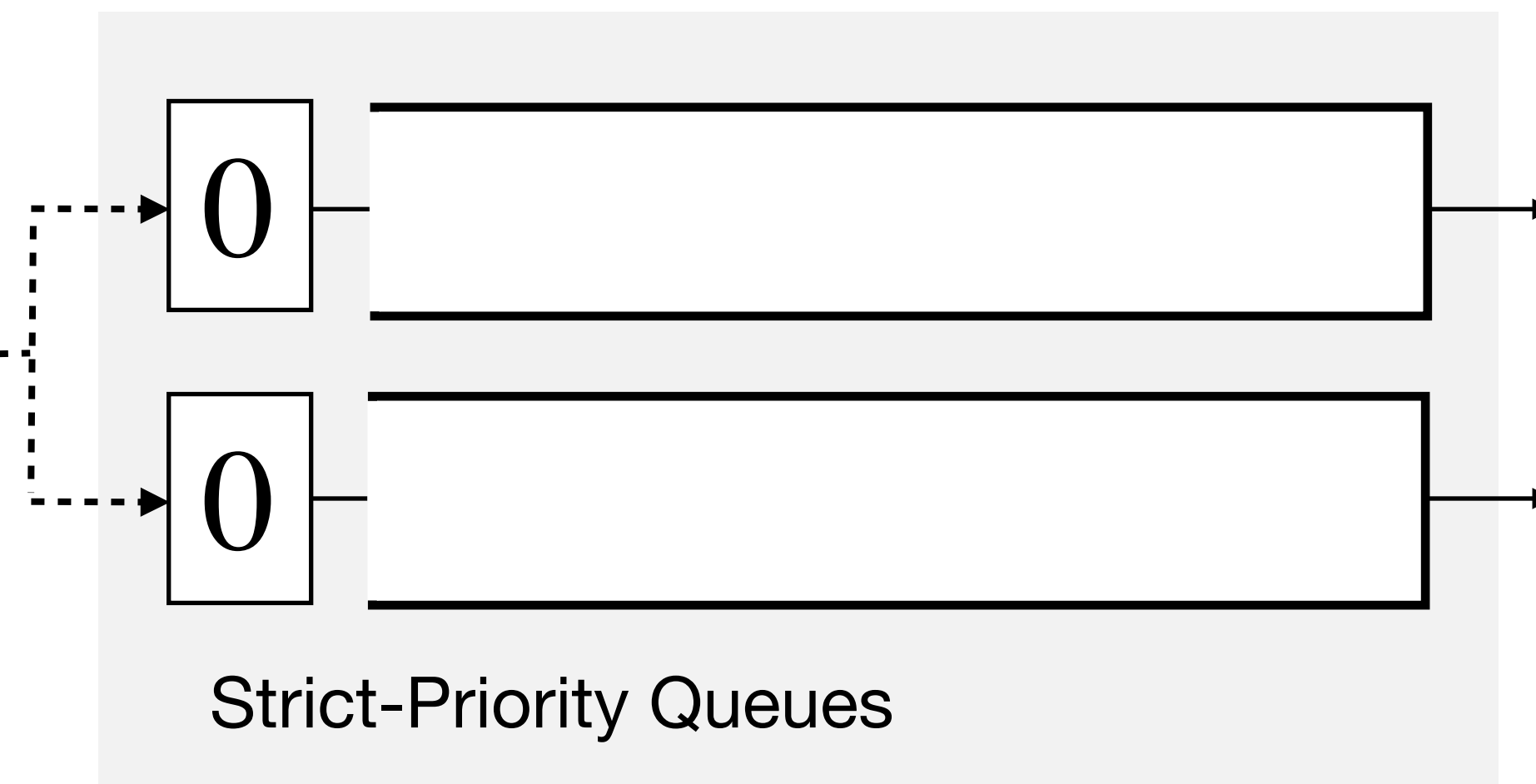
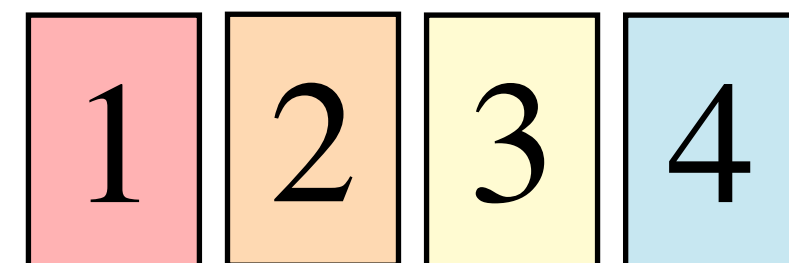
*Unpifoness* ( $U$ ) quantifies the scheduling errors

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

## Initialization

Zero traffic knowledge  
Queue bounds set to zero

Input sequence

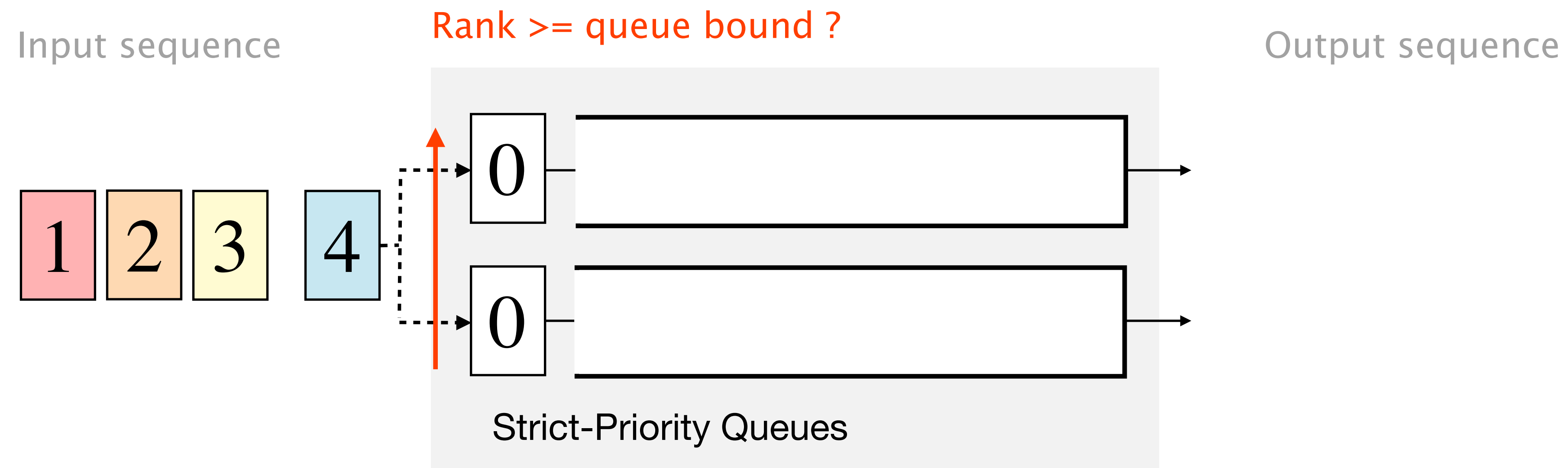


Output sequence

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

## Initialization

Zero traffic knowledge  
Queue bounds set to zero

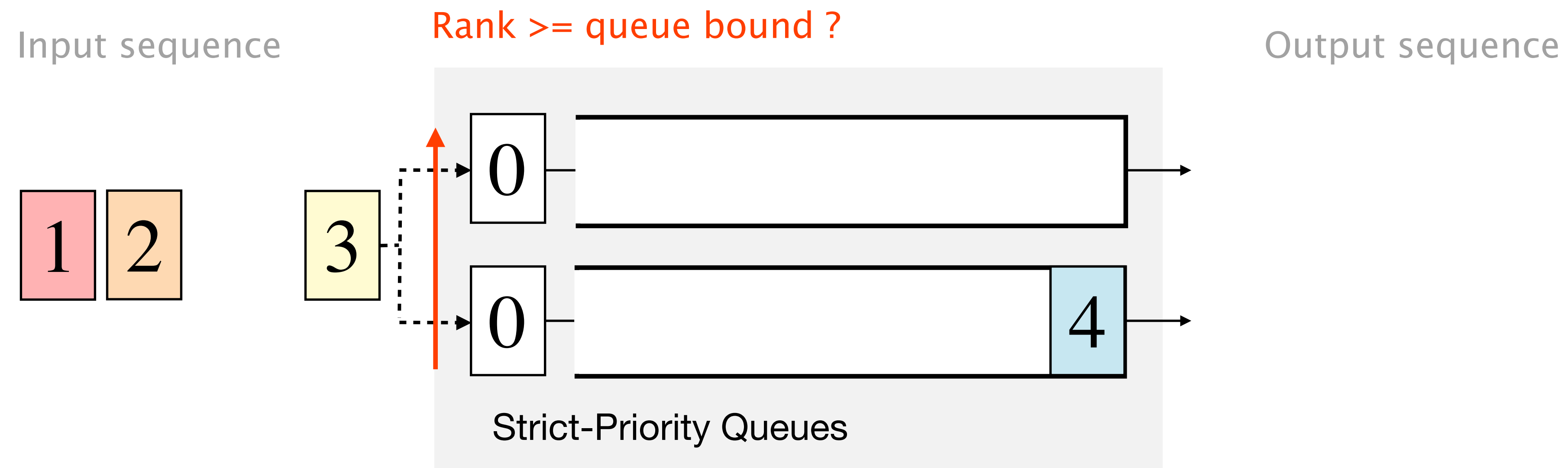


# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Initialization

Zero traffic knowledge

Queue bounds set to zero



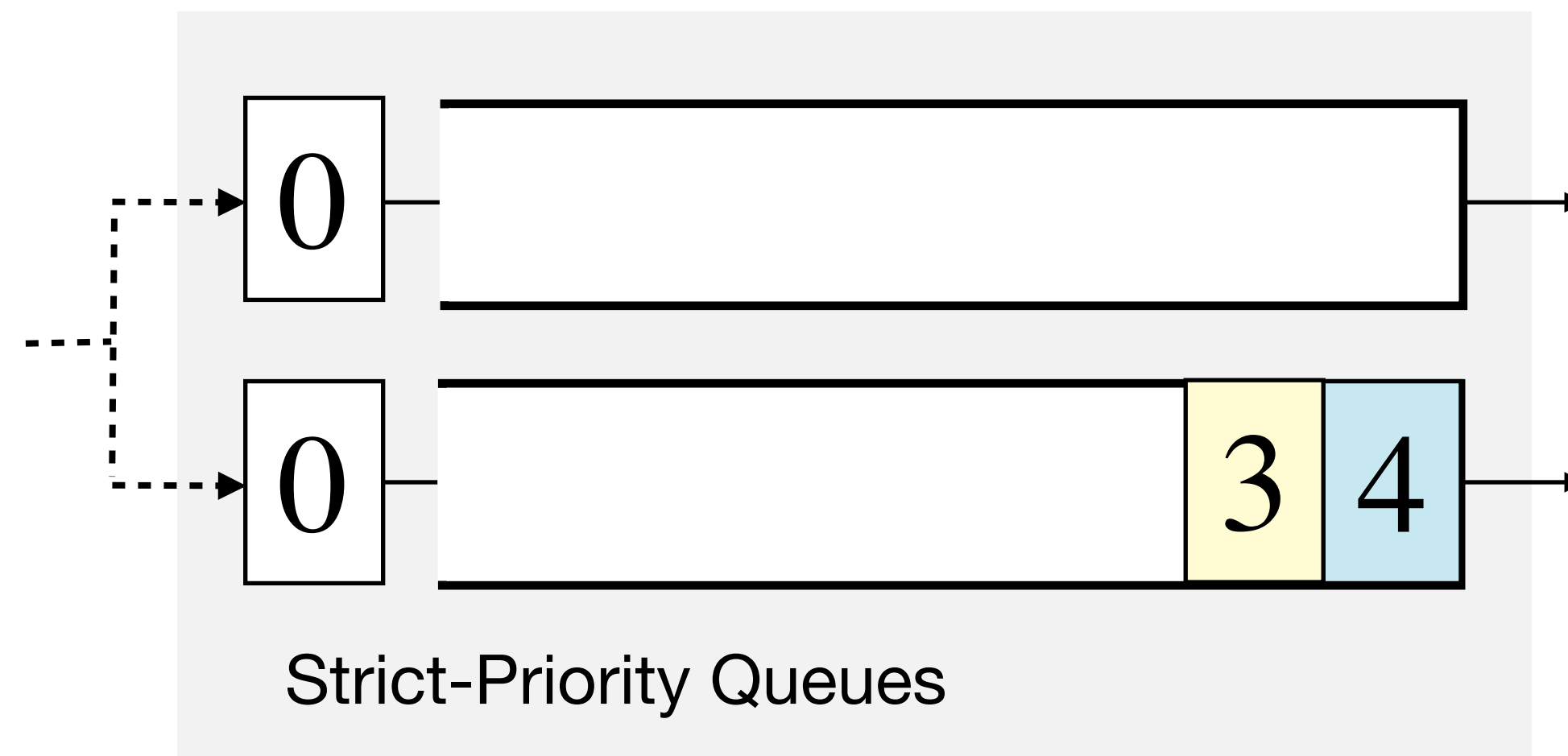
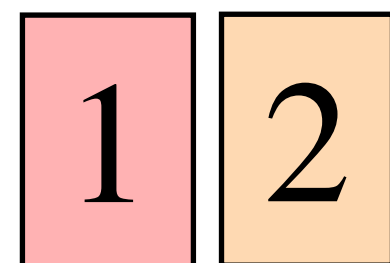
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Initialization

Zero traffic knowledge

Queue bounds set to zero

Input sequence



Output sequence

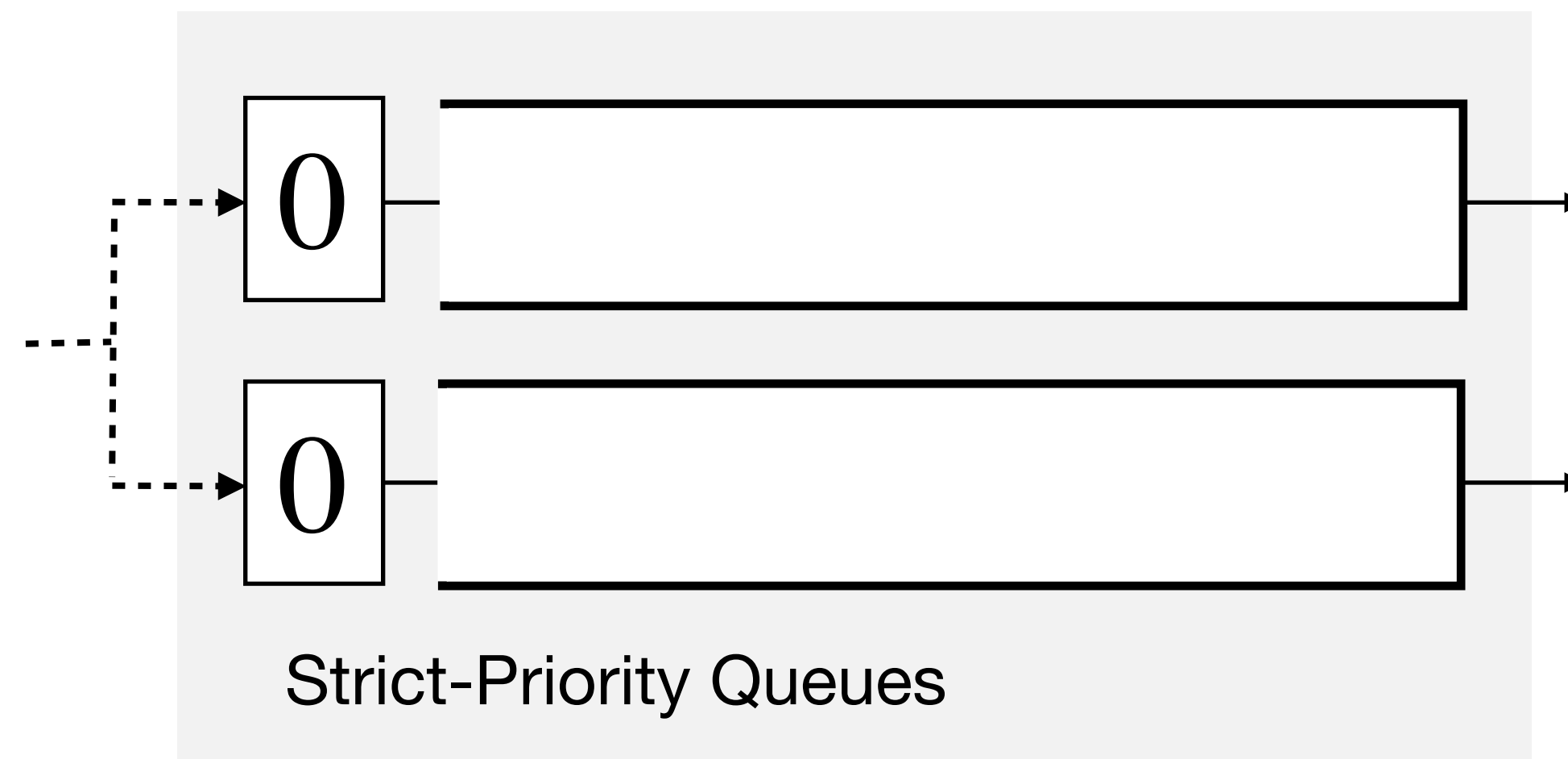
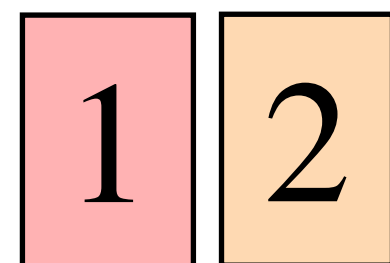
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Initialization

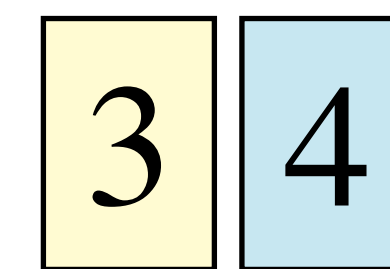
Zero traffic knowledge

Queue bounds set to zero

Input sequence



Output sequence



scheduling error

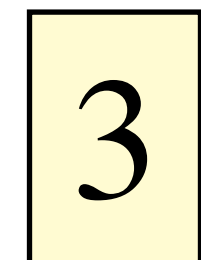
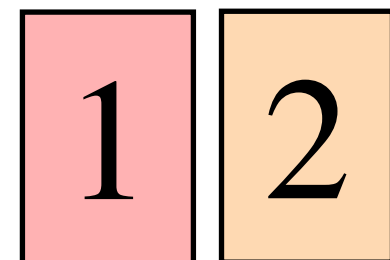
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Initialization

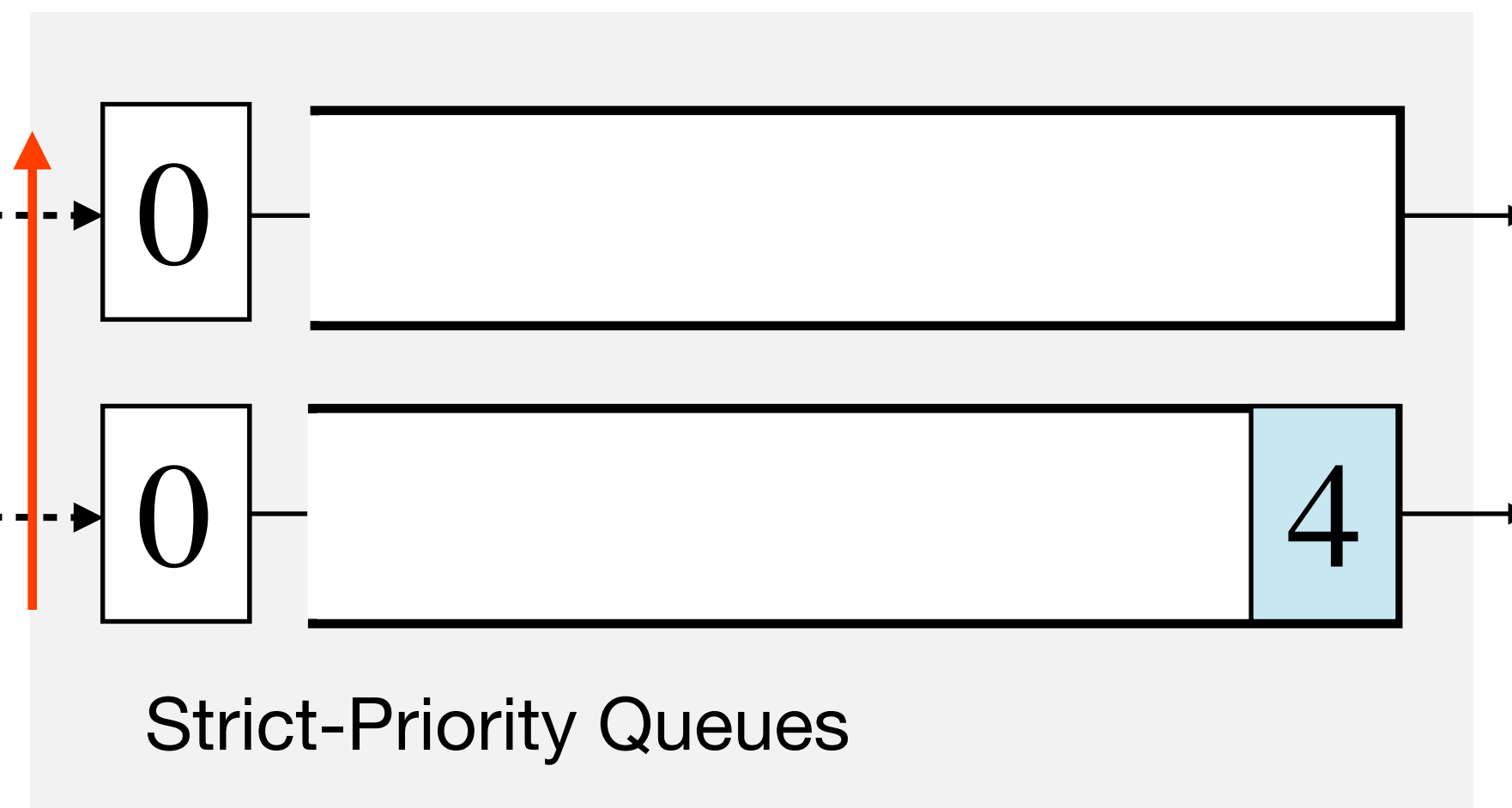
Zero traffic knowledge

Queue bounds set to zero

Input sequence



Rank  $\geq$  queue bound ?



Output sequence

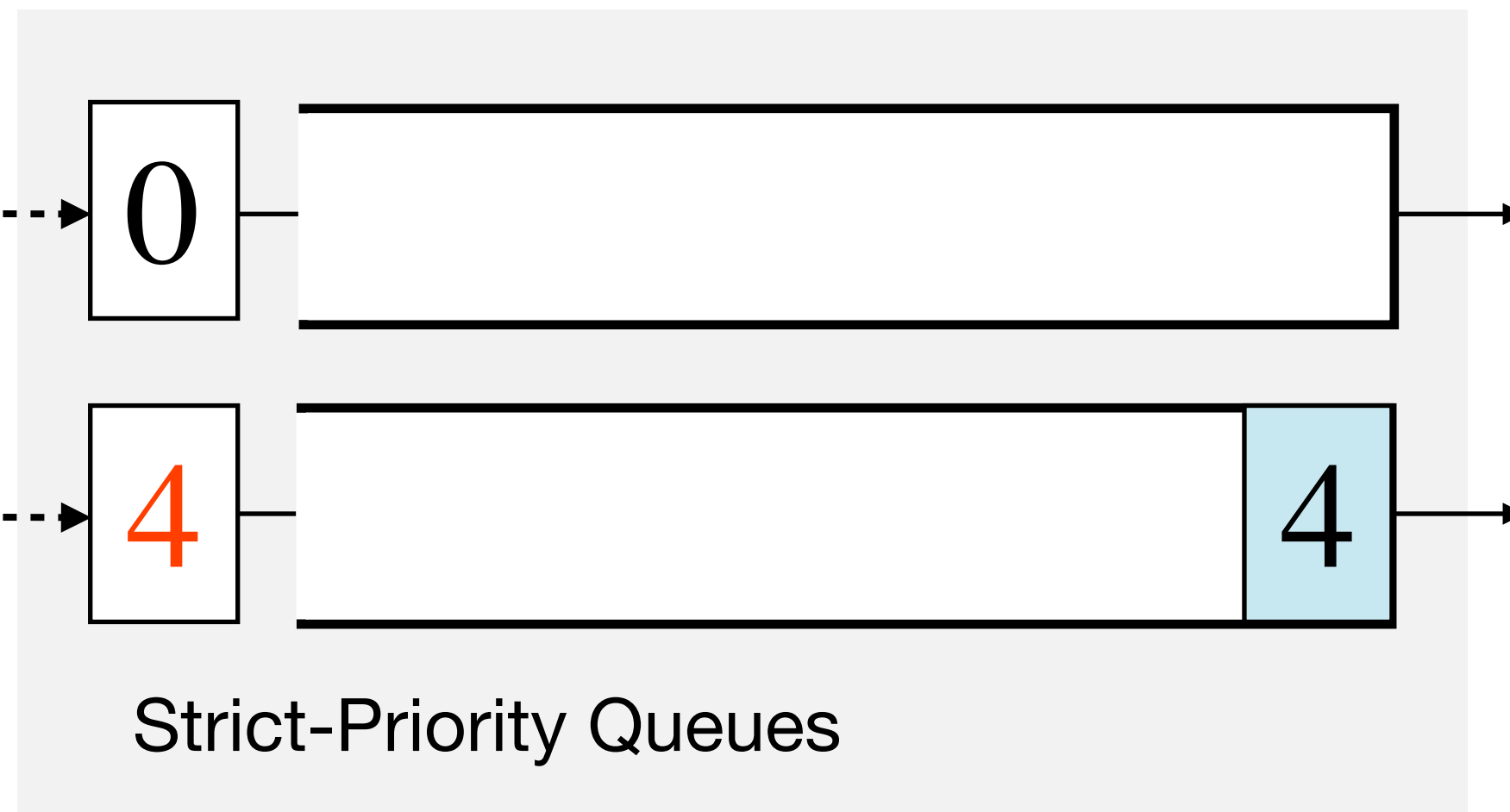
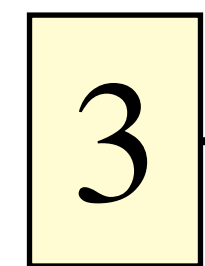
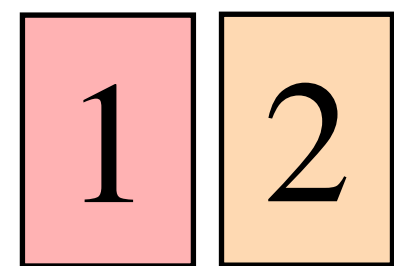
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues





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

Input sequence



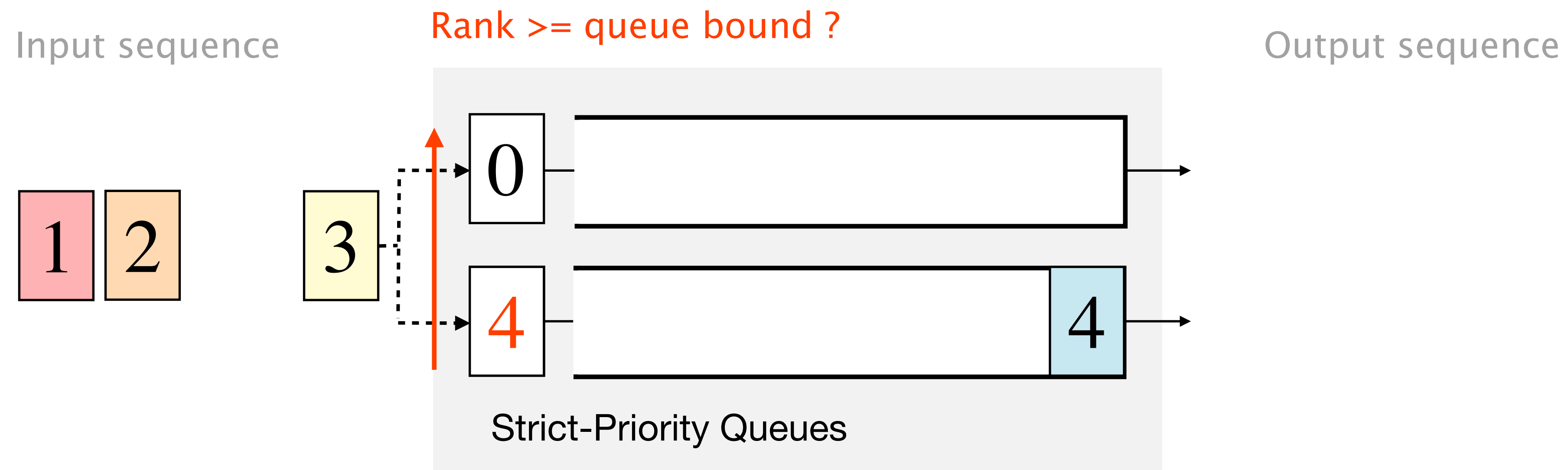
Output sequence

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

## Push-up

(future low-rank packets to higher-priority queues)

After enqueue, **queue bound** set to the **rank of the packet** enqueued



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

## Push-up

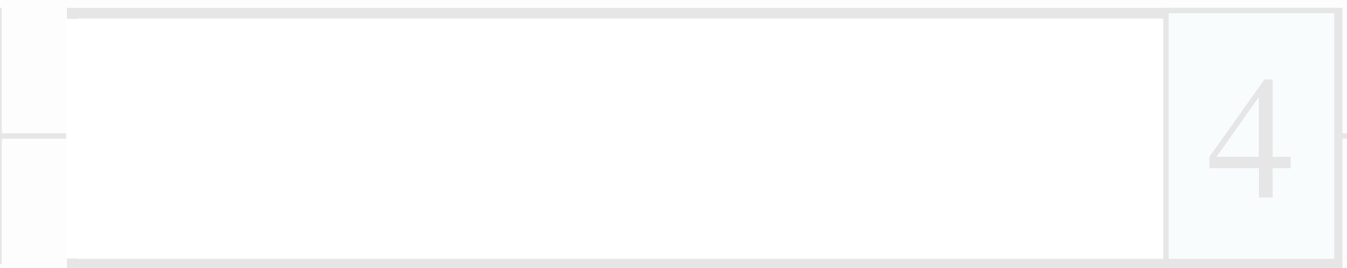
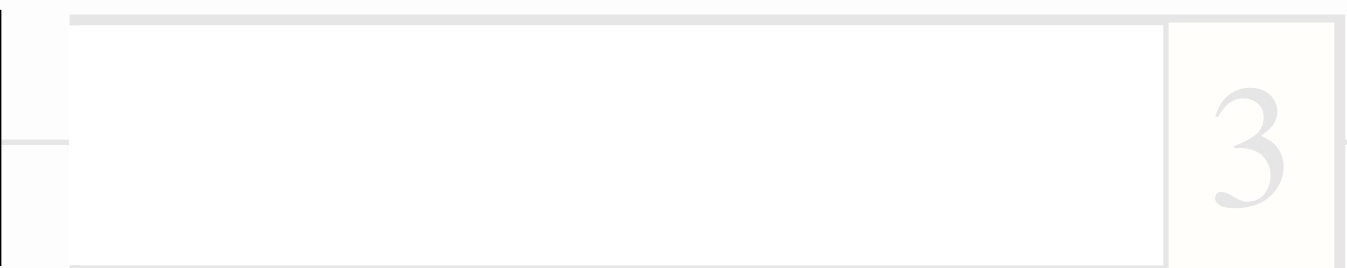
(future low-rank packets to higher-priority queues)

After enqueue, **queue bound** set to the **rank of the packet** enqueued

Input sequence



Push-up

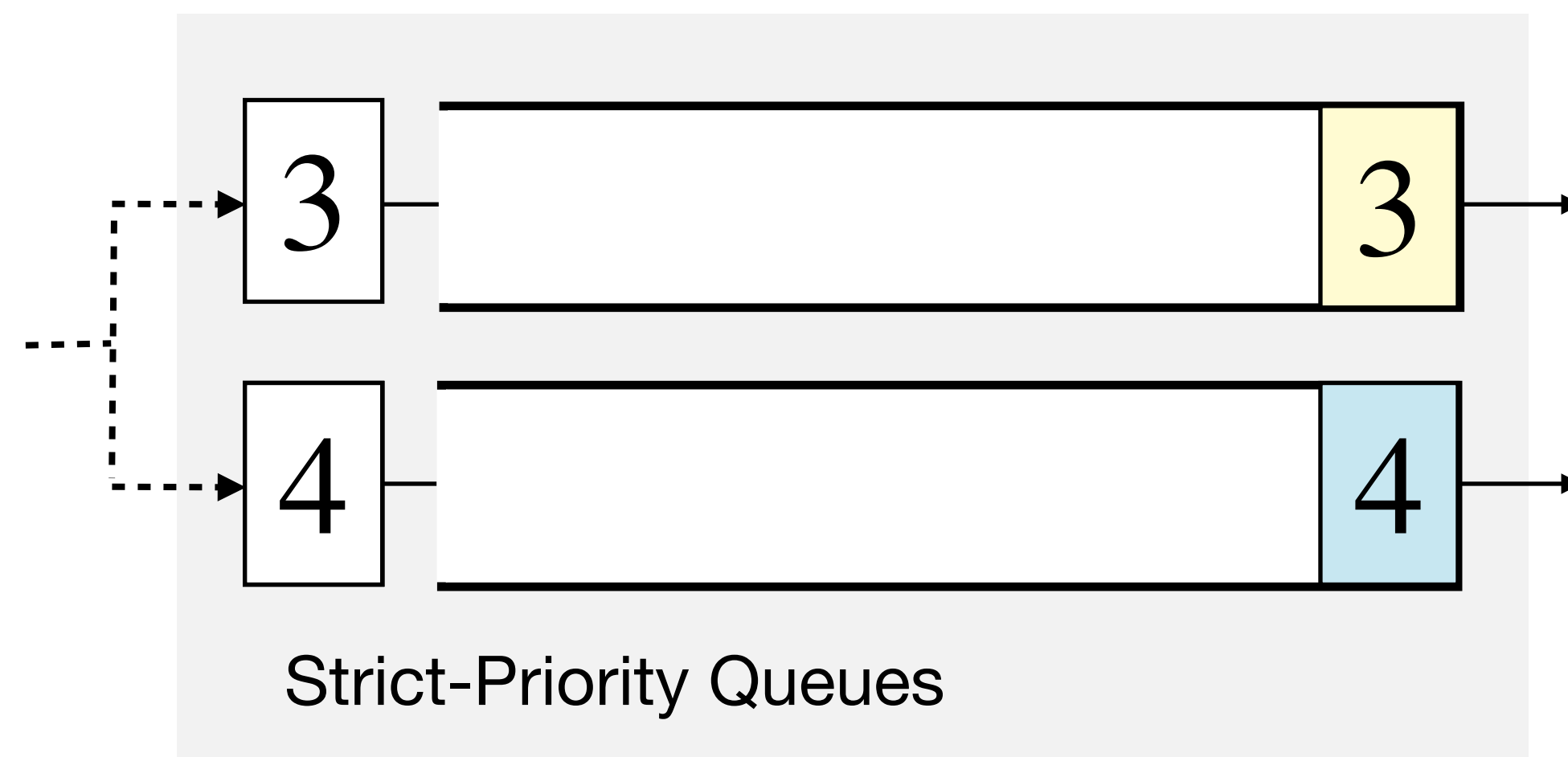
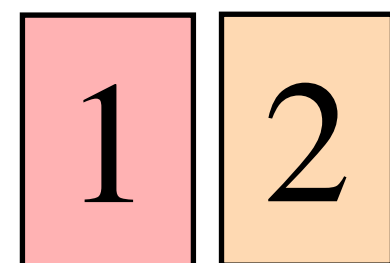


Strict-Priority Queues

Output sequence

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

Input sequence



Output sequence

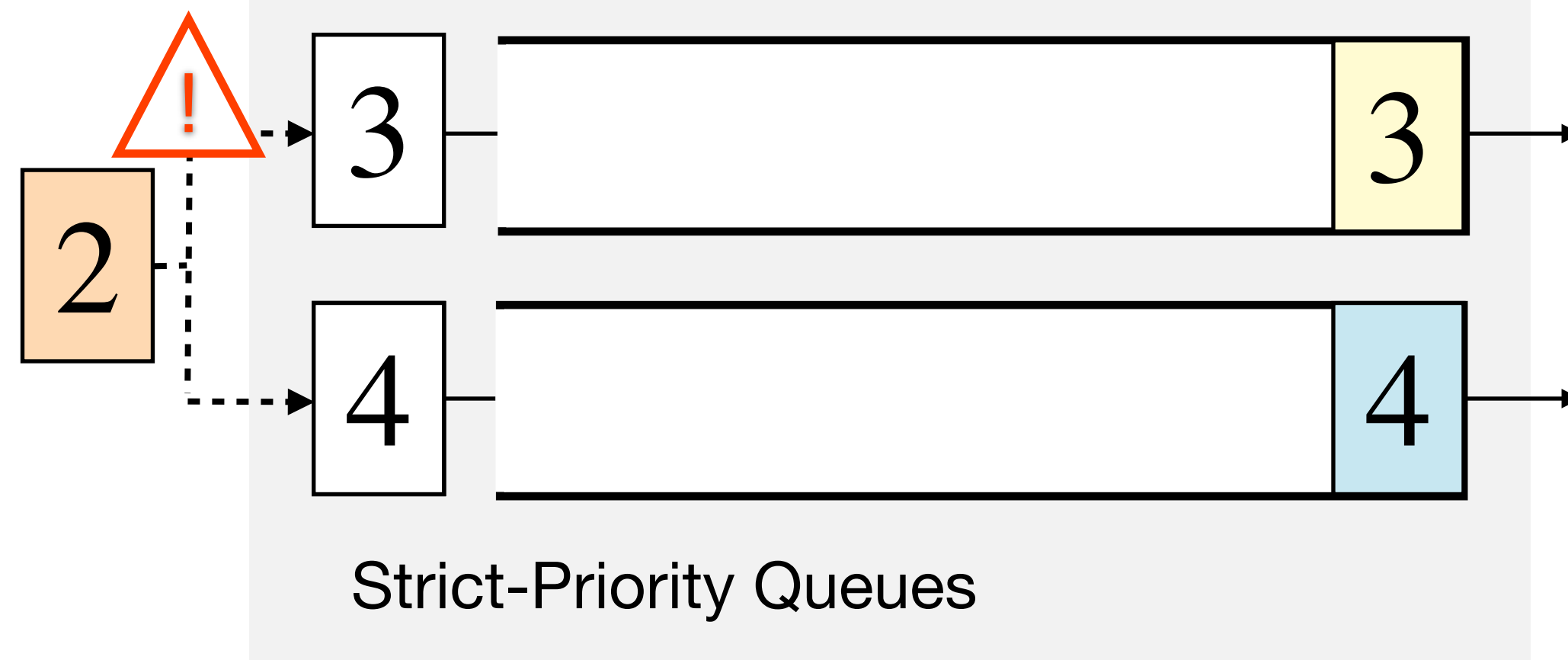
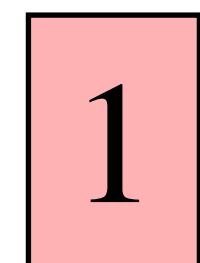
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Push-up

(future low-rank packets to higher-priority queues)

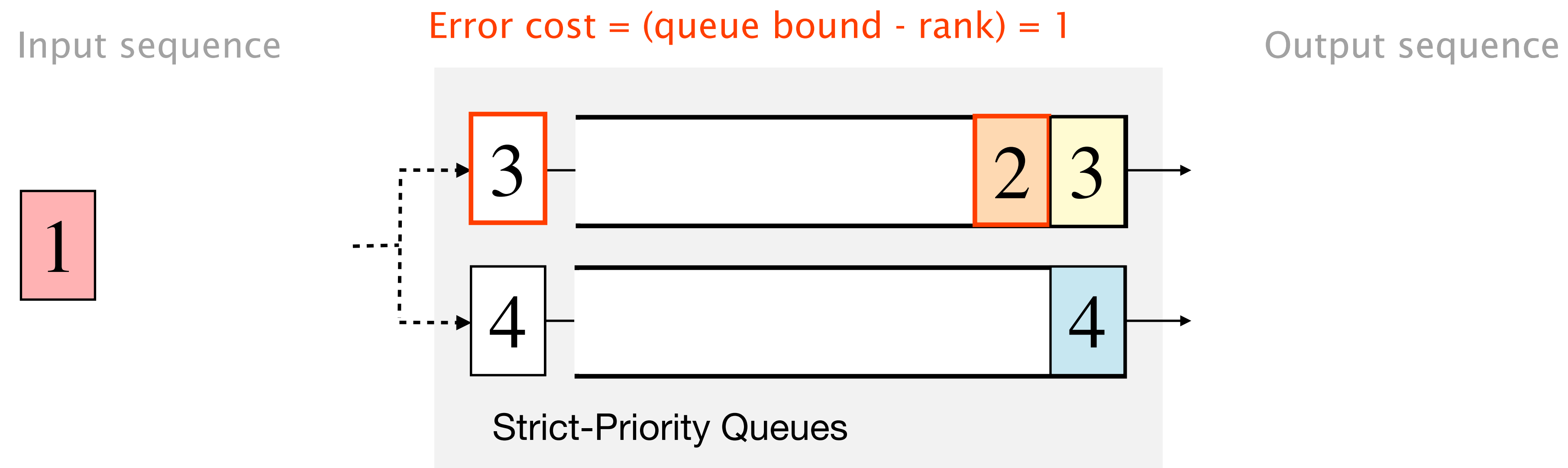
After enqueue, **queue bound** set to the **rank of the packet** enqueued

Input sequence

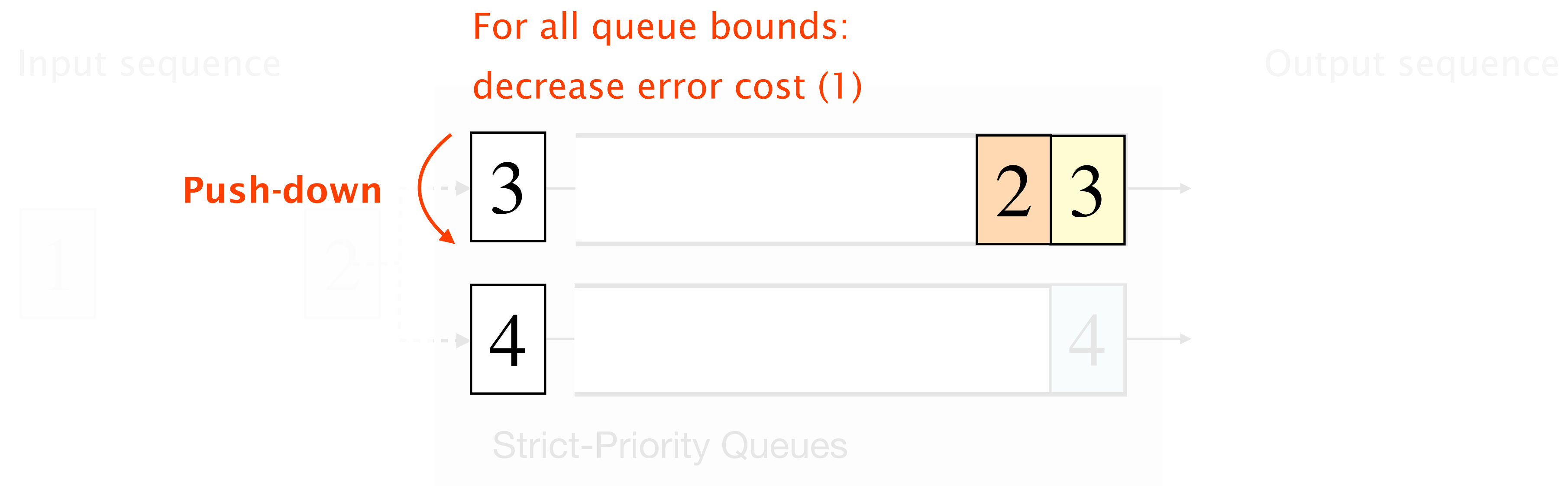


Output sequence

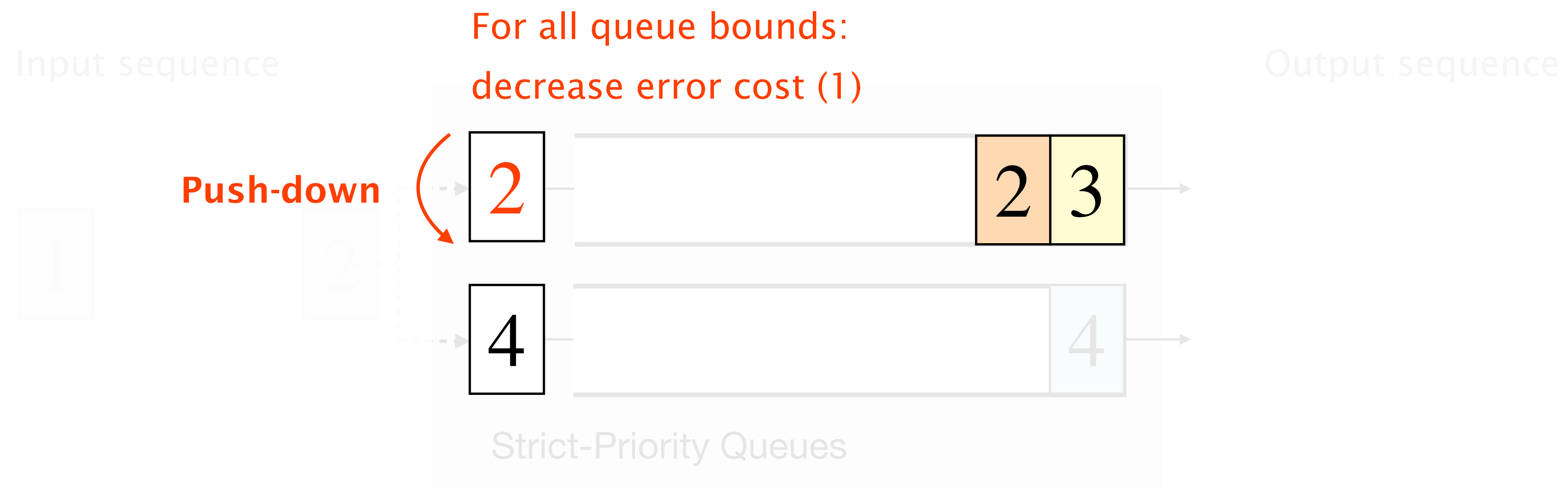
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues



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

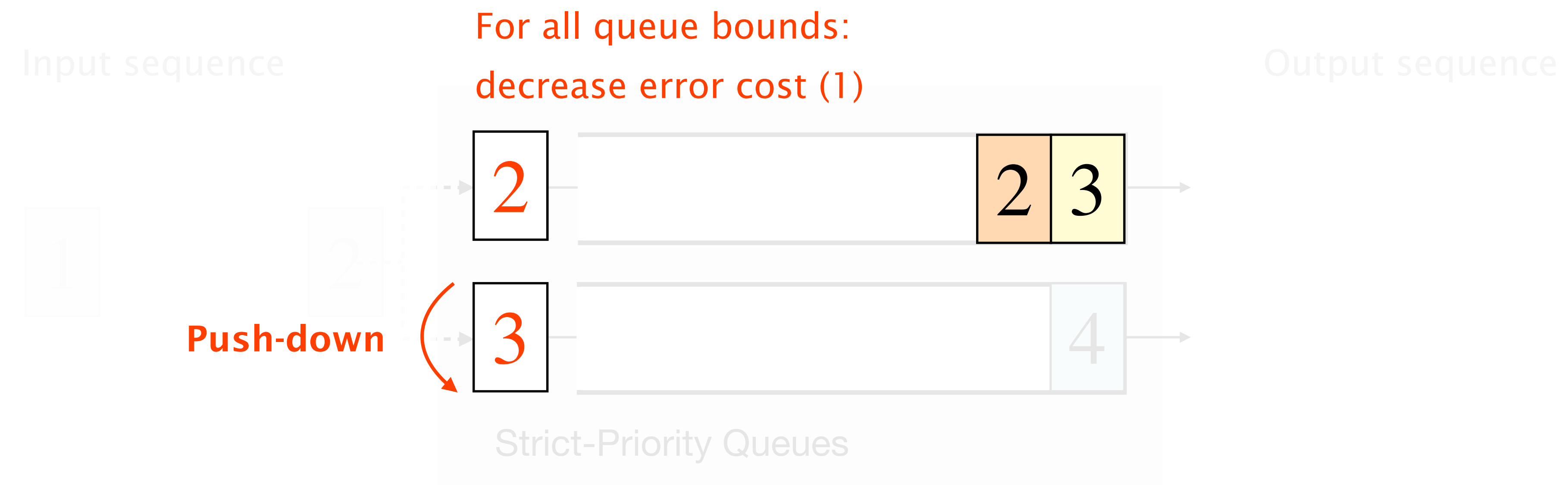


# SP-PIFO adapts the mapping of packet ranks to strict-priority queues





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



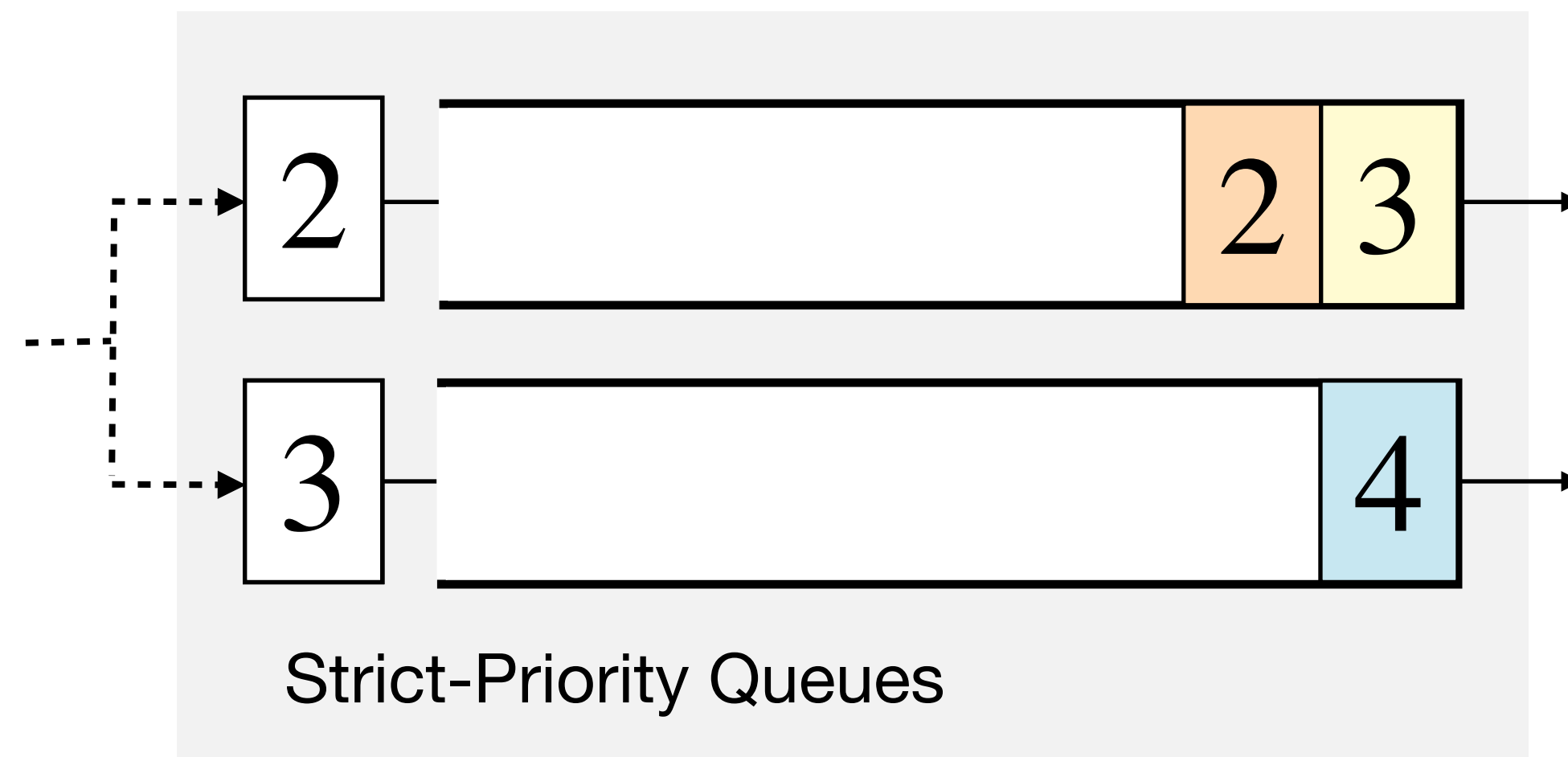
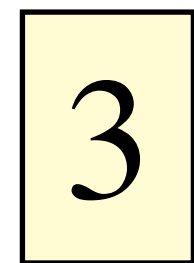
# SP-PIFO adapts the mapping of packet ranks to strict-priority queues

## Push-down

(future high-rank packets to lower-priority queues)

After potential error detected, all **queue bounds decreased** the **error cost**

Input sequence

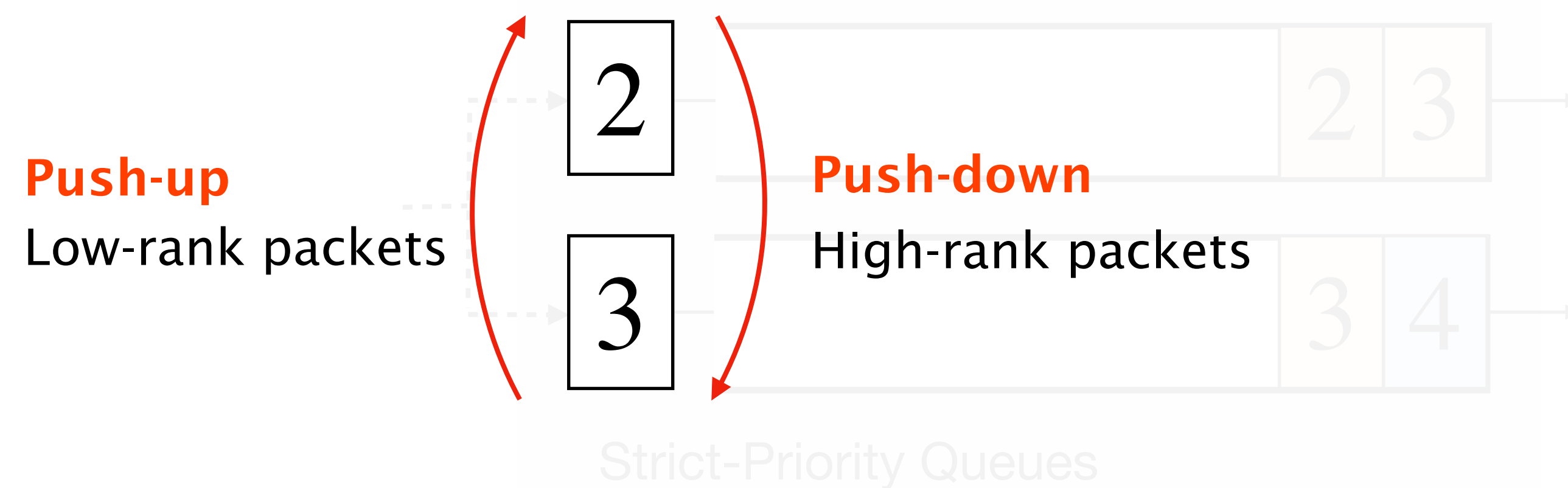


Output sequence

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

**Objective** Find optimal queue bounds  $q^*$   
That minimize the expected loss  $U$  for all ranks

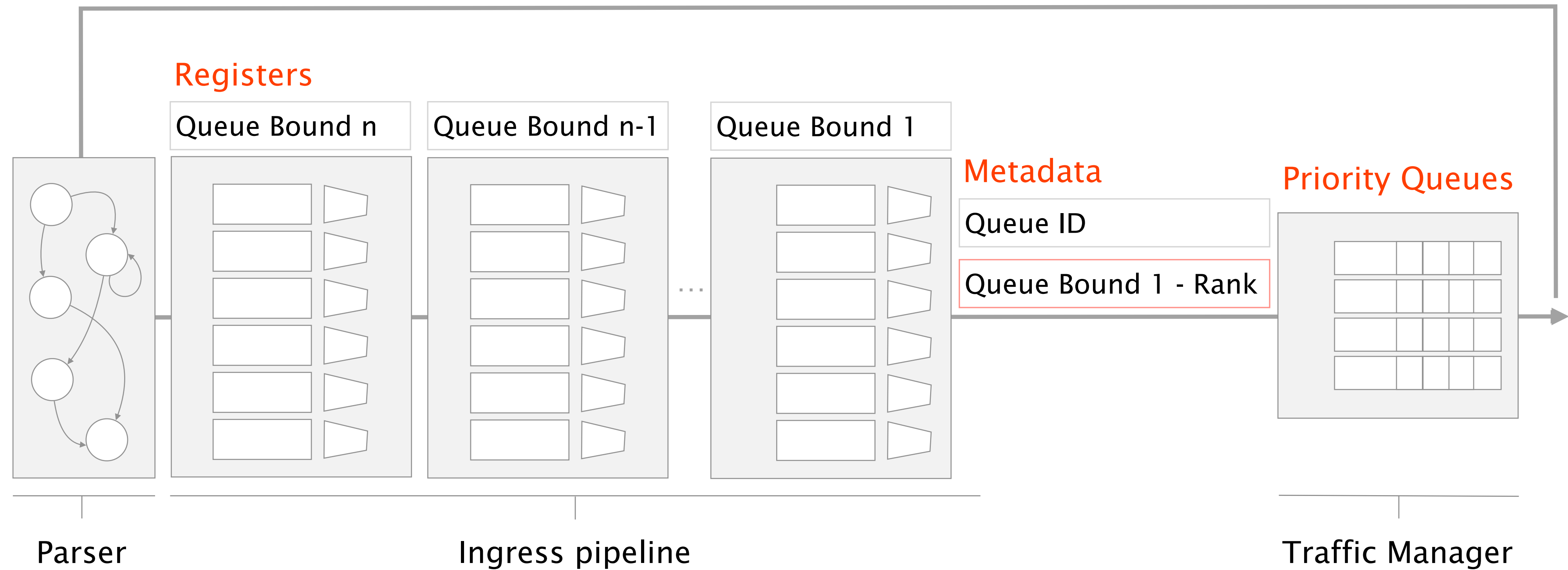
**Result** Packet-level adaptation of  $q$



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

- 1      **Adaptation design**  
How does it work
- 2      **Implementation**  
How can it be deployed
- 3      **Evaluation**  
How well does it perform

# SP-PIFO has been fully implemented on Barefoot Tofino



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

- 1      **Adaptation design**  
How does it work
- 2      **Implementation**  
How can it be deployed
- 3      **Evaluation**  
How well does it perform

# Evaluation

## Question

How well does SP-PIFO approximate well-known scheduling objectives under realistic traffic workloads?

## Scheduling objectives

Minimizing Flow Completion Time

pFabric\* (8 queues)

Ranks are set to the **remaining flow size**

Max-min fairness

Start-Time Fair Queuing (32 queues)

Ranks based on a **fluid model**

\* without starvation prevention

# Methodology

## Packet-level simulator

We integrated SP-PIFO in Netbench  
[SIGCOMM 2017]

## Topology

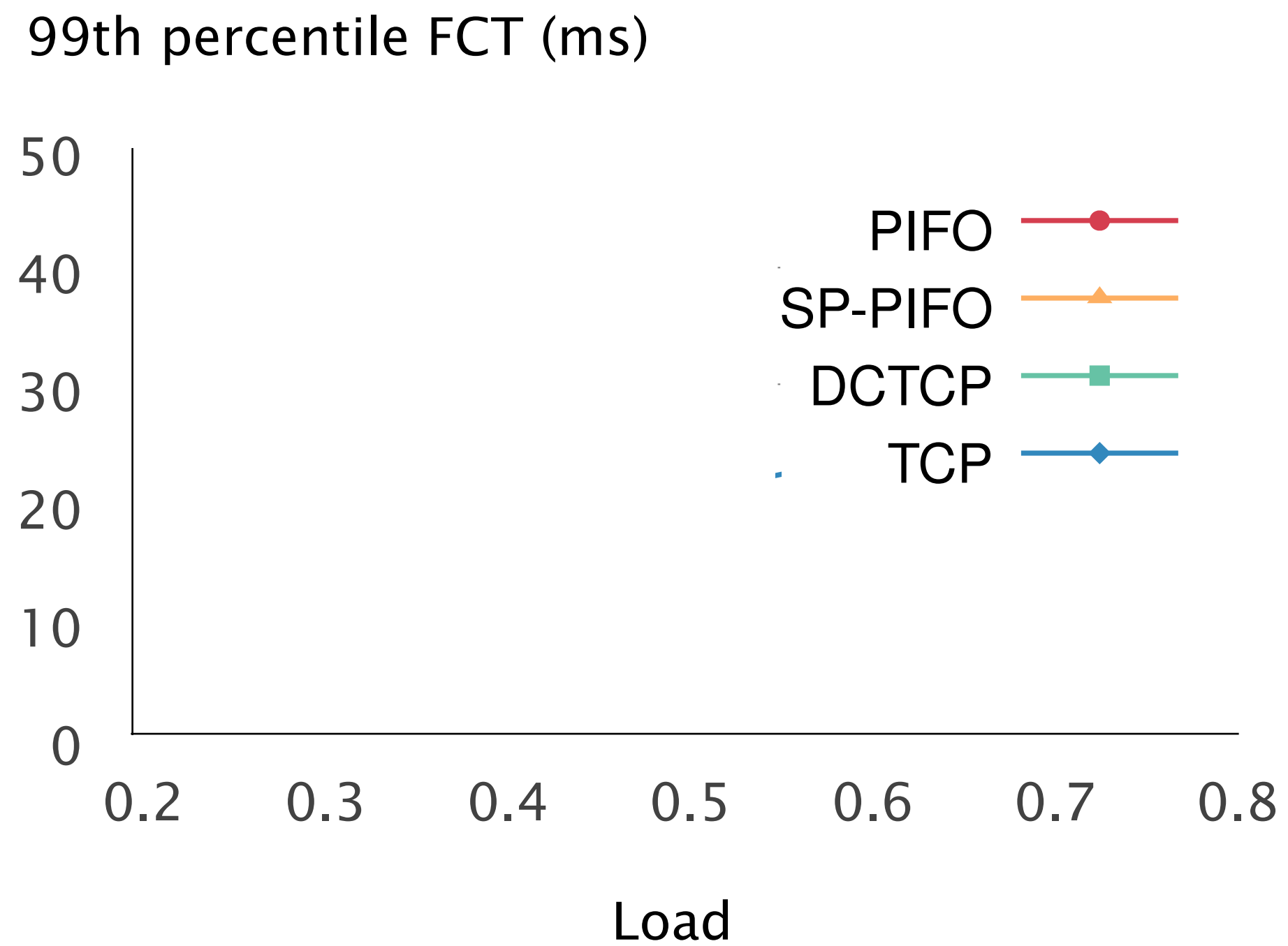
We use a leaf-spine topology with 144 servers,  
links of 1Gbps and 4Gbps

## Realistic workloads

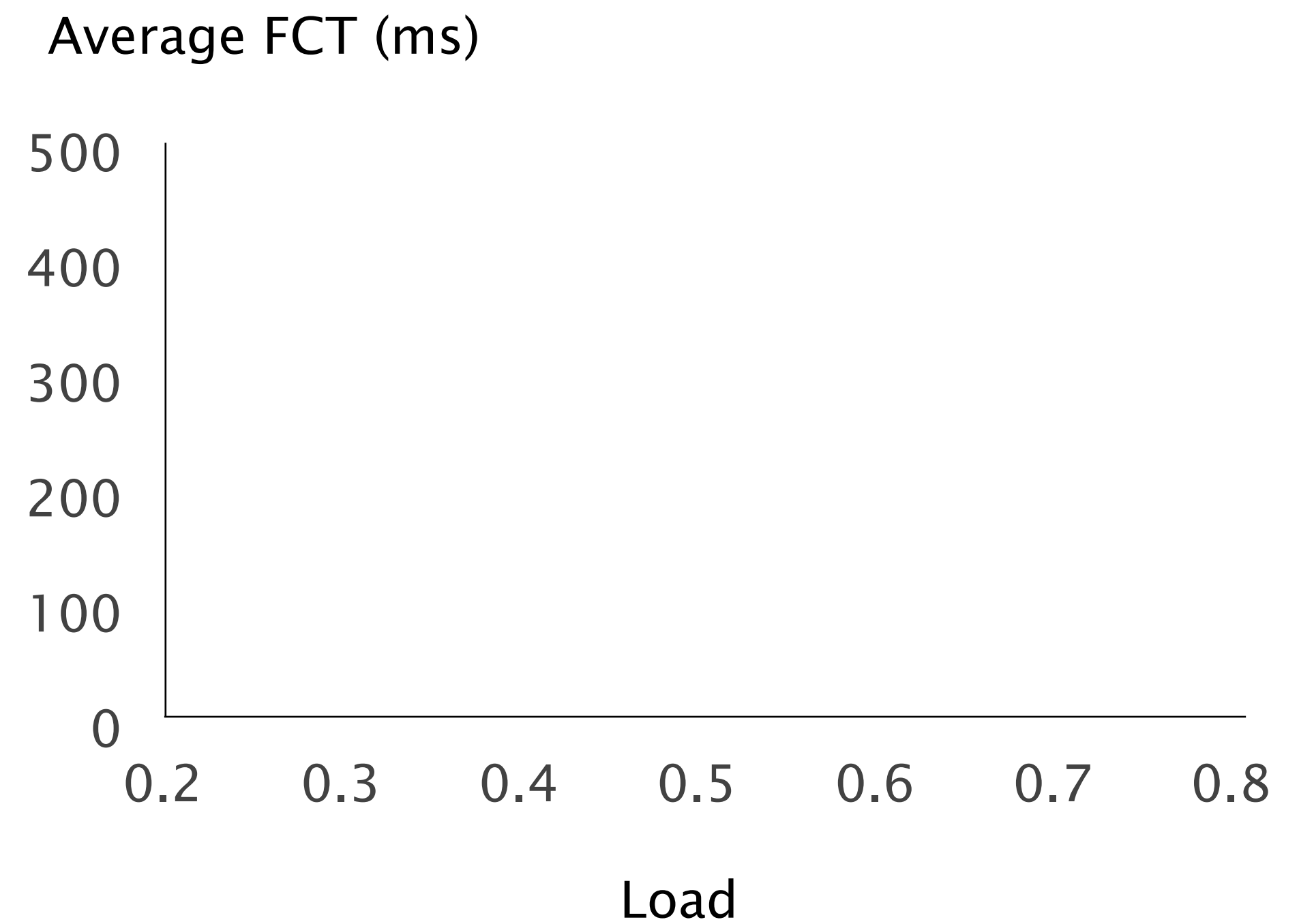
We generate traffic following pFabric  
web-search workload



# SP-PIFO closely approximates pFabric, minimizing FCTs for both small and big flows



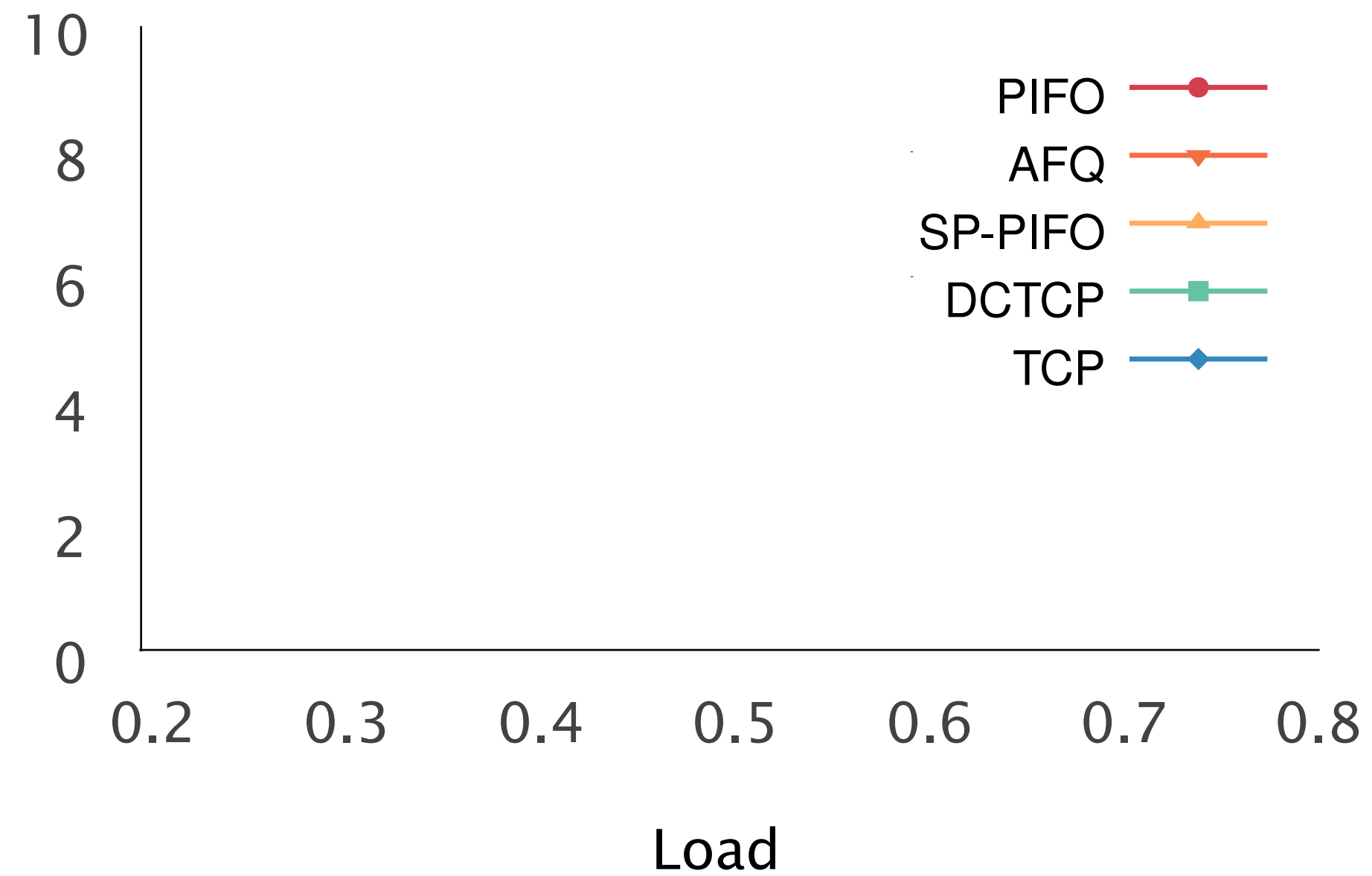
Small flows <100KB



Big flows  $\geq 1$  MB

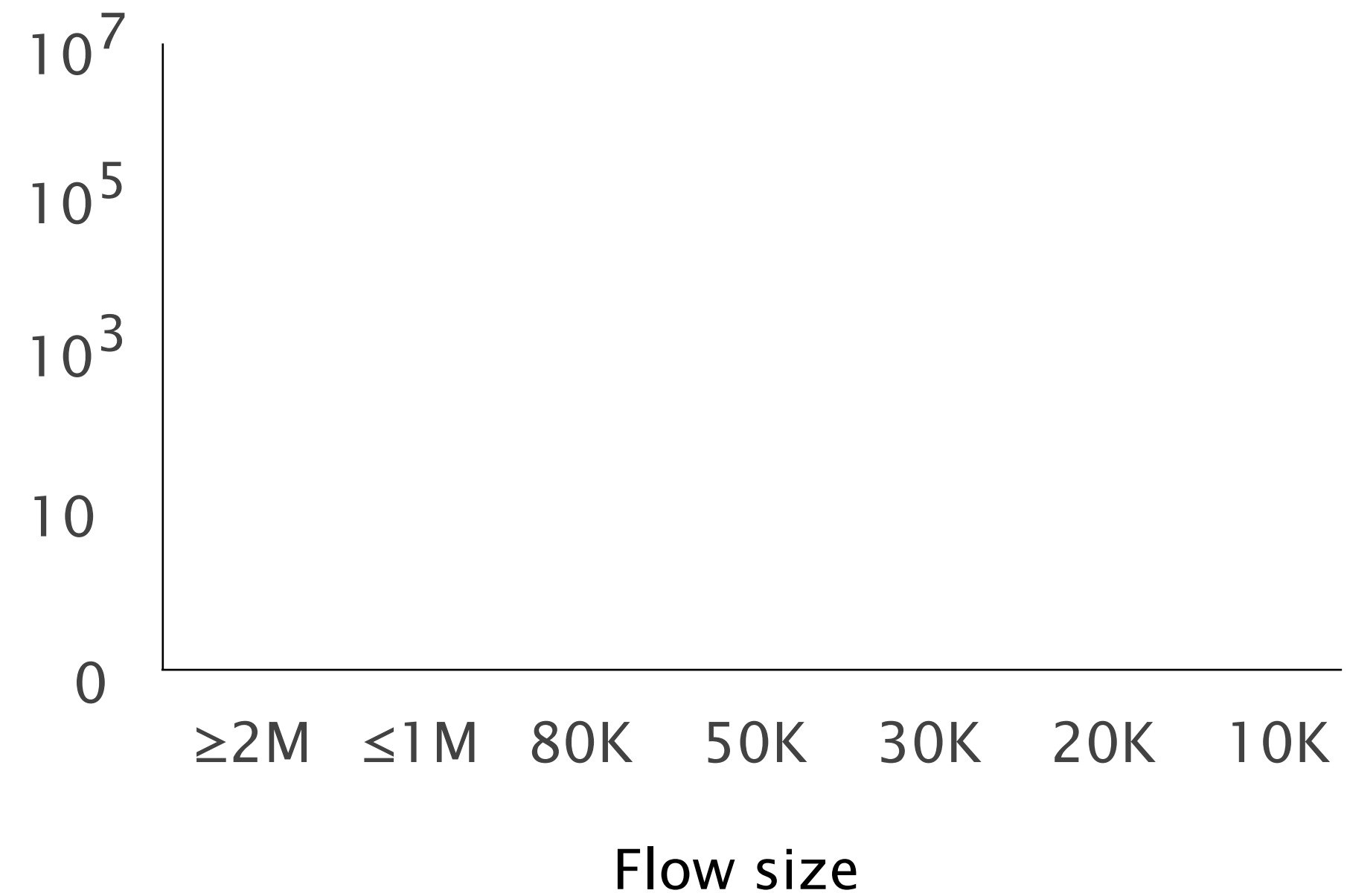
# SP-PIFO closely approximates state-of-the-art fair-queuing algorithms

Average FCT (ms)



Small flows <100KB

Average FCT (ms)



All flows @ Load 0.7

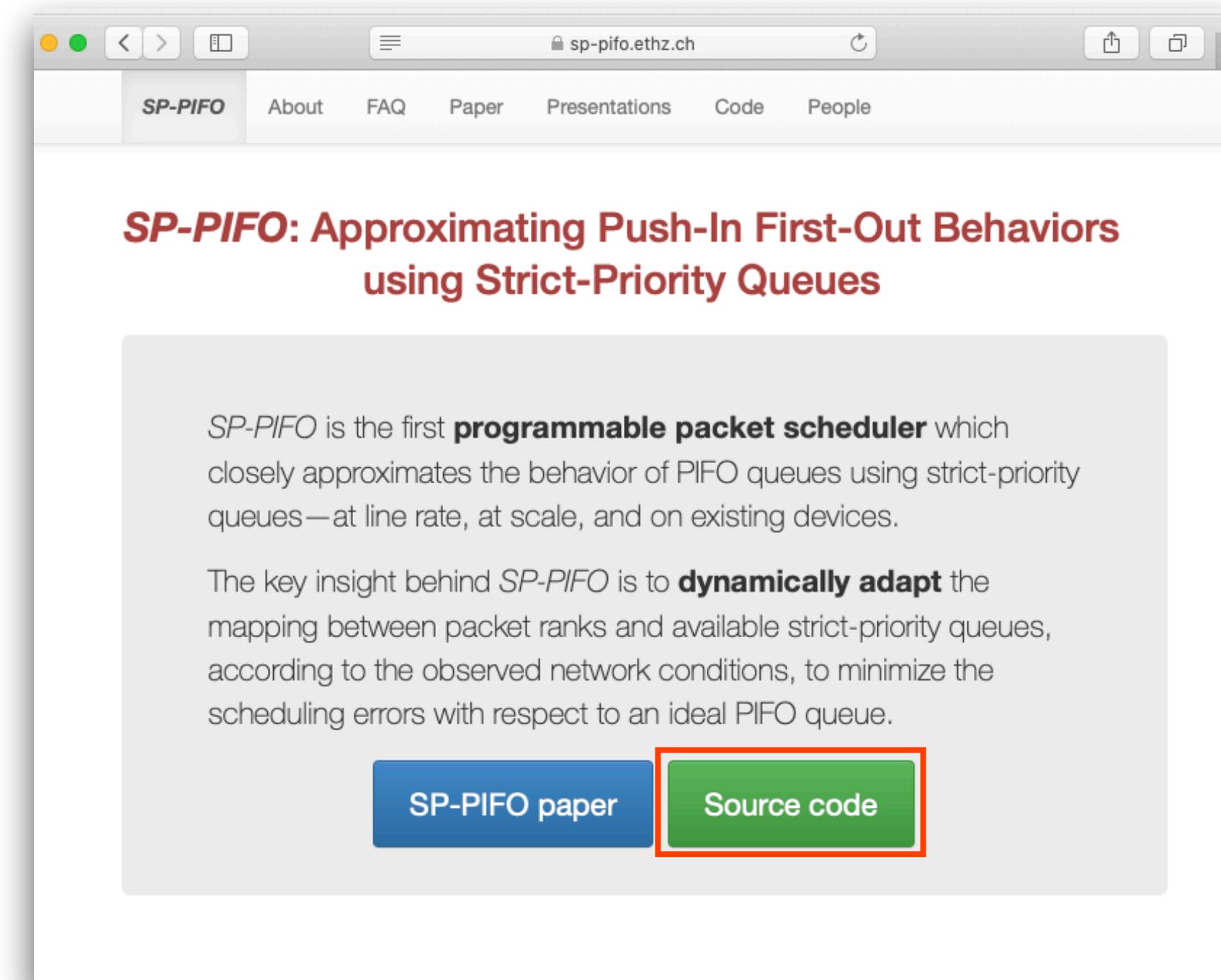
# Check our website!

[sp-pifo.ethz.ch](http://sp-pifo.ethz.ch)

SP-PIFO characterization,  
comparison with gradient

Hardware evaluation on  
Barefoot Tofino

Limitations and  
future improvements



All the code is available  
All our experiments are reproducible

# SP-PIFO: Making scheduling programmable, today!

SP-PIFO approximates the behavior of PIFO queues at line rate, at scale and on existing devices

It adapts the mapping between packet ranks and strict-priority queues to minimize the scheduling errors

It reacts per-packet to traffic variations, without traffic knowledge required

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

[sp-pifo.ethz.ch](http://sp-pifo.ethz.ch)



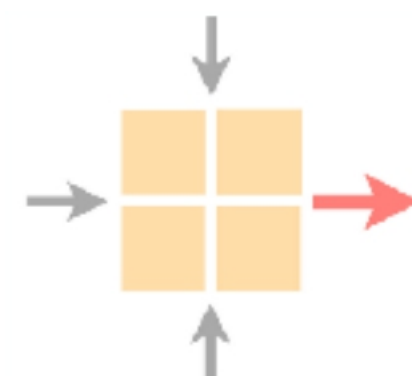
Albert Gran Alcoz



Alexander Dietmüller



Laurent Vanbever



Networked Systems  
ETH Zürich — seit 2015

**ETH** zürich