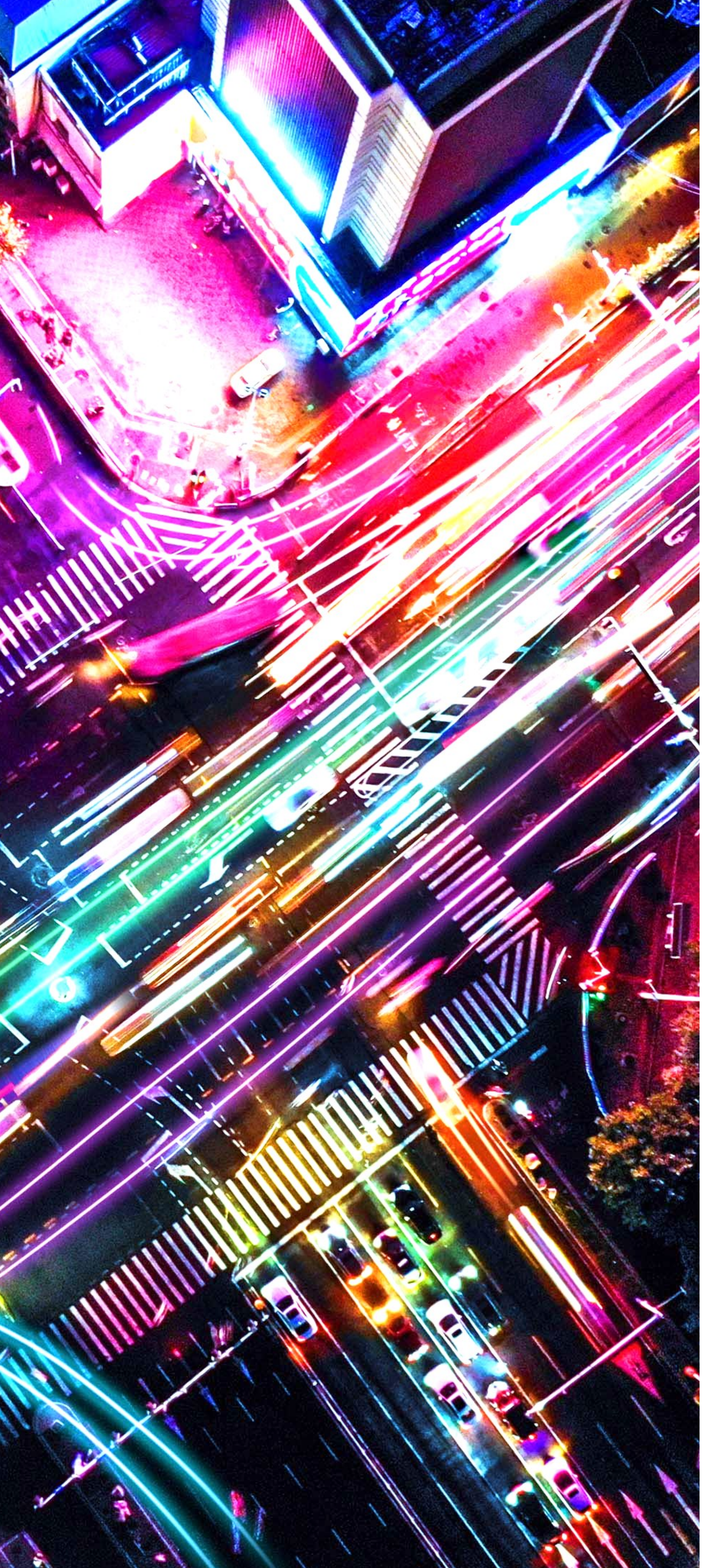


In-Network Congestion Management for Security and Performance

Albert Gran Alcoz

nsg.ee.ethz.ch

Cloudflare

November 2023

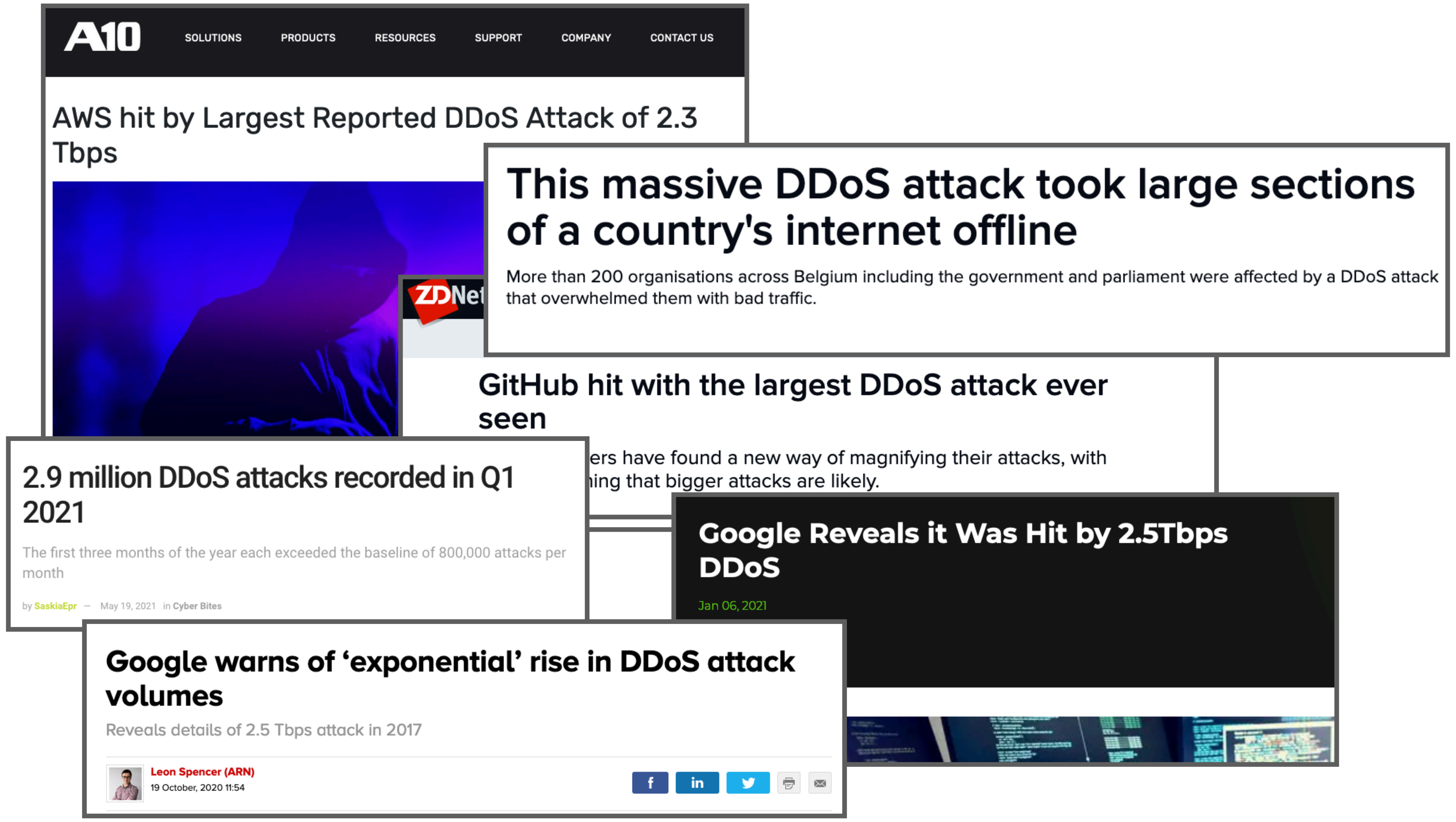
Why do we care about congestion?

Why do we care about congestion?

Without congestion management, the Internet collapses

Why do we care about congestion?

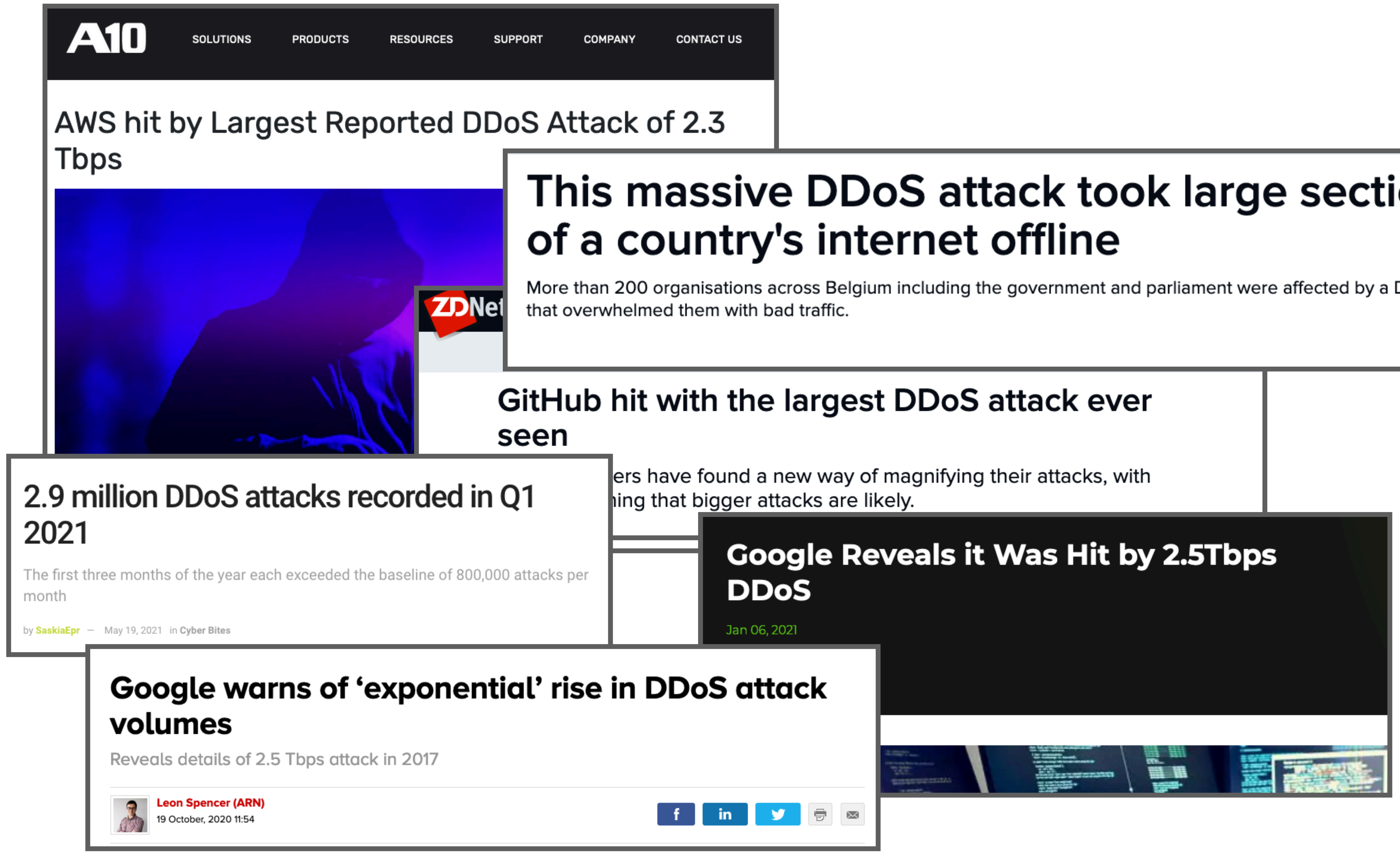
Without congestion management, the Internet collapses



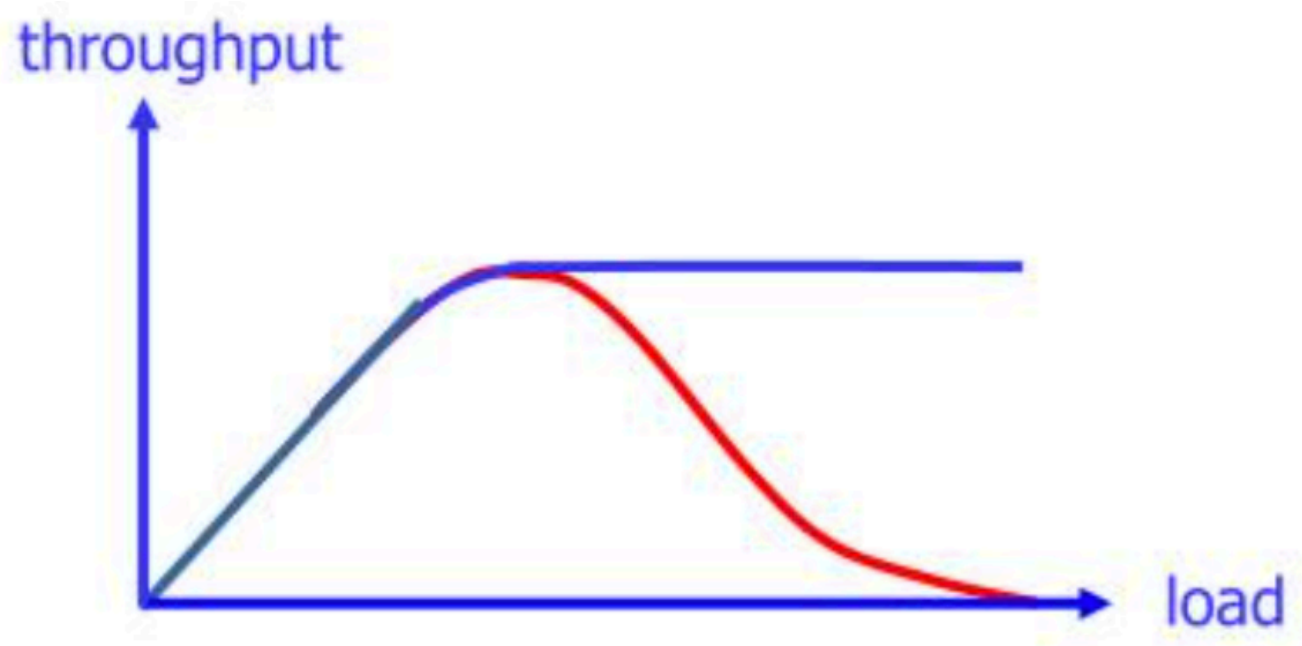
DDoS attacks

Why do we care about congestion?

Without congestion management, the Internet collapses



DDoS attacks



Congestion collapse (1986)

Congestion management is more than congestion control

Flow Control

Congestion Control

Host CC

Admission Control

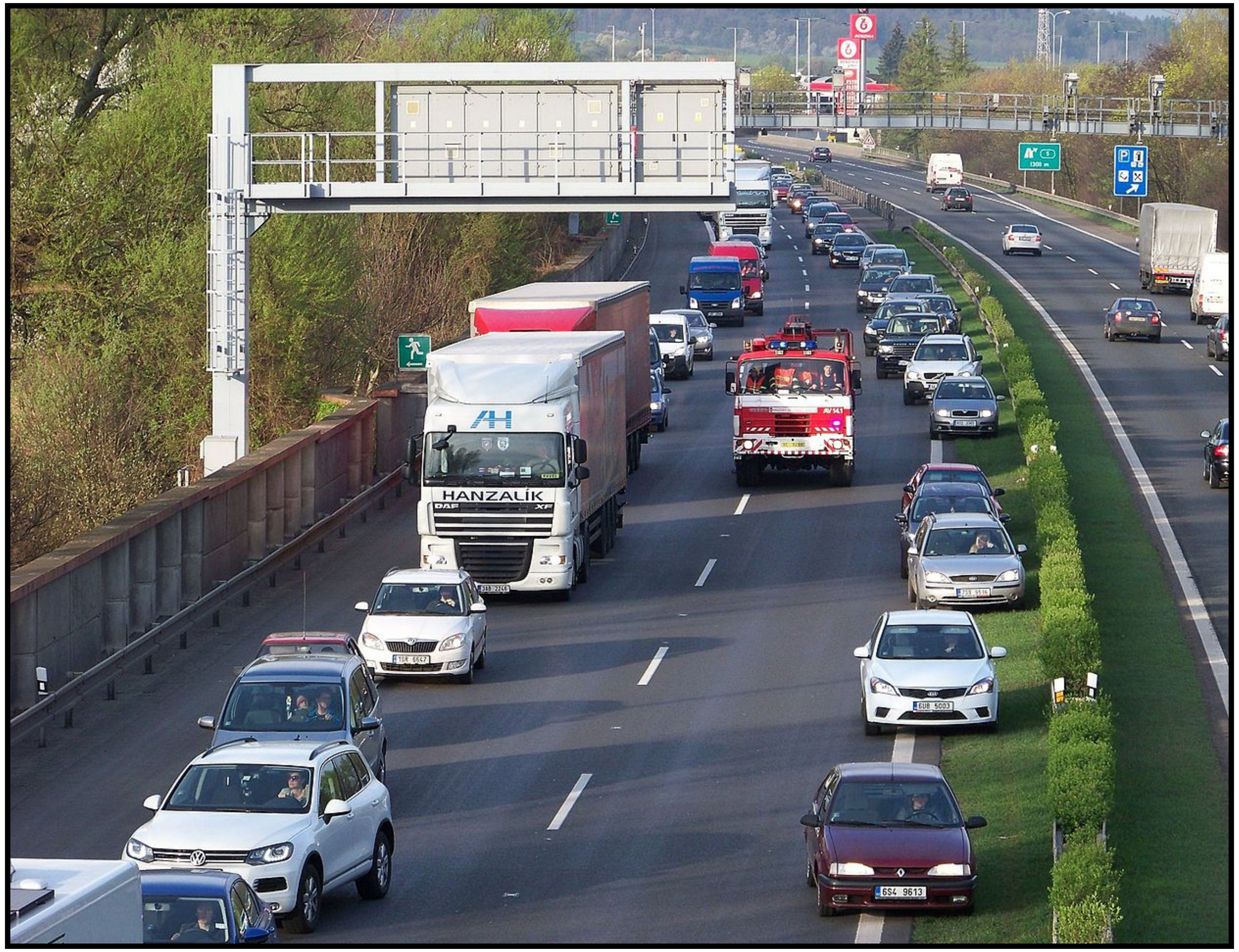
Packet Scheduling

Active Queue Management

Buffer Management


EXIT-2
गाजीपुर फल एवं
सब्जी मार्केट
Ghazipur Fruit &
Vegetable Market
मेरठ
Meerut
वैशाली
Vaishali
NORMAL SPEED MEETS EVERY NEED


EXIT-2
गाजीपुर फल एवं सब्जी मार्केट
Ghazipur Fruit & Vegetable Market
मेरठ Meerut
वैशाली Vaishali
NORMAL SPEED MEETS EVERY NEED


HANZALIK

SP-PIFO: Programmable Scheduling, Today — NSDI ‘20

ACC-Turbo: Mitigating Pulse-wave DDoS with Programmable Scheduling — SIGCOMM ‘22

QVISOR: Virtualizing Scheduling Policies — HotNets ‘23

SP-PIFO: Programmable Scheduling, Today | NSDI ‘20

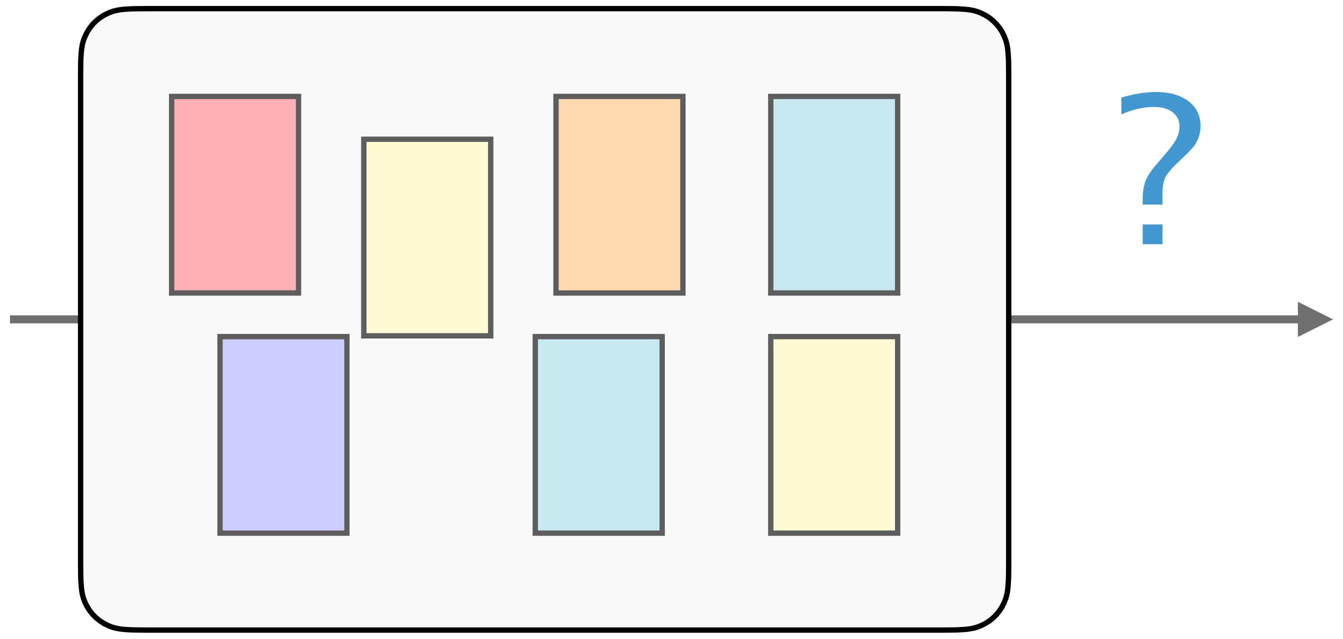
ACC-Turbo: Mitigating Pulse-wave DDoS with Programmable Scheduling | SIGCOMM ‘22

QVISOR: Virtualizing Scheduling Policies | HotNets ‘23

Packet scheduling

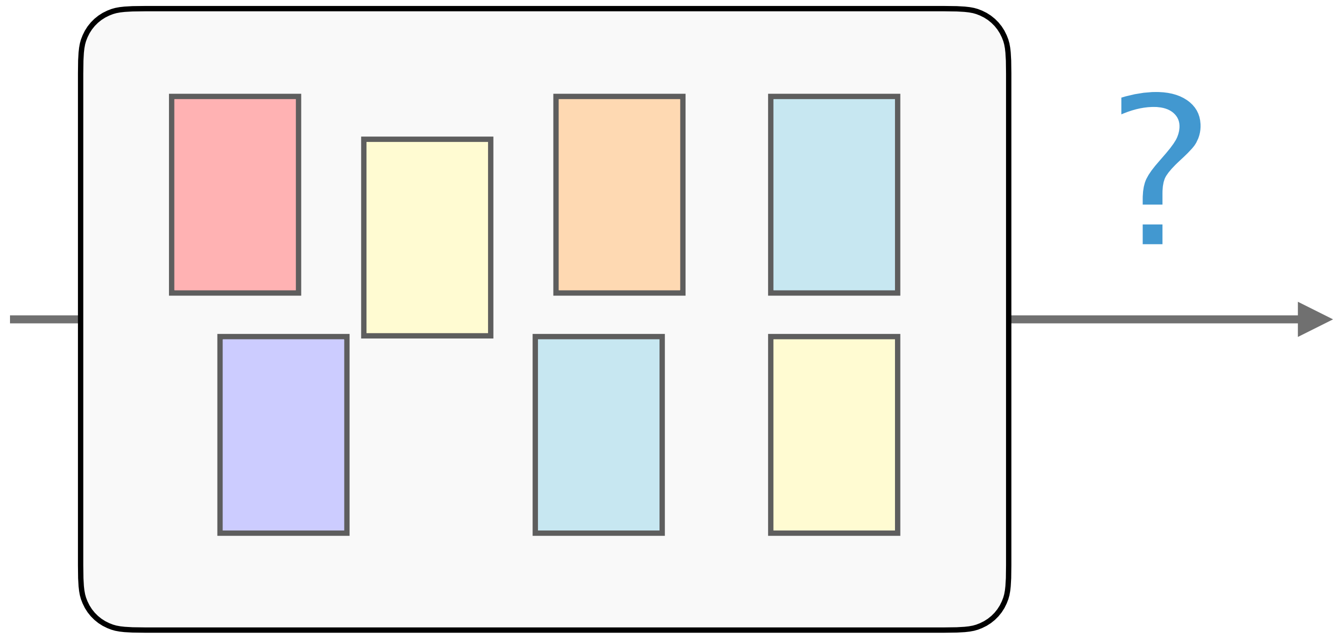
What packet next

and *when?*



Packet scheduling

What packet next and *when?*



Minimize tail latency FIFO+

Prioritize packets with higher queuing time

Minimize FCTs SRPT, PIAS, pFabric

Prioritize packets from short flows

Enforce fairness WRR, (S)FQ, WFQ

One packets from each class at a time

Quite unfortunately…

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 mechanisms to achieve fairness [16, 29, 32], to schemes that help reduce tail latency [15] or flow completion time [7], and this short list barely scratches the surface of past and

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 packet schedulers so that such algorithms could be more easily deployed (as argued in [33]; in fact, it was the eloquent argument in this paper that caused us to initially ask the question about universality).

Quite unfortunately…

a universal scheduling algorithm does *not* exist

"You can't have *everything* you want,

but you can have *anything* you want"

Generality

Universal packet scheduler

Flexibility

Customized algorithms

Quite unfortunately…

a universal scheduling algorithm does *not* exist

“You can’t have *everything* you want,

but you can have *anything* you want”

Generality

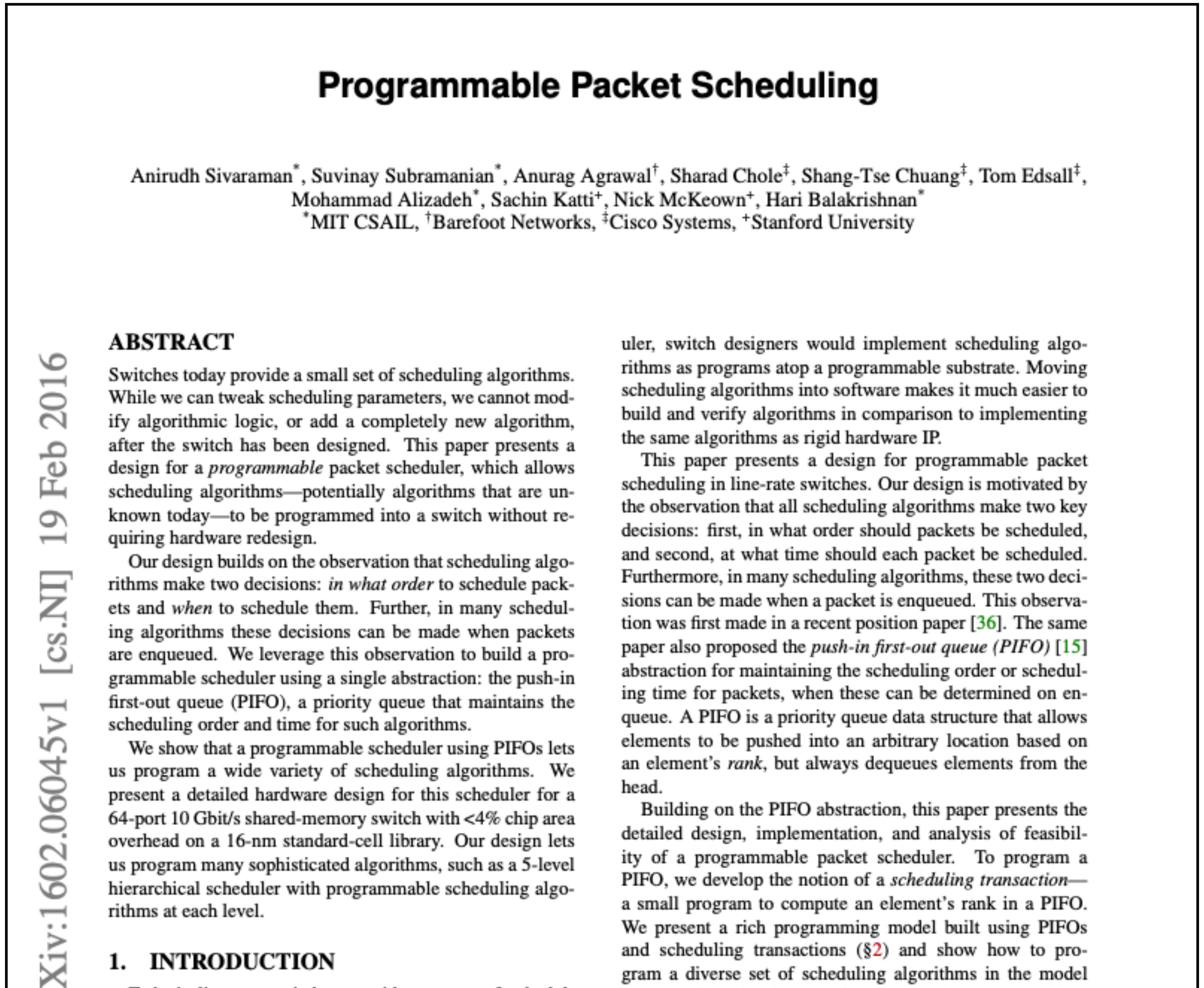
Universal packet scheduler

Programmable

Scheduling

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

SIGCOMM'16



Programmable Packet Scheduling

Anirudh Sivaraman*, Suvinay Subramanian*, Anurag Agrawal†, Sharad Chole‡, Shang-Tse Chuang‡, Tom Edsall‡, Mohammad Alizadeh*, Sachin Katti+, Nick McKeown+, Hari Balakrishnan*
*MIT CSAIL, †Barefoot Networks, ‡Cisco Systems, +Stanford University

ABSTRACT

Switches today provide a small set of scheduling algorithms. While we can tweak scheduling parameters, we cannot modify algorithmic logic, or add a completely new algorithm, after the switch has been designed. This paper presents a design for a *programmable* packet scheduler, which allows scheduling algorithms—potentially algorithms that are unknown today—to be programmed into a switch without requiring hardware redesign.

Our design builds on the observation that scheduling algorithms make two decisions: *in what order* to schedule packets and *when* to schedule them. Further, in many scheduling algorithms these decisions can be made when packets are enqueued. We leverage this observation to build a programmable scheduler using a single abstraction: the push-in first-out queue (PIFO), a priority queue that maintains the scheduling order and time for such algorithms.

We show that a programmable scheduler using PIFOs lets us program a wide variety of scheduling algorithms. We present a detailed hardware design for this scheduler for a 64-port 10 Gbit/s shared-memory switch with <4% chip area overhead on a 16-nm standard-cell library. Our design lets us program many sophisticated algorithms, such as a 5-level hierarchical scheduler with programmable scheduling algorithms at each level.

1. INTRODUCTION

uler, switch designers would implement scheduling algorithms as programs atop a programmable substrate. Moving scheduling algorithms into software makes it much easier to build and verify algorithms in comparison to implementing the same algorithms as rigid hardware IP.

This paper presents a design for programmable packet scheduling in line-rate switches. Our design is motivated by the observation that all scheduling algorithms make two key decisions: first, in what order should packets be scheduled, and second, at what time should each packet be scheduled. Furthermore, in many scheduling algorithms, these two decisions can be made when a packet is enqueued. This observation was first made in a recent position paper [36]. The same paper also proposed the *push-in first-out queue (PIFO)* [15] abstraction for maintaining the scheduling order or scheduling time for packets, when these can be determined on enqueue. A PIFO is a priority queue data structure that allows elements to be pushed into an arbitrary location based on an element's *rank*, but always dequeues elements from the head.

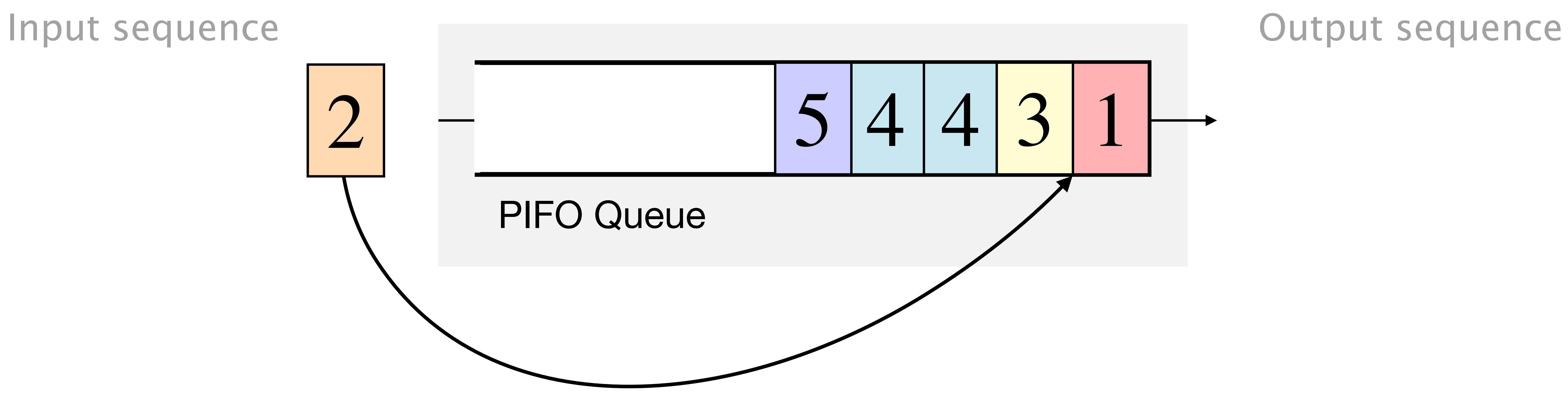
Building on the PIFO abstraction, this paper presents the detailed design, implementation, and analysis of feasibility of a programmable packet scheduler. To program a PIFO, we develop the notion of a *scheduling transaction*—a small program to compute an element's rank in a PIFO. We present a rich programming model built using PIFOs and scheduling transactions (§2) and show how to program a diverse set of scheduling algorithms in the model

Push-In First-Out (PIFO) queues

enable programmable packet scheduling

Pushes packets into arbitrary locations (Packet ranks)

Drains packets from the head

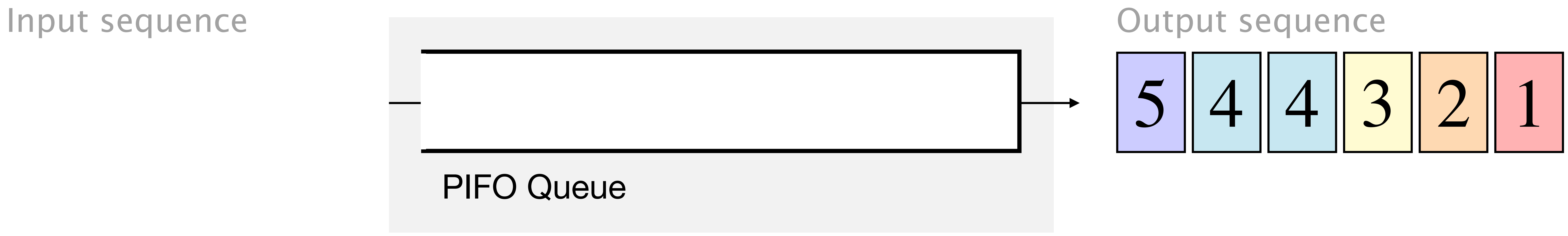


Push-In First-Out (PIFO) queues

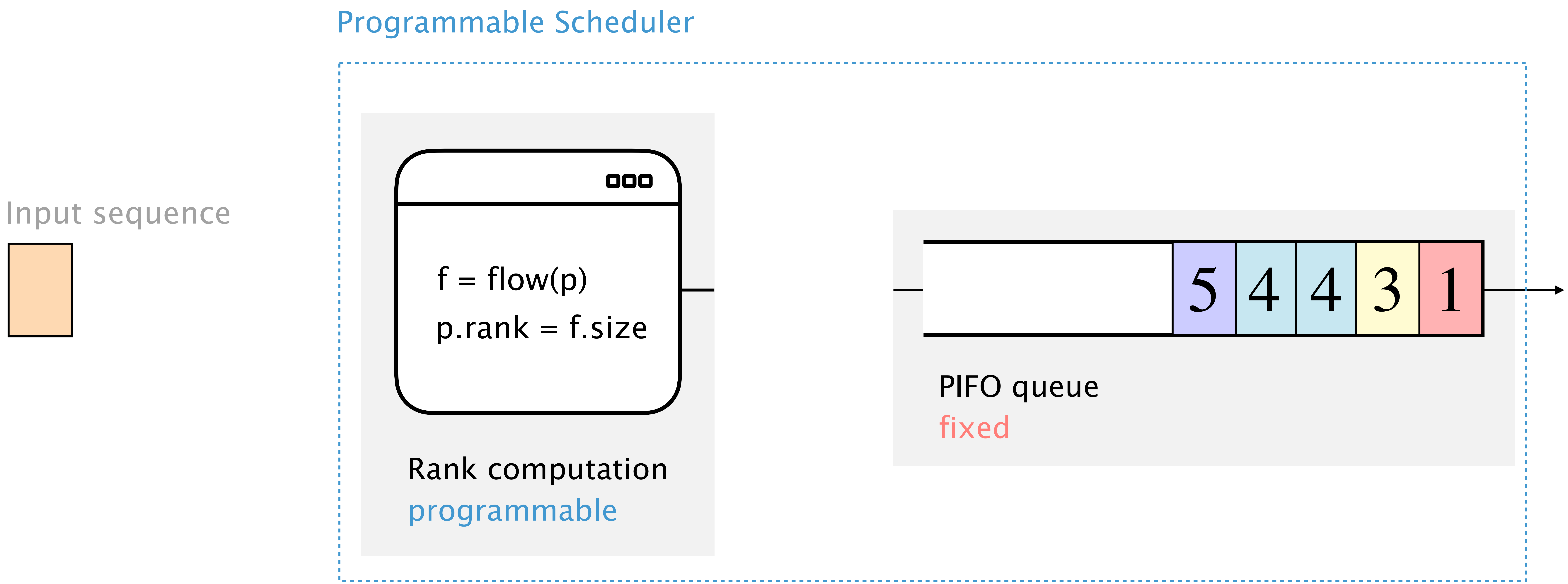
enable programmable packet scheduling

Pushes packets into arbitrary locations (Packet ranks)

Drains packets from the head

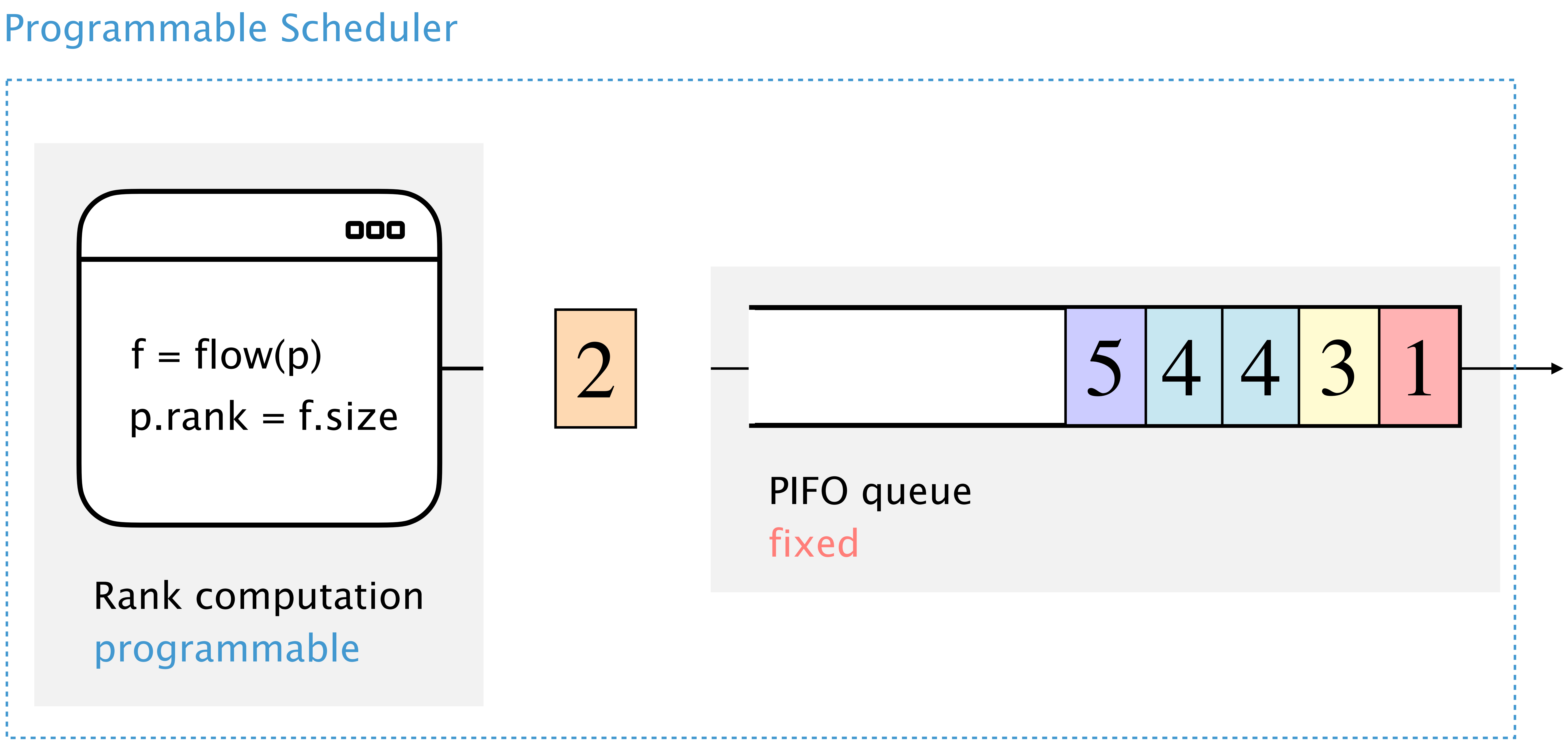


Push-In First-Out (PIFO) queues enable programmable packet scheduling

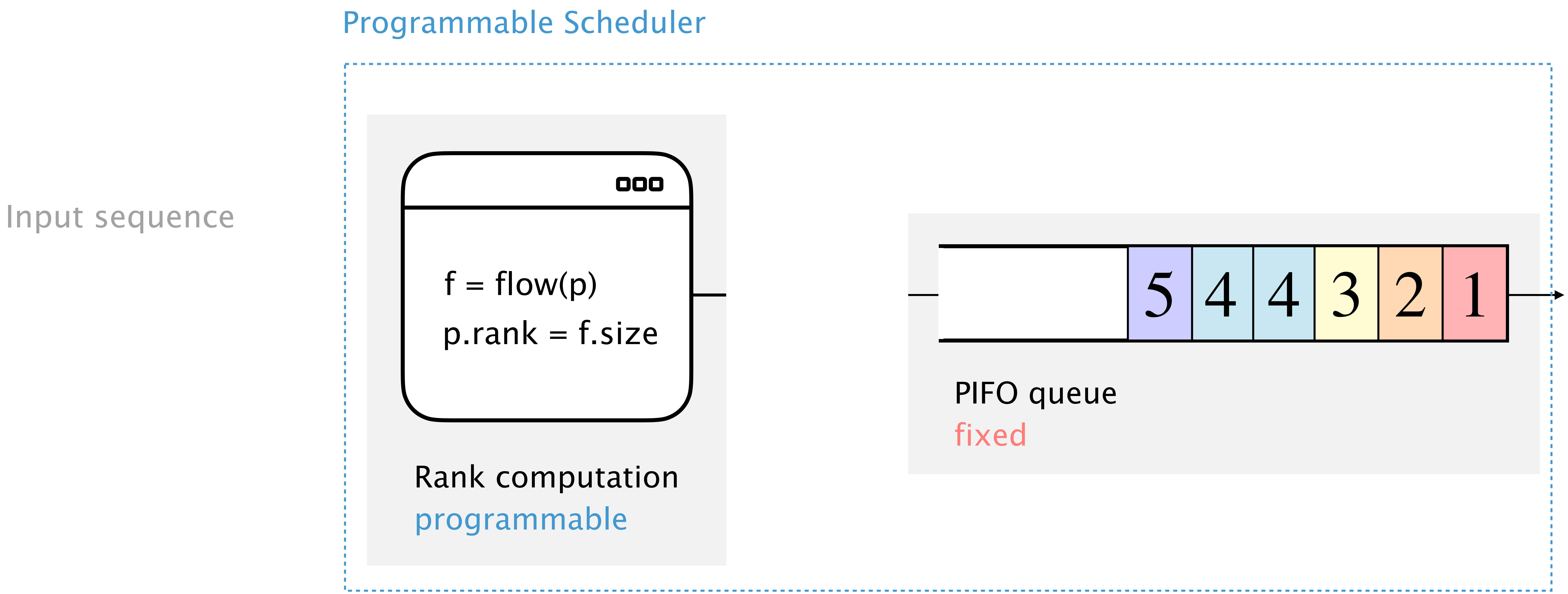


Push-In First-Out (PIFO) queues enable programmable packet scheduling

Input sequence



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



Implementing PIFO queues in hardware is challenging

Existing proposal…

Scalability	supports ~1k flows and ~10 Gbps
Flexibility	assumes monotonically increasing ranks

Moreover…

Deployability	implementing ASICs takes years

Can we approximate PIFO queues…

at line rate,

at scale, and

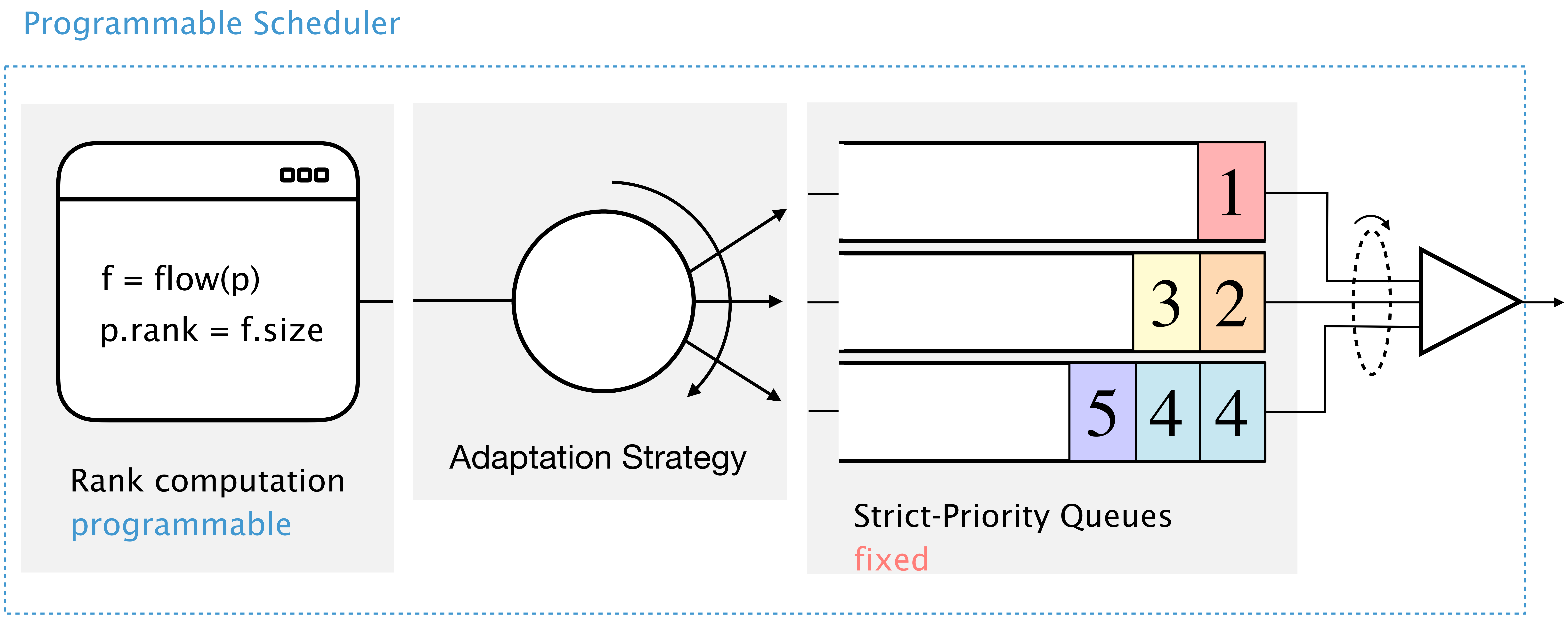
on existing devices?

Introducing…

SP-PIFO

A *deployable, scalable and flexible* PIFO approximation

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



SP-PIFO: Programmable Scheduling, Today — NSDI ‘20

ACC-Turbo: Mitigating Pulse-wave DDoS with Programmable Scheduling — SIGCOMM ‘22

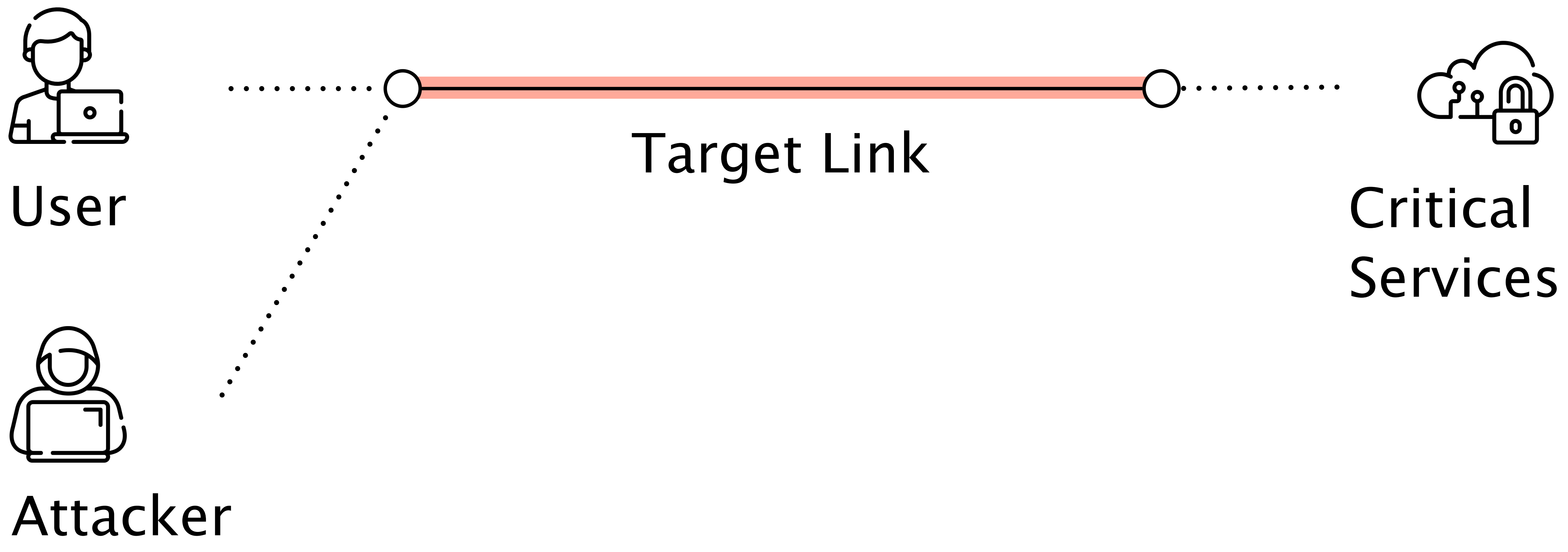
QVISOR: Virtualizing Scheduling Policies — HotNets ‘23

Pulse-wave DDoS attacks are a new type of network-layer DDoS attack

Target
a critical link

Volumetric
(Gbps)

Multiple
attack vectors

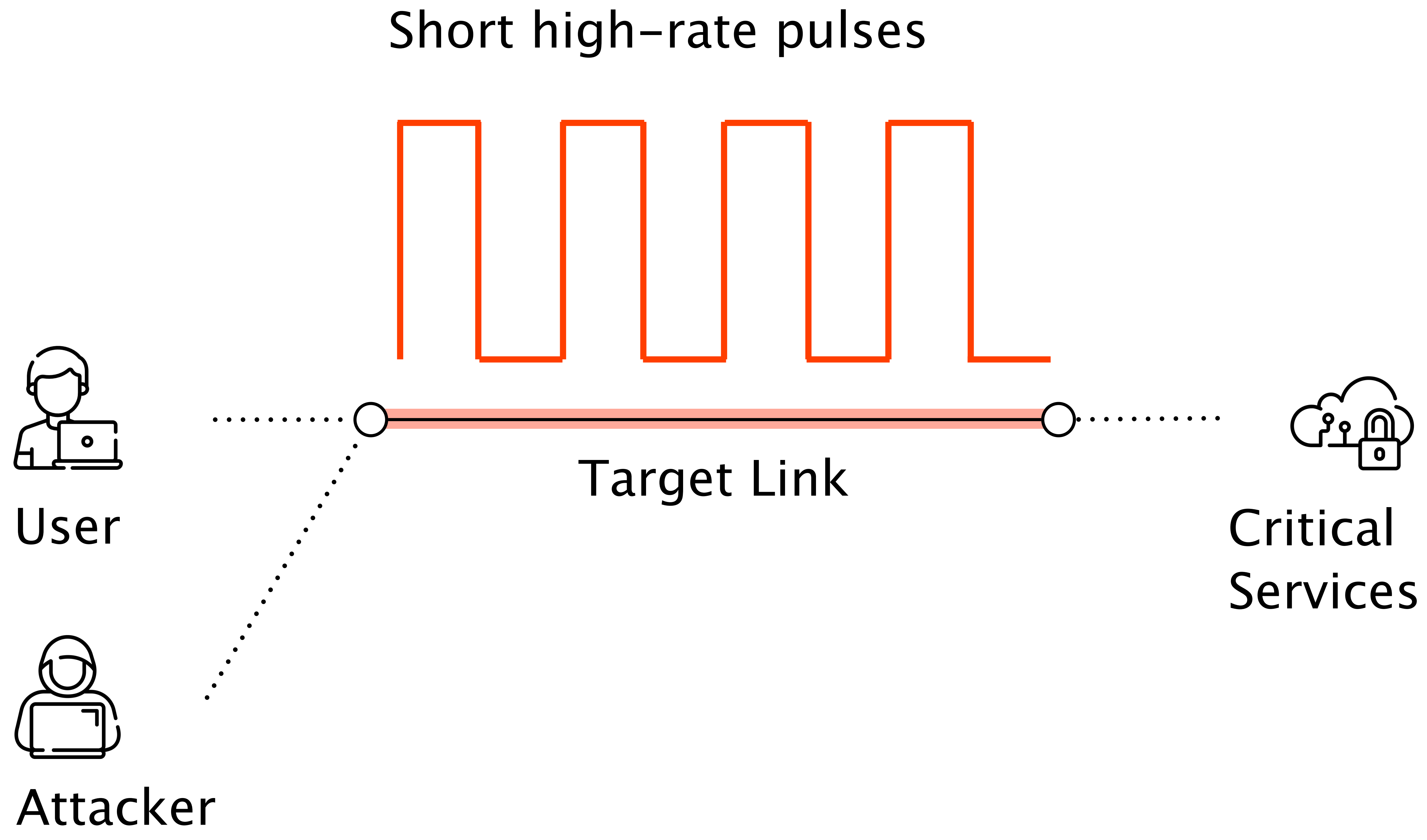


Pulse-wave DDoS attacks are a new type of network-layer DDoS attack

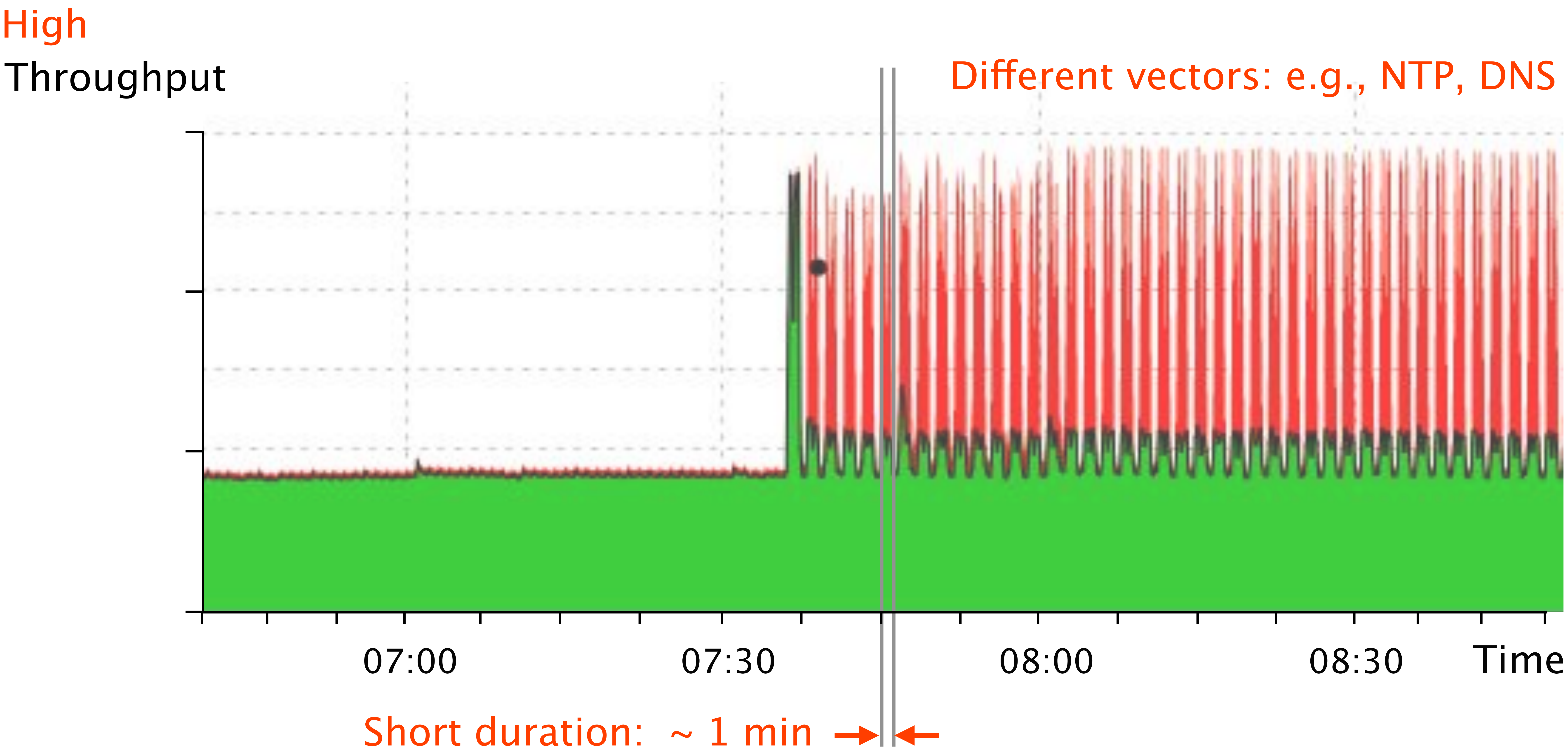
Target
a critical link

Volumetric
(Gbps)

Multiple
attack vectors



Pulse-wave DDoS attacks are composed of short-duration high-rate traffic pulses

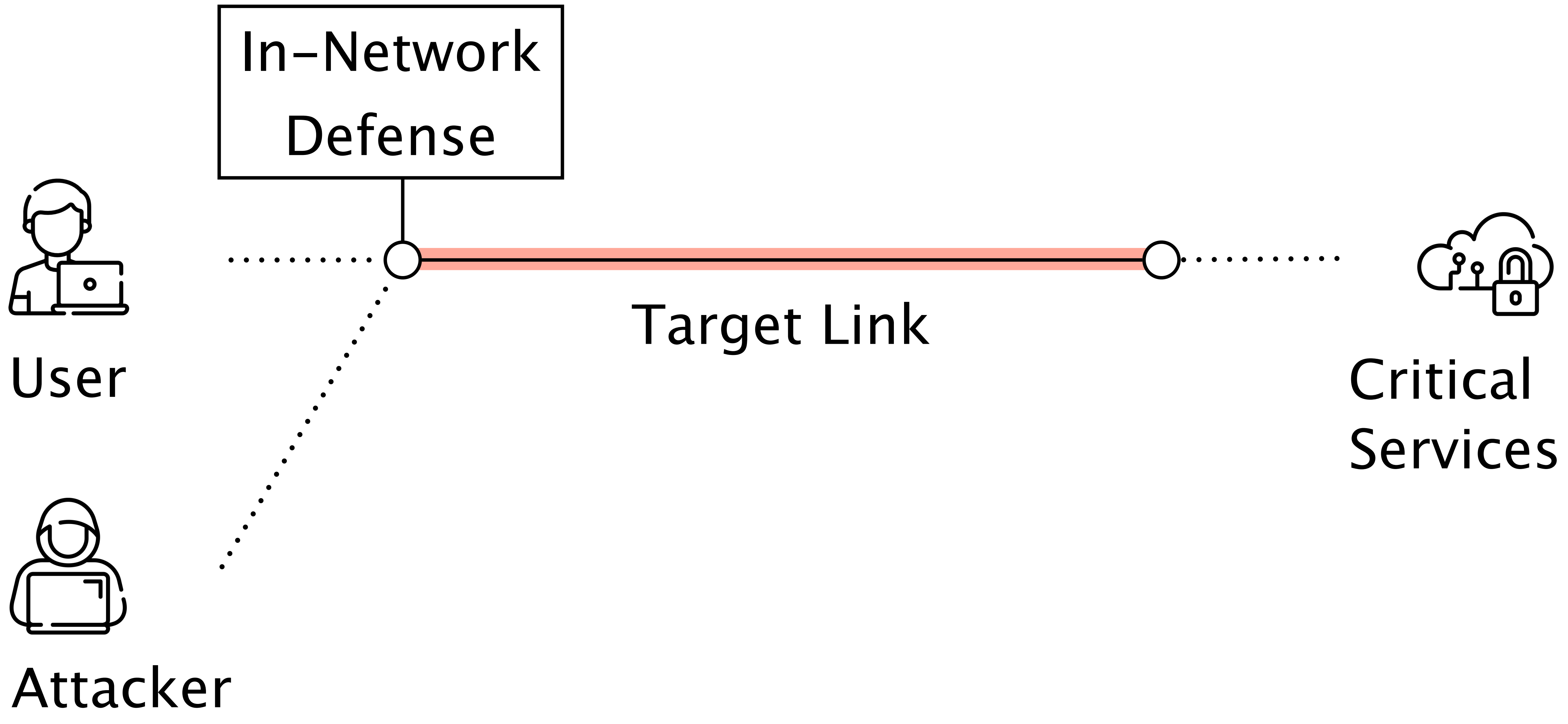


Damian Menscher, Google 2021

Pulse-wave DDoS attacks exploit the limitations of existing defenses

Narrow
attack coverage

Signature-based
Access-control lists



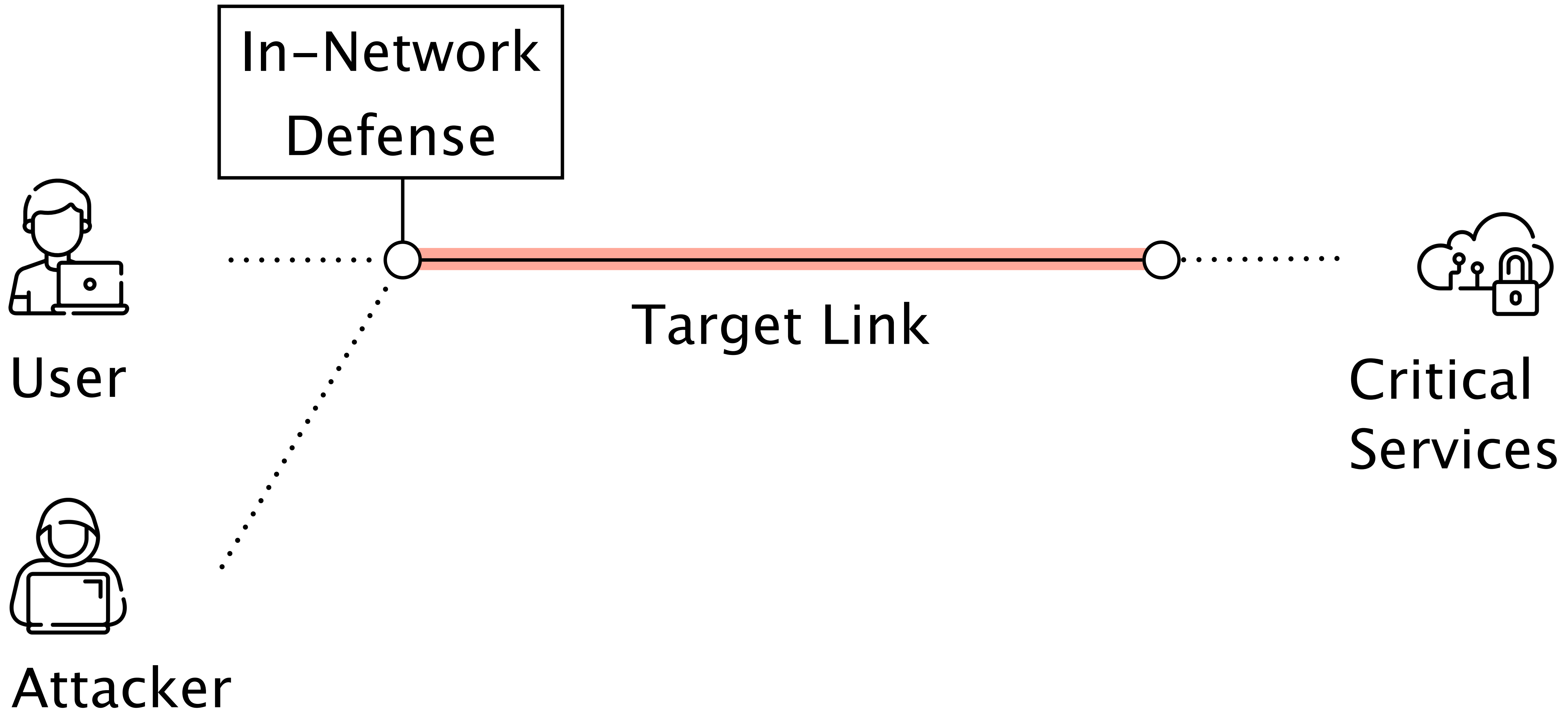
Pulse-wave DDoS attacks exploit the limitations of existing defenses

Narrow attack coverage

Filter-based

Rerouting-based

Drastic mitigation

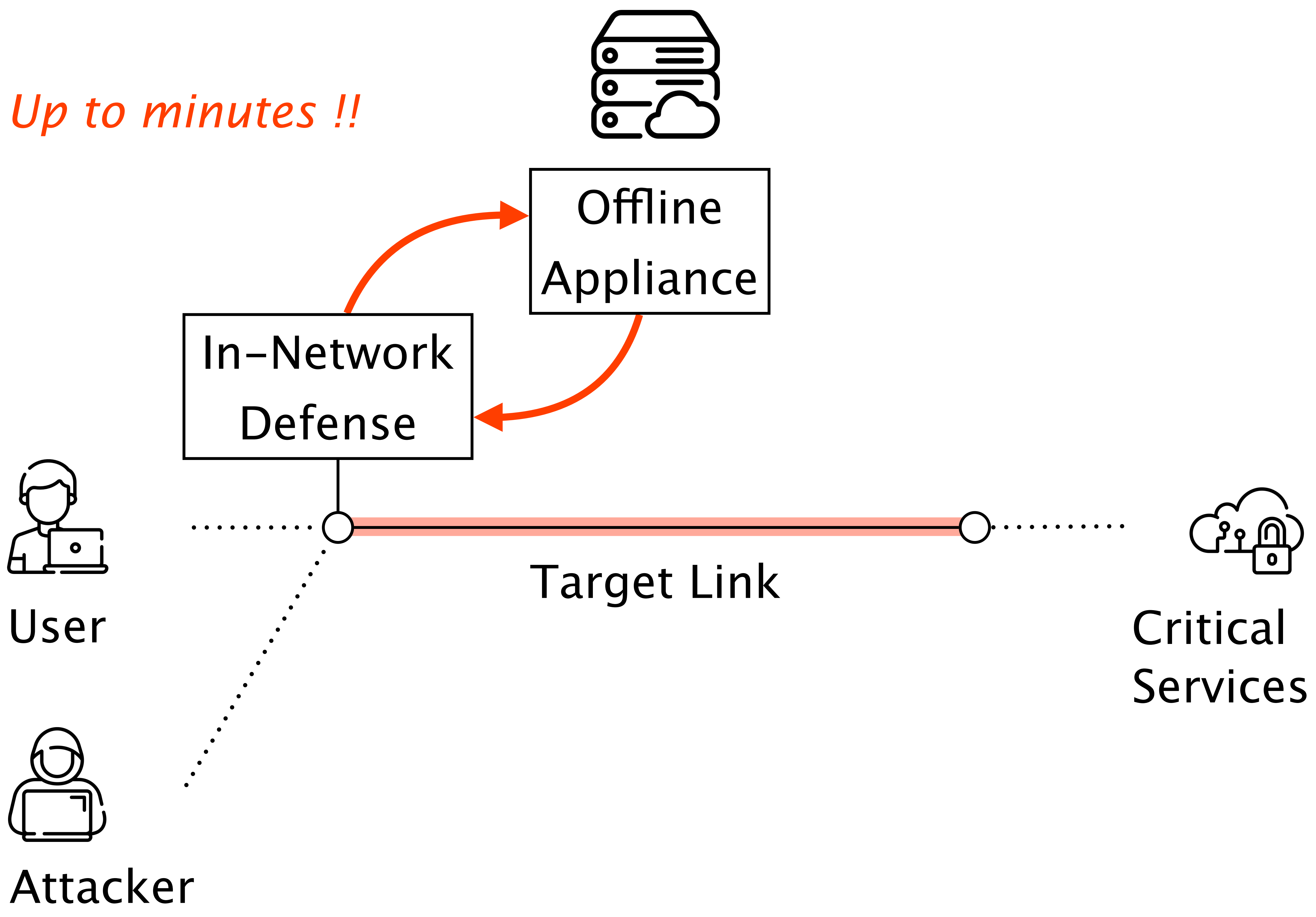


Pulse-wave DDoS attacks exploit the limitations of existing defenses

Narrow attack coverage

Drastic mitigation

Slow reaction time



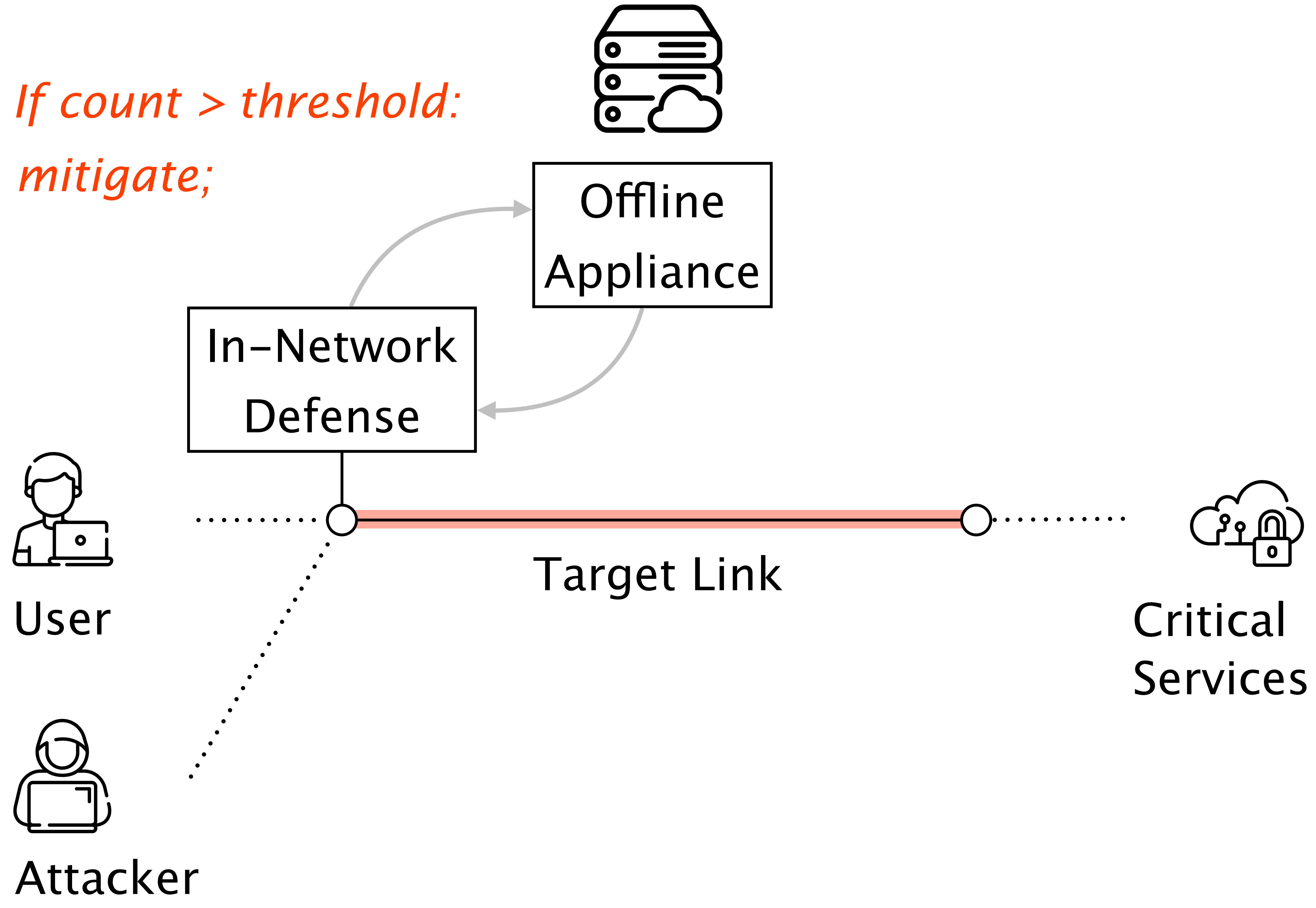
Pulse-wave DDoS attacks exploit the limitations of existing defenses

Narrow attack coverage

Drastic mitigation

Slow reaction time

Risk of misconfiguration



A pulse-wave DDoS defense needs to be …

Narrow attack coverage	**Generic** detection
Drastic mitigation	**Safe** mitigation
Slow reaction time	**Fast** reaction
Risk of misconfiguration	**Automated** configuration

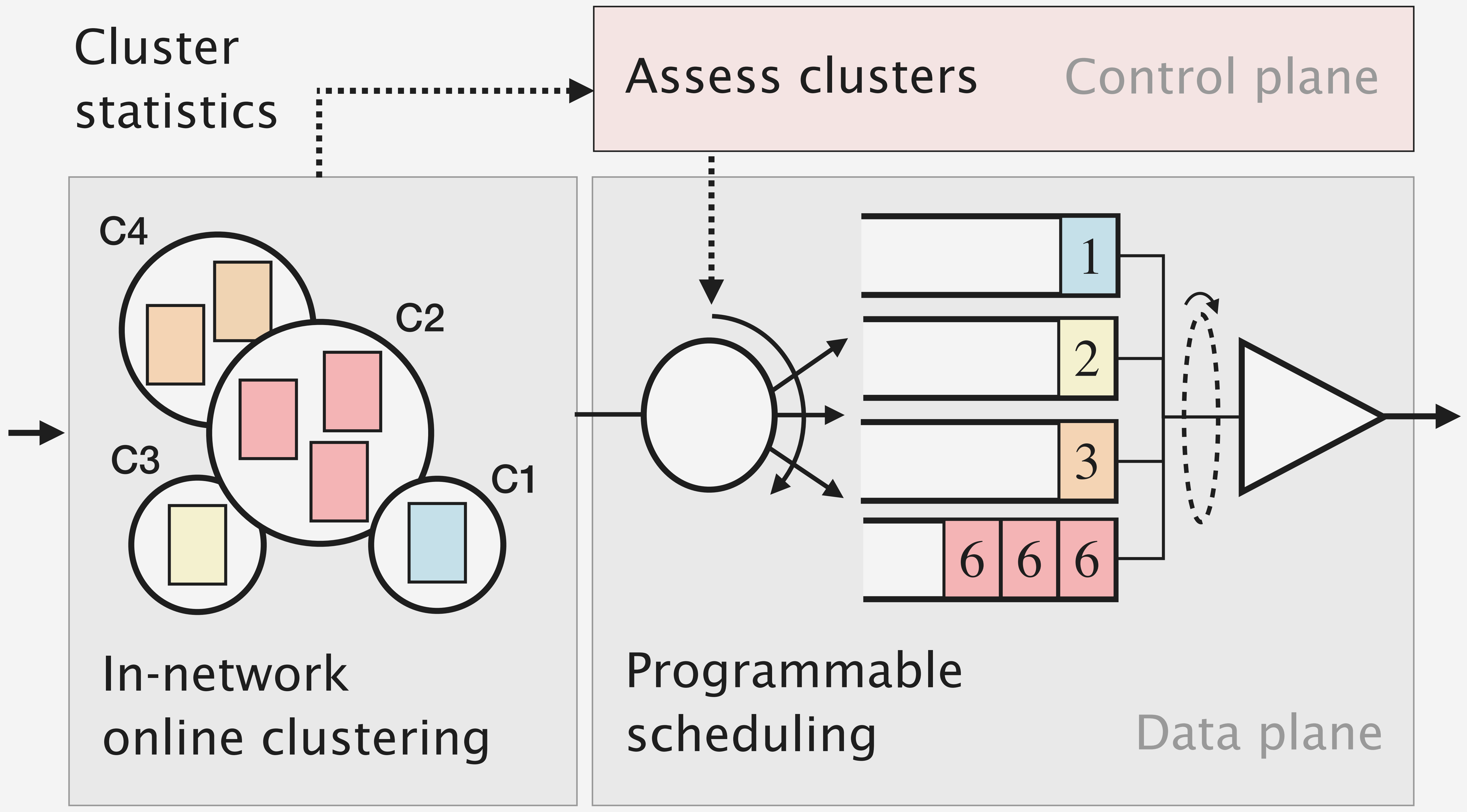
Introducing…

ACC-Turbo

A *generic, safe, fast, and automated* DDoS defense

ACC-Turbo

2022



How to automatically mitigate inferred attacks?

Programmable scheduling

ACC-Turbo deprioritizes malicious clusters

… leverages the whole uncertainty spectrum

with fine-grained scheduling policies

… is safe

only drops under congestion

… does not require activation

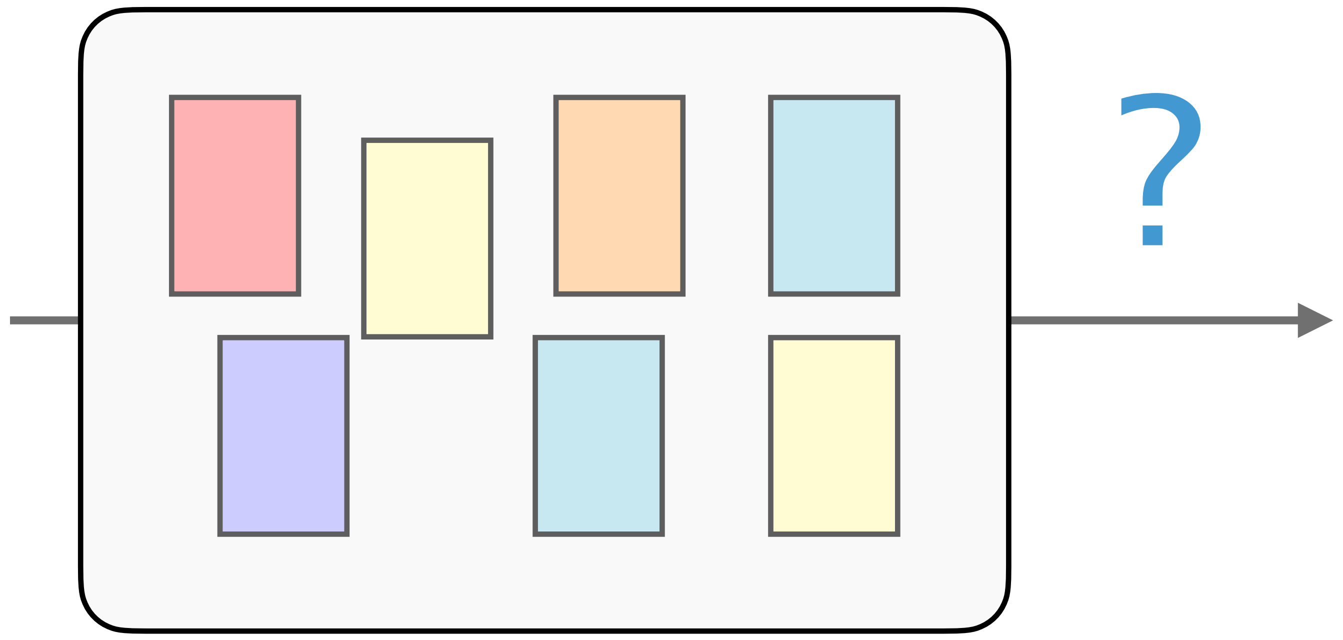
can be always-on

SP-PIFO: Making Scheduling Programmable Today	NSDI '20
ACC-Turbo: Mitigating Pulse-wave DDoS with Programmable Scheduling	SIGCOMM '22
QVISOR: Virtualizing Scheduling Policies	HotNets '23

Packet scheduling

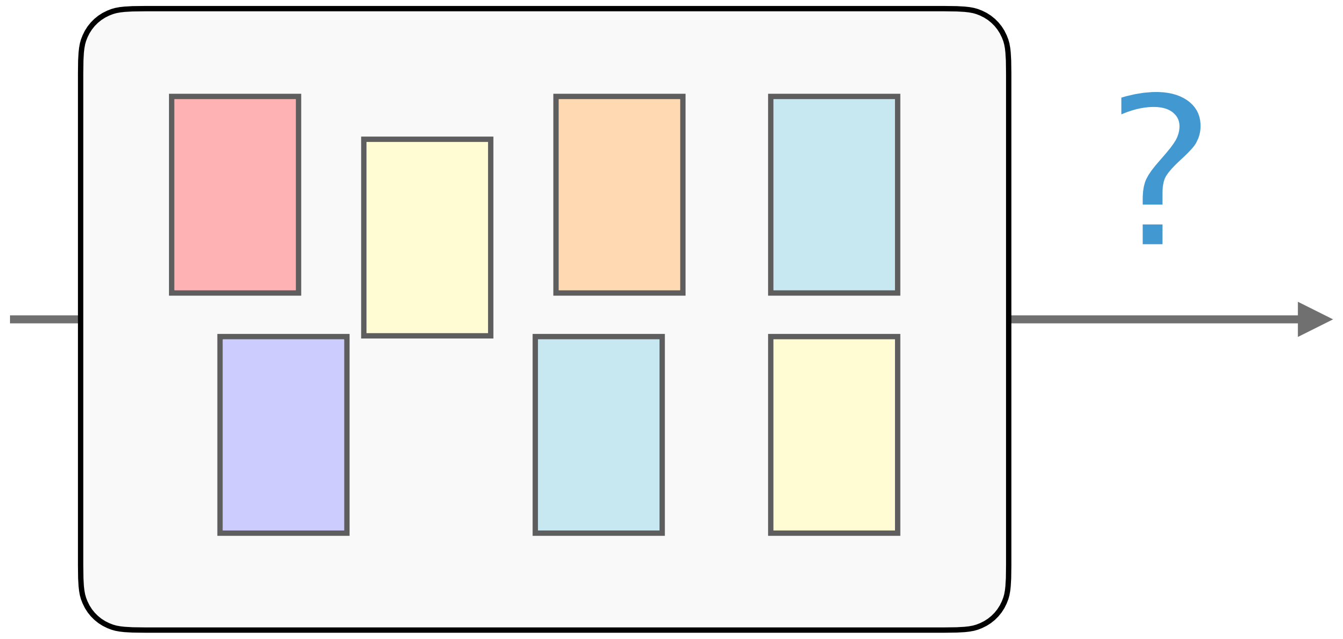
What packet next

and *when?*



Packet scheduling

What packet next
and *when?*



Minimize tail latency   FIFO+

Prioritize packets with higher queuing time

Minimize FCTs   SRPT, PIAS, pFabric

Prioritize packets from short flows

Enforce fairness   WRR, (S)FQ, WFQ

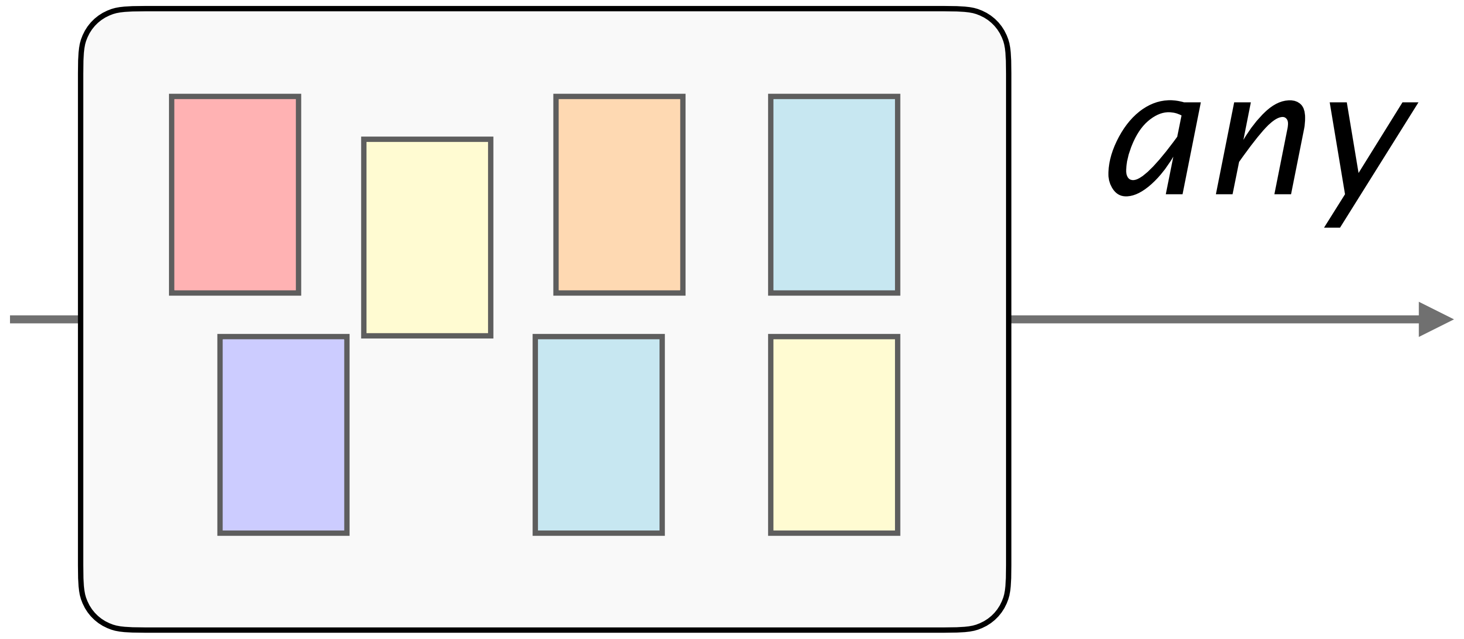
One packets from each class at a time

With programmable scheduling,

we can program *any* policy

What packet next

and *when?*



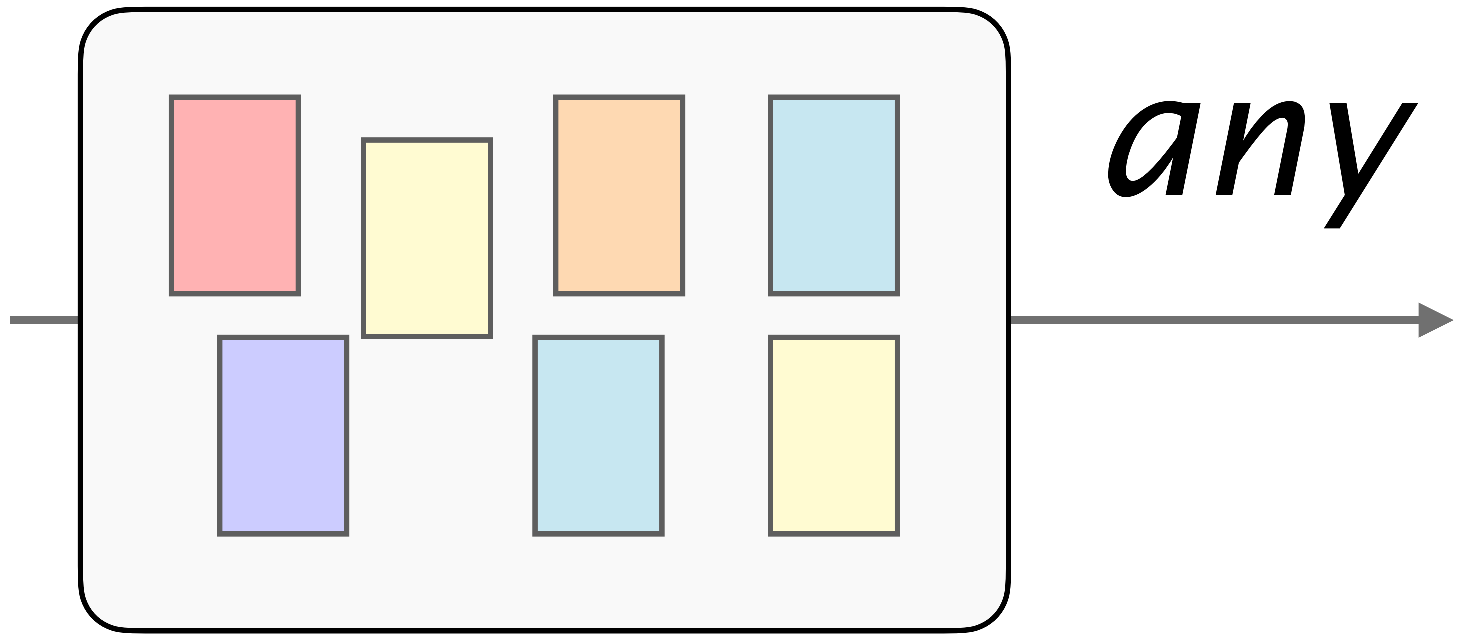
Which one?

With programmable scheduling,

we can program *any* policy

What packet next

and *when?*



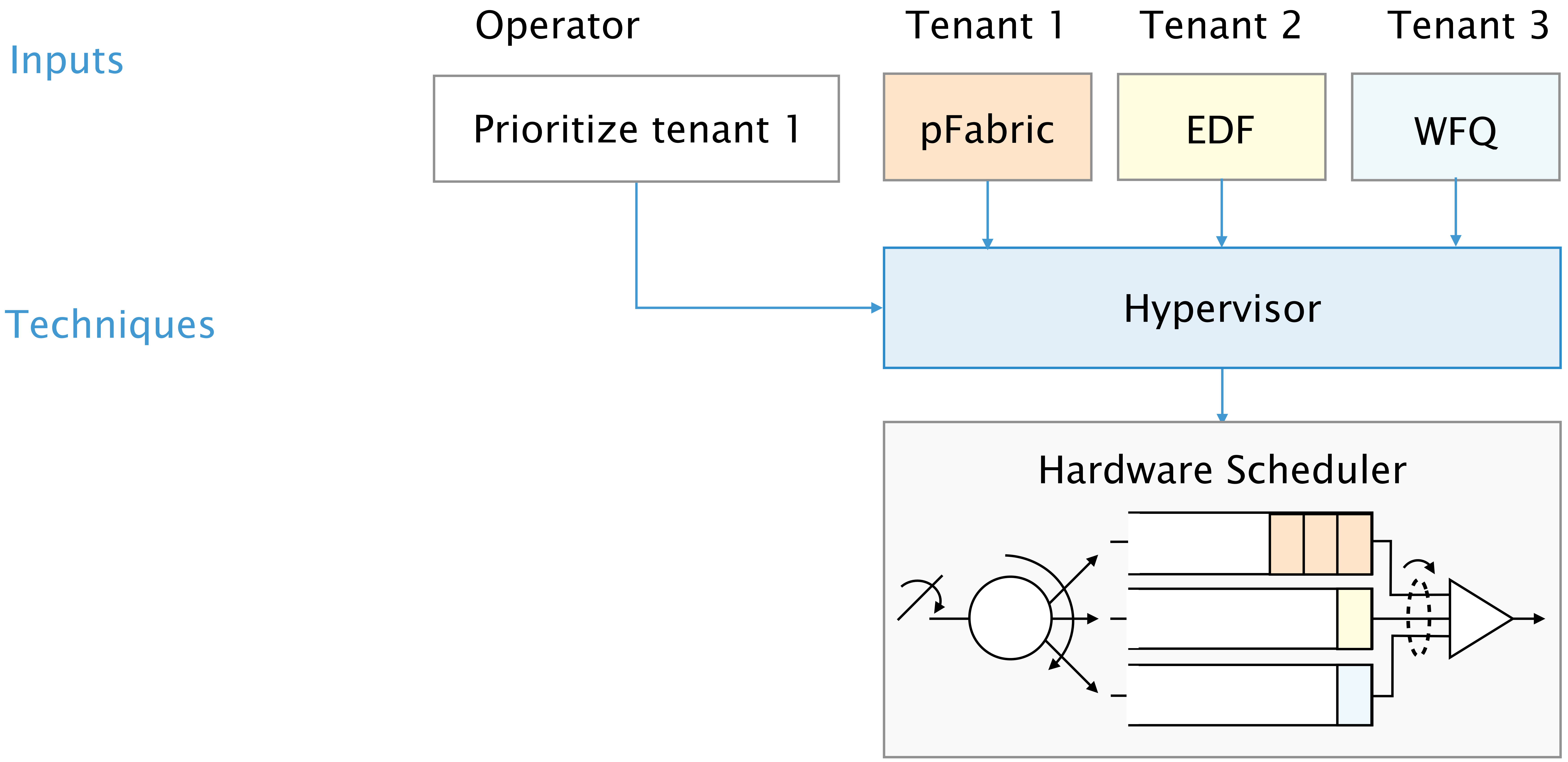
Which one(s)?

Introducing…

QVISOR

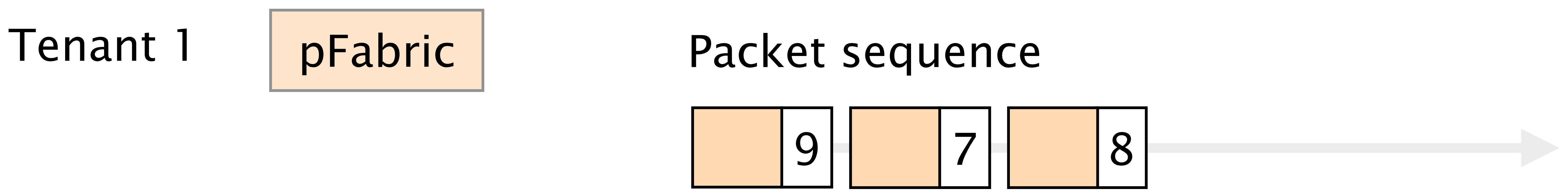
A *packet scheduling* hypervisor

What would it take to run multiple scheduling algorithms?



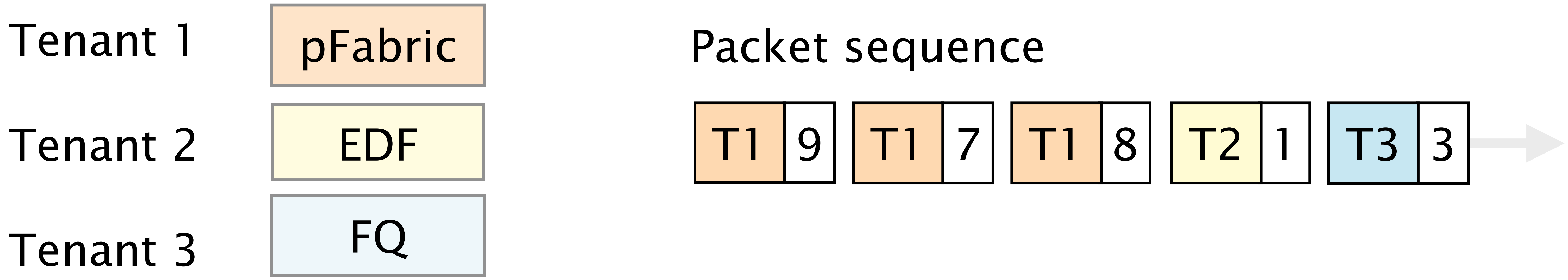
Tenants have the illusion that their traffic is scheduled by a PIFO queue

Tenants label each packet with a rank and the tenant ID



Tenants have the illusion that their traffic is scheduled by a PIFO queue

Tenants label each packet with a rank and the tenant ID



Tenants have the illusion that their traffic is scheduled by a PIFO queue

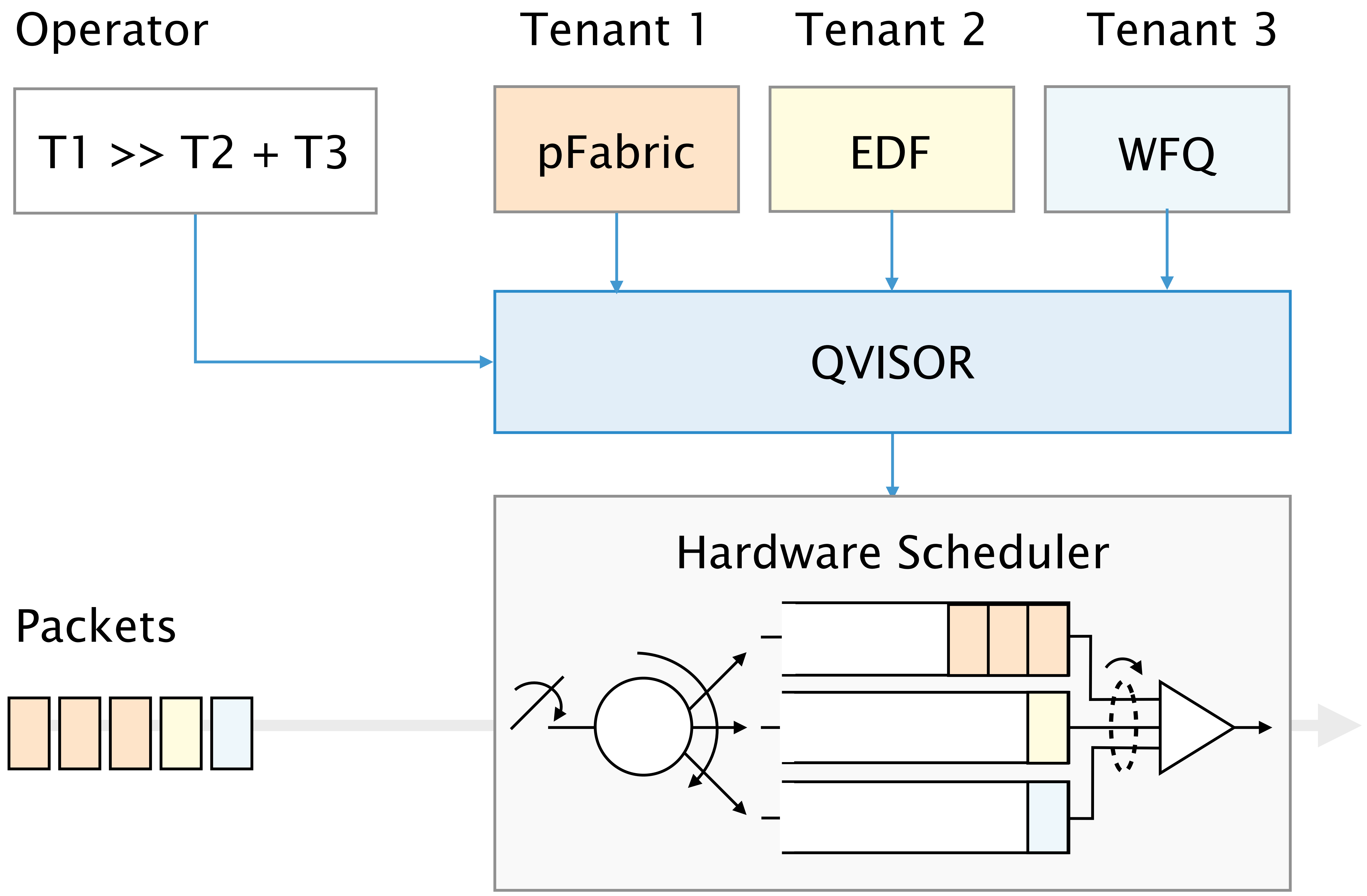
Operators define their policy with a composition language

>>	Strict priority
>	Best-effort priority
+	Sharing

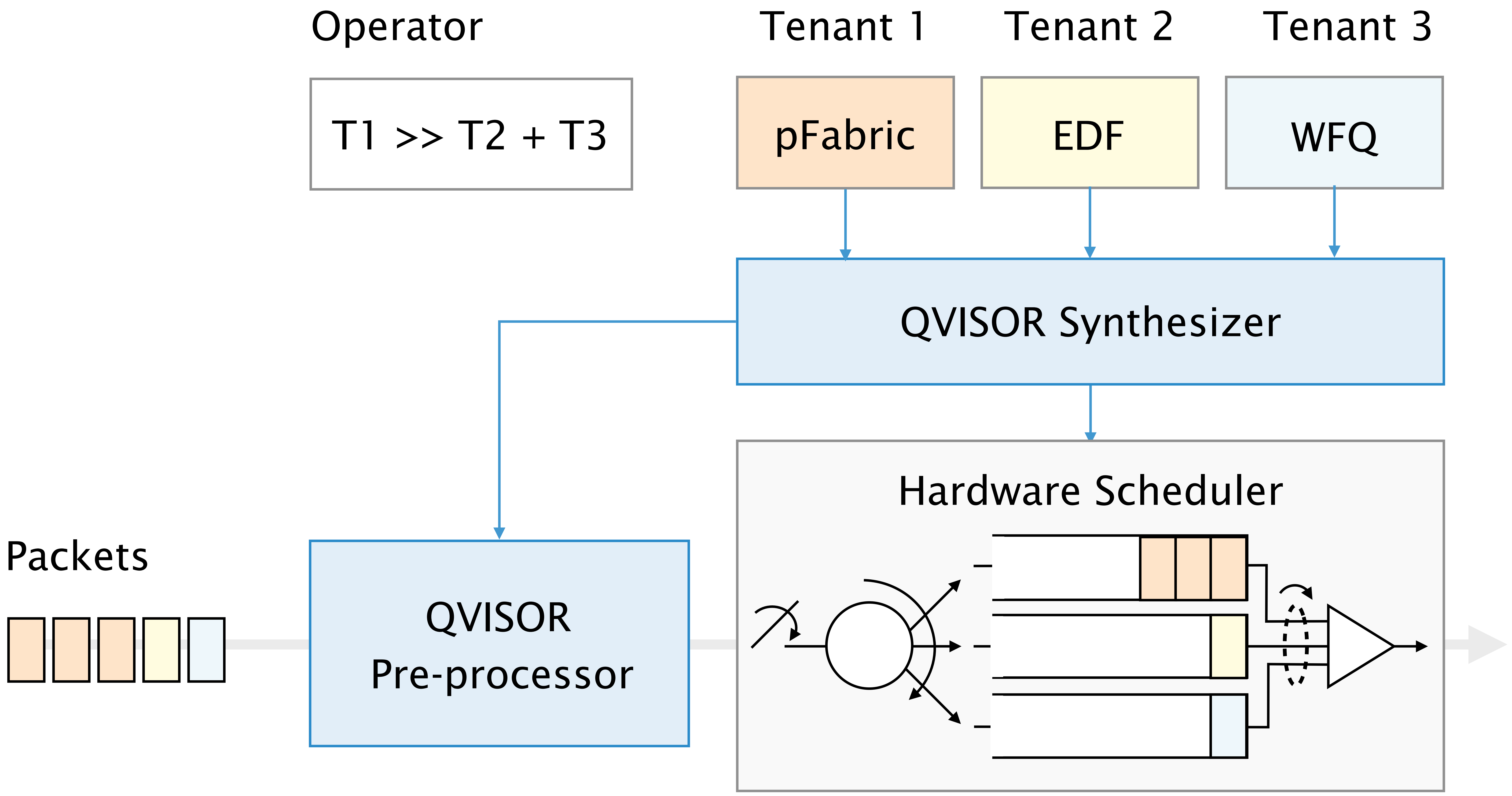
Policy:

T1 >> T2 + T3

QVISOR takes as input the policies from the tenants and the operator



QVISOR synthesizes a joint scheduling function and deploys it to hardware



QVISOR's synthesizer generates a set of rank-transformation functions

Currently, the synthesizer supports two operation types

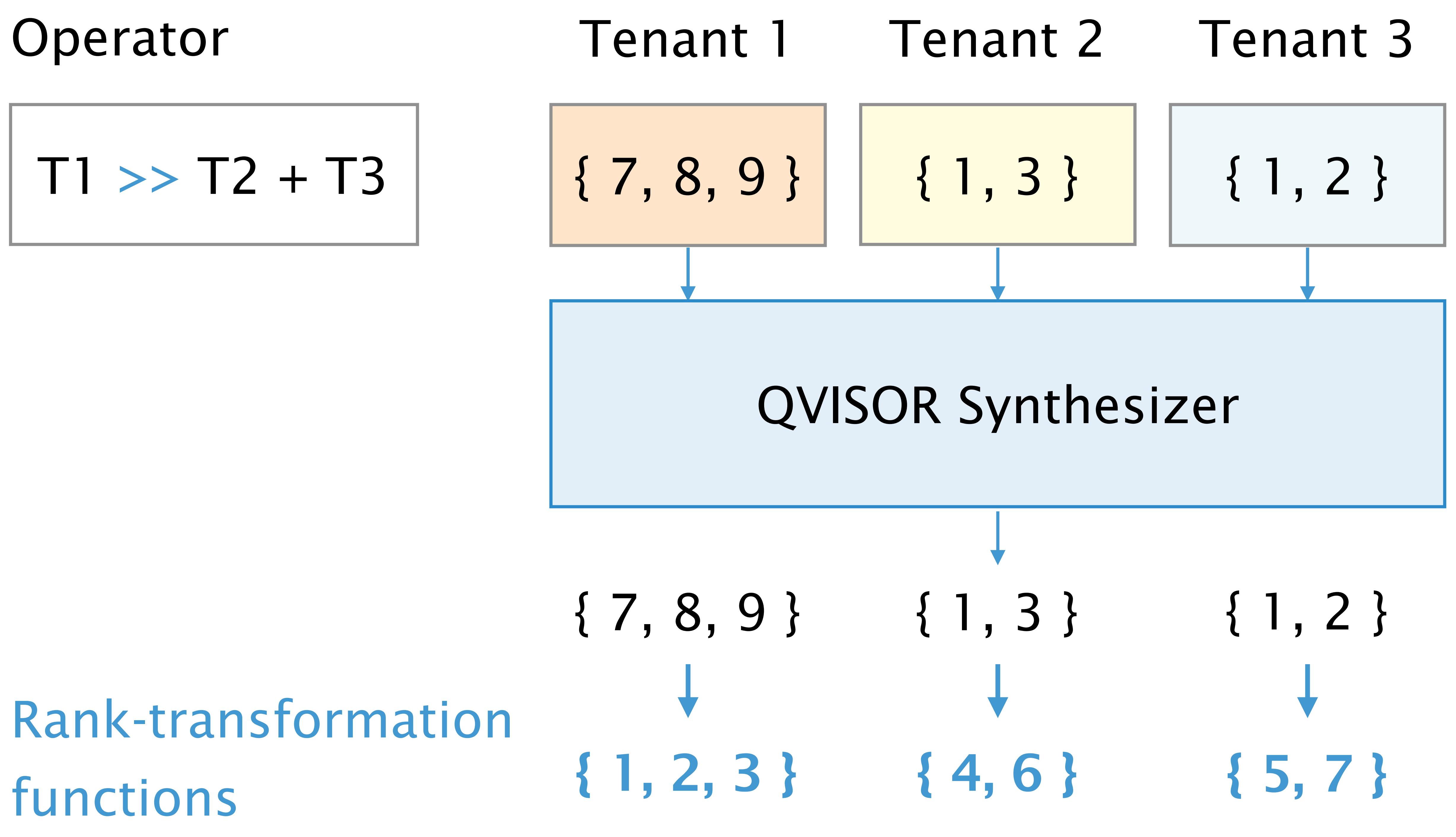
Rank normalizations

{ 700, 800, 900 } ⟶ { 7, 8, 9 }

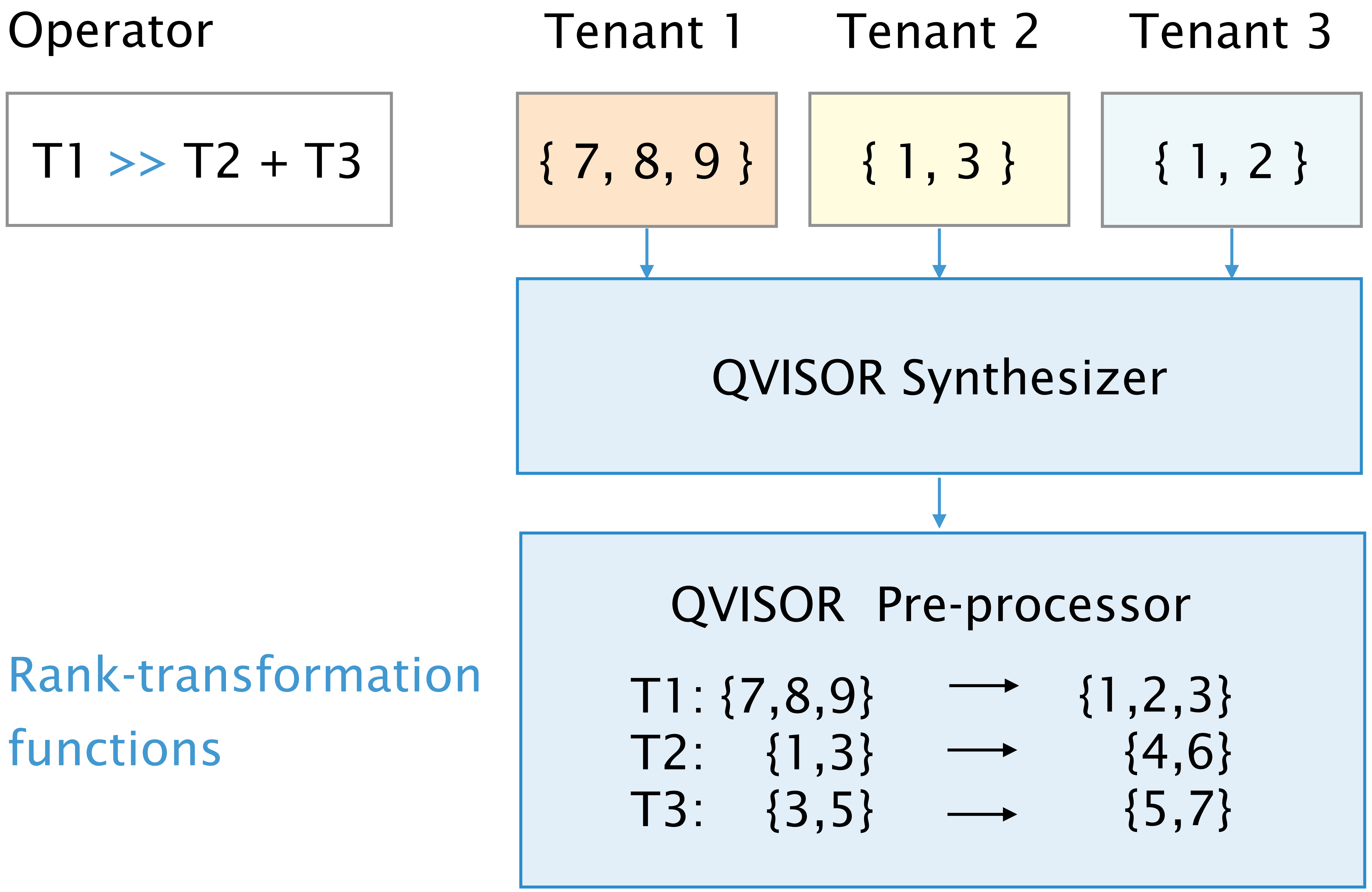
Rank shifts

{ 7, 8, 9 } ⟶ { 1, 2, 3 }

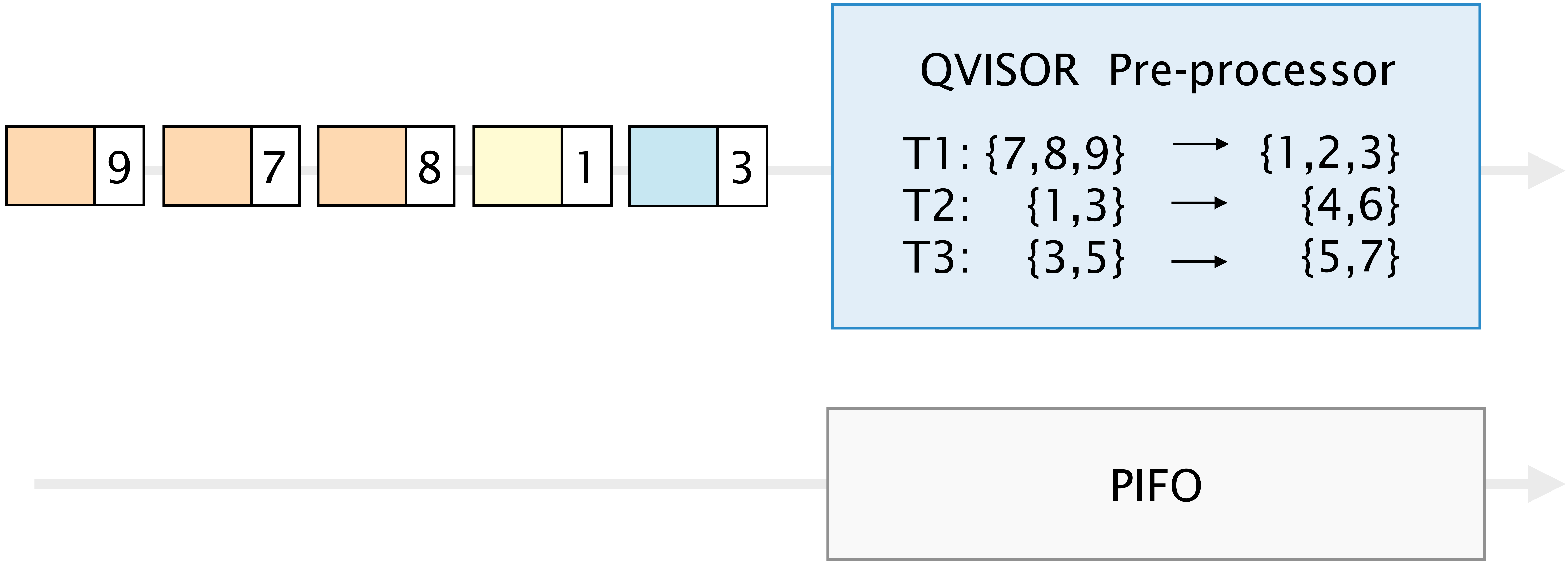
QVISOR's synthesizer generates a set of rank-transformation functions



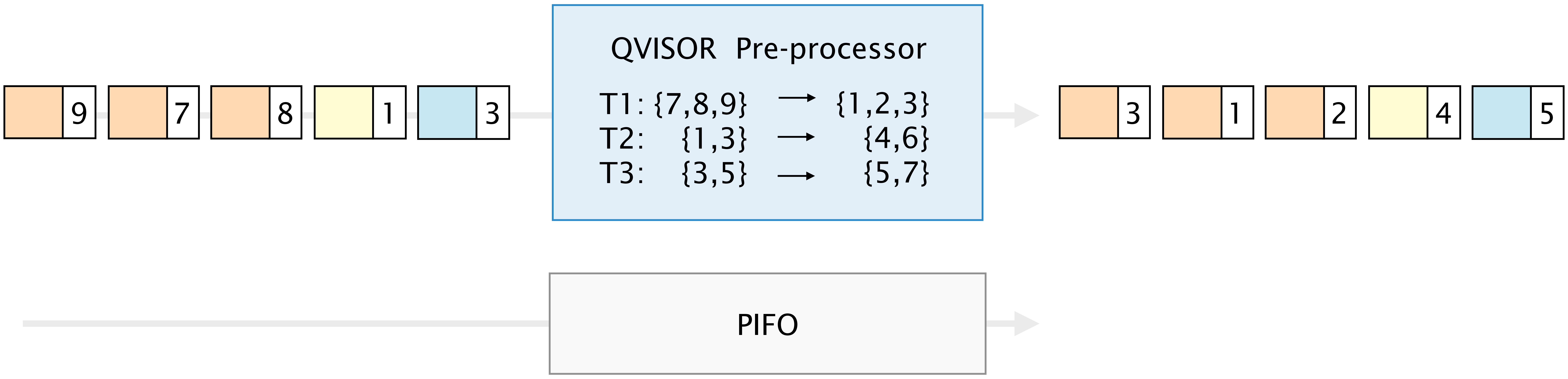
QVISOR's synthesizer generates a set of rank-transformation functions



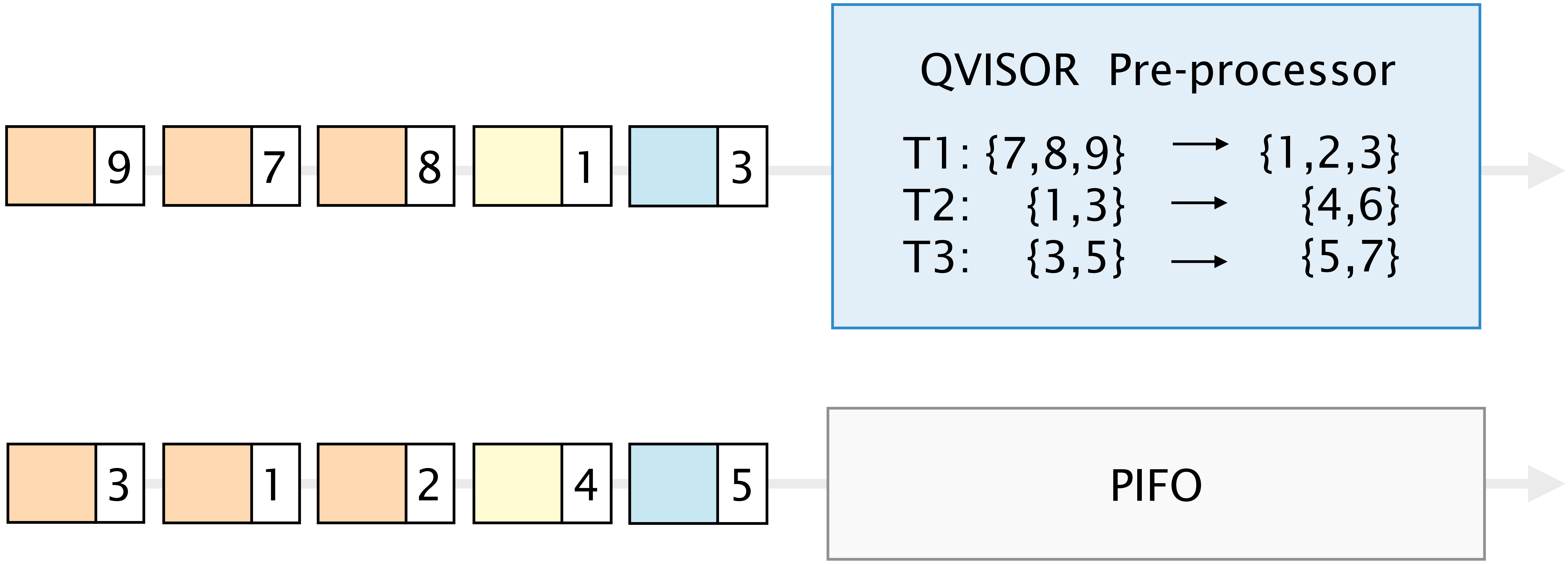
QVISOR's pre-processor applies the transformation functions at line rate in the data plane



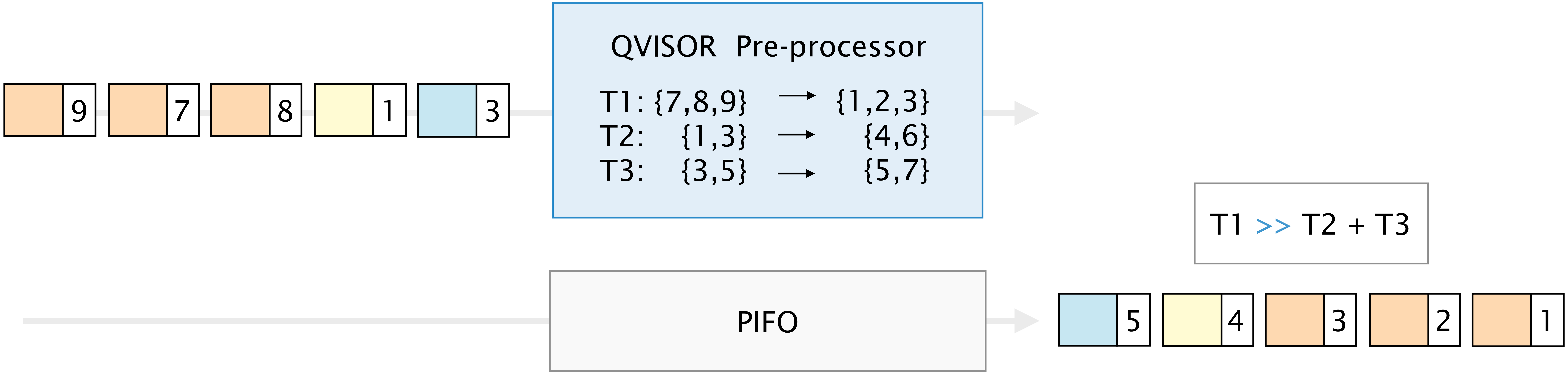
QVISOR's pre-processor applies the transformation functions at line rate in the data plane



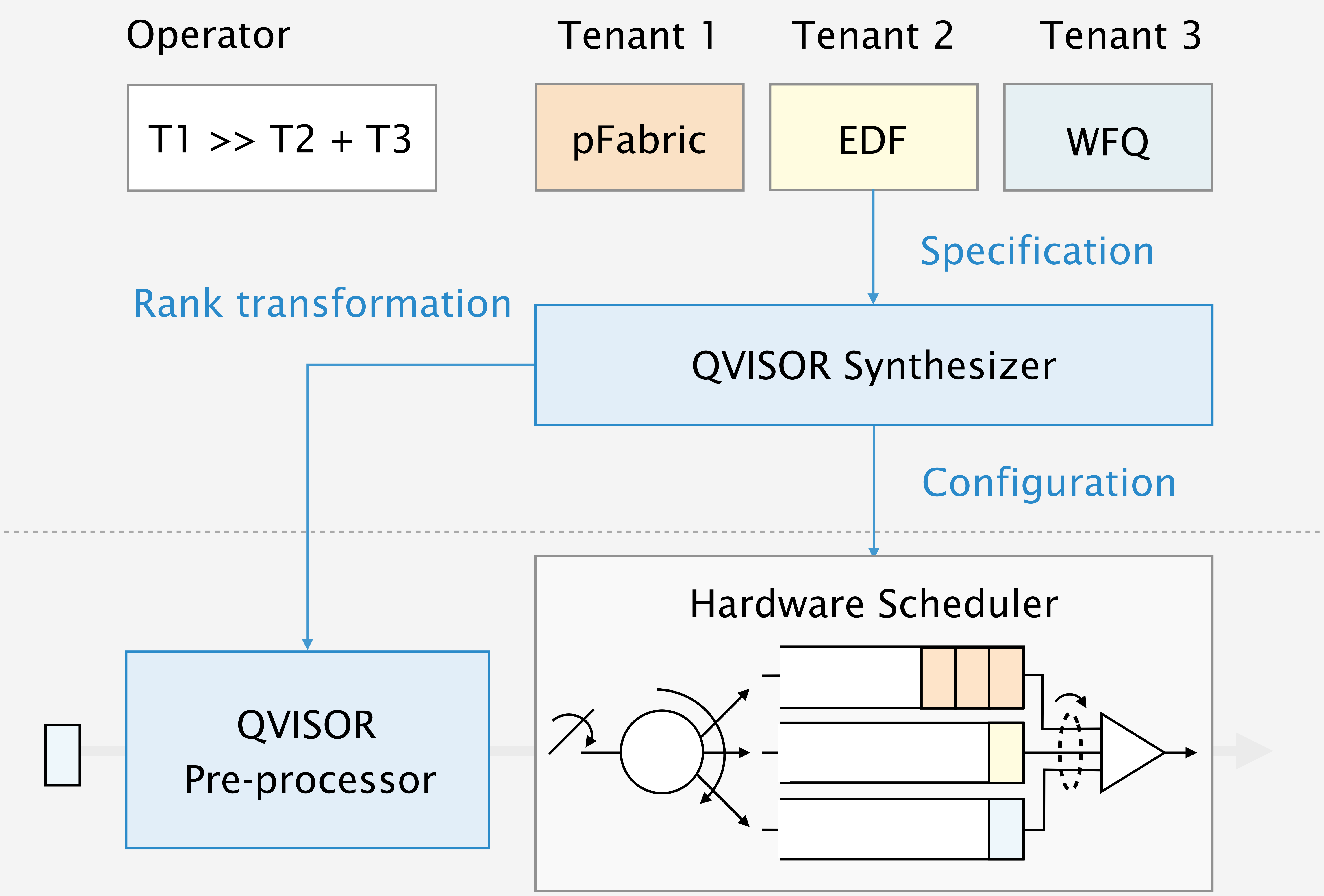
QVISOR's pre-processor applies the transformation functions at line rate in the data plane



QVISOR's pre-processor applies the transformation functions at line rate in the data plane



QVISOR


Operator
T1 >> T2 + T3
Tenant 1
pFabric
Tenant 2
EDF
Tenant 3
WFQ
Specification
Rank transformation
QVISOR Synthesizer
Configuration
Hardware Scheduler
QVISOR
Pre-processor

SP-PIFO: Programmable Scheduling, Today	NSDI ‘20
ACC-Turbo: Mitigating Pulse-wave DDoS with Programmable Scheduling	SIGCOMM ‘22
QVISOR: Virtualizing Scheduling Policies	HotNets ‘23