Programmable Packet Scheduling at Line Rate

Anirudh Sivaraman, Suvinay Subramanian, Mohammad Alizadeh, Sharad Chole, Shang-Tse Chuang, Anurag Agrawal, Hari Balakrishnan, Tom Edsall, Sachin Katti, Nick McKeown

Massachusetts Institute of Technology

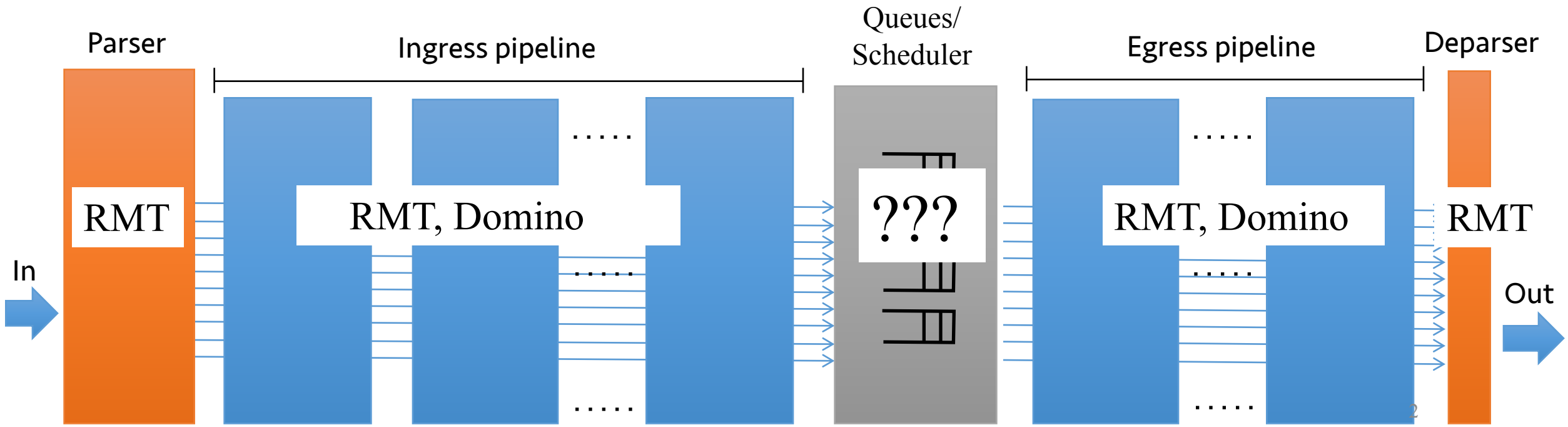
BAREFOOT NETWORKS

CISCO

Stanford University

Programmable scheduling at line rate

- Motivation: Can't deploy new schedulers in production networks
- The status quo in line-rate switches



The scheduler is still fixed

Why is programmable scheduling hard?

- Many algorithms, yet no consensus on abstractions, cf.
  - Parse graphs for parsing
  - Match-action tables for forwarding
  - Packet transactions for data-plane algorithms
- Scheduler has tight timing requirements
  - Can’t simply use an FPGA/CPU

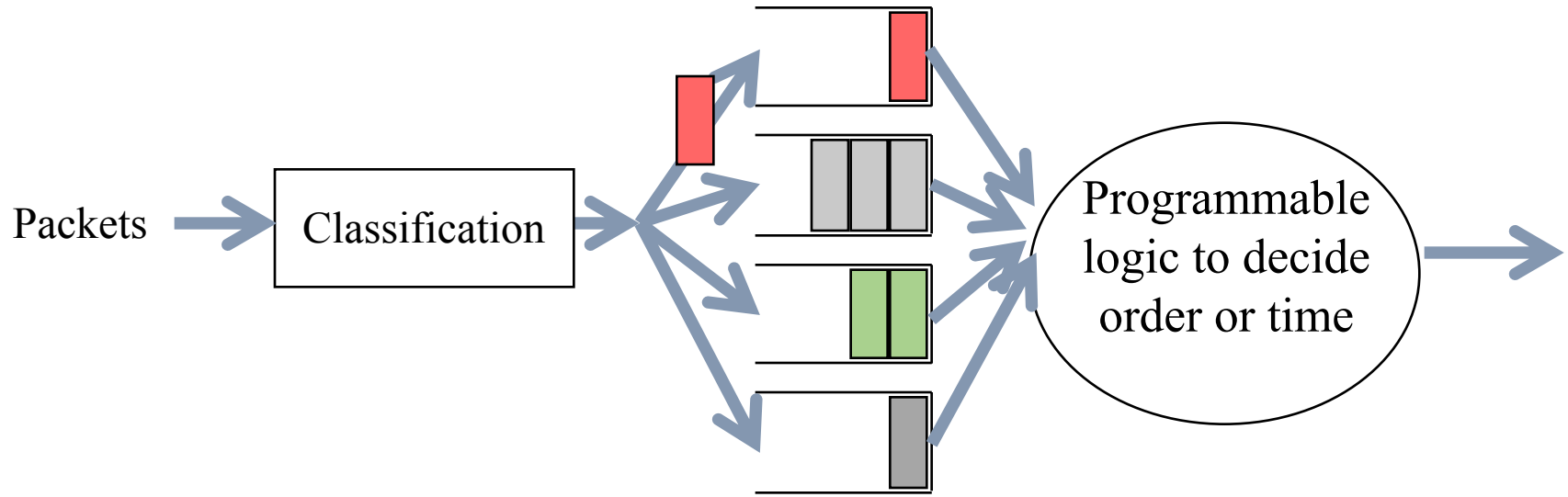
Need expressive abstraction that can run at line rate

What does the scheduler do?

It decides

- In what **order** are packets sent
  - e.g., FCFS, priorities, weighted fair queueing
- At what **time** are packets sent
  - e.g., Token bucket shaping

A strawman programmable scheduler



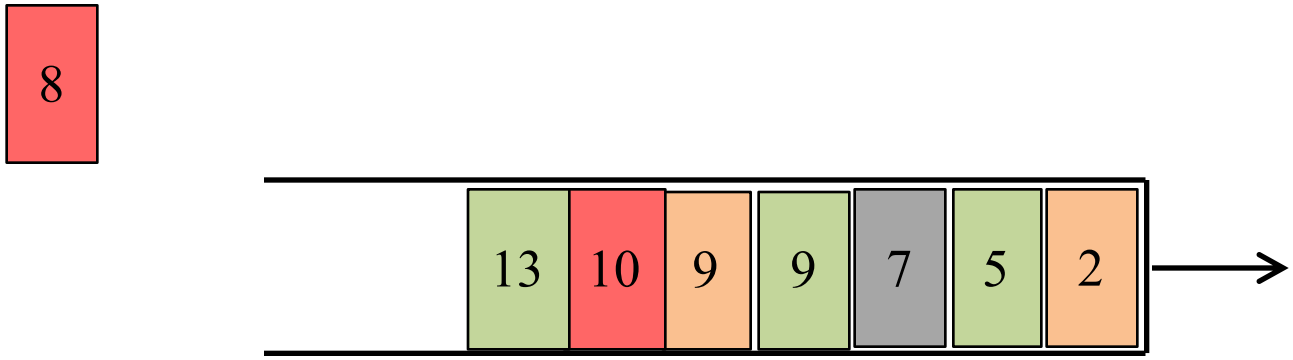
- Very little time on the dequeue side => limited programmability
- Can we move programmability to the enqueue side instead?

The Push-In First-Out Queue

Key observation

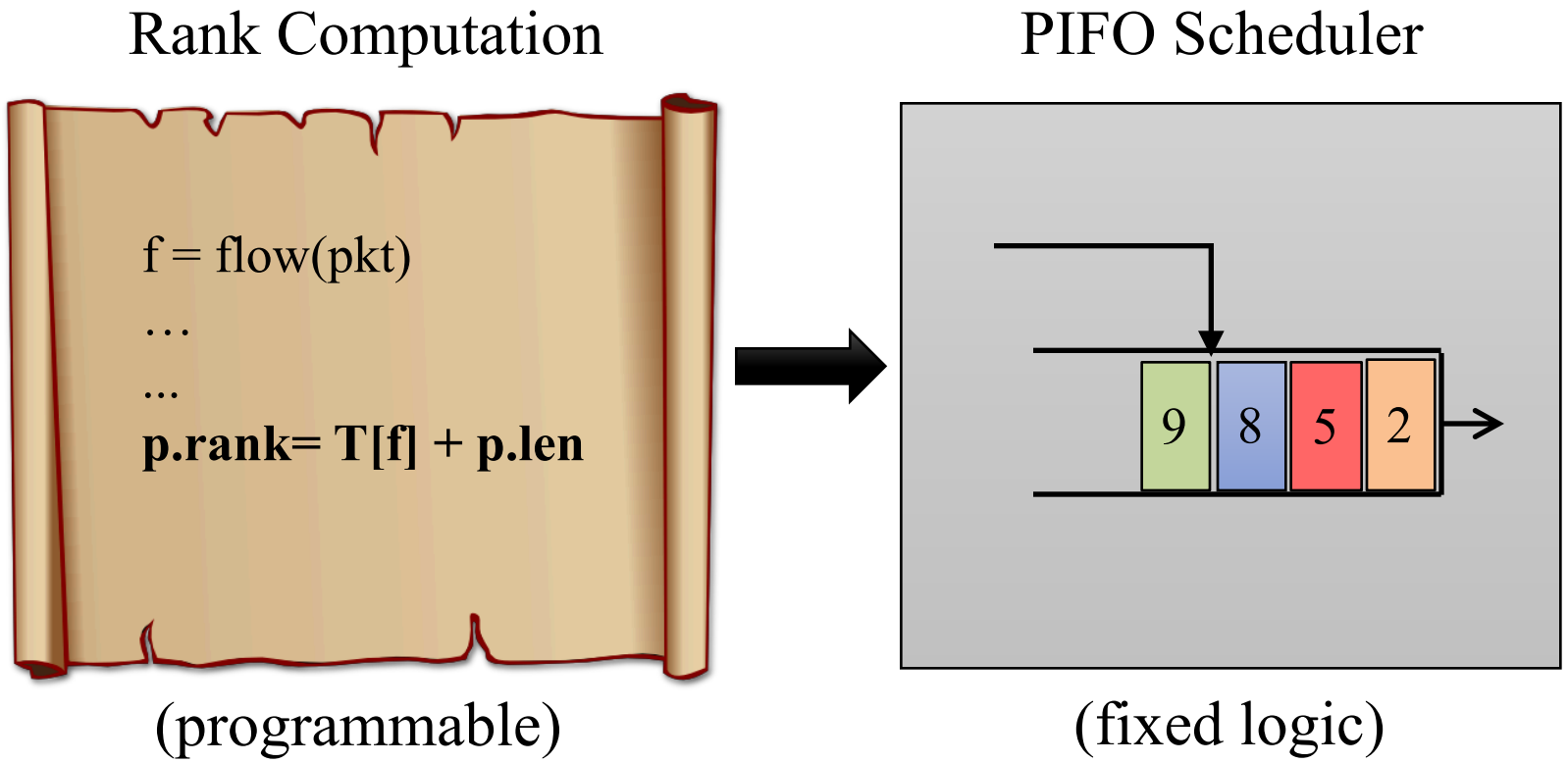
- In many cases, relative order of buffered packets does not change
- i.e., a packet's place in the scheduling order is known at enqueue

The Push-In First-Out Queue (PIFO): Packets are pushed into an arbitrary location based on a **rank**, and dequeued from the head

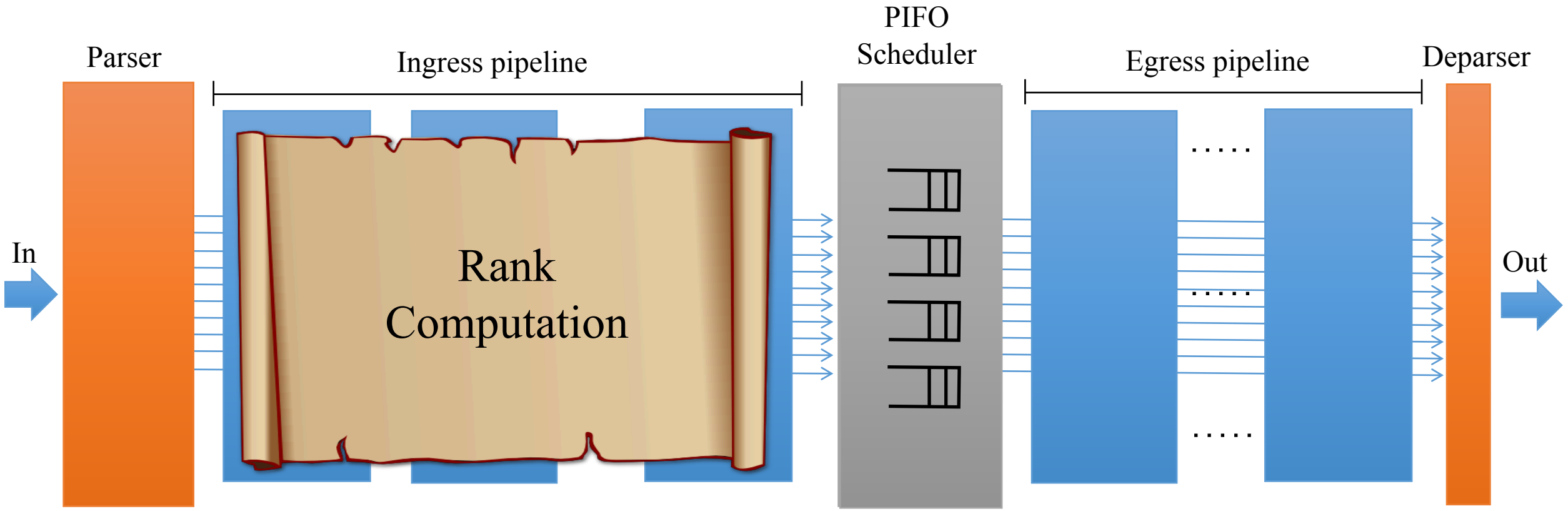


A programmable scheduler

To program the scheduler, program the rank computation

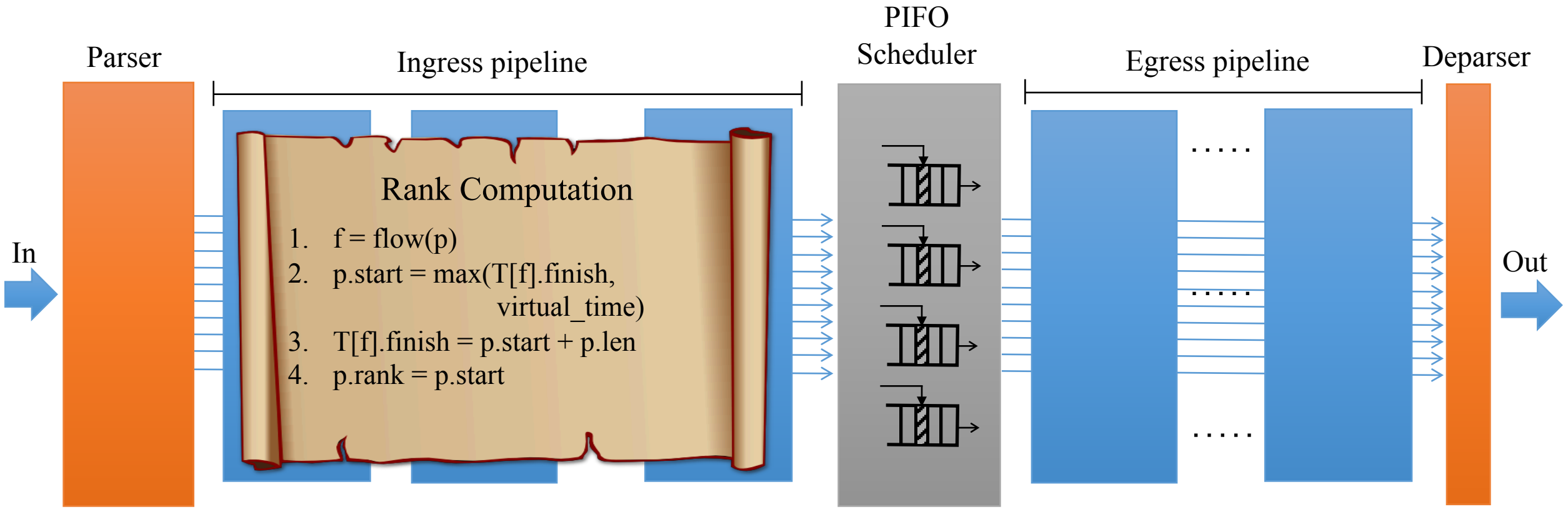


A programmable scheduler

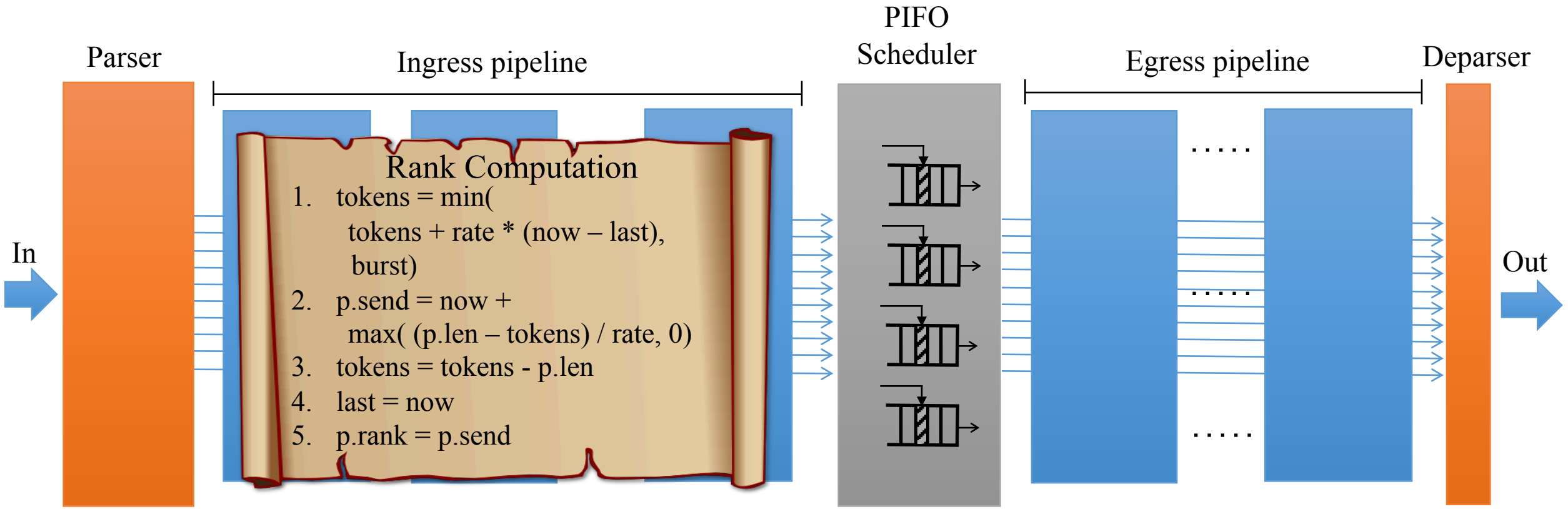


Rank computation is a packet transaction (Domino, SIGCOMM' 16)

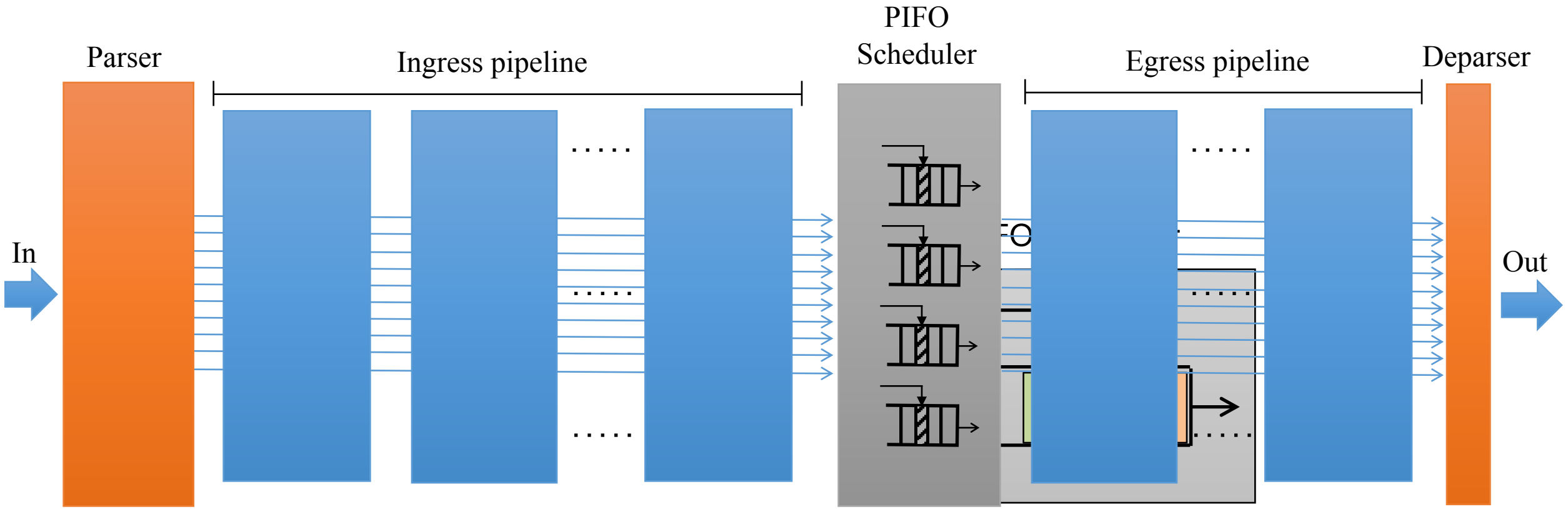
Fair queuing



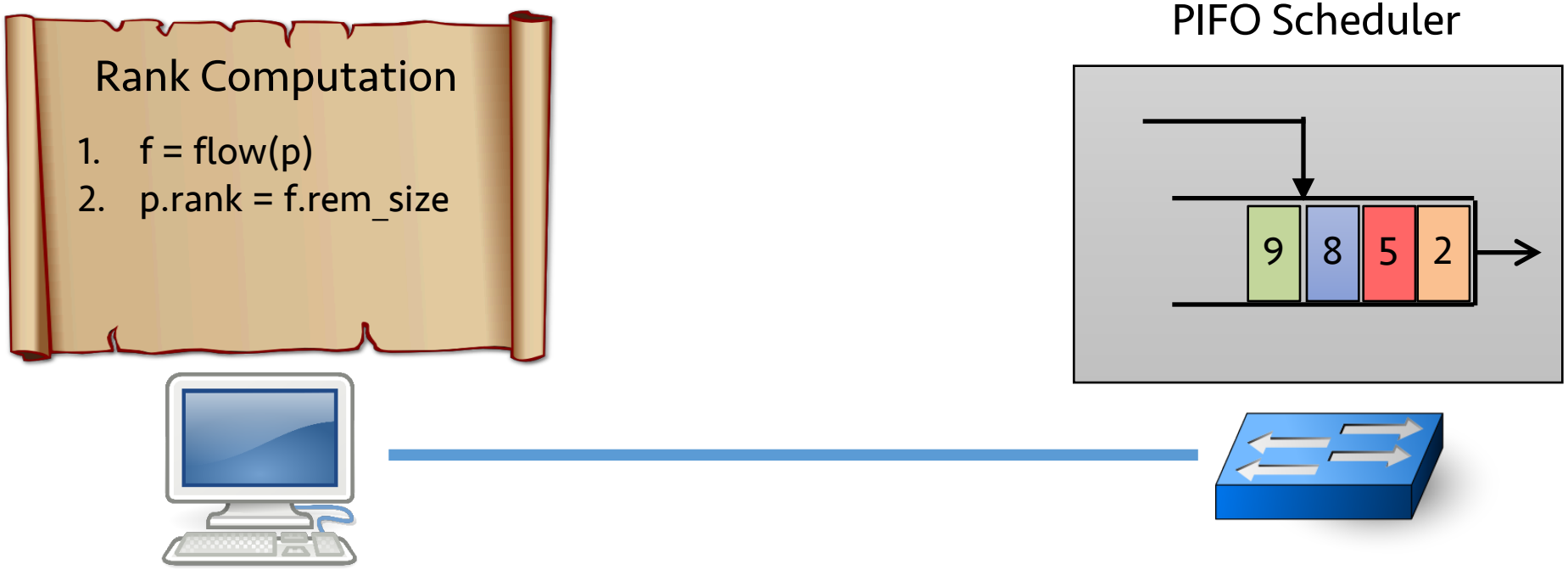
Token bucket shaping



Shortest remaining flow size

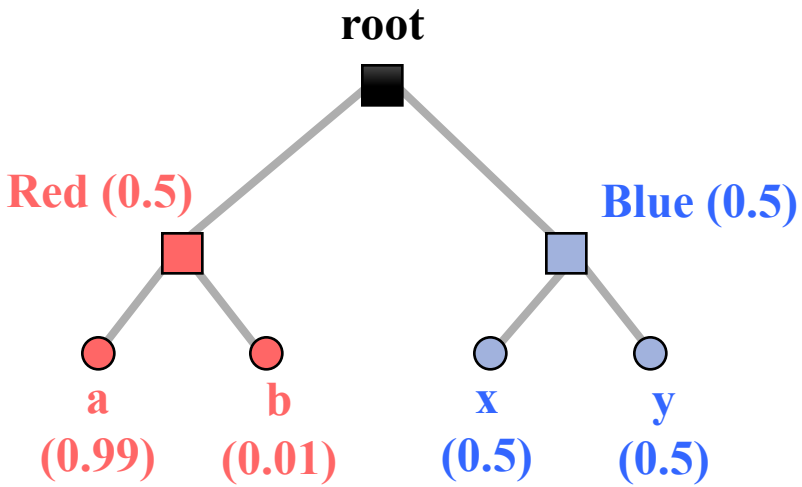


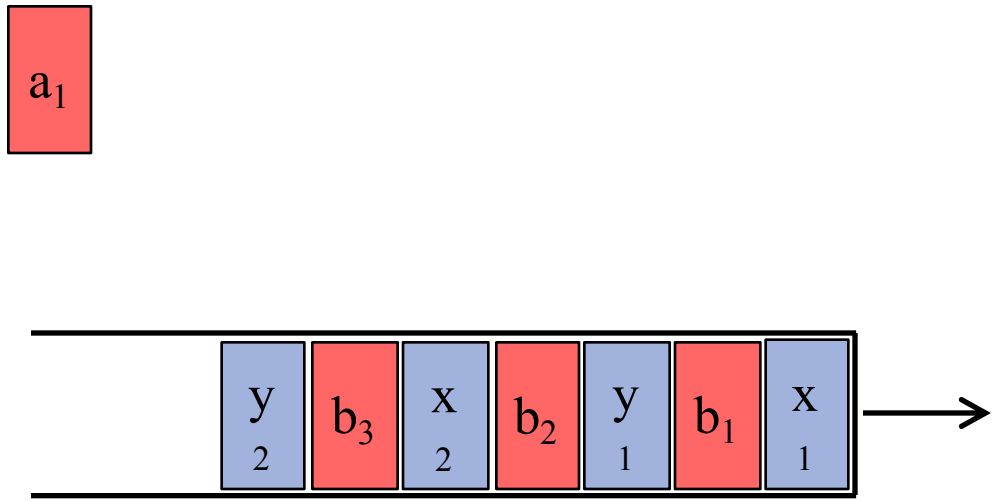
Shortest remaining flow size



Beyond a single PIFO

Hierarchical
Packet Fair Queuing

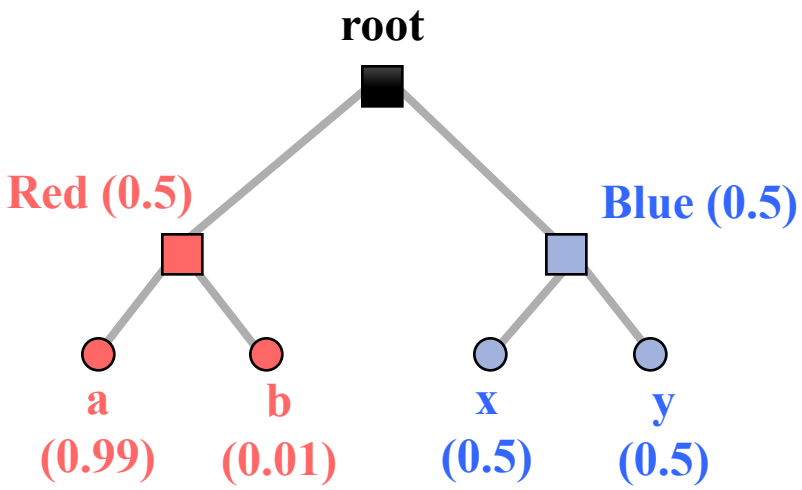


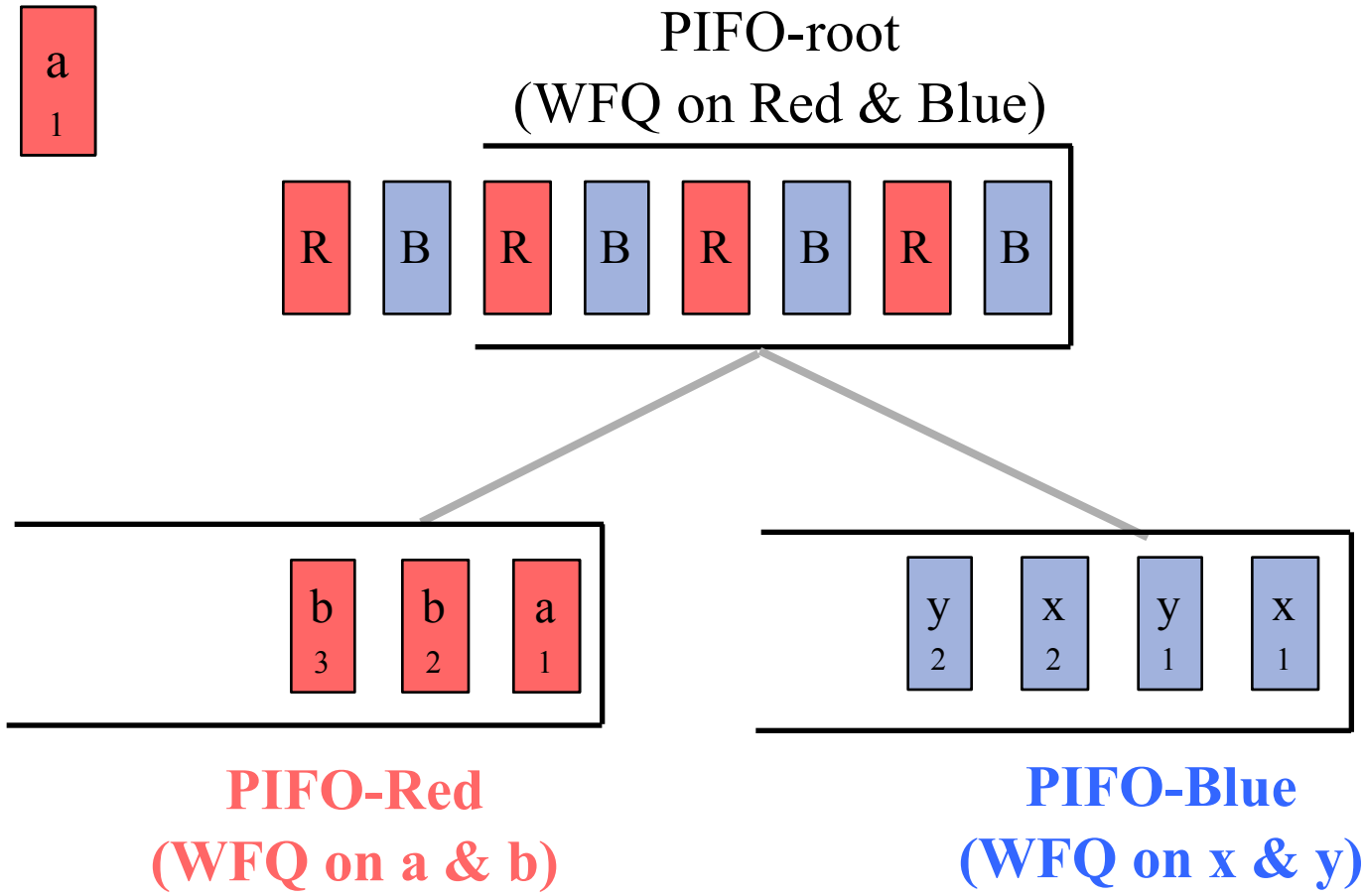


Hierarchical scheduling algorithms need hierarchy of PIFOs

Tree of PIFOs

Hierarchical
Packet Fair Queuing





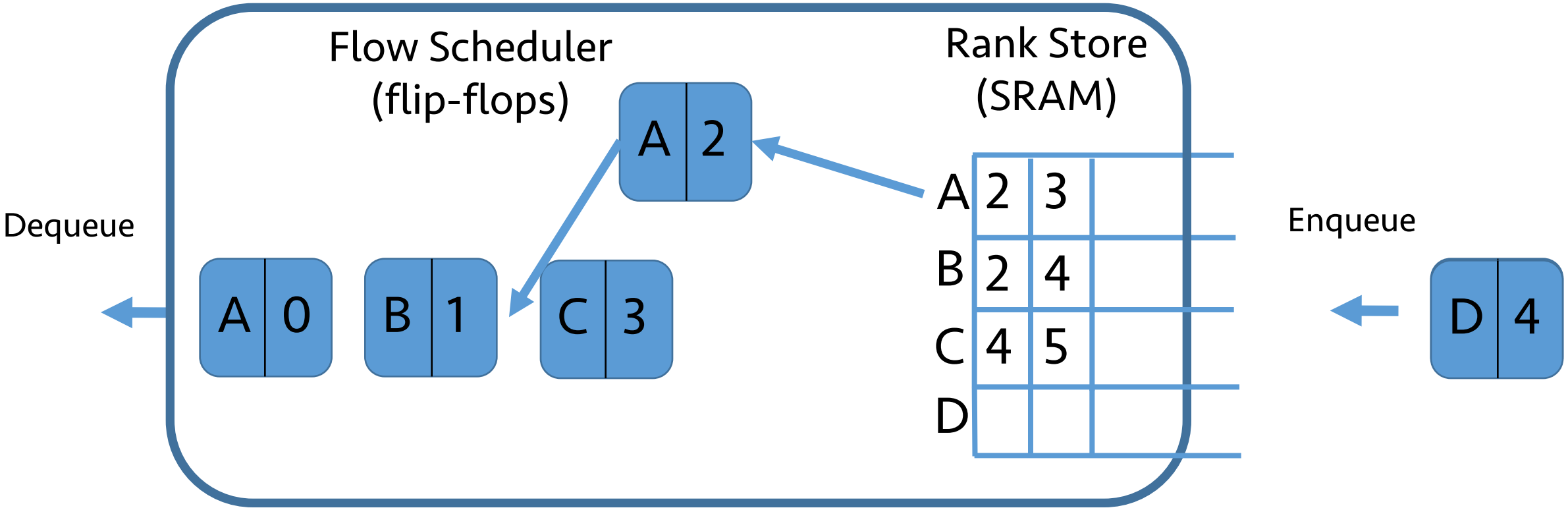
Expressiveness of PIFOs

- Fine-grained priorities: shortest-flow first, earliest deadline first, service-curve EDF
- Hierarchical scheduling: HPFQ, Class-Based Queuing
- Non-work-conserving algorithms: Token buckets, Stop-And-Go, Rate Controlled Service Disciplines
- Least Slack Time First
- Service Curve Earliest Deadline First
- Minimum and maximum rate limits on a flow
- **Cannot express some scheduling algorithms, e.g., output shaping.**

PIFO in hardware

- Performance targets for a shared-memory switch
  - 1 GHz pipeline (64 ports * 10 Gbit/s)
  - 1K flows/physical queues
  - 60K packets (12 MB packet buffer, 200 byte cell)
  - Scheduler is shared across ports
- Naive solution: flat, sorted array is infeasible
- Exploit observation that ranks increase within a flow

A single PIFO block



- 1 enqueue + 1 dequeue per clock cycle
- Can be shared among multiple logical PIFOs

Hardware feasibility

- The rank store is just a bank of FIFOs (well-understood design)
- Flow scheduler for 1K flows meets timing at 1GHz on 16-nm transistor library
  - Continues to meet timing until 2048 flows, fails timing at 4096
- 7 mm^2 area for 5-level programmable hierarchical scheduler
  - < 4% for a typical chip.

Related work

- PIFO: Used in theoretical work by Chuang et. al. in the 90s

- Universal Packet Scheduling (UPS): Uses LSTF to replay all schedules, end point sets slack
  - Assumes fixed switches => cannot express fair queueing, shaping
  - Assumes single priority queue => cannot express hierarchies

Conclusion

- Programmable scheduling at line rate is within reach
- Two benefits:
  - Express new schedulers for different performance objectives
  - Express existing schedulers as software, not hardware
- Code: http://web.mit.edu/pifo

Backup slides

Limitations of PIFOs

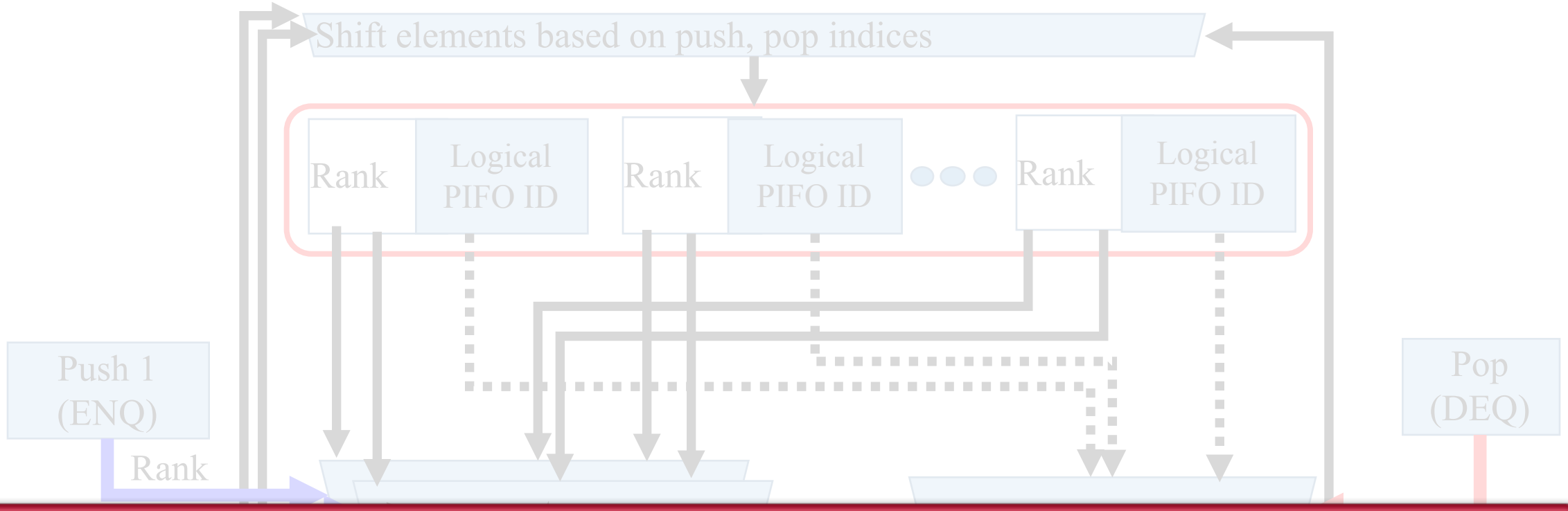
- Output shaping: PIFOs rate limit input to a queue, not output
- Shaping and scheduling are coupled.

PIFO mesh

Proposal: scheduling in P4

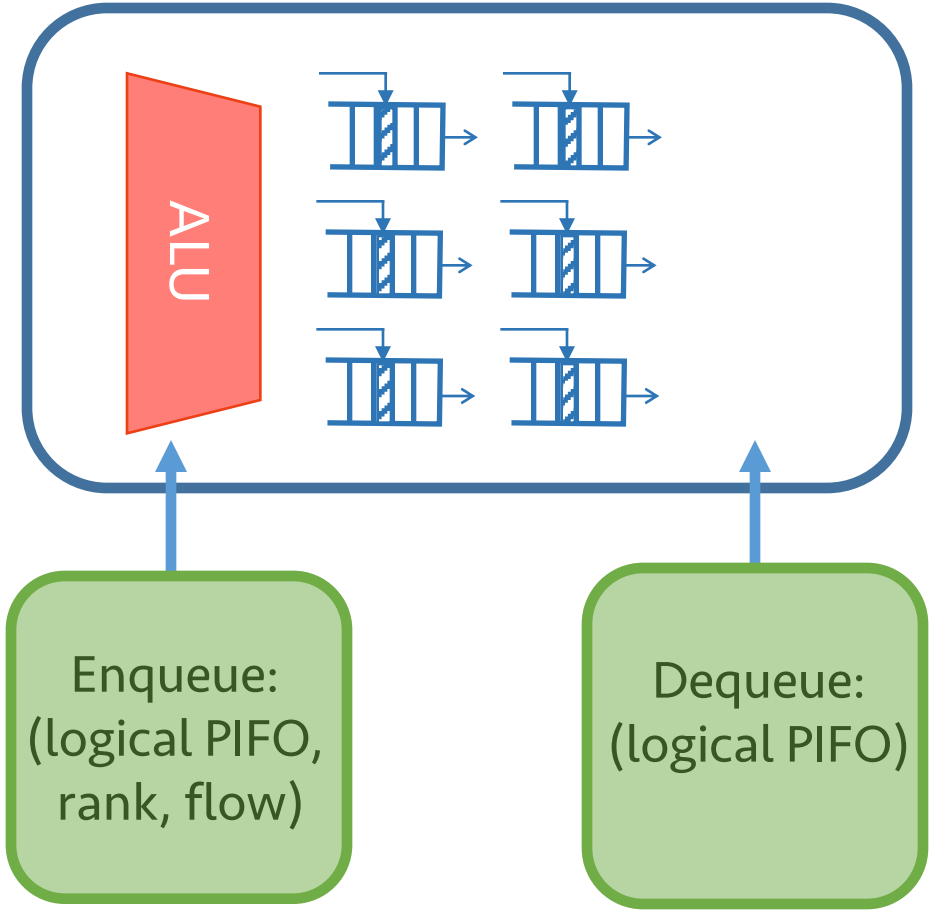
- Currently not modeled at all, blackbox left to vendor
- Only part of the switch that isn't programmable
- PIFOs present a candidate
- Concurrent work on Universal Packet Scheduling also requires a priority queue that is identical to a PIFO

Hardware implementation

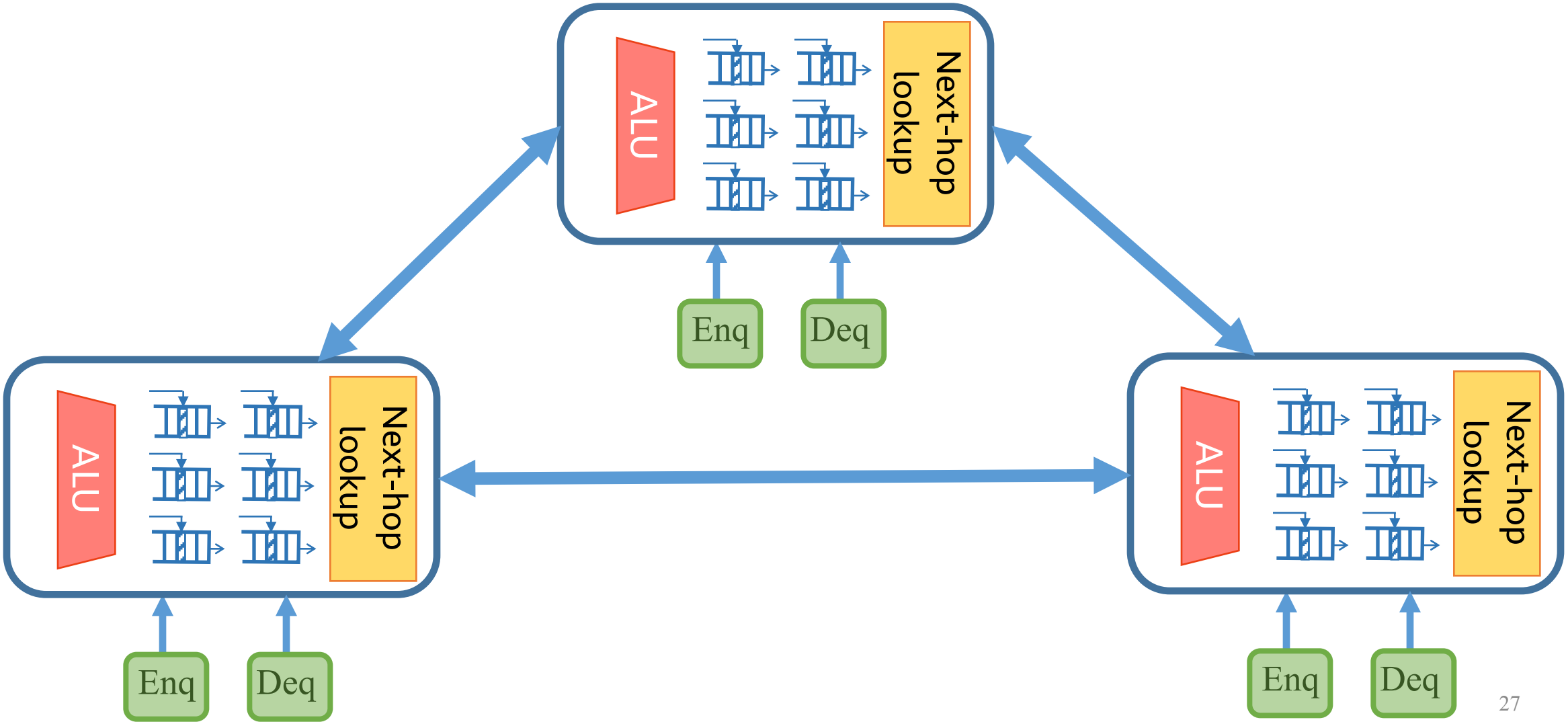


- Meets timing (1 GHz) for up to 2048 flows at 16 nm
- Less than 4% area overhead (~7 mm^2) for 5-level scheduler

A PIFO block



A PIFO mesh


ALU
Next-hop lookup
Enq
Deq
ALU
Next-hop lookup
Enq
Deq
ALU
Next-hop lookup
Enq
Deq

Proposal: scheduling in P4

- Need to model a PIFO (or priority queue) in P4

- Requires an extern instance to model a PIFO
  - Can start by including it in a target-specific library
  - Later migrate to standard library if there's sufficient interest
  - Section 16 of P4v1.1

- Transactions themselves can be compiled down to P4 code using the Domino DSL for stateful algorithms.

Hardware feasibility of PIFOs

- Number of flows handled by a PIFO affects timing.

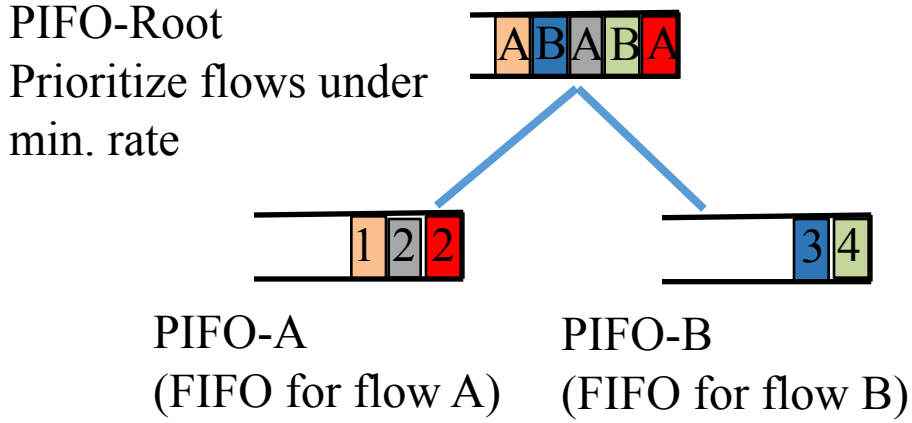
- Number of logical PIFOs within a PIFO, priority and metadata width, and number of PIFO blocks only increases area.

Composing PIFOs: min. rate guarantees

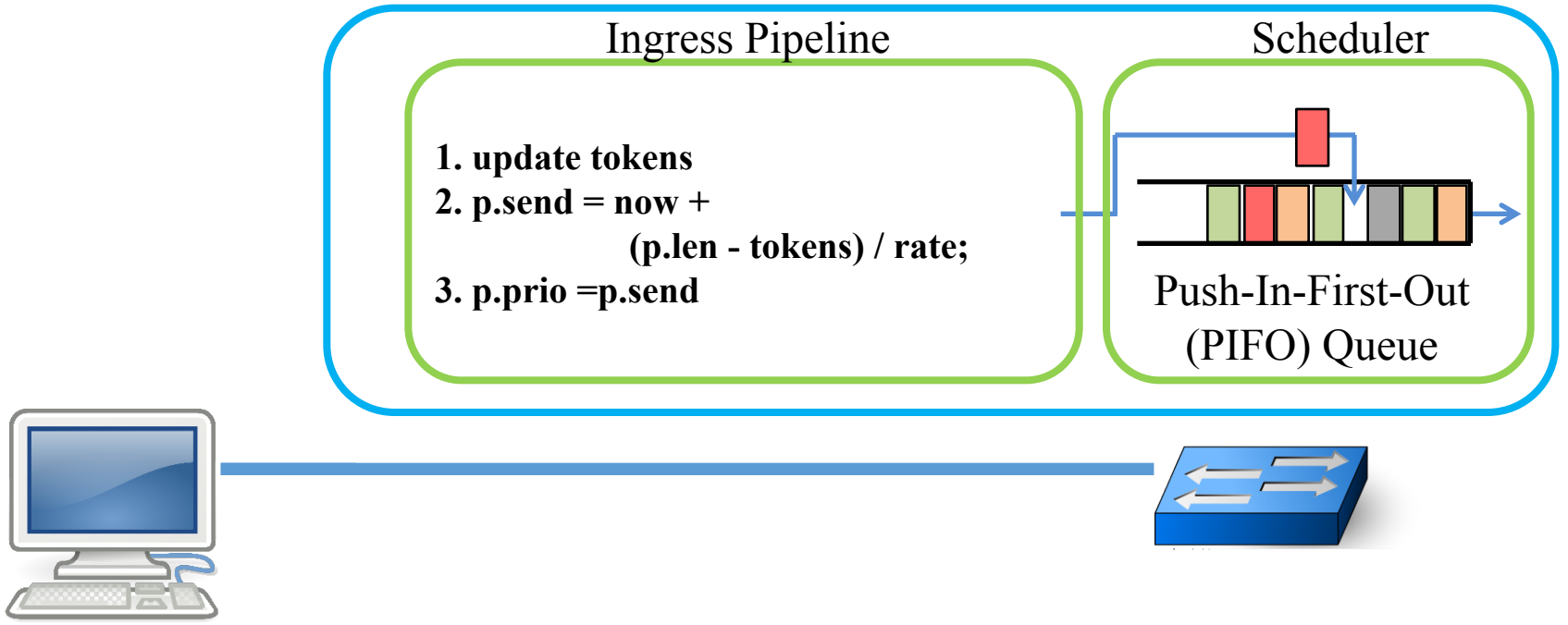
Minimum rate guarantees:

Provide each flow a guaranteed rate provided the sum of these guarantees is below capacity.

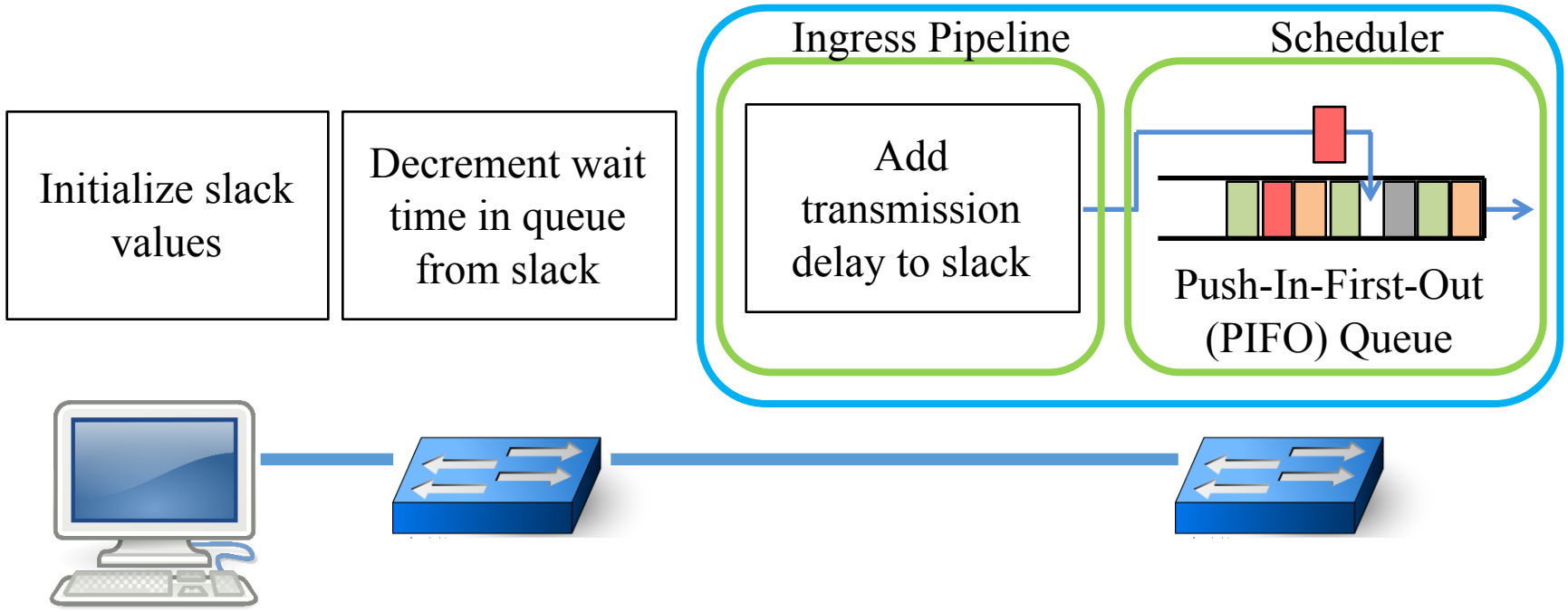
Composing PIFOs



Traffic Shaping



LSTF


Initialize slack values
Decrement wait time in queue from slack
Ingress Pipeline
Add transmission delay to slack
Scheduler
Push-In-First-Out (PIFO) Queue

The PIFO abstraction in one slide

- PIFO: A sorted array that let us insert an entry (packet or PIFO pointer) into a PIFO based on a programmable priority
- Entries are always dequeued from the head
- If an entry is a packet, dequeue and transmit it
- If an entry is a PIFO, dequeue it, and continue recursively