Programmable Packet Scheduling at Line Rate

Mohammad Alizadeh

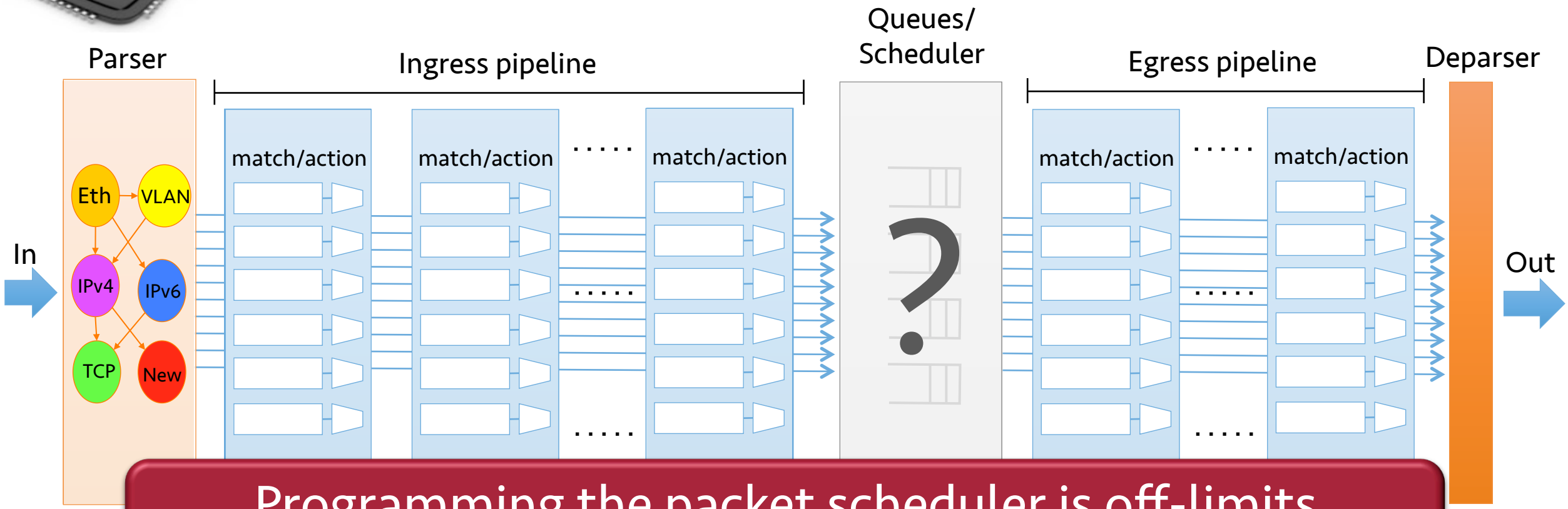
Massachusetts Institute of Technology

Joint work with

Anirudh Sivaraman
(MIT)

- MIT: Suvinay Subramanian, Hari Balakrishnan
- Cisco: Da Chuang, Sharad Chole, Tom Edsall
- Barefoot: Anurag Agrawal
- Stanford: Sachin Katti, Nick McKeown

Programmable switching chips



Programming the packet scheduler is off-limits for today's switching chips

Why is programmable scheduling hard?

- Decades of scheduling algorithms, but no consensus on abstractions for scheduling; in contrast to
  - Parse graphs for parsing
  - Match-action tables for forwarding

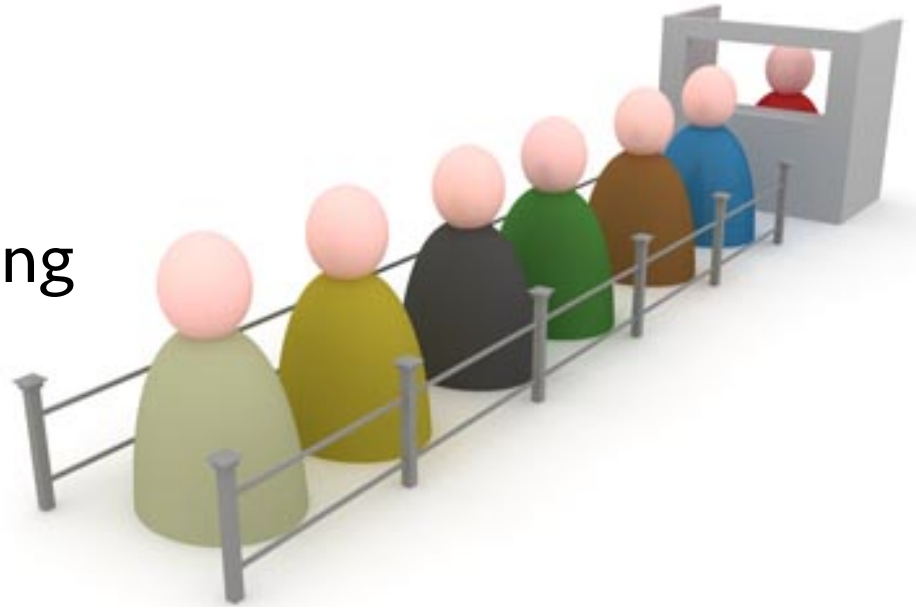
- The scheduler has very tight timing requirements
  - One decision per clock cycle is typical

Need expressive abstraction that can be implemented 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

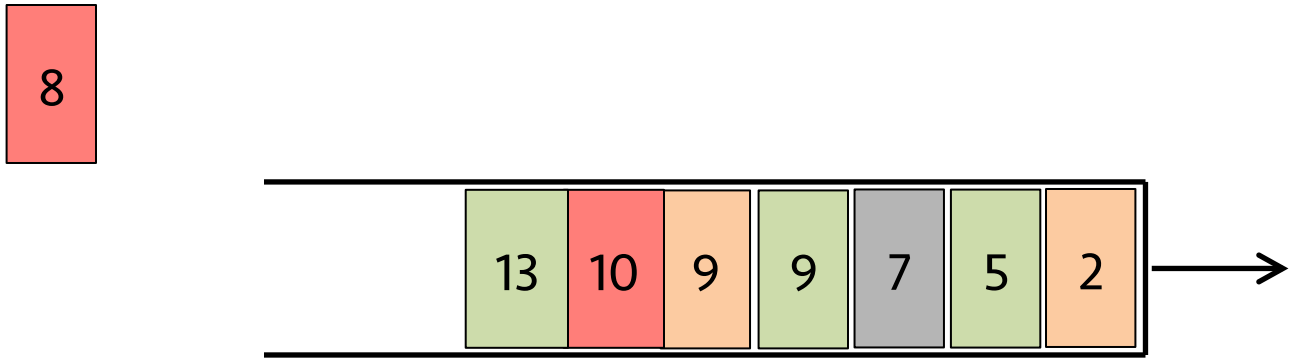


Key observation

- In many algorithms, the scheduling order/time can be determined on enqueue
- i.e., relative order of buffered packets does not change

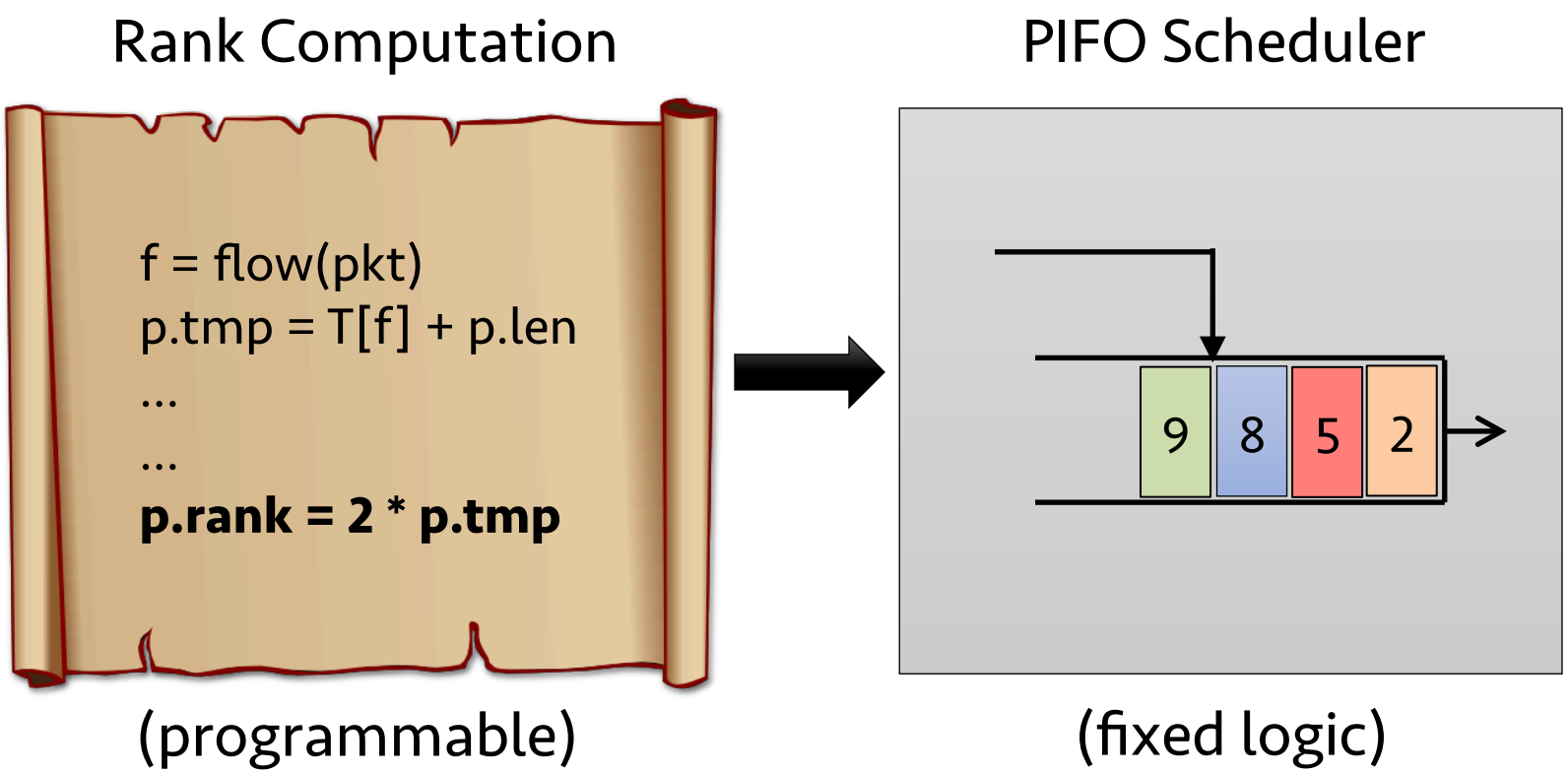
The Push-In First-Out Queue

- Packets are pushed into an arbitrary location based on a **rank** number, and dequeued from the head
  - First used as a proof construct by Chuang et. al. in the 90s
  - Also a powerful construct for programmable scheduling

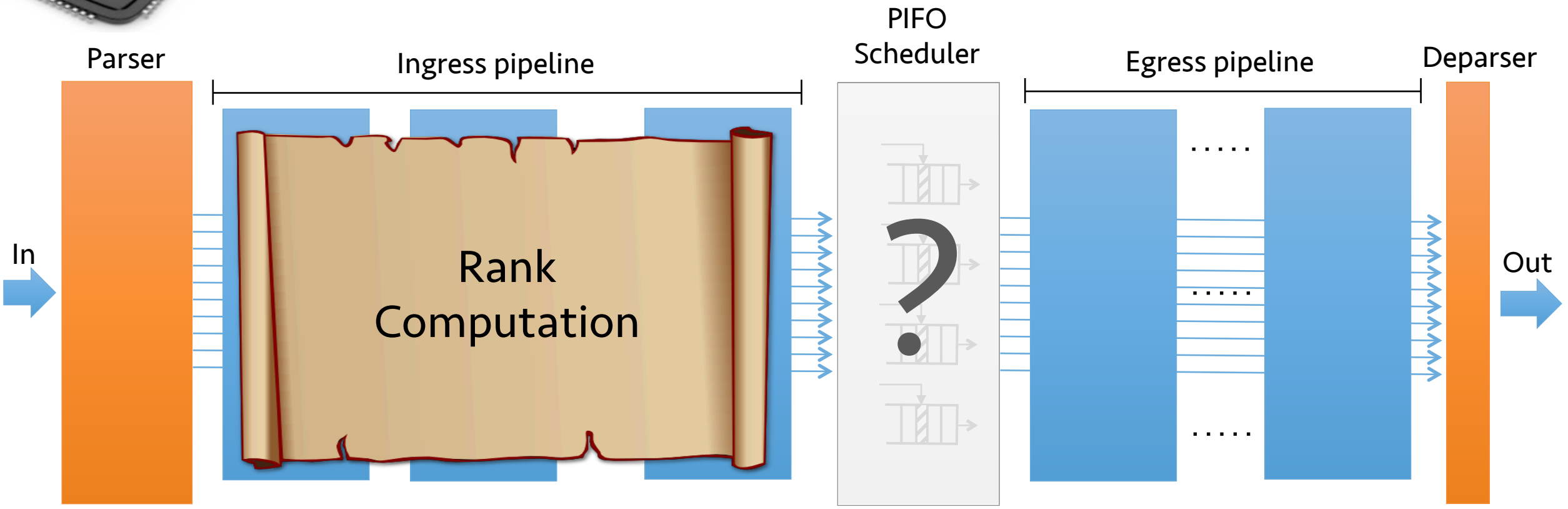


A programmable scheduler

To program the scheduler, program the rank computation

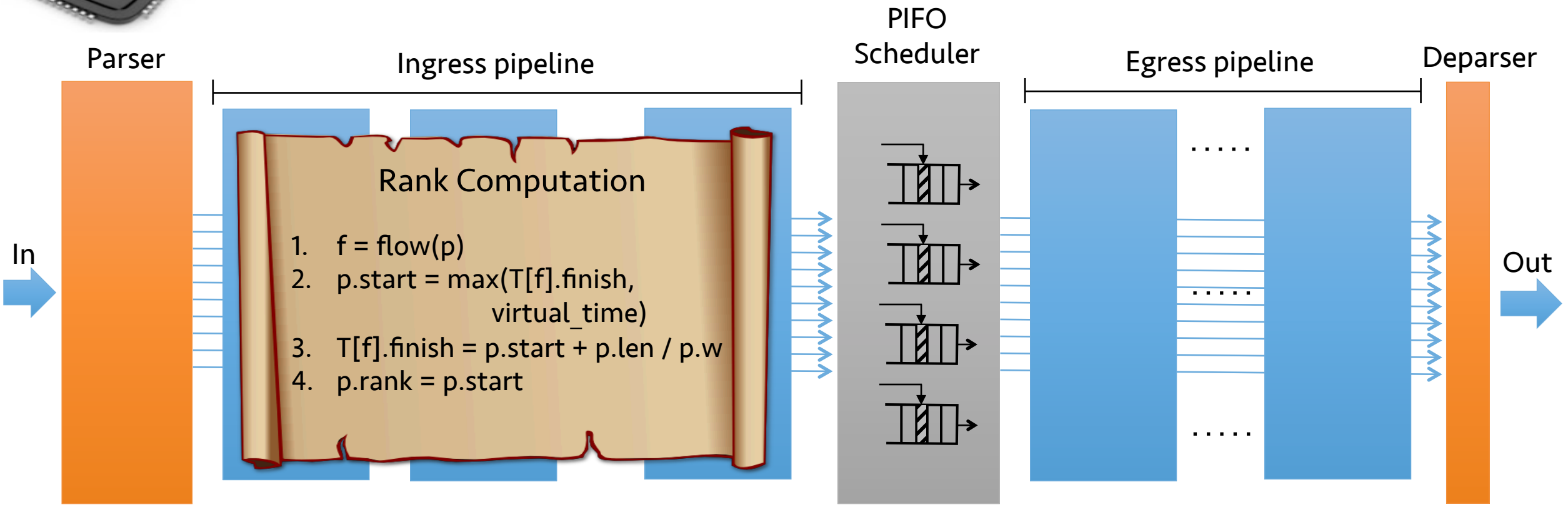


A programmable scheduler

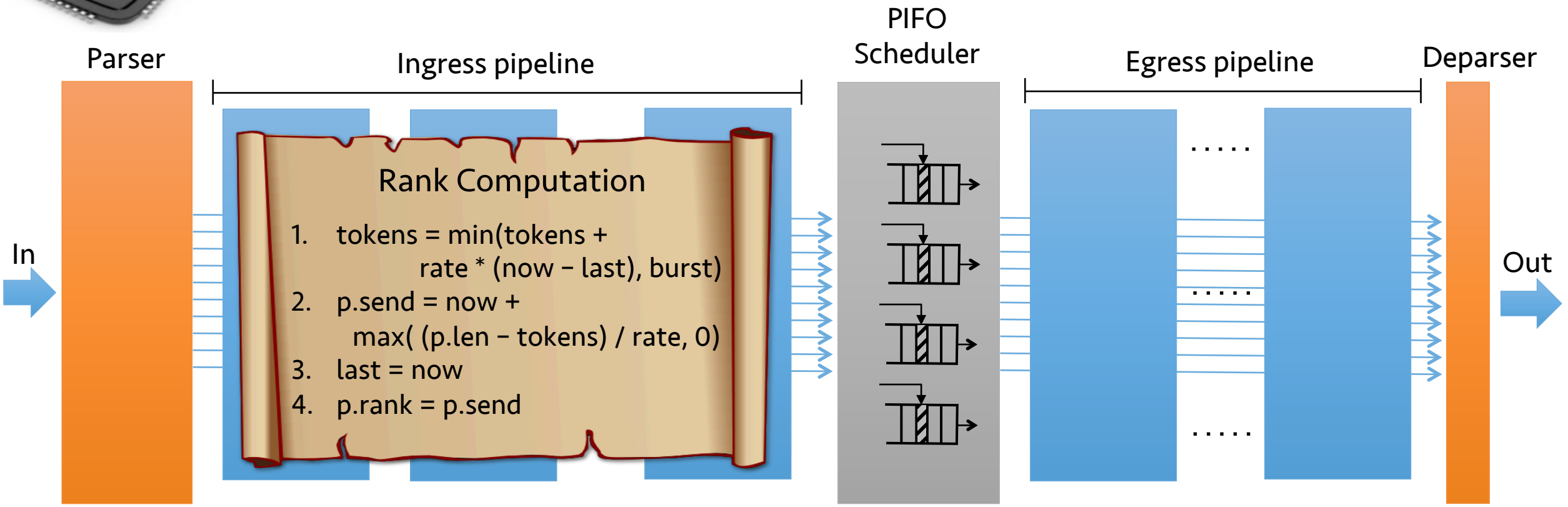


- Rank computation expressed in P4, or compiled to P4 using Domino DSL

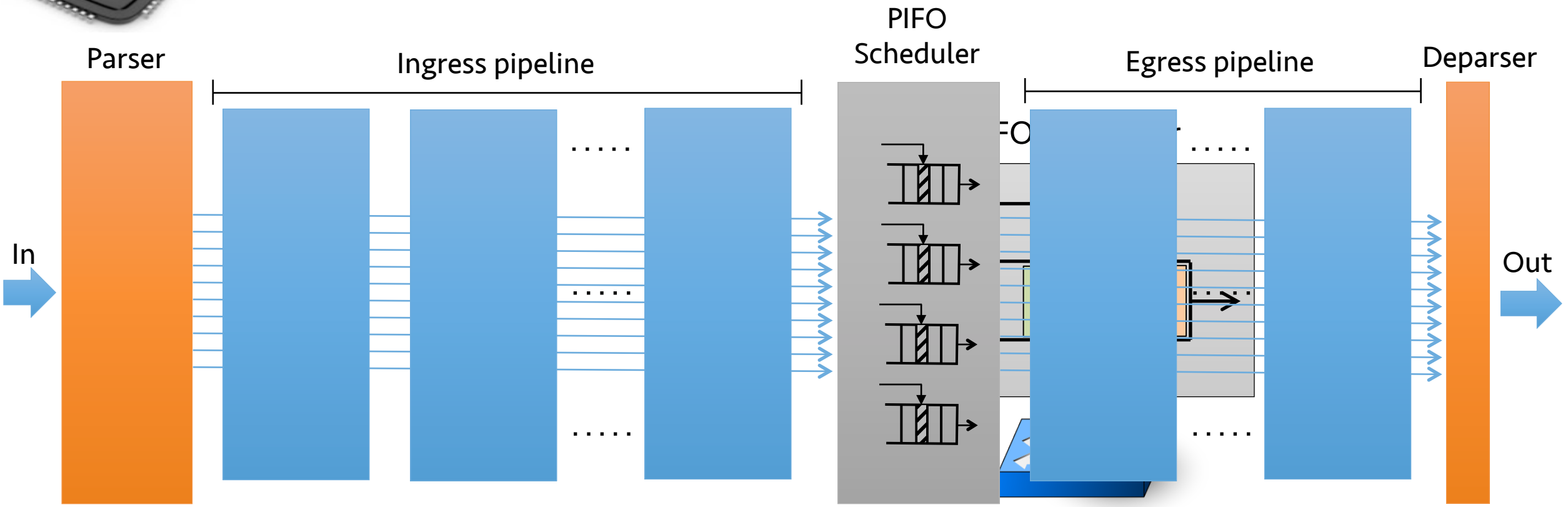
Weighted Fair Queuing



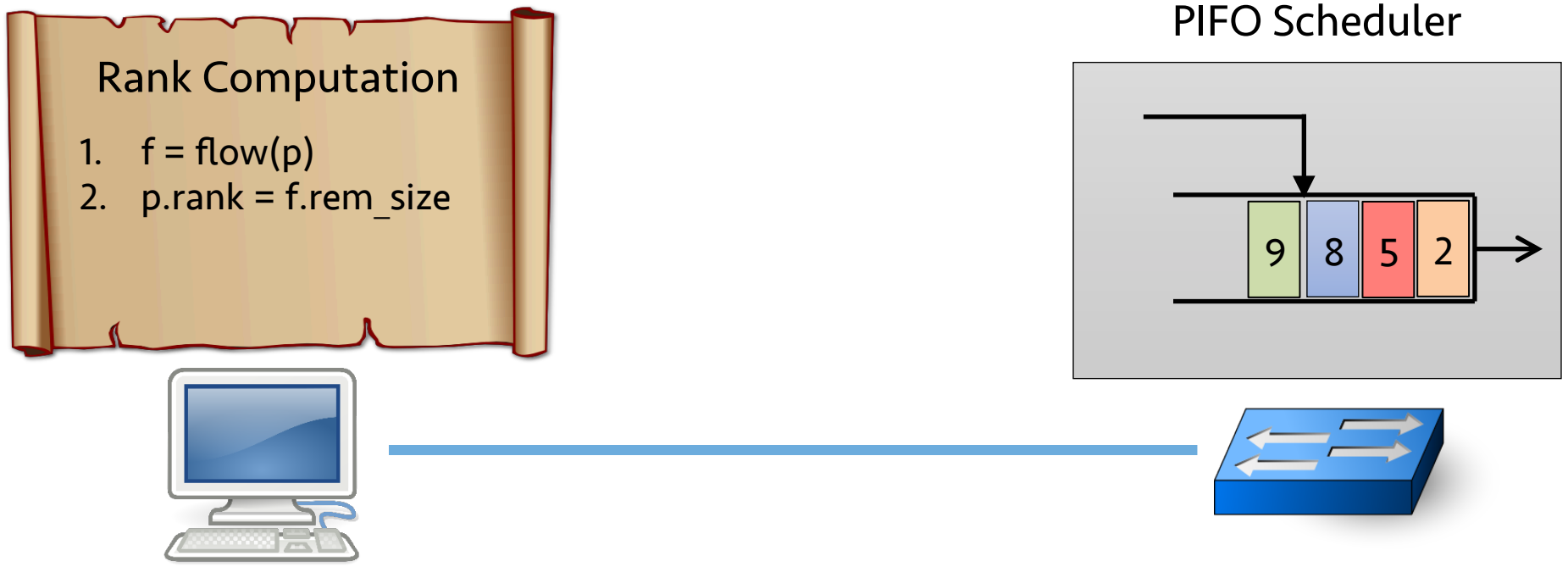
Token bucket shaping



pFabric (SRPT)

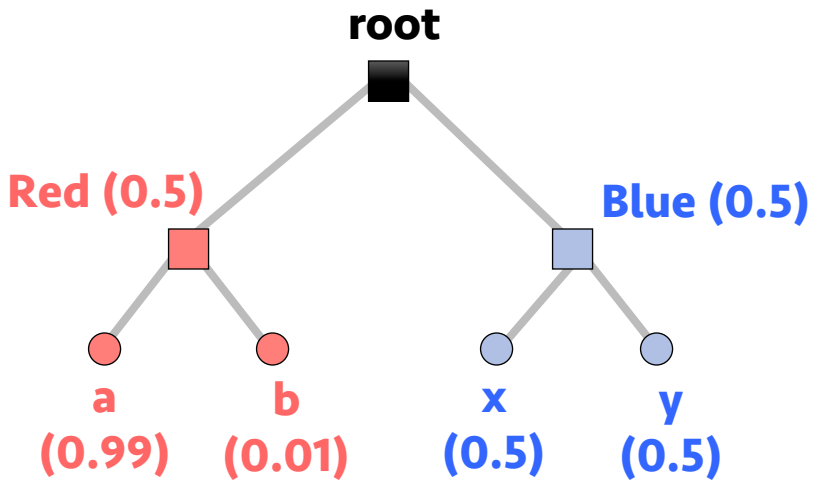


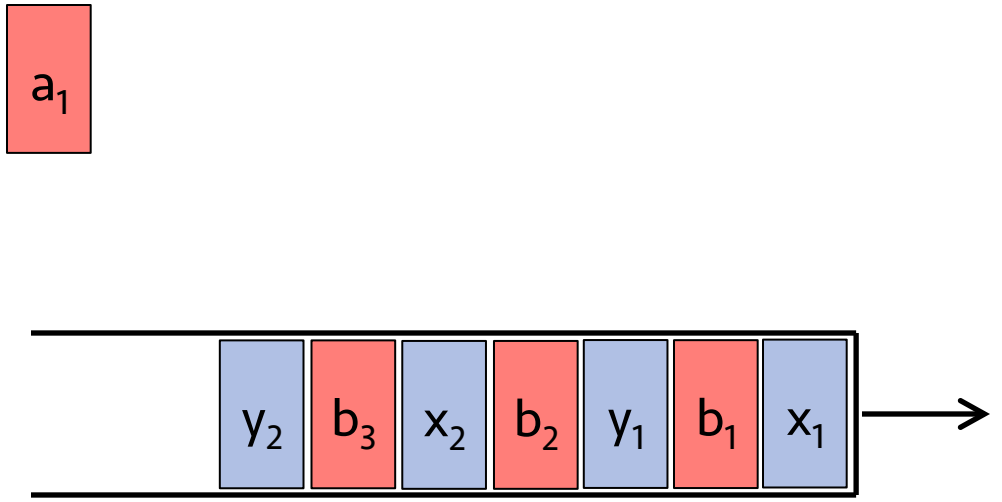
pFabric (SRPT)



Beyond a single PIFO

Hierarchical Packet Fair Queuing

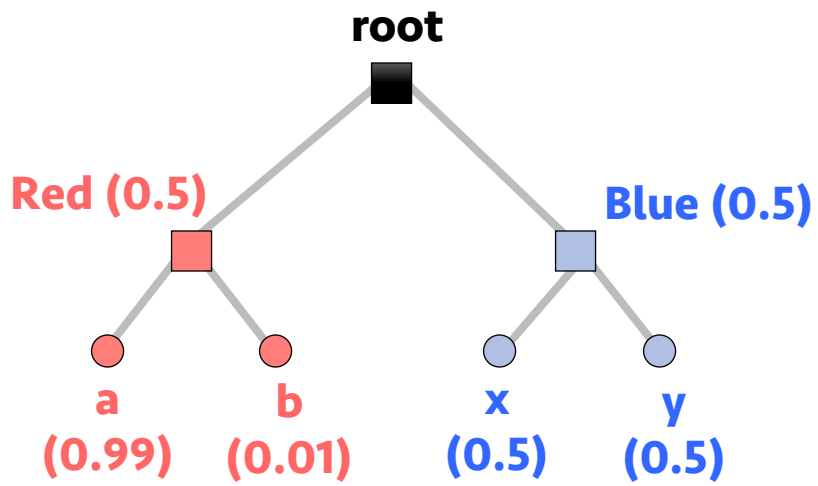


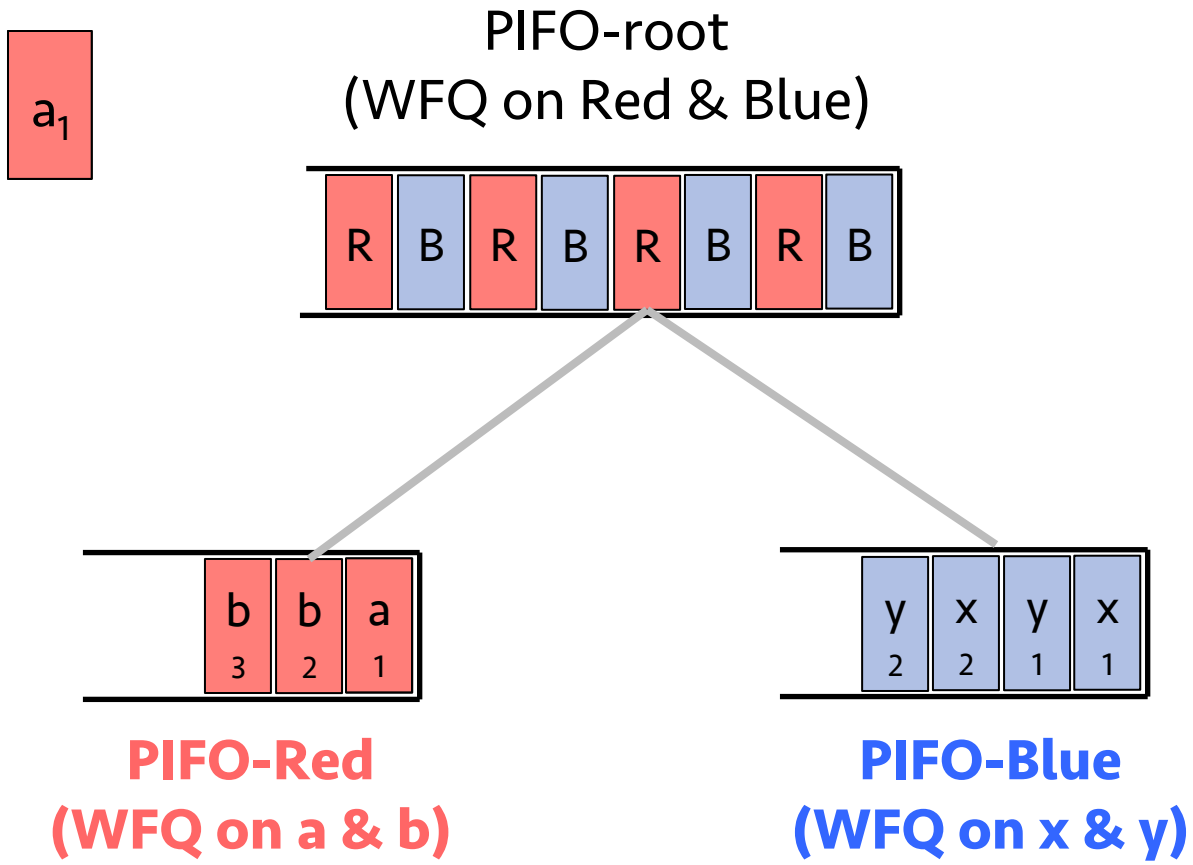


Hierarchical scheduling algorithms need hierarchy of PIFOs

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

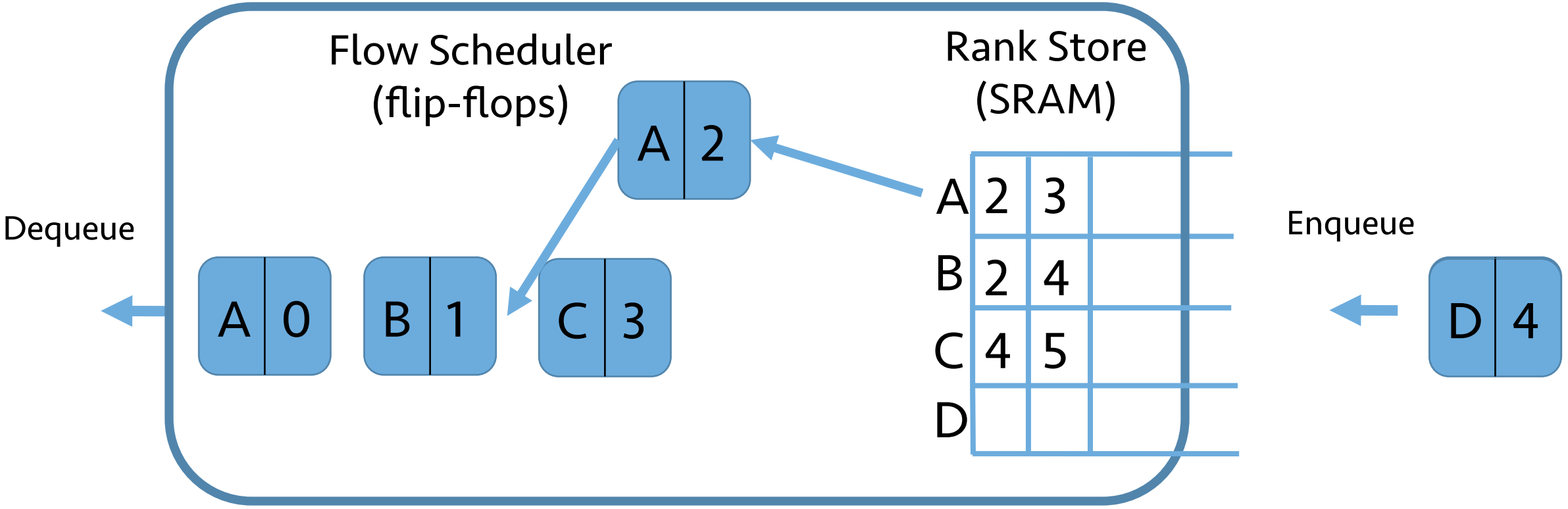
PIFO in hardware

- Performance requirements, based on standard single-chip shared-memory switch (e.g., Broadcom Trident)
  - 1 GHz pipeline
  - 1K flows/physical queues
  - 60K packets (12 MB packet buffer, 200 byte cell)

- Naive solution: flat, sorted array, doesn't scale

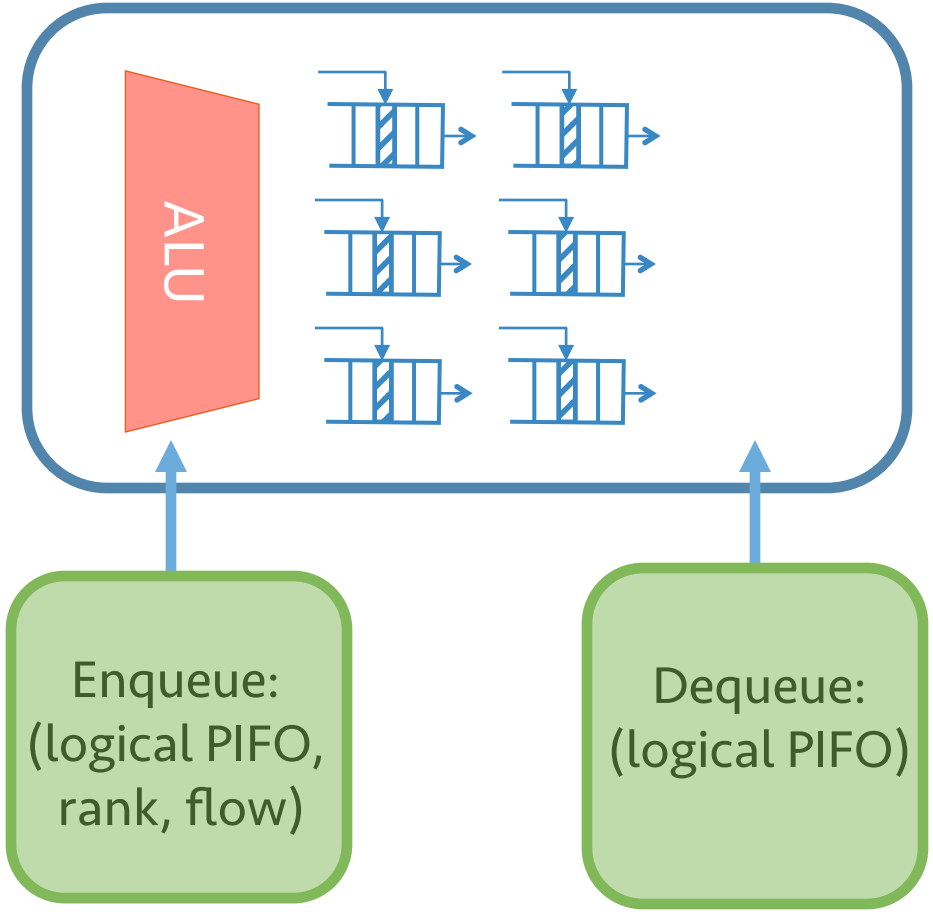
- Scalable solution: use fact that ranks increase within a flow

A PIFO block

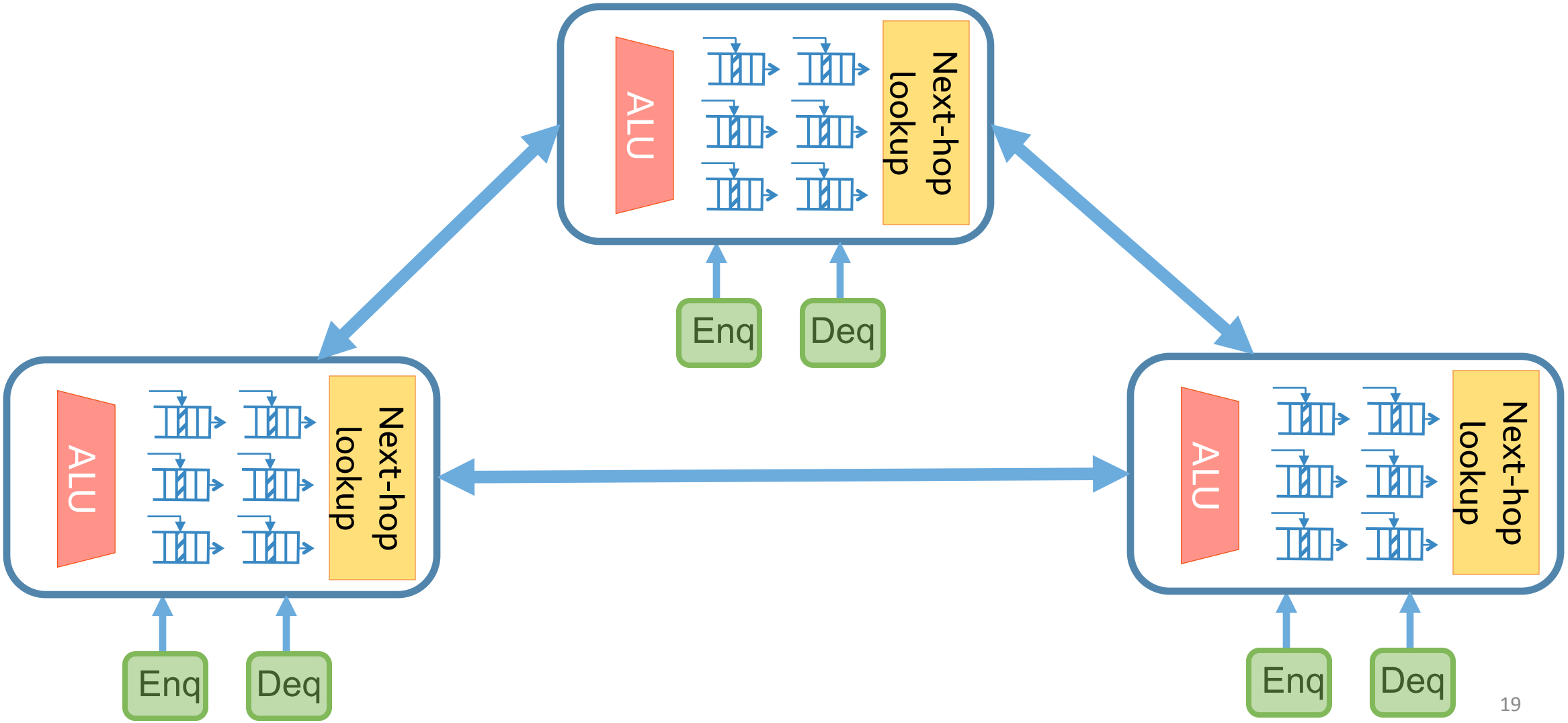


- 1 enqueue + 1 dequeue per clock cycle
- Can be shared among multiple logical PIFOs

A PIFO block



A PIFO mesh


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

Hardware feasibility

- The Rank store is just a bank of FIFOs (stable hardware IP)
- 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
- E.g., < 4% area overhead to program 5-level hierarchies (5-block PIFO mesh)

Summary & Next Steps

- PIFO is a promising abstraction for packet scheduling
  - Can express a wide range of algorithms
  - Can be implemented at line rate with modest overhead
- Preprints of paper appearing at SIGCOMM 2016:
  - http://arxiv.org/abs/1602.06045
- Next steps
  - FPGA implementations, PIFO-capable switching chips
  - Language support in P4 (extern objects?)
- Would love feedback