Programmable Packet Scheduling at Line Rate

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

(CSE 294 Winter 2017)

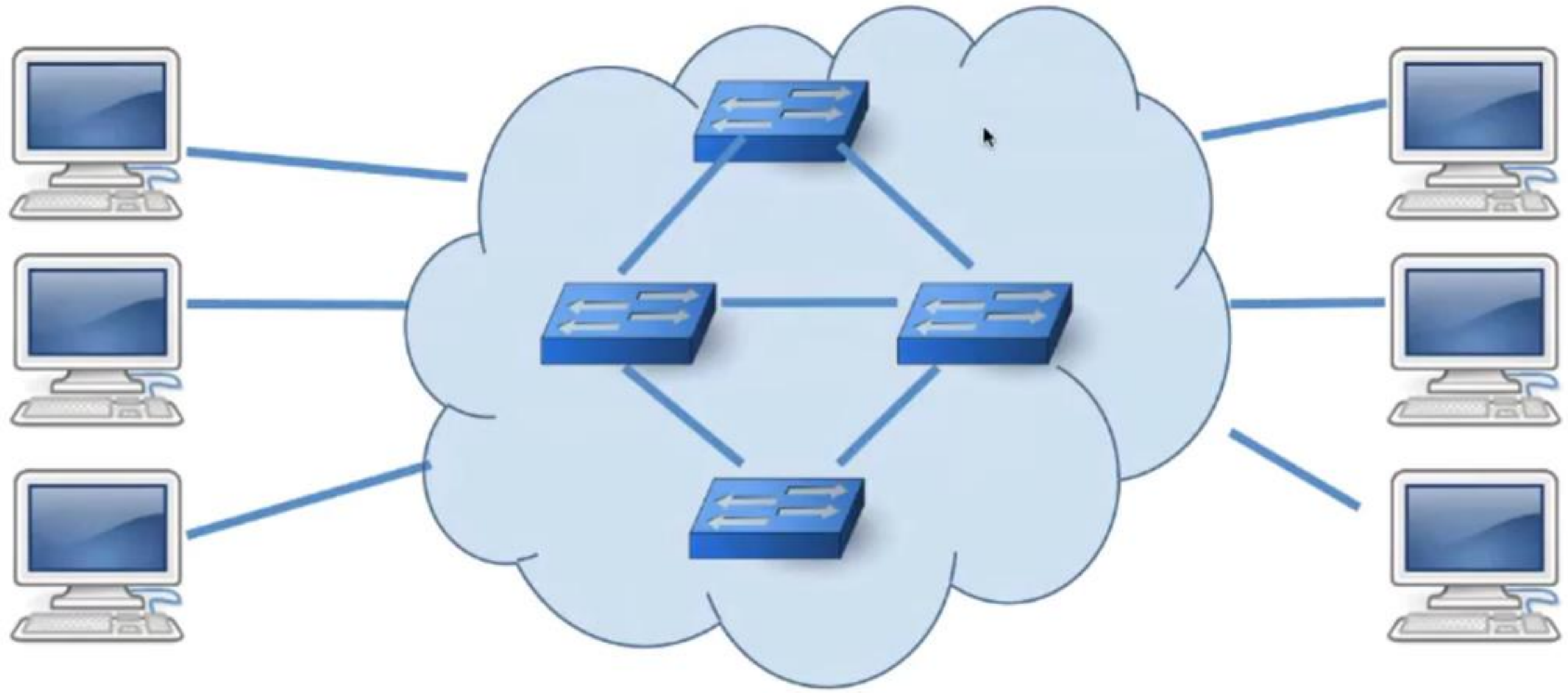
Massachusetts Institute of Technology

BAREFOOT NETWORKS

CISCO

Stanford University

Traditional networking



Fixed (simple) switches and programmable (smart) end points

This is showing signs of age …

- Switch features tied to ASIC design cycles (2-3 years)
  - Long lag time for new protocol formats (IPv6, VXLAN)

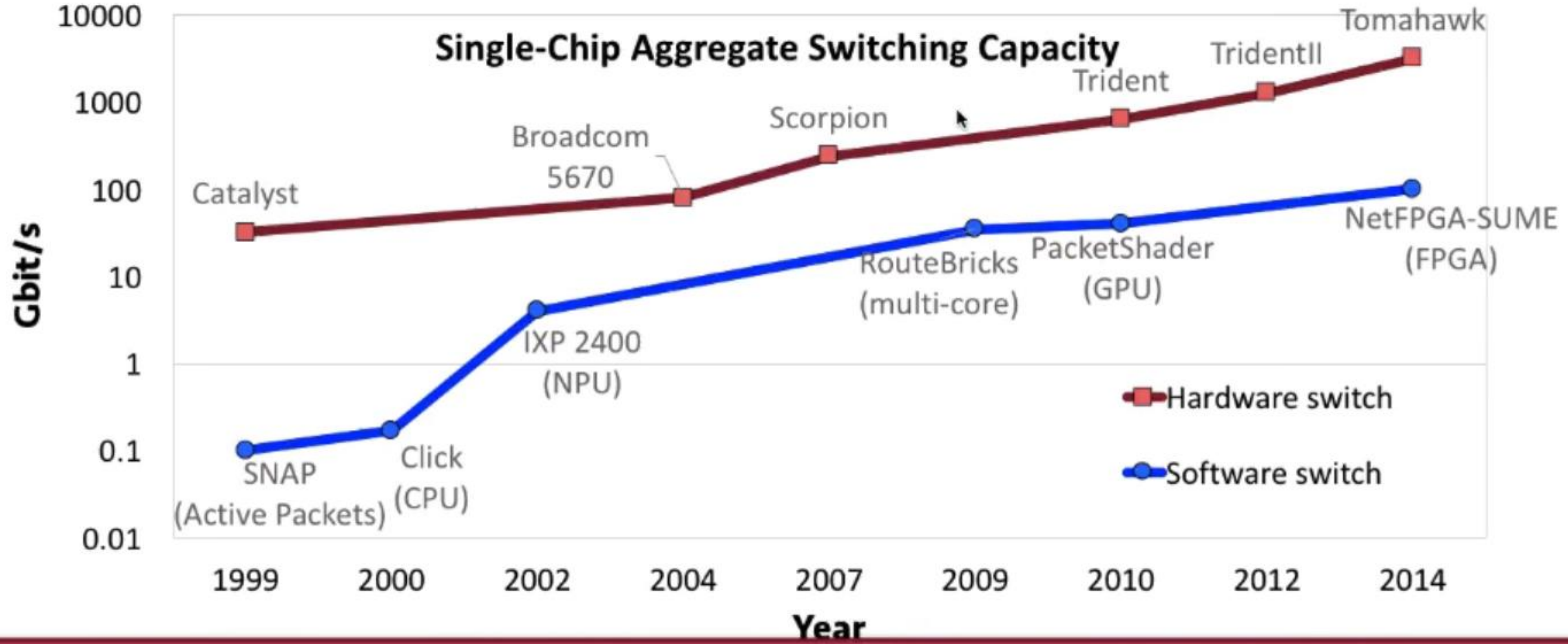
- Operators (esp. in datacenters) need more control over switches
  - Access control, load balancing, bandwidth sharing, measurement

- Many switch algorithms never make it to production

The quest for programmable switches

- Early switches built out of minicomputers, which were sufficient
    - IMPs (1969): Honeywell DDP-516
    - Fuzzball (1971): DEC LSI-11
    - Stanford multiprotocol switch (1981): DEC PDP 11
    - Proteon / MIT C gateway (1980s): DEC MicroVAX II

The quest for programmable switches



Software switches (CPUs, NPUs, GPUs, FPGAs) are 10—100x slower

The vision: programmability at line rate

- Performance of fastest, fixed-function switches (> 1 Tbit/s)

- More programmable than fixed-function switches
  - Much more than OpenFlow/SDN, which only programs routing/control plane.
  - …, but less than software switches

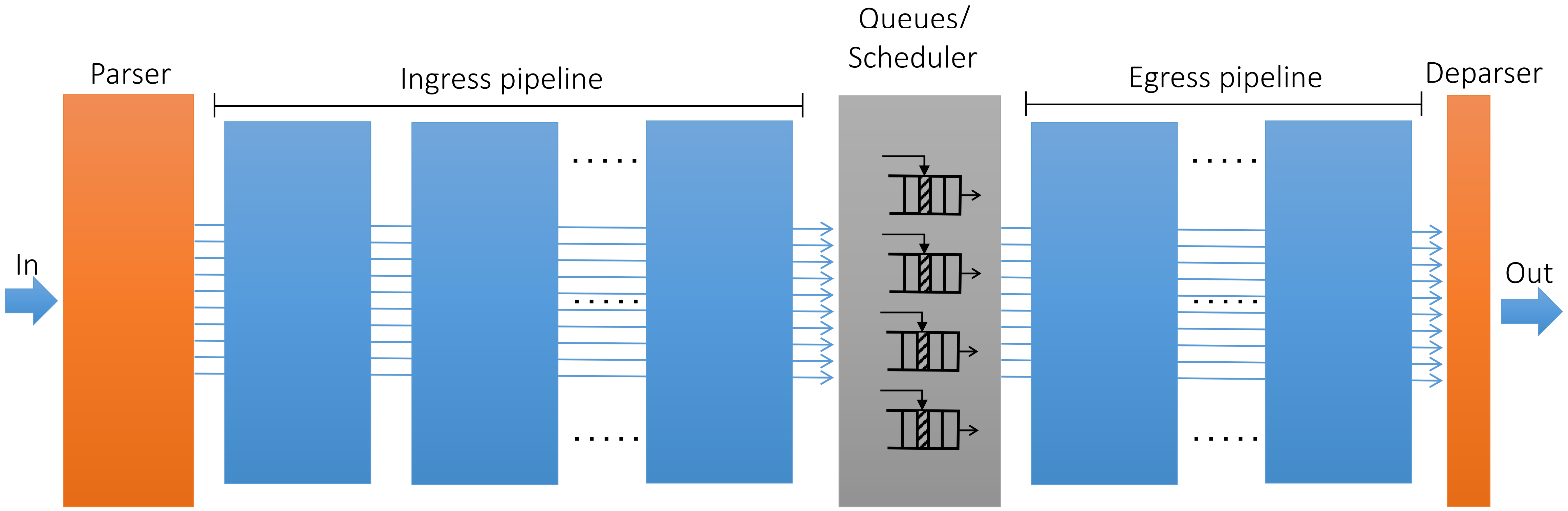
- Such programmable chips are emerging: Tofino, FlexPipe, Xpliant
  - As are languages such as P4 to program them

Programmable scheduling at line rate

- Programmable: Can we express a new scheduling algorithm?

- Line-rate: Highest capacity supported by a communication standard
  - Can we put a CPU or an FPGA in the fast path? No!
  - Software Switch will not work – not fast enough for line rate

Architecture of a switch

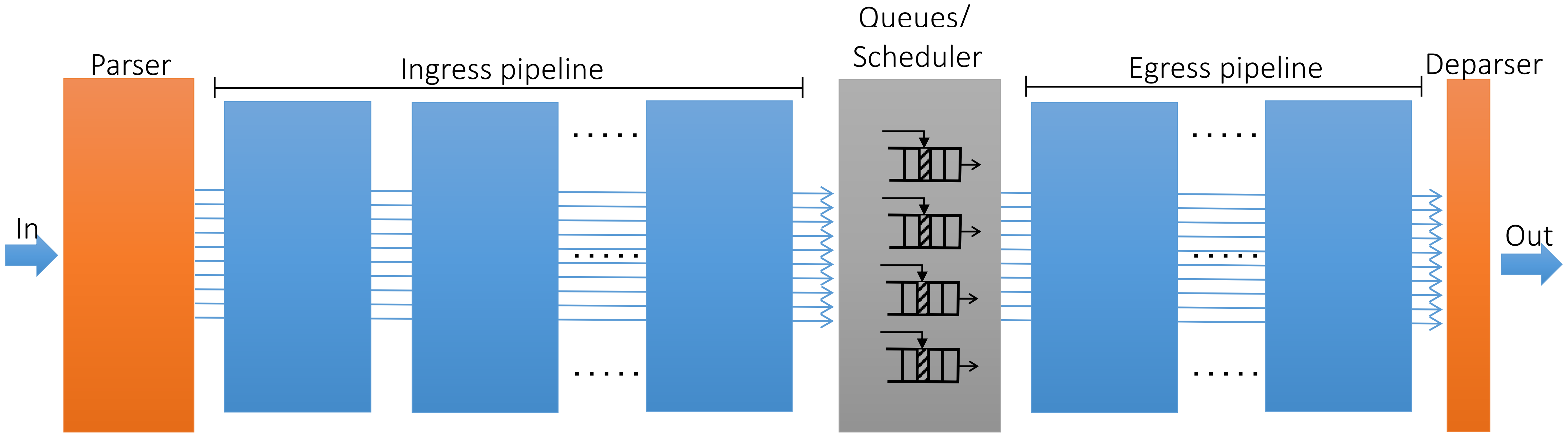


Parser : Turns bytes into packets.
Ingress pipeline: Lookup tables operates on these packets. (Match Action)
Scheduler: a set of output queues, typically a few for each port.
Egress pipeline: process packets after they are scheduled
Deparser: turn packets back into bytes on the wire.

State of the art in line-rate programmability.

One of the goals of Software-Defined Networking is to make commodity line-rate switches programmable.

Is that enough for programmable scheduling?



Why is programmable scheduling harder?

- 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

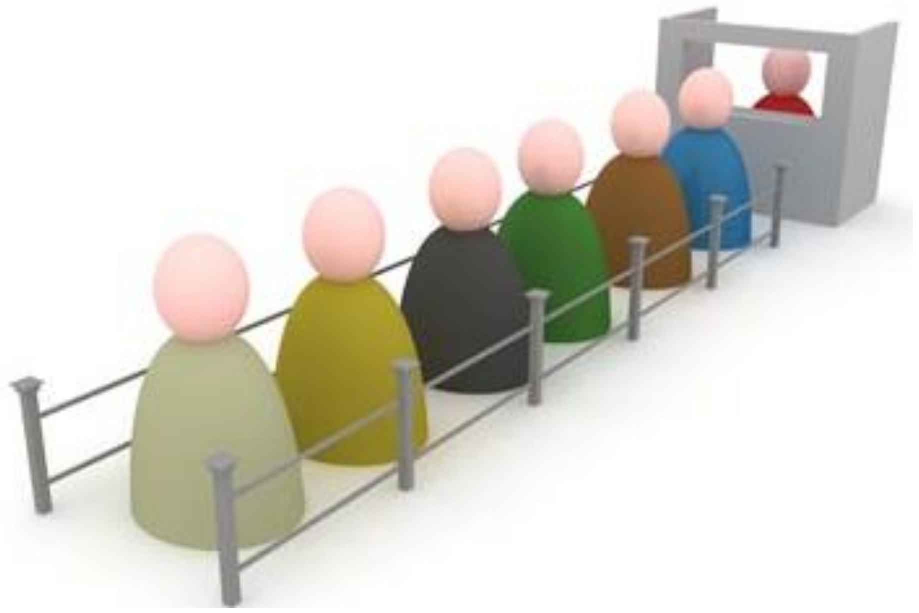
Need expressive abstraction that can run at line rate

At a very high level…

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



So a programmable scheduler will have to provide a way to flexibly specify both order and time.

Key ideas from Scheduling

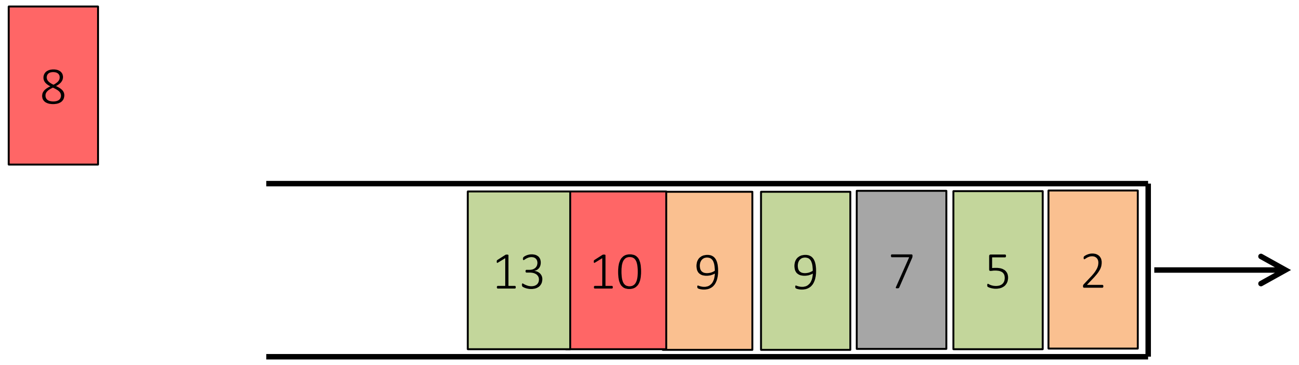
- Many algorithms (not all) determine transmission order at packet arrival
- Relative order of packet transmissions of packets in the queue doesn’t change with future arrivals
- Examples:
    - SJF: Order determined by flow size
    - FCFS: Order determined by arrival time

That’s where the paper comes in…

- This paper provides an abstraction for scheduling and shows that it can be implemented in hardware.

The Push-In First-Out Queue

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



The Push-In First-Out Queue

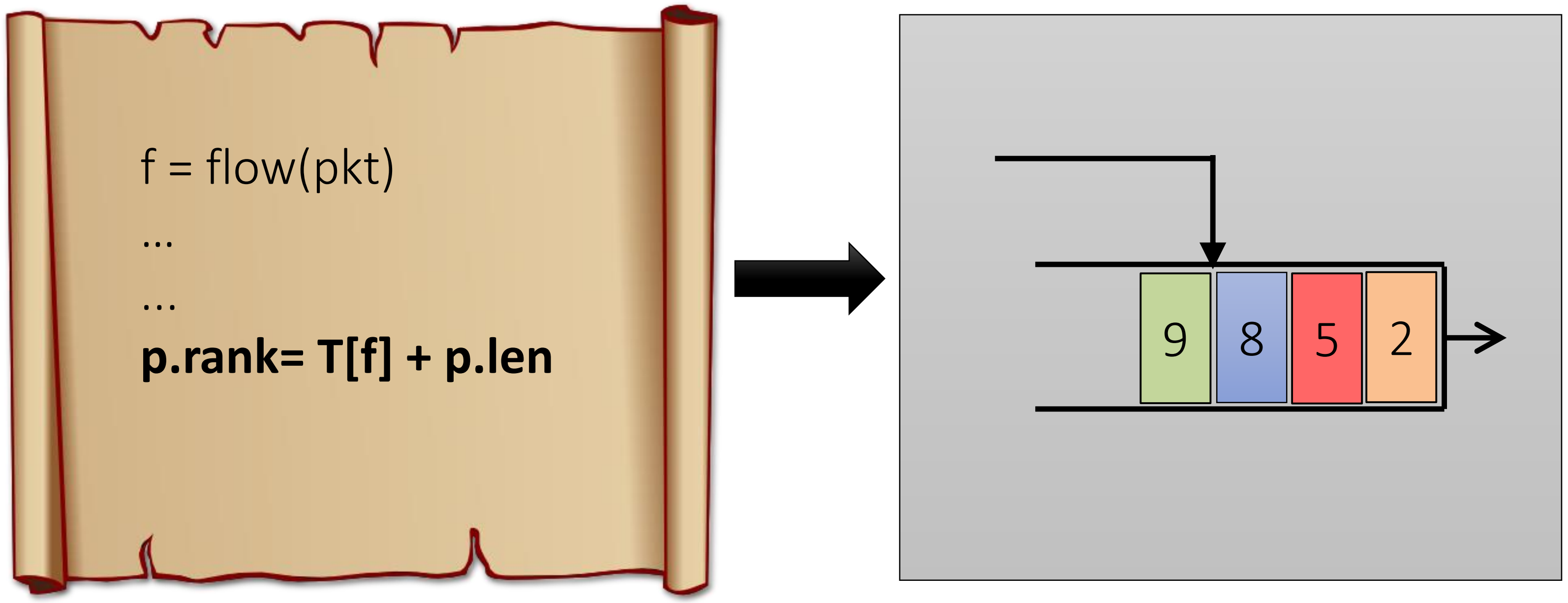
- PIFO: A sorted array that let us insert an entry (packet or PIFO pointer) into a PIFO based on a programmable priority
- Entries are always dequeued from the head
- If an entry is a packet, dequeue and transmit it
- If an entry is a PIFO, dequeue it, and continue recursively

A programmable scheduler

To program the scheduler, program the rank computation

Rank Computation

PIFO Scheduler

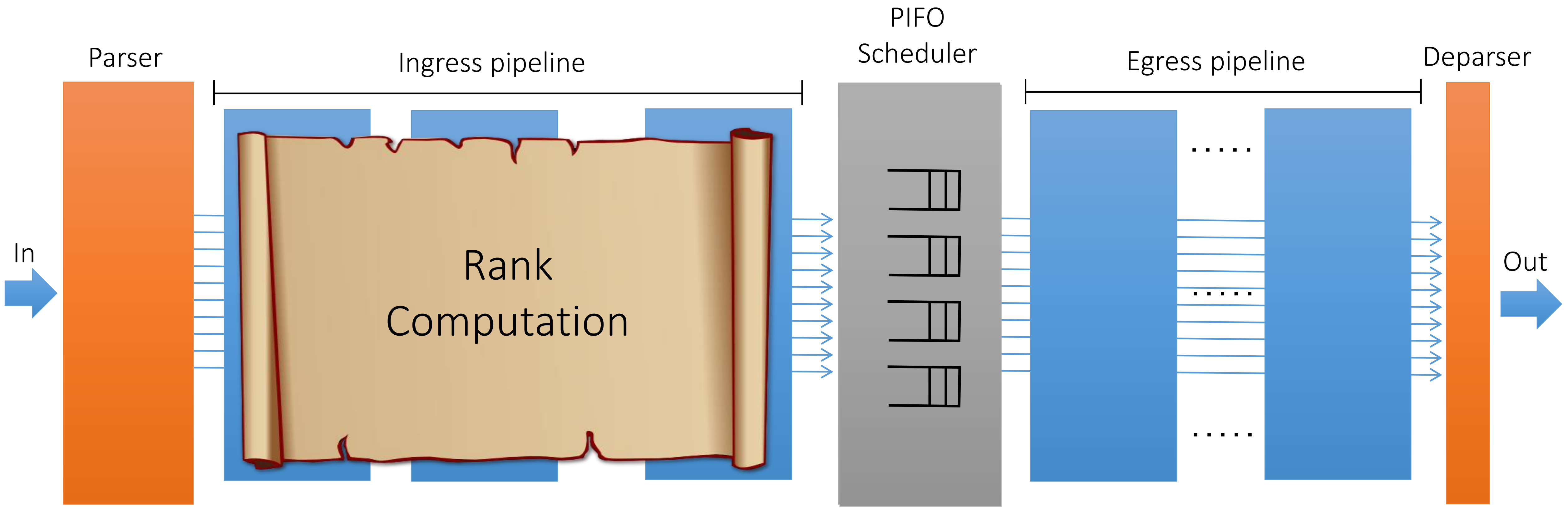


```
f = flow(pkt)
...
...
p.rank= T[f] + p.len
```

(programmable)

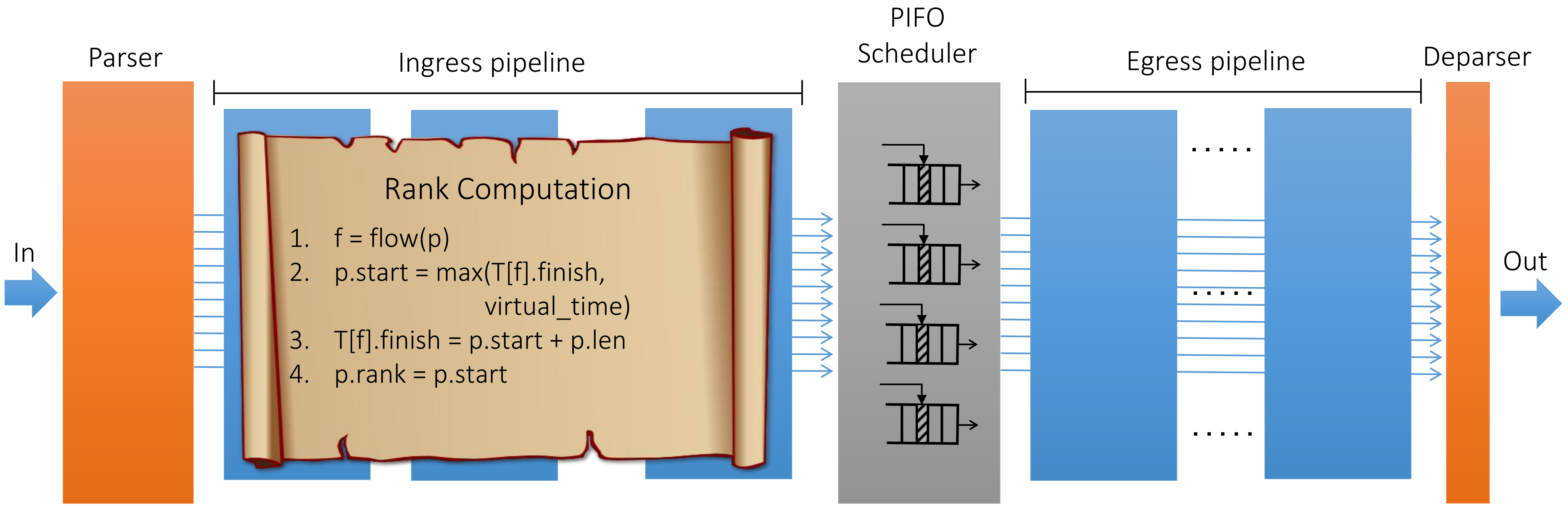
(fixed logic)

A programmable scheduler



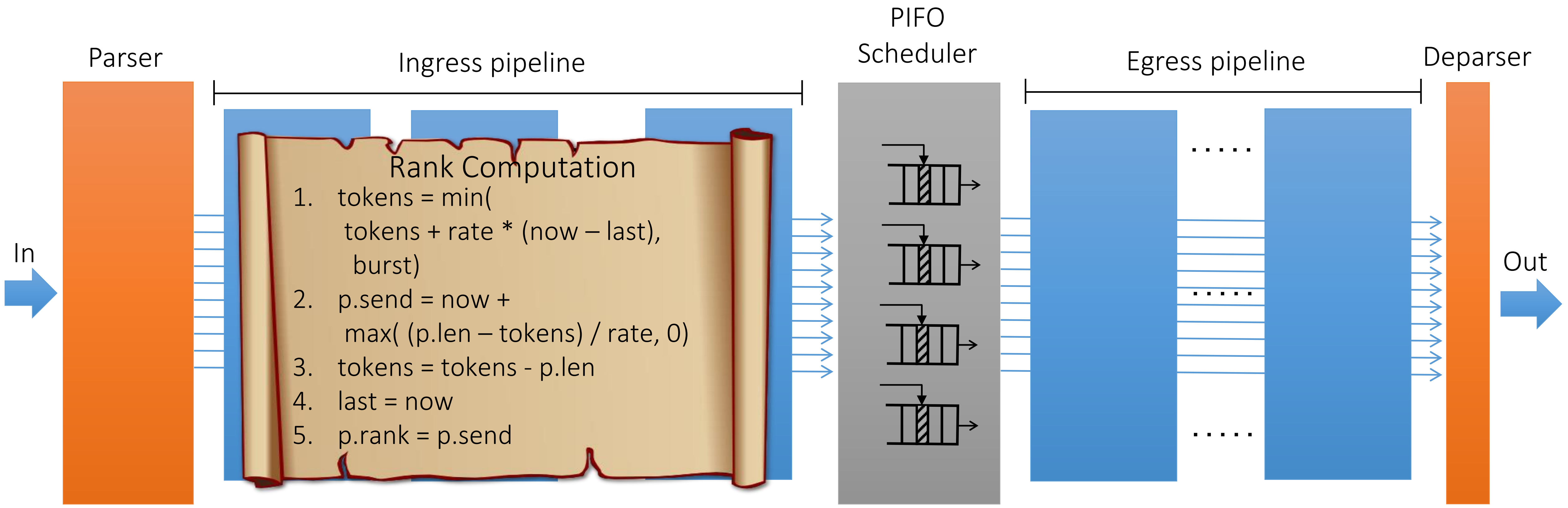
Rank computation is a packet transaction

Fair queuing



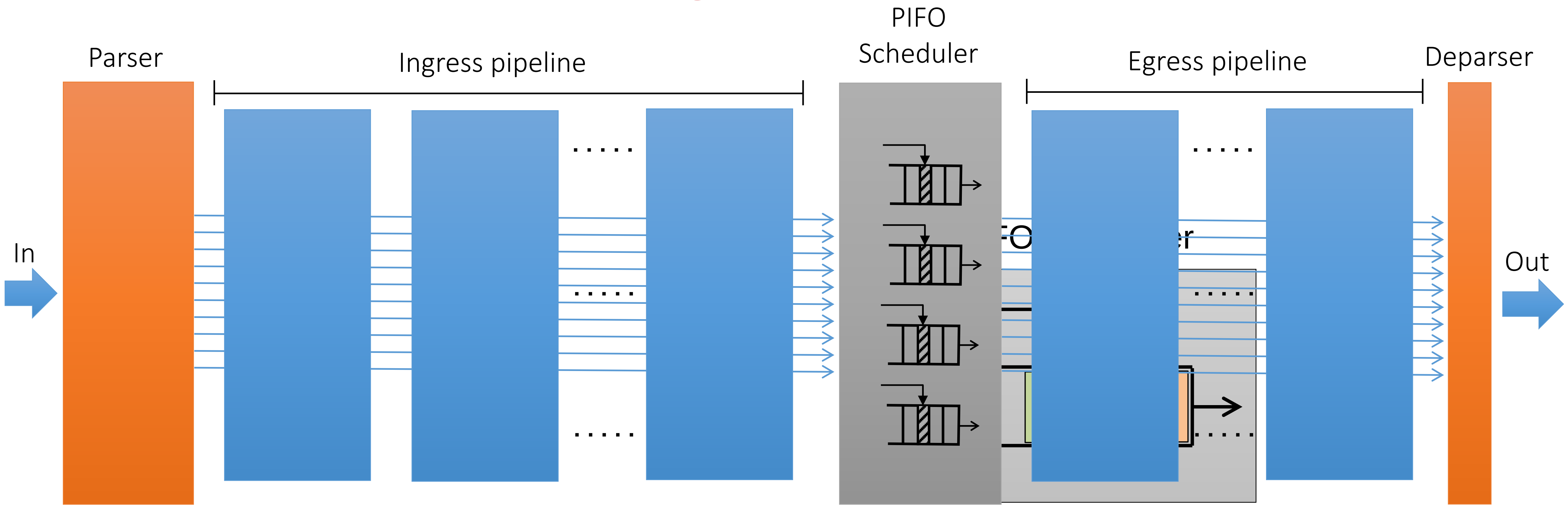
```
1.  f = flow(p)
2.  p.start = max(T[f].finish,
                  virtual_time)
3.  T[f].finish = p.start + p.len
4.  p.rank = p.start
```

Token bucket shaping



```
1. tokens = min(
     tokens + rate * (now – last),
      burst)
2. p.send = now +
     max( (p.len – tokens) / rate, 0)
3. tokens = tokens - p.len
4. last = now
5. p.rank = p.send
```

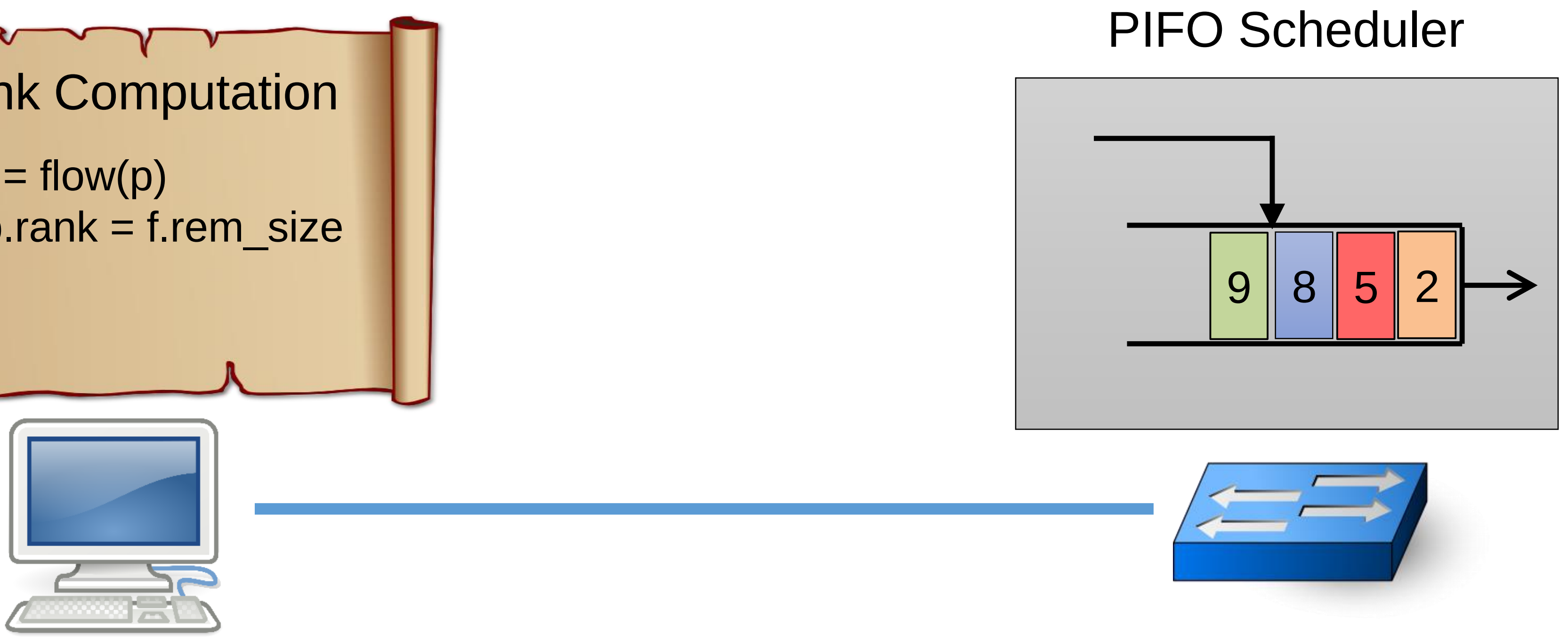
Shortest remaining flow size



Shortest remaining flow size

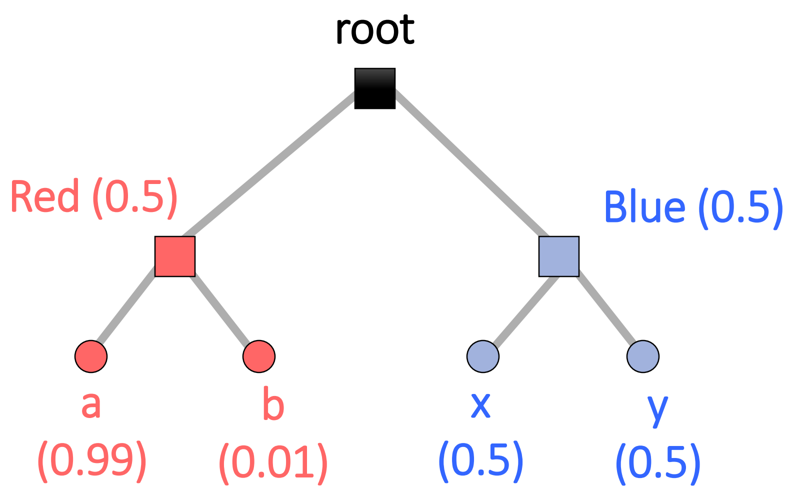
Rank Computation

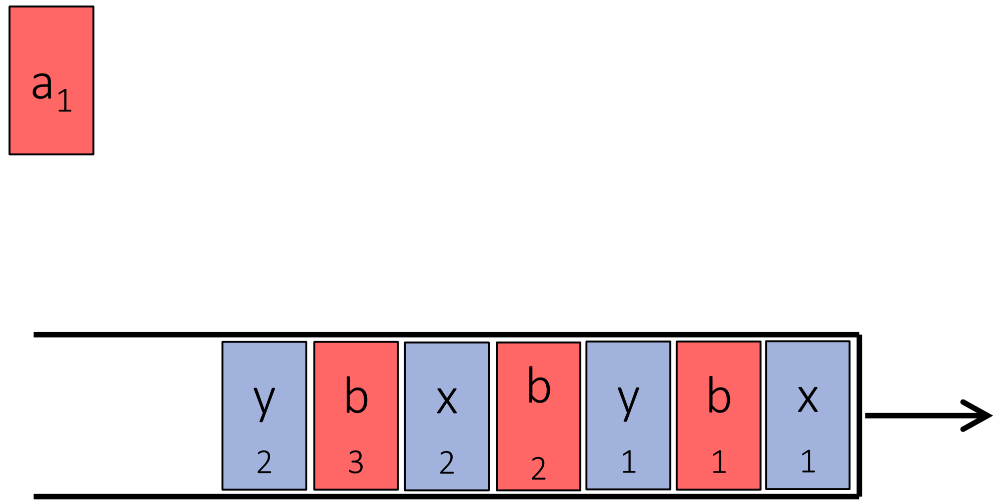
1. f = flow(p)
2. p.rank = f.rem_size



Beyond a single PIFO

Hierarchical
Packet Fair Queuing (HPFQ)

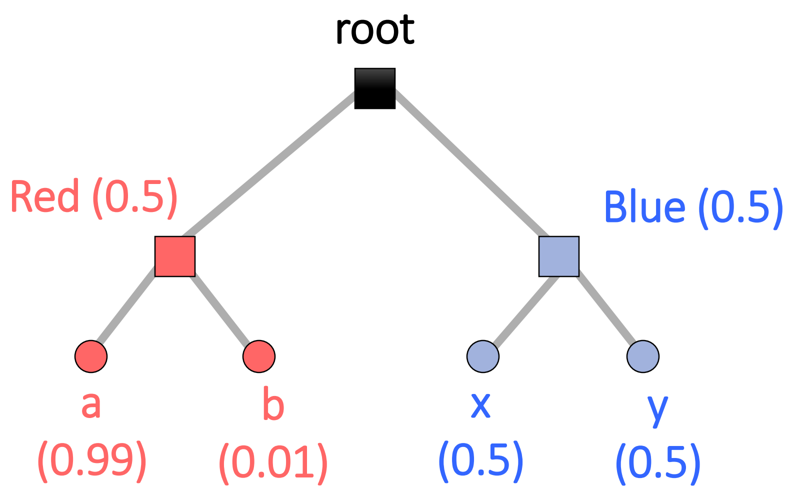


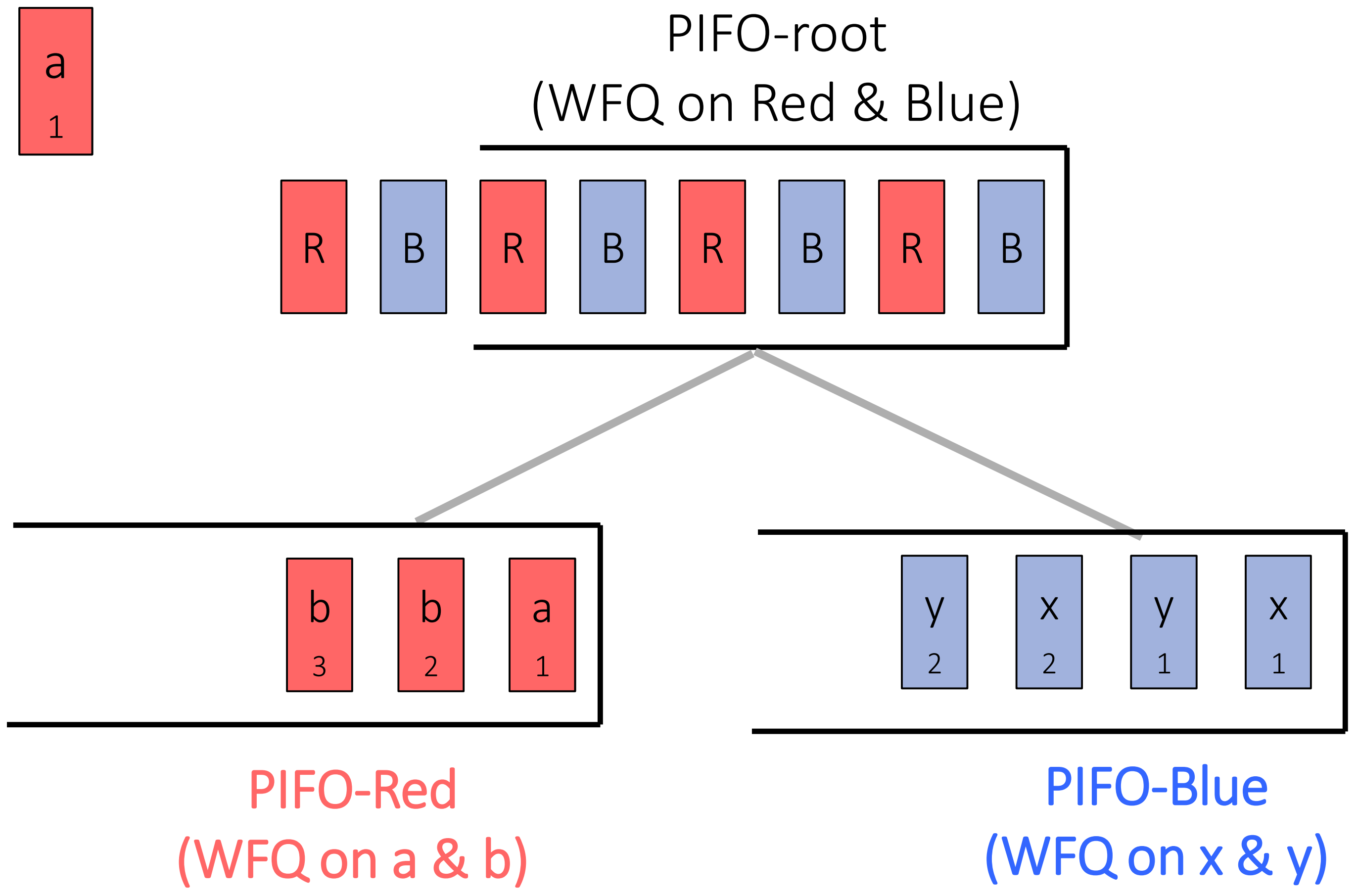


Hierarchical scheduling algorithms need hierarchy of PIFOs

Tree of PIFOs

Hierarchical
Packet Fair Queuing (HPFQ)





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.**

Is a PIFO Feasible

- What is the target?
- 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

PIFO in Hardware

- Shared across all the ports
- Naive solution: flat, sorted array is infeasible
  - Array of size 60000 & 60000 parallel comparators (☹)
- Exploit observation that ranks increase within a flow

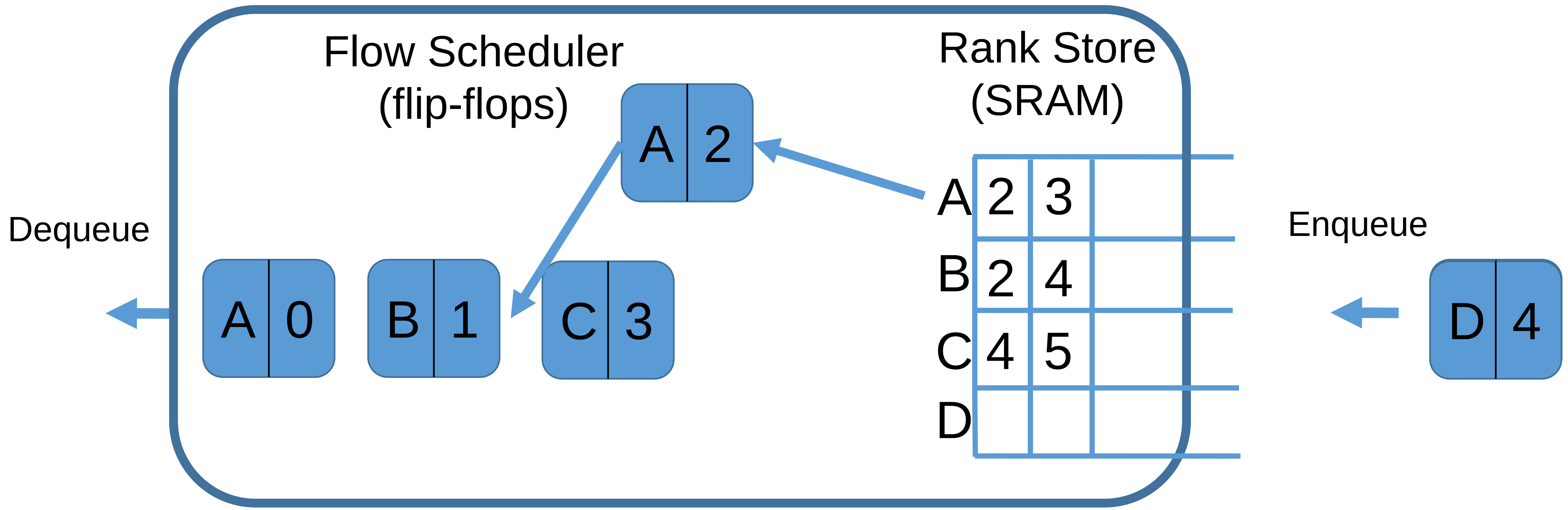
PIFO in Hardware

- Implementing Scheduling and Shaping transactions
  - Rank computation done using Domino (SIGCOMM 2016)
  - No loops in Domino(Not required for rank computation)

- Implementing Tree of PIFOs
  - Use full mesh of PIFO blocks.

- Compiler to auto-config this mesh from a scheduling tree

A single PIFO block



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

Hardware snapshot of Flow Scheduler

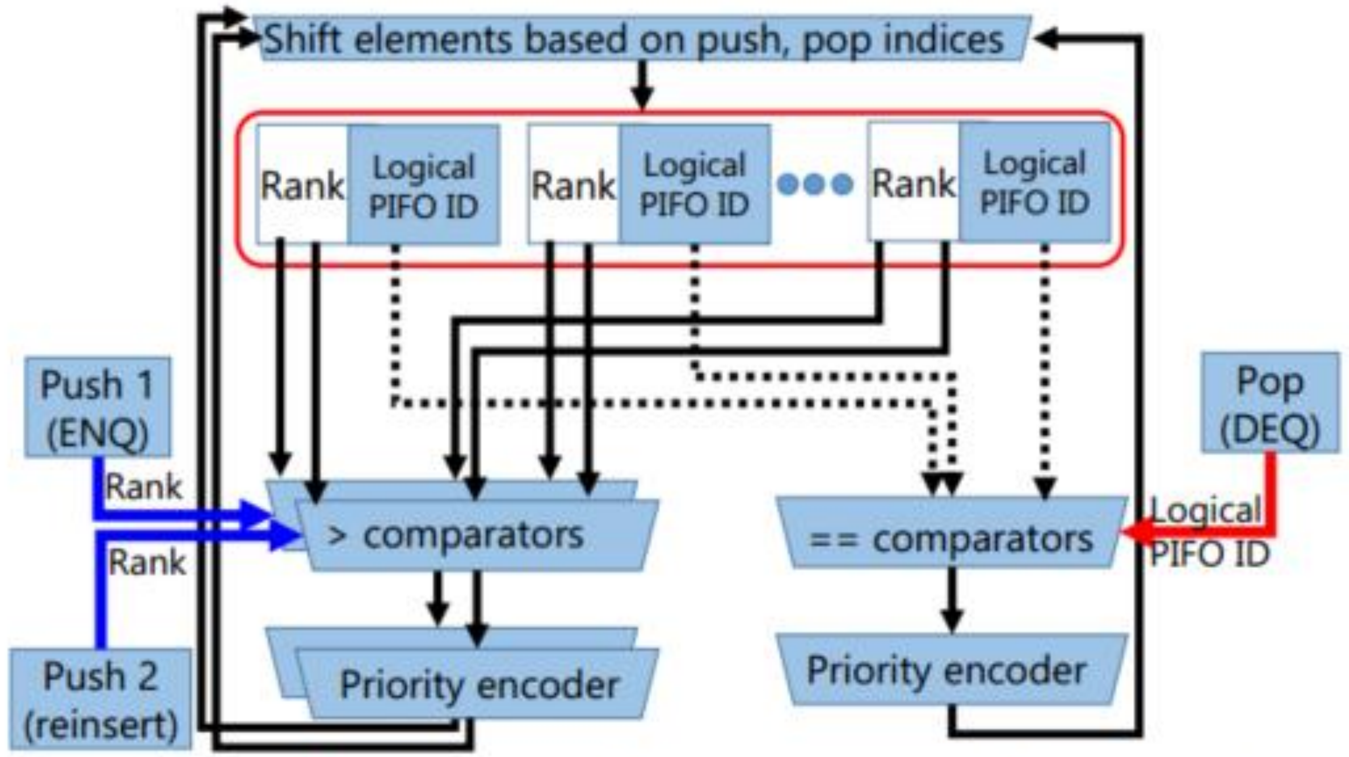


Figure 14: Hardware implementation of flow scheduler. Each element in the flow scheduler is connected to two > comparators (2 pushes) and one == comparator (1 pop).

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.

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

References

- Code: http://web.mit.edu/pifo
- Towards Programmable Packet Scheduling : Hotnets 2015
- Programmable Packet Scheduling At Line rate: SIGCOMM 2016
- First few slides are borrowed from Anirudh’s tech talk at Google.

Observations from 294 Discussion

- Barefoot Networks is a startup founded by Nick McKeon for creating programmable network switches.
- Paper has many hi-profile authors from academia and industry.
- While DRR, SRPT(Shortest Remaining Processing Time) are present in network switches and it is possible to modify their coefficients, to add a new scheduling algorithm is not possible in switches today.
  - Where do we used Hierarchical Routing? It used when we to sub-divide the network traffic into classes.

Observations from 294 Discussion

- Style for evaluating this paper. Few questions that arose were:
  - Is saving on chip area really important considering Moore’s law? How much do companies really save? Is it that bad to have 60000 comparators?
  - Are we the right audience to discuss the hardware feasibility of this paper? ISCAA people?
  - What are the other ways they could have evaluated their PIFO abstraction?