Abstractions for programming line-rate routers

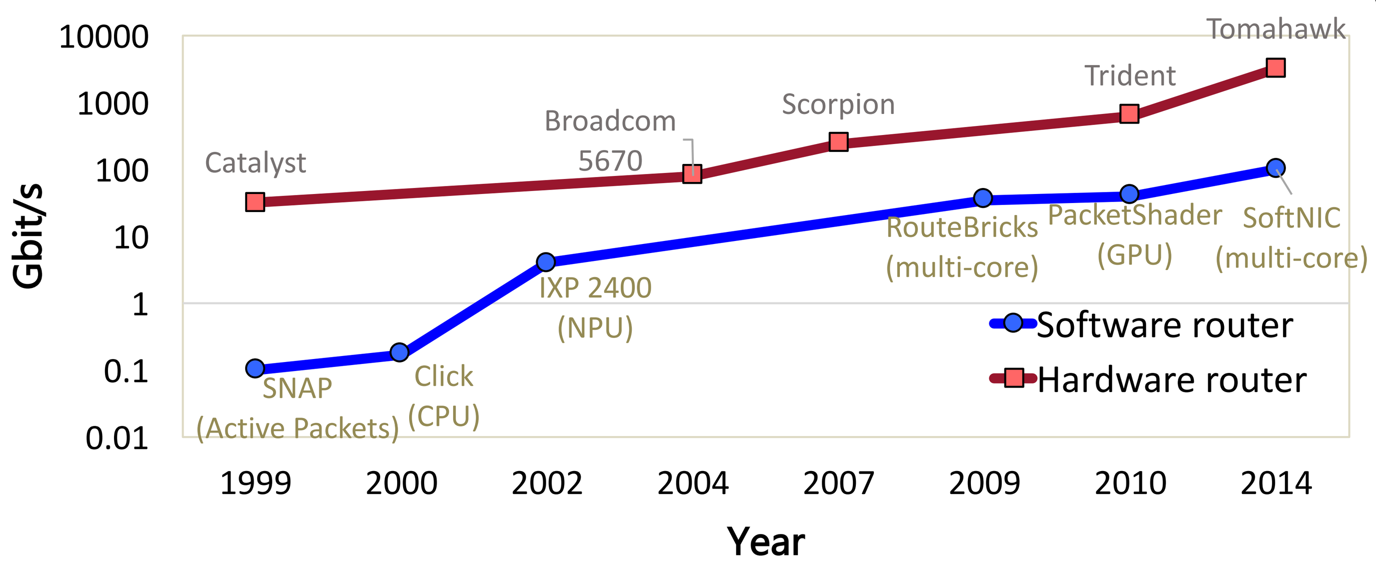
Anirudh Sivaraman, Suvinay Subramanian, Mohammad Alizadeh, Alvin Cheung, Anurag Agrawal, Hari Balakrishnan, Mihai Budiu, Sharad Chole, Shang-Tse Chuang, Tom Edsall, Sachin Katti, Changhoon Kim, Steve Licking, Nick McKeown, George Varghese

Massachusetts Institute of Technology · University of Washington Seattle · Barefoot Networks · vmware · Stanford University · Cisco · Microsoft Research

Programming router data planes at line rate

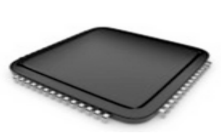
- Programming: Can we implement a new data-plane algorithm?
  - AQM
  - Scheduling
  - Congestion control
  - Load balancing
- Line rate: Highest speed supported by dedicated hardware

The evolution of programmable routers

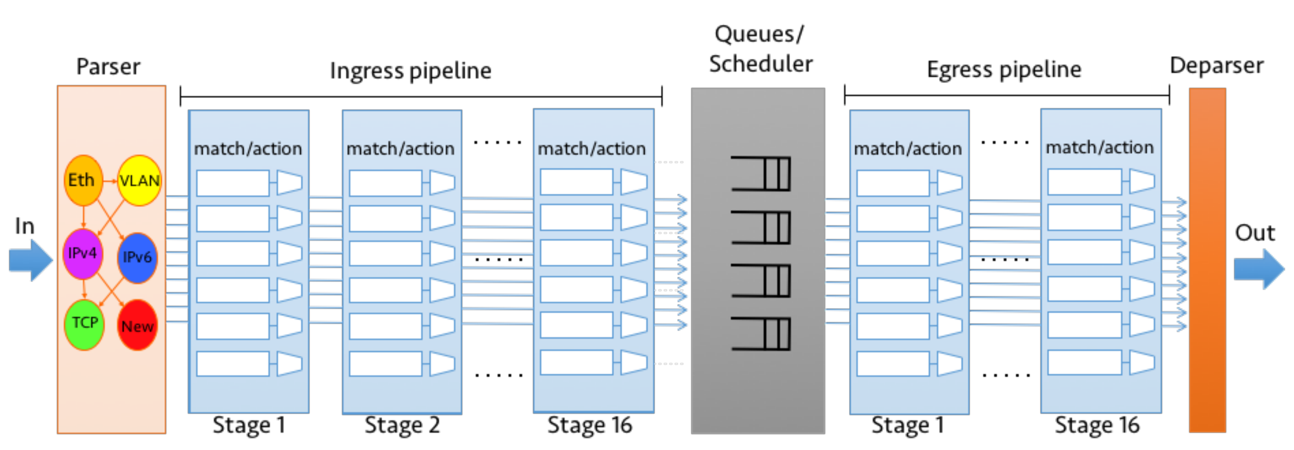


10—100X gap between hardware and software routers

Programmable router chips



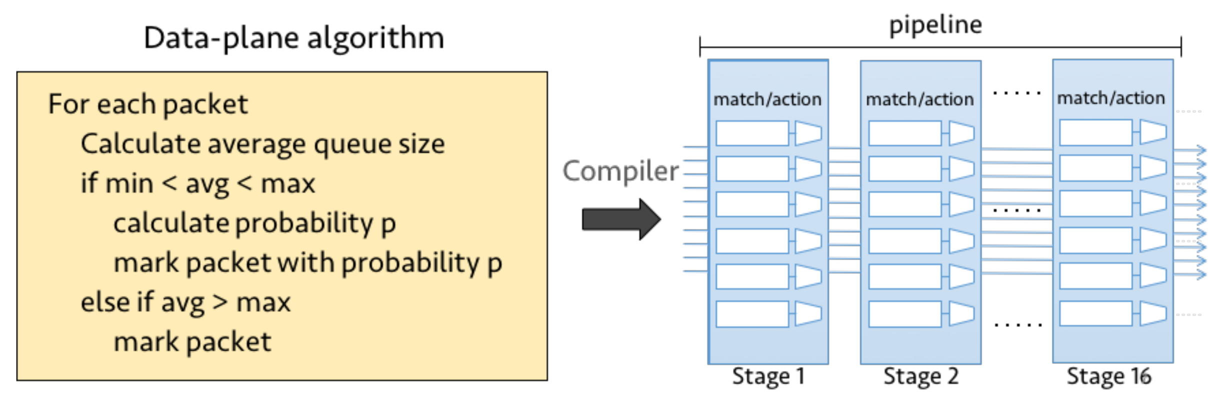
- Same performance as fixed-function chips,
- Some programmability



Two abstractions for line-rate programming

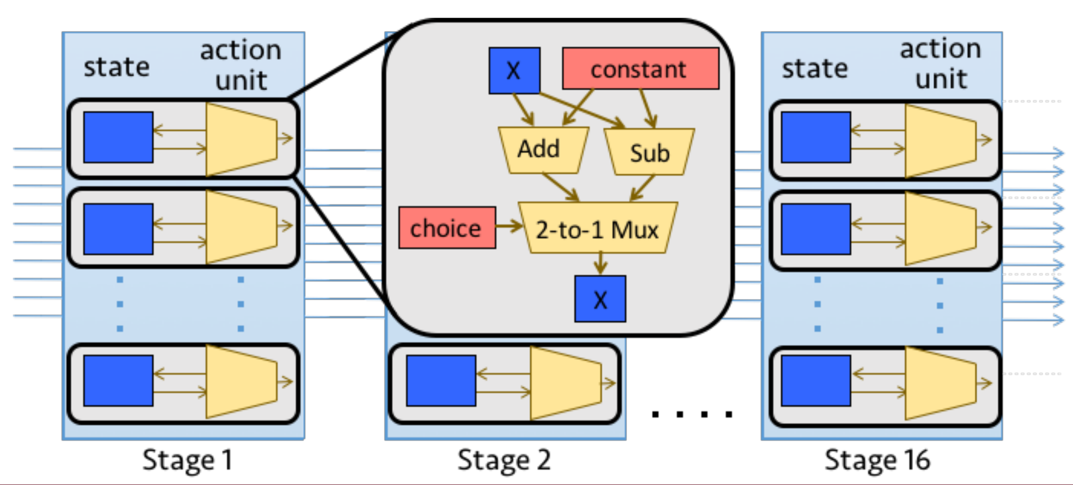
- Packet transactions
  - Stateful algorithms
  - C-like syntax
  - Automatically compile to a router pipeline
- Push-In First-Out (PIFO) queues
  - Expresses many scheduling algorithms
  - Modest area overhead

Packet transactions: high-level data-plane programming



Program in C-like DSL, compile to run at line-rate

A machine model for line-rate routers



Atom = action unit + state
A router's atoms form its instruction set

A family of atoms

Least Expressive → Most Expressive

Atom	Description	Area overhead
R/W	Read or write state	0.04%
RAW	Read, add, and write back	0.07%
PRAW	Predicated version of RAW	0.13%
IfElseRAW	2 RAWs, one each when a predicate is true or false	0.16%
Sub	IfElseRAW with a stateful subtraction capability	0.24%
Nested	4-way predication (nests 2 IfElseRAWs)	0.58%
Pairs	Update a pair of state variables	0.96%

Compilation results

Algorithm	LOC	Stages	Max. atoms/stage	Min. Atom Required
Bloom filter	29	4	3	R/W
Heavy hitter detection	35	10	9	RAW
Rate Control Protocol	23	6	2	PRAW
Flowlet switching	37	3	3	PRAW
Sampled NetFlow	18	4	2	IfElseRAW
HULL	26	7	1	Sub
Adaptive Virtual Queue	36	7	3	Nested
CONGA	32	4	2	Pairs
CoDel	57	15	3	Doesn't map

Programmable scheduling is 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

Need an expressive abstraction that can run at line rate

What does the scheduler do?

It decides in what order packets are sent

- E.g., FCFS, priorities, weighted fair-queueing

Key observation

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

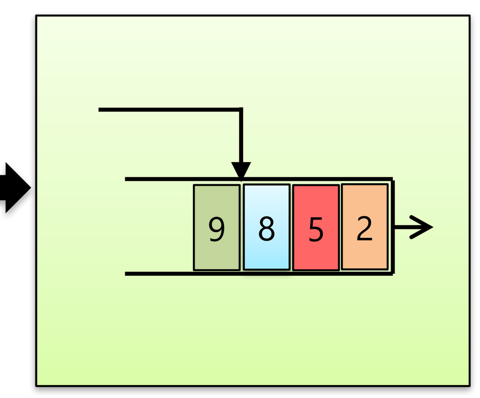
The Push-In First-Out Queue

Rank Computation

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

(programmable)

PIFO Scheduler

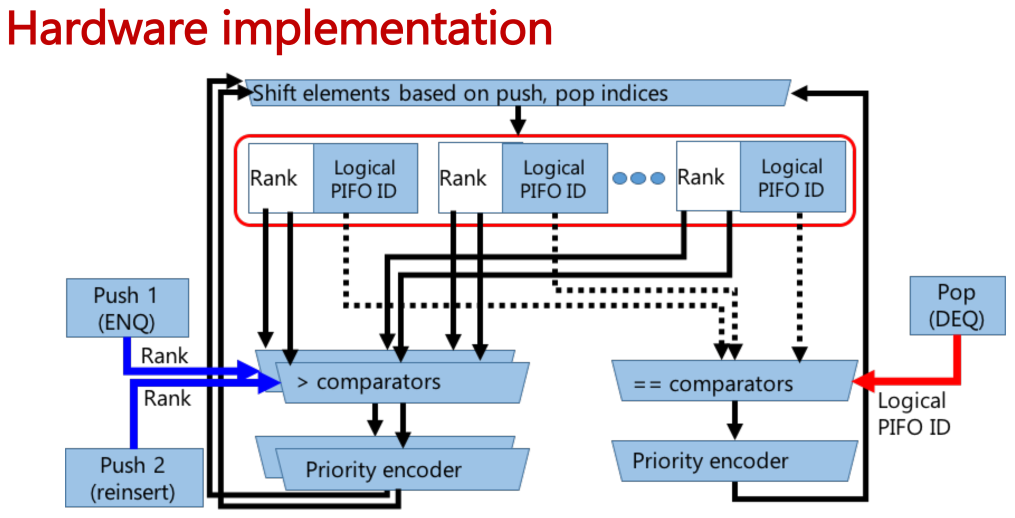


(fixed logic)

Rank computation is a packet transaction (e.g., WFQ, strict priorities, shaping)

Generalizes to hierarchical scheduling)

Hardware implementation



Meets timing @ 1 GHz
Incurs 4% area overhead