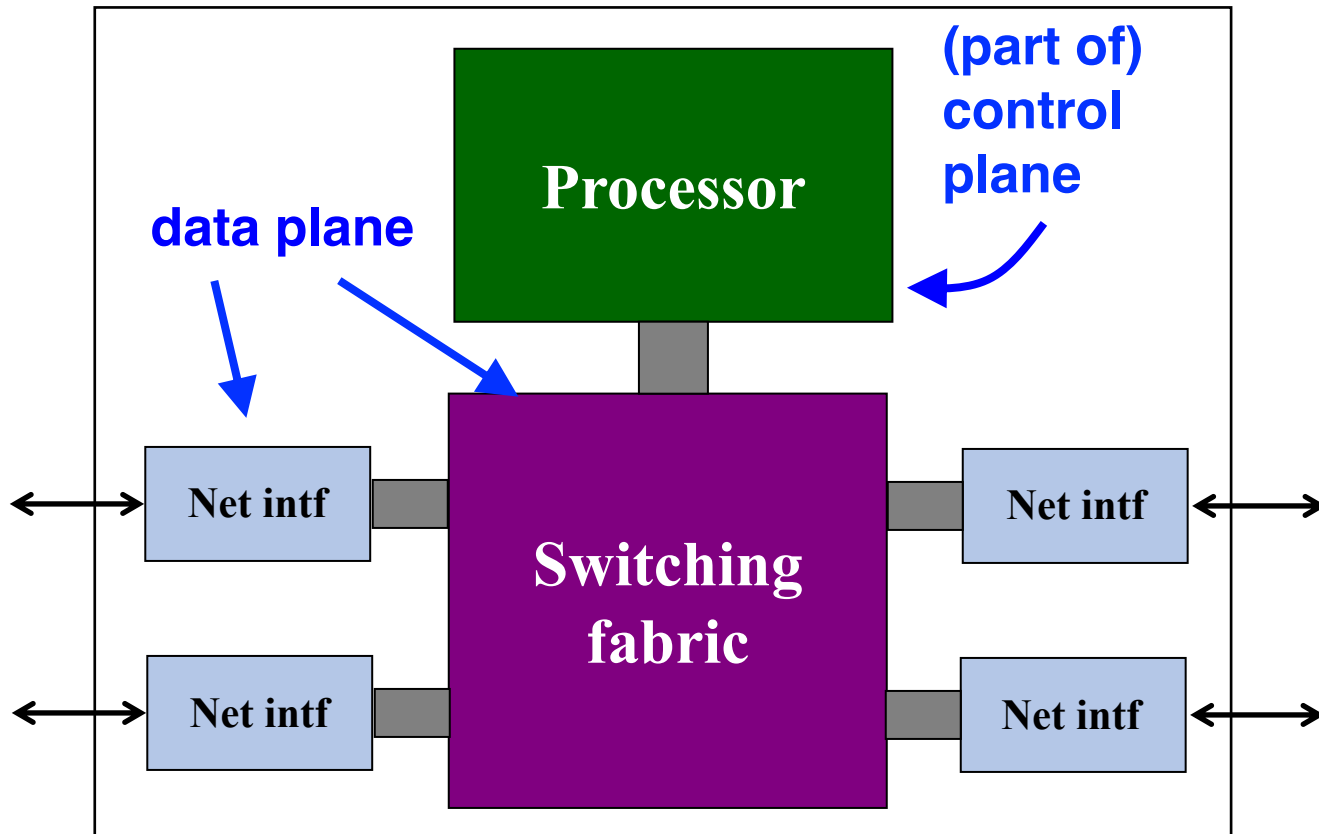


Router Design: An Overview

Lecture 15, Computer Networks (198:552)

Fall 2019

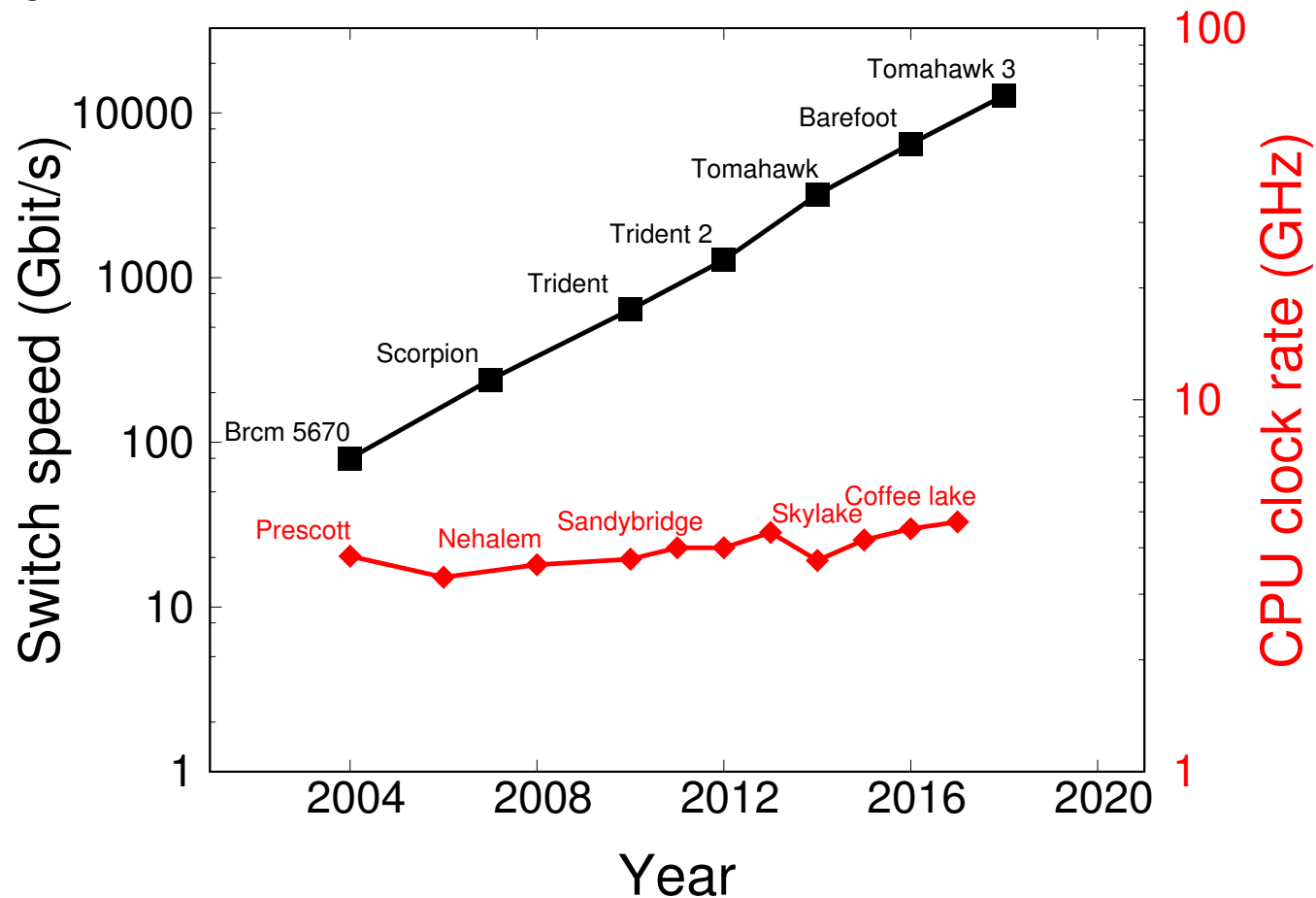
The router data plane



- Data plane implements per-packet decisions
 - On behalf of control & management planes
- Forward packets at high speed
- Manage contention for switch/link resources

Requirements on router data planes

- Speed!



Inherently
parallel workload

➔ Leverage
hardware
parallelism!

Requirements for router data planes

- Speed
- Chip area → size
- Power
- Port density
- Programmability



Overview of router functionality

- Historically evolving, multiple concurrent router designs
 - Many commonalities
 - Today: broad look at two router designs
 - MGR: router from the late 1990s
 - RMT: router from the late 2010s
- Mechanisms implemented:
 - Packet receive/transmit from/to physical interfaces
 - Packet and header parsing
 - Packet lookup and modification: ingress & egress processing
 - High-speed switching fabric to connect different interfaces
 - Traffic management: fair sharing, rate limiting, prioritization
 - Buffer management: admission into switch memory

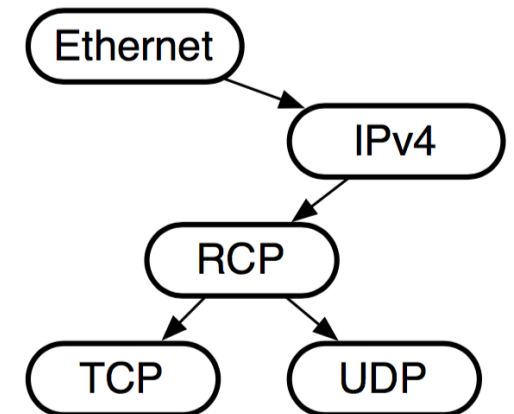
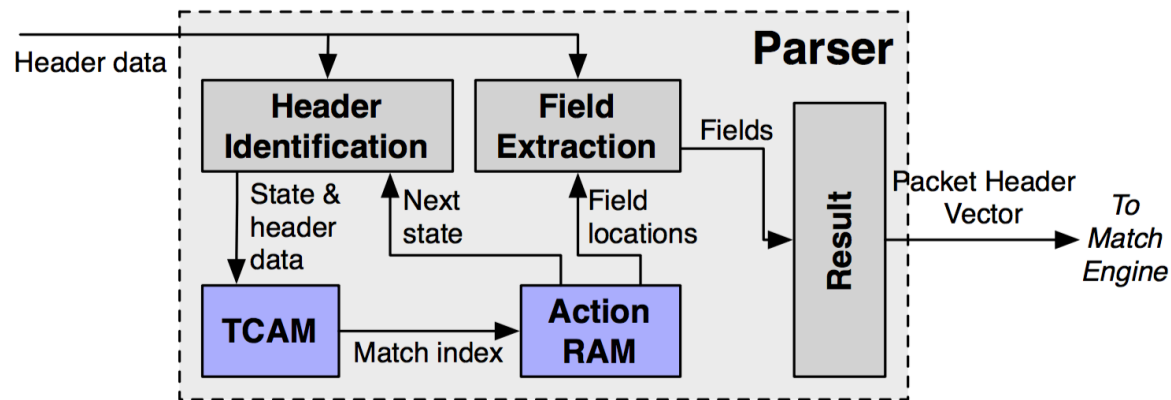
Life of a packet

(1) Receive data at line cards

- Circuitry to interface with physical medium: CoAx, optical
 - SerDes/IO modules: serialize/deserialize data from the wire
 - Network interfaces keep getting faster: more parallelism
 - but stay the same size (Moore's law is alive here, for now)
- Multiple network interfaces on a single line card
 - Component detachable from the rest of the switch
 - Ex: upgrade multiple 10 Gbit/s interfaces to 40 Gbit/s in one shot
- Preliminary header processing possible
 - MGR: convert link-layer headers to standard format

(2) Packet parsing

- Extract header fields: branching, looped processing
 - Ex: Determine transport-level protocol based on IP protocol type
 - Ex: Multiple encapsulations of VLAN or MPLS headers
- Outcome: parse graph and data in the parsed regions
- MGR: done in software using bit slicing of header memory
- RMT: programmable packet parsing *in hardware*



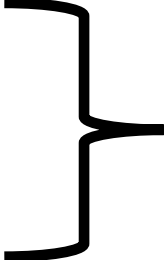
(2) Packet parsing

- Key principle: Separate the packet header and payload
 - Conserve bandwidth for data read/written inside switch!
- Header continues on to packet lookup/modification
- Payload sits on a buffer until router knows what to do with the packet
 - Buffer could be on the ingress line card (MGR)
 - But more commonly a buffer shared between line cards (RMT)

Things that routers are expected to do...

- RFC 1812: Forward pkts using route lookup, but also ...
- Update TTL: `ttl -= 1`
- Update IP checksum (Q: why?)
- IP to link layer mappings across networks (why?)
- Rewrite link layer source address
- Special processing (IP options): source route, record route, ...
- Fragmentation of packets
- Multicast
- Handle QoS assurances, if any

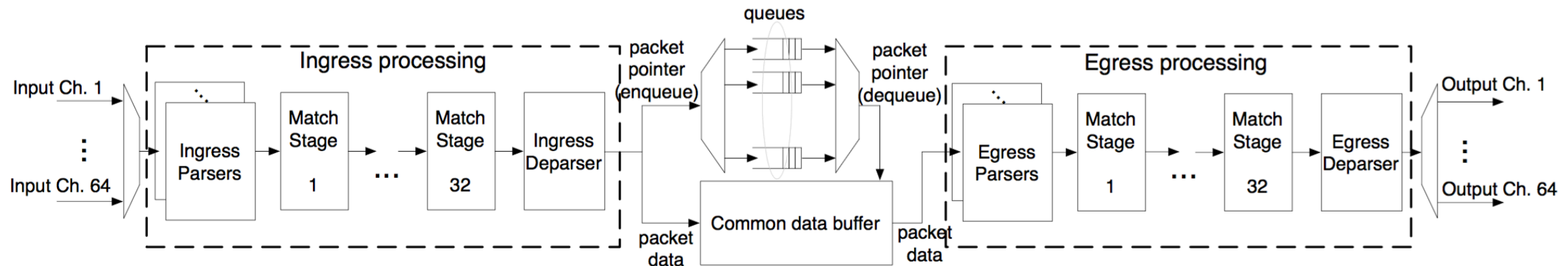
(3) Packet lookup

- Typical structure: Sequence of tables (Ex: L2, L3, ACL tables)
 - Exact match lookup
 - Longest prefix match
 - Wildcard lookups

Interesting algorithmic problems!
- Outcome: a (set of) output ports, possible header rewrites
- Wide range of table sizes (# entries) and widths (headers)
- Header modifications possible (we saw examples earlier)
 - TTL decrements, IP checksum re-computation
 - Encapsulate/decapsulate tunneling headers (MPLS, NV-GRE, ...)
 - MAC source address rewrite

(3) Packet lookup in RMT: *Pipelined parallelism*

- Different functionalities (ex: L2, L3) in different table stages
- Highly parallel over packets (1 packet/stage): high throughput
- Pipeline circuitry *clocked* at a high rate: ex: RMT@1 GHz
- MGR: software with memory access non-determinism
- RMT: deterministic hardware pipeline stages

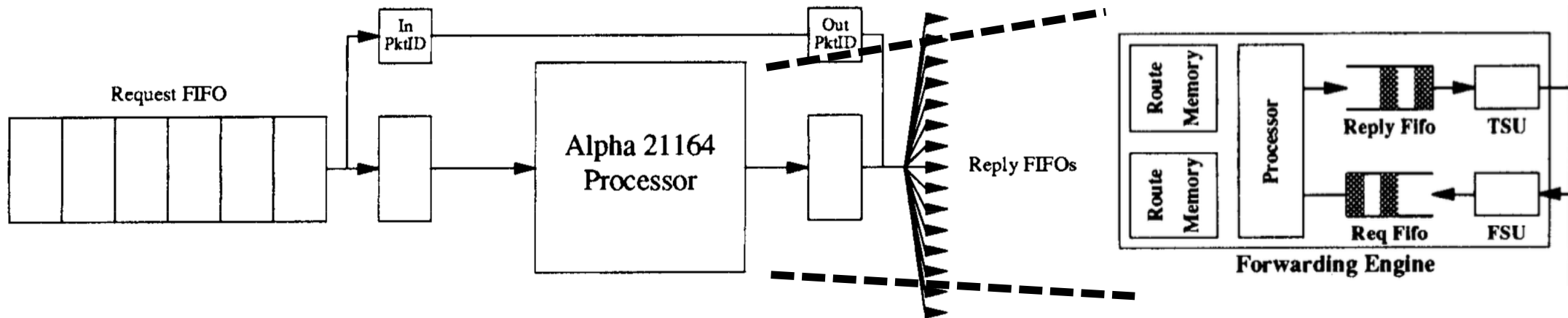


(3) Packet lookup in MGR

- A forwarding engine card separate from line cards
 - Scale forwarding and interface capacity separately
- Use Alpha 21164 (a 415MHz generic processor)
- Programmed in assembly

(3) MGR: Memory layout matters

- Try to fit all code into local instruction cache
- **Local cache** of routes for fast route lookup
 - Why might route caches work in the Internet?
- Far-away external memory stores full forwarding table
 - Accessed through a dedicated bus

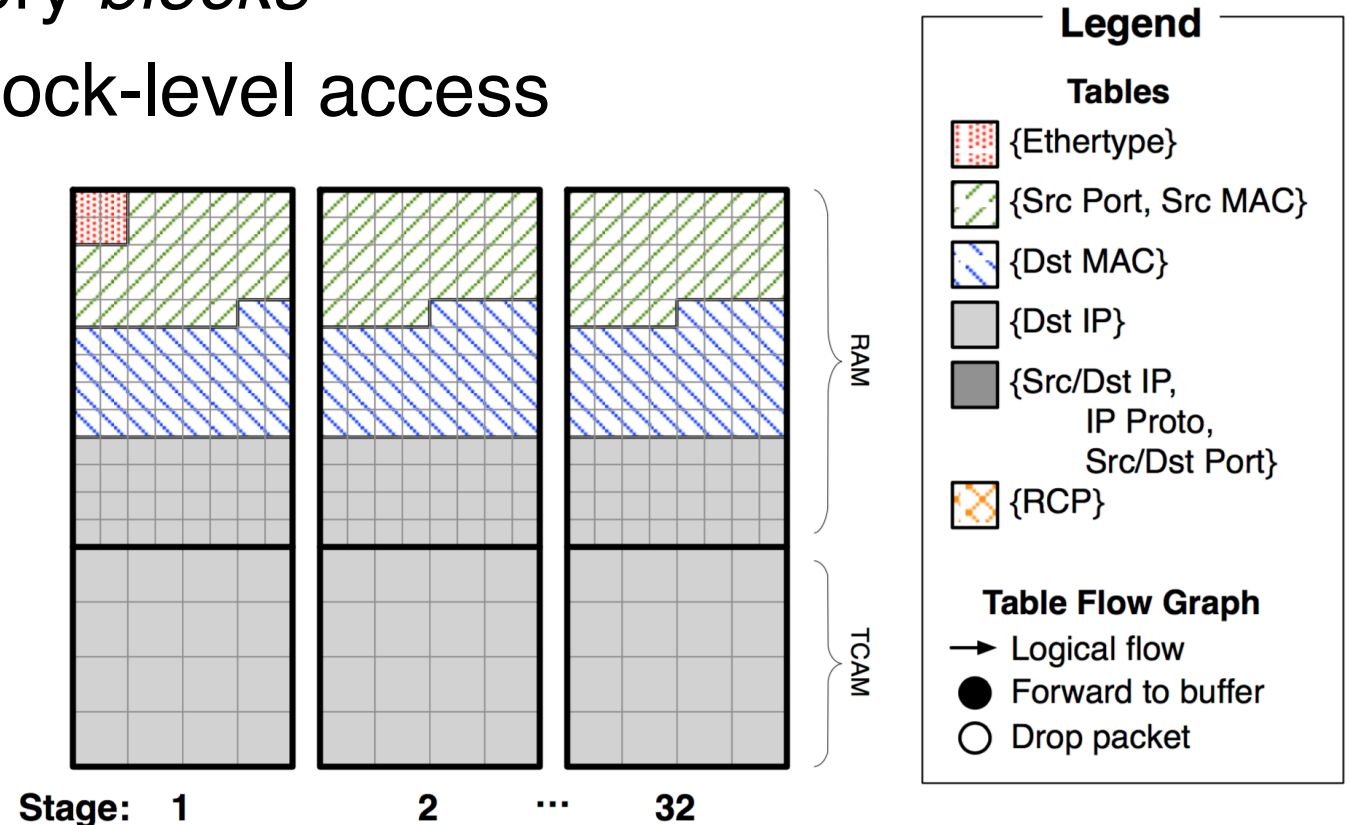


(3) Packet lookup in MGR

- Many micro-optimizations to improve performance
- **Separate fast path from slow path** (optimize the common case)
 - ARP lookup
 - Fragmentation
 - Error handling
- Separate packet classification from QoS
- Reduce data flowing through the processor memory bus
 - Packet headers separated from payload
 - Packet IDs not normally read from/written to in the normal case
- Two copies of table in ext memory to support seamless updates

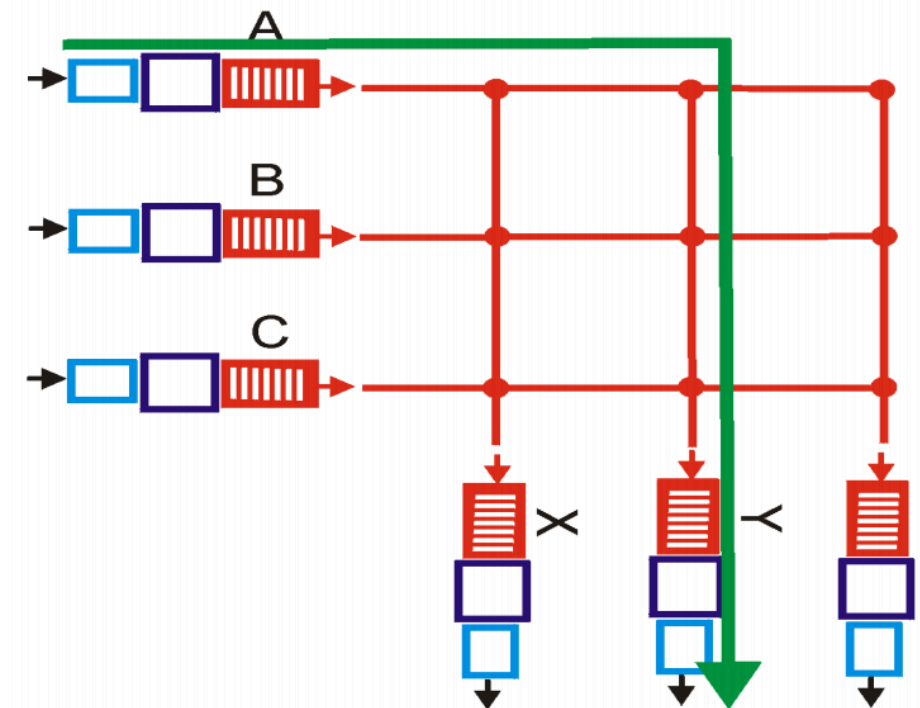
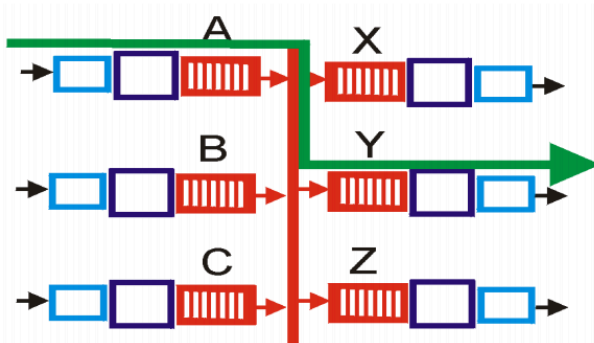
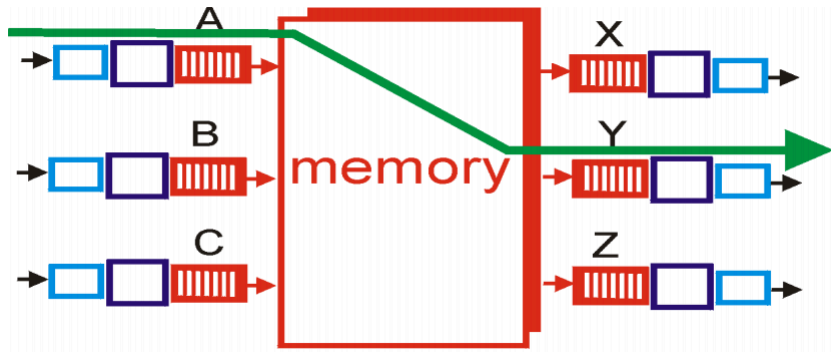
(3) RMT: Memory layout matters

- RMT: flexible partitioning of memory across SRAM and TCAM
- Numerous fixed size memory *blocks*
- Circuitry for independent block-level access
- Deterministic access times
 - All of it is SRAM or TCAM
- Interesting compiler issues
 - “Packing” tables



(4) Interconnect/Switching Fabric

- Move headers and packet from one interface to another
- Kinds of fabrics: memory, bus, crossbar



(4) Crossbars: The scheduling problem

- Demands from port i to port j
- Can one utilize fabric capacity regardless of demand pattern?
 - Blocking vs. nonblocking
- MGR considers strategies:
 - Greedy, wavefront, **block wavefront**
 - Need to address **fairness** issues

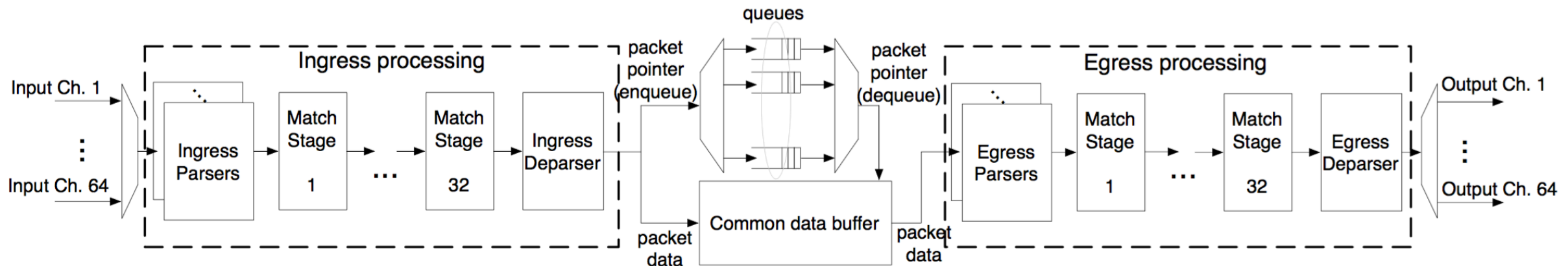
	1	2	3	4	5	6
1	1	0	1	0	1	1
2	1	0	1	1	0	0
3	1	0	1	0	1	1
4	0	1	1	1	0	0
5	1	1	1	0	1	1
6	1	1	1	0	1	1

(4) RMT switching fabric

- RMT uses shared memory as the fabric to hold packet headers and payloads between any two interfaces
- Tradeoff
 - More wires and power
 - But implement traffic and buffer management in one place

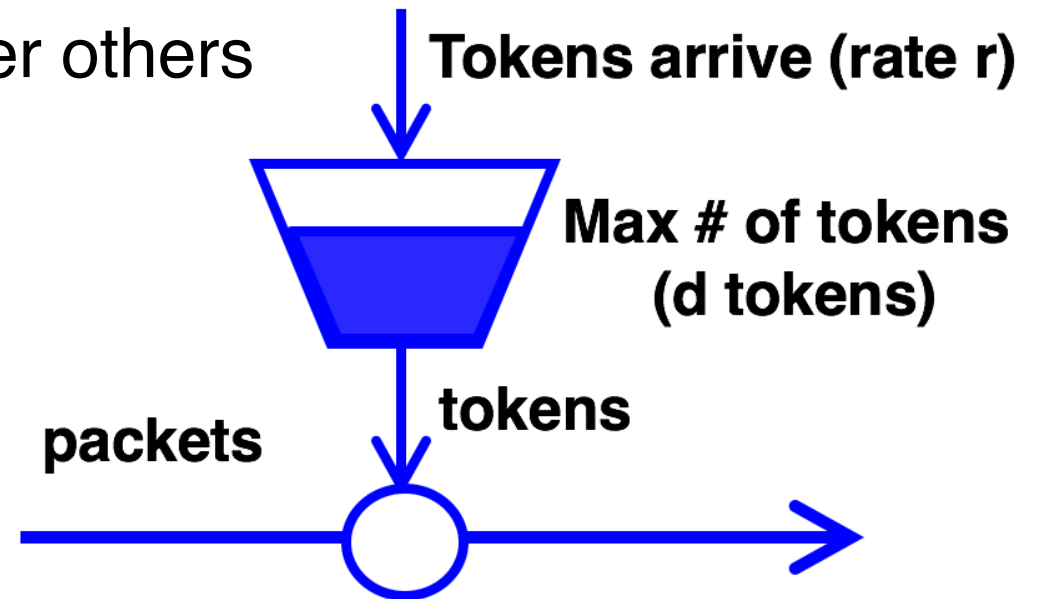
(5) Queueing: Traffic management

- Where should the packets not currently serviced wait?
- Input-queued vs. output-queued
- HOL blocking? Suppose port 1 wants to send to both 2 and 3
 - But port 2 is clogged
 - Port 1's packets towards port 3 should not be delayed



(5) Queueing: Traffic Management

- Better to have queues represent output port contention
- Scheduling policies:
 - Fair queueing across ports
 - Strict prioritization of some ports over others
 - Rate limiting per port!



(5) Queueing: Buffer Management

- Typical buffer management: Tail-drop

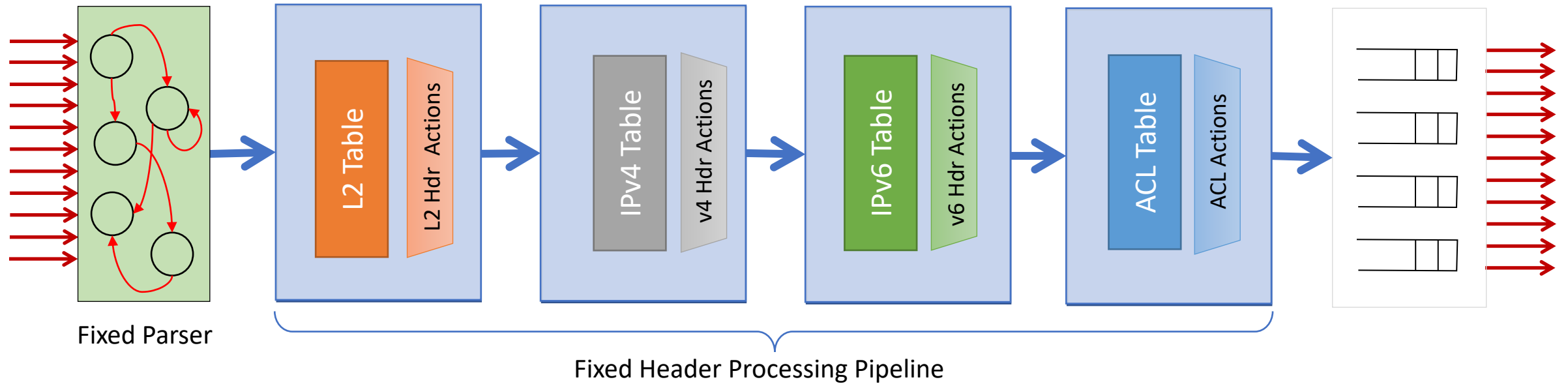


- How should buffer memory be partitioned across ports?
 - Static partitioning: if port 1 has no packets, don't drop port 2
 - Shared memory with dynamic partitioning
- However, need to share fairly:
 - If output port 1 is congested, why should port 2 traffic suffer?
- Algorithmic problems in dynamic memory sizing across ports

(6) Egress processing

- Combine headers with payload for transmission
 - Need to incorporate header modifications
 - ... also called “deparsing”
- Multicast: egress-specific packet processing
 - Ex: source MAC address
- Multicast makes almost everything inside the switch (interconnect, queueing, lookups) more complex

Fixed function pipeline



Protocol Independent Switch Arch. (PISA)

