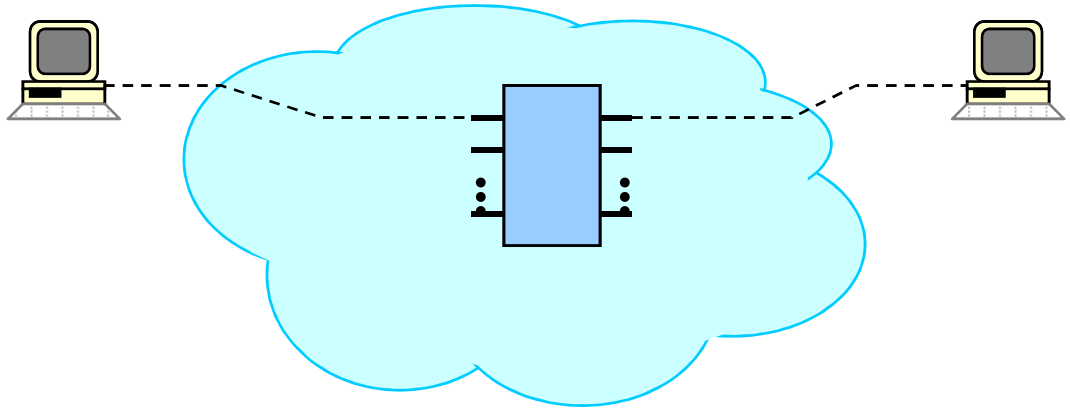
Lecture 5:
Router Architecture

CS 598: Advanced Internetworking

Matthew Caesar

February 8, 2011

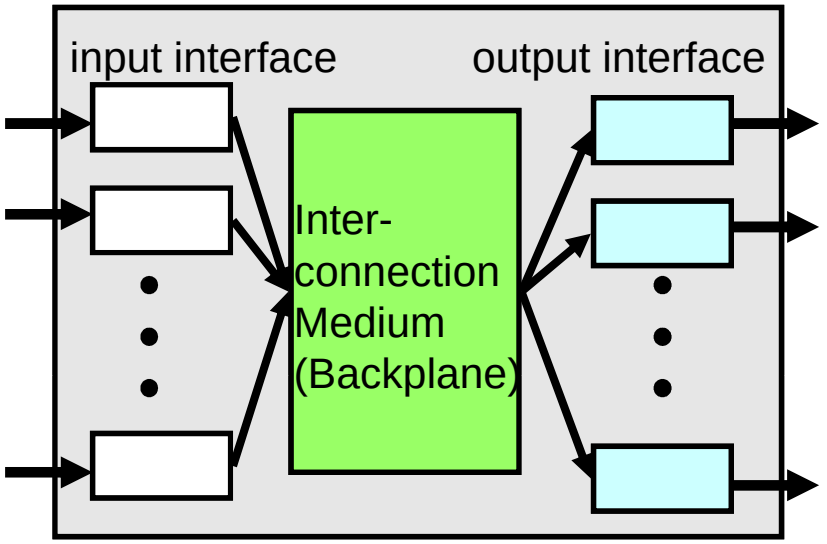
IP Router



- A router consists
  - A set of input interfaces at which packets arrive
  - A se of output interfaces from which packets depart
- Router implements two main functions
  - Forward packet to corresponding output interface
  - Manage congestion

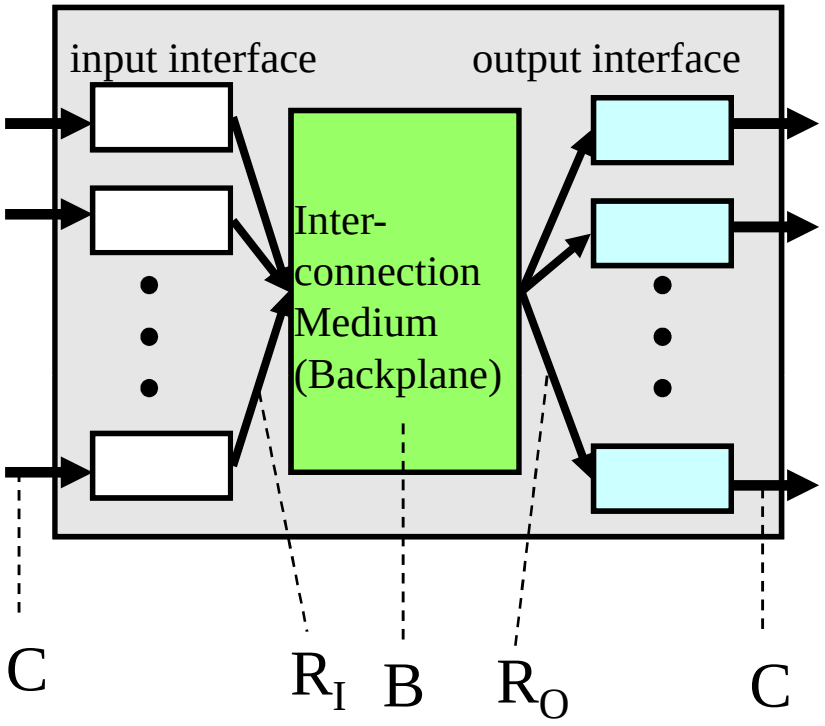
Generic Router Architecture

- Input and output interfaces are connected through a backplane
- A backplane can be implemented by
  - Shared memory
    - Low capacity routers (e.g., PC-based routers)
  - Shared bus
    - Medium capacity routers
  - Point-to-point (switched) bus
    - High capacity routers



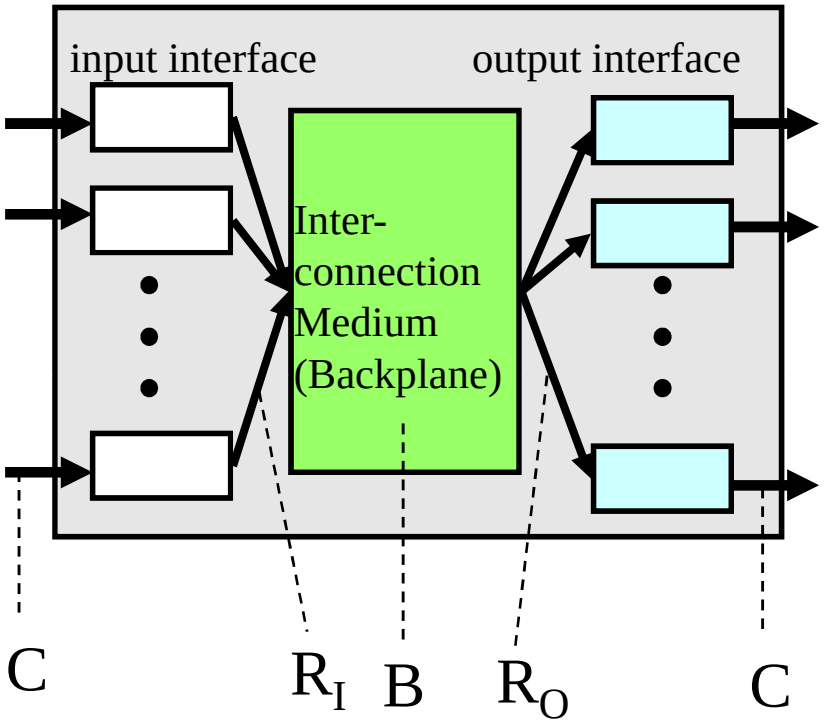
Speedup

- C – input/output link capacity
- R_I – maximum rate at which an input interface can send data into backplane
- R_O – maximum rate at which an output can read data from backplane
- B – maximum aggregate backplane transfer rate
- Back-plane speedup: B/C
- Input speedup: R_I/C
- Output speedup: R_O/C



Function division

- Input interfaces:
  - Must perform packet forwarding – need to know to which output interface to send packets
  - May enqueue packets and perform scheduling
- Output interfaces:
  - May enqueue packets and perform scheduling

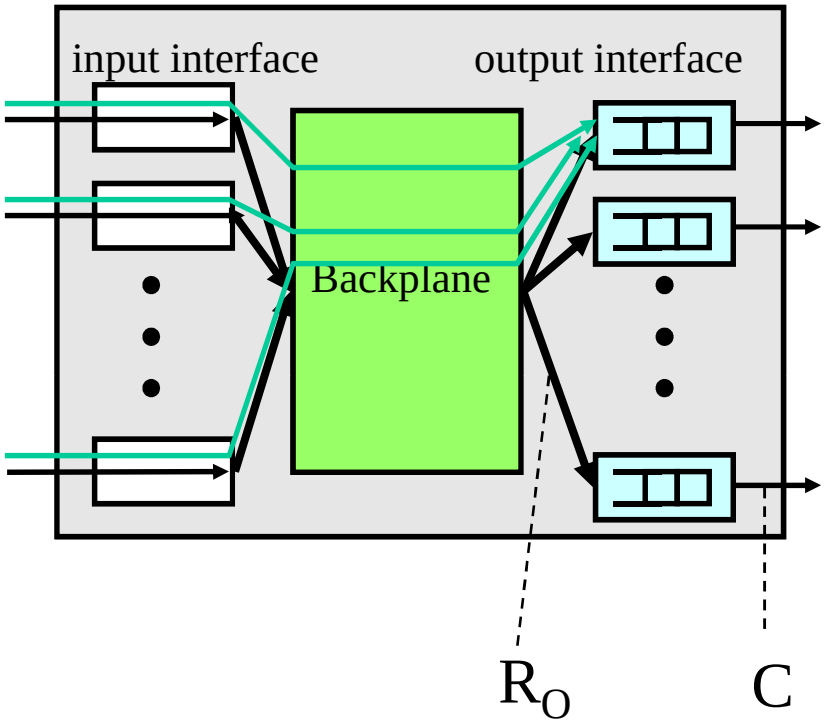


Three Router Architectures

- Output queued
- Input queued
- Combined Input-Output queued

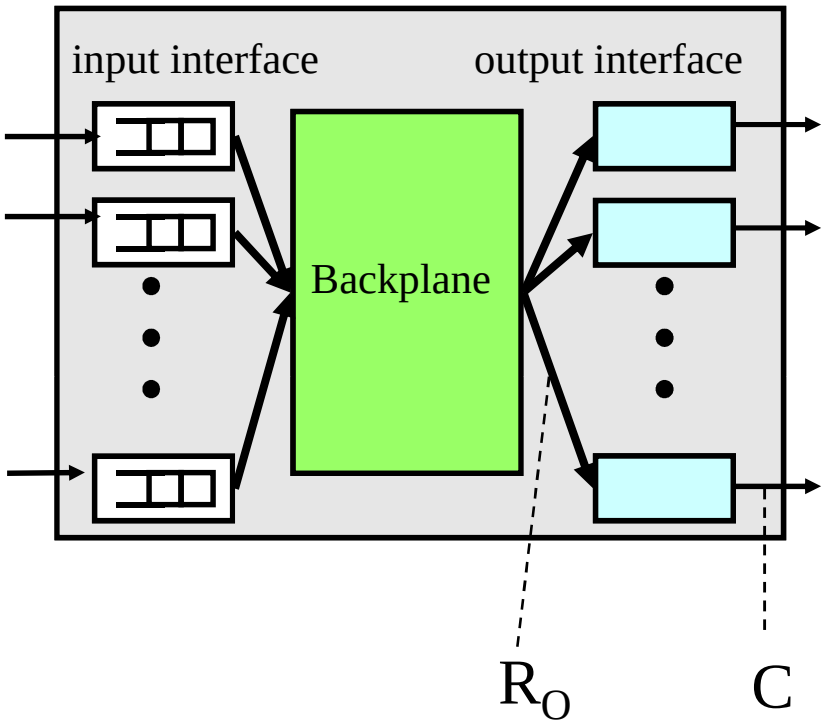
Output Queued (OQ) Routers

- Only output interfaces store packets
- Advantages
  - Easy to design algorithms: only one congestion point
- Disadvantages
  - Requires an output speedup of N, where N is the number of interfaces → not feasible



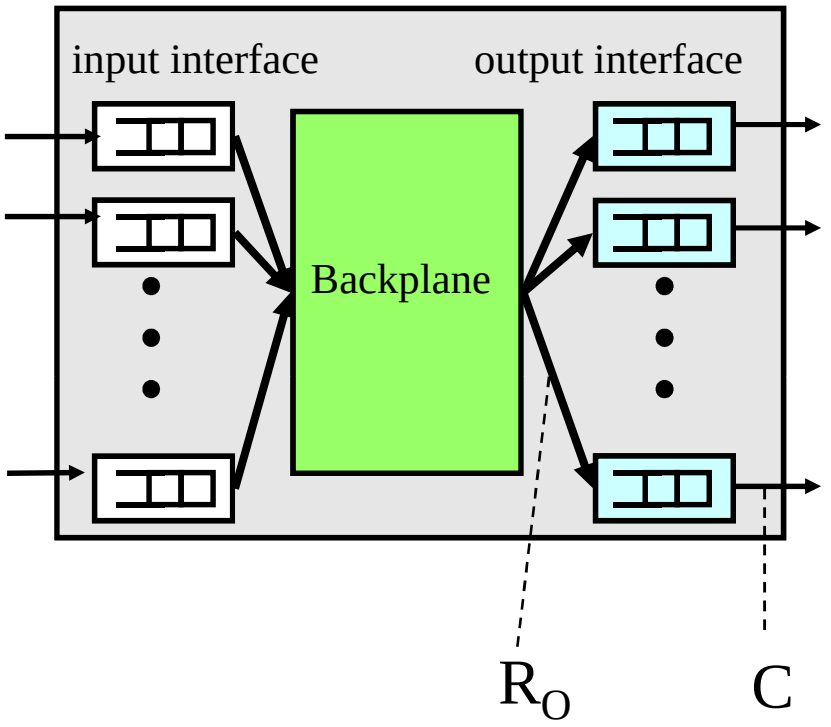
Input Queueing (IQ) Routers

- Only input interfaces store packets
- Advantages
  - Easy to built
    - Store packets at inputs if contention at outputs
  - Relatively easy to design algorithms
    - Only one congestion point, but not output…
    - need to implement backpressure
- Disadvantages
  - Hard to achieve utilization → 1 (due to output contention, head-of-line blocking)
    - However, theoretical and simulation results show that for realistic traffic an input/output speedup of 2 is enough to achieve utilizations close to 1



Combined Input-Output Queueing (CIOQ) Routers

- Both input and output interfaces store packets
- Advantages
  - Easy to built
    - Utilization 1 can be achieved with limited input/output speedup (<= 2)
- Disadvantages
  - Harder to design algorithms
    - Two congestion points
    - Need to design flow control
  - Note: results show that with a input/output speedup of 2, a CIOQ can emulate any work-conserving OQ [G+98,SZ98]

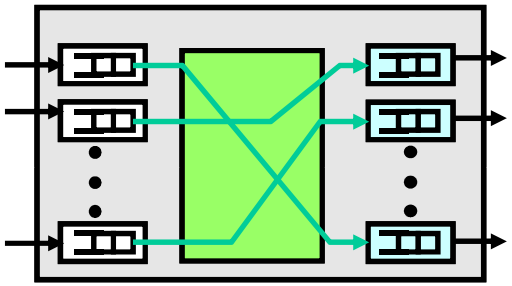


Generic Architecture of a High Speed Router Today

- Combined Input-Output Queued Architecture
  - Input/output speedup <= 2
- Input interface
  - Perform packet forwarding (and classification)
- Output interface
  - Perform packet (classification and) scheduling
- Backplane
  - Point-to-point (switched) bus; speedup N
  - Schedule packet transfer from input to output

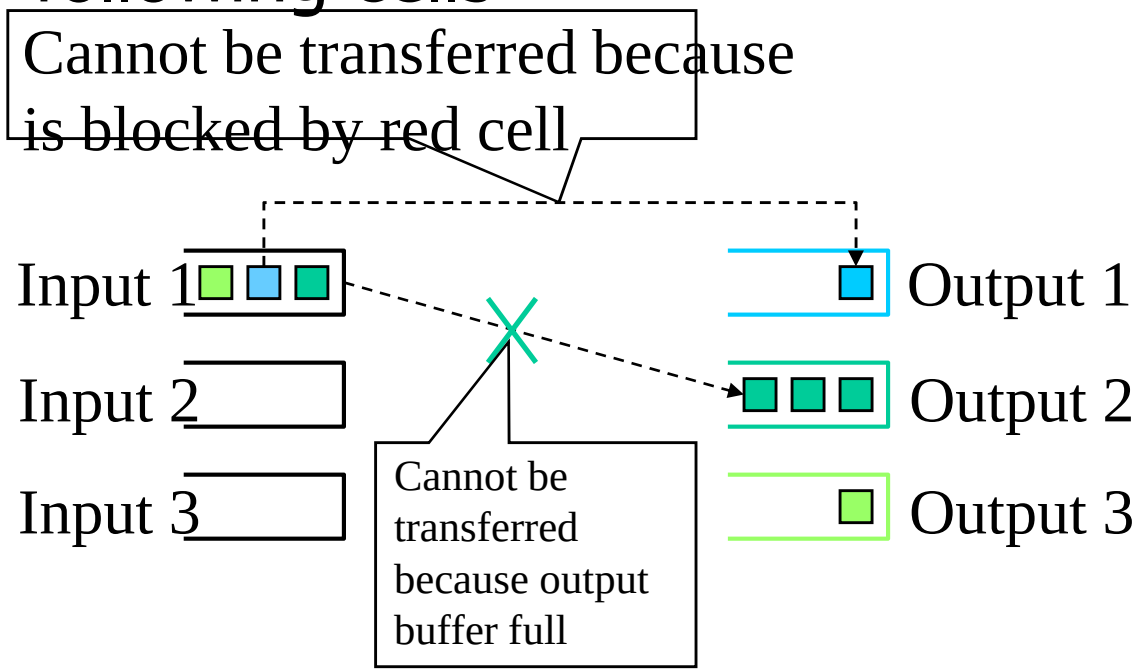
Backplane

- Point-to-point switch allows to simultaneously transfer a packet between any two disjoint pairs of input-output interfaces
- Goal: come-up with a schedule that
  - Meet flow QoS requirements
  - Maximize router throughput
- Challenges:
  - Address head-of-line blocking at inputs
  - Resolve input/output speedups contention
  - Avoid packet dropping at output if possible
- Note: packets are fragmented in fix sized cells (why?) at inputs and reassembled at outputs
  - In Partridge et al, a cell is 64 B (what are the trade-offs?)



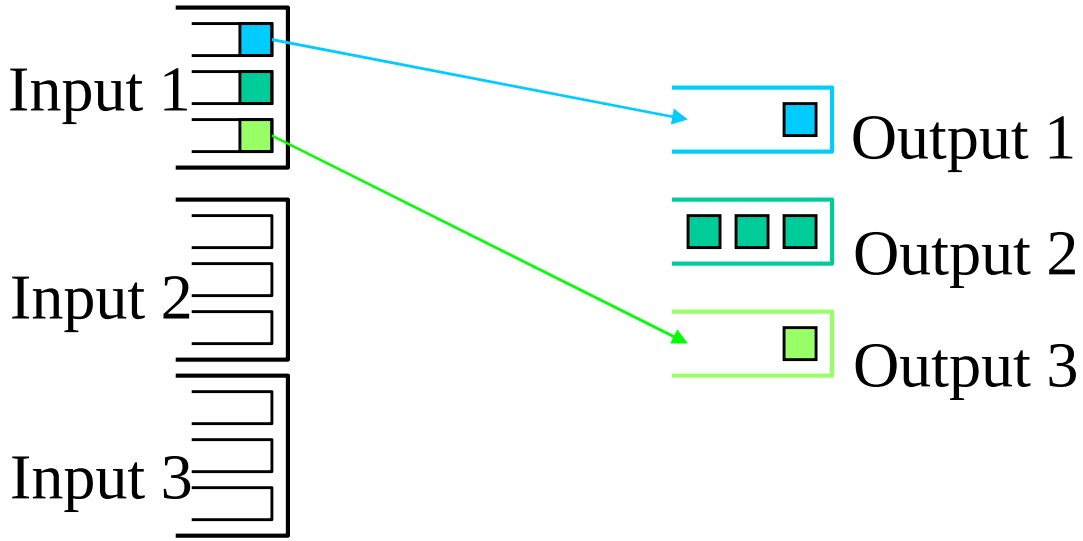
Head-of-line Blocking

- The cell at the head of an input queue cannot be transferred, thus blocking the following cells



Solution to Avoid Head-of-line Blocking

- Maintain at each input N virtual queues, i.e., one per output



Cell transfer

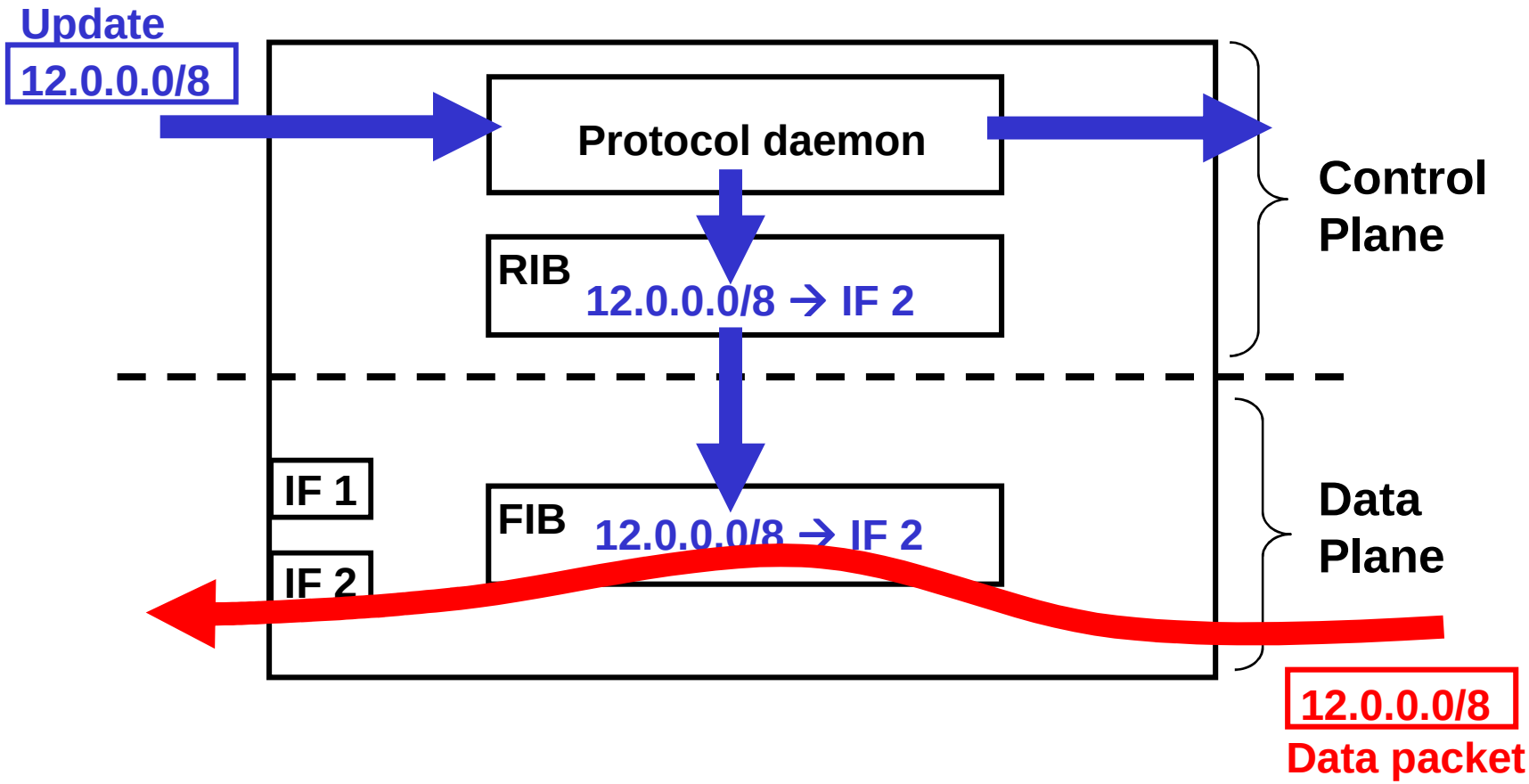
- Schedule:
  - Ideally: find the maximum number of input-output pairs such that:
    - Resolve input/output contentions
    - Avoid packet drops at outputs
    - Packets meet their time constraints (e.g., deadlines), if any
- Example
  - Assign cell preferences at inputs, e.g., their position in the input queue
  - Assign cell preferences at outputs, e.g., based on packet deadlines, or the order in which cells would depart in a OQ router
  - Match inputs and outputs based on their preferences
- Problem:
  - Achieving a high quality matching complex, i.e., hard to do in constant time

Routing vs. Forwarding

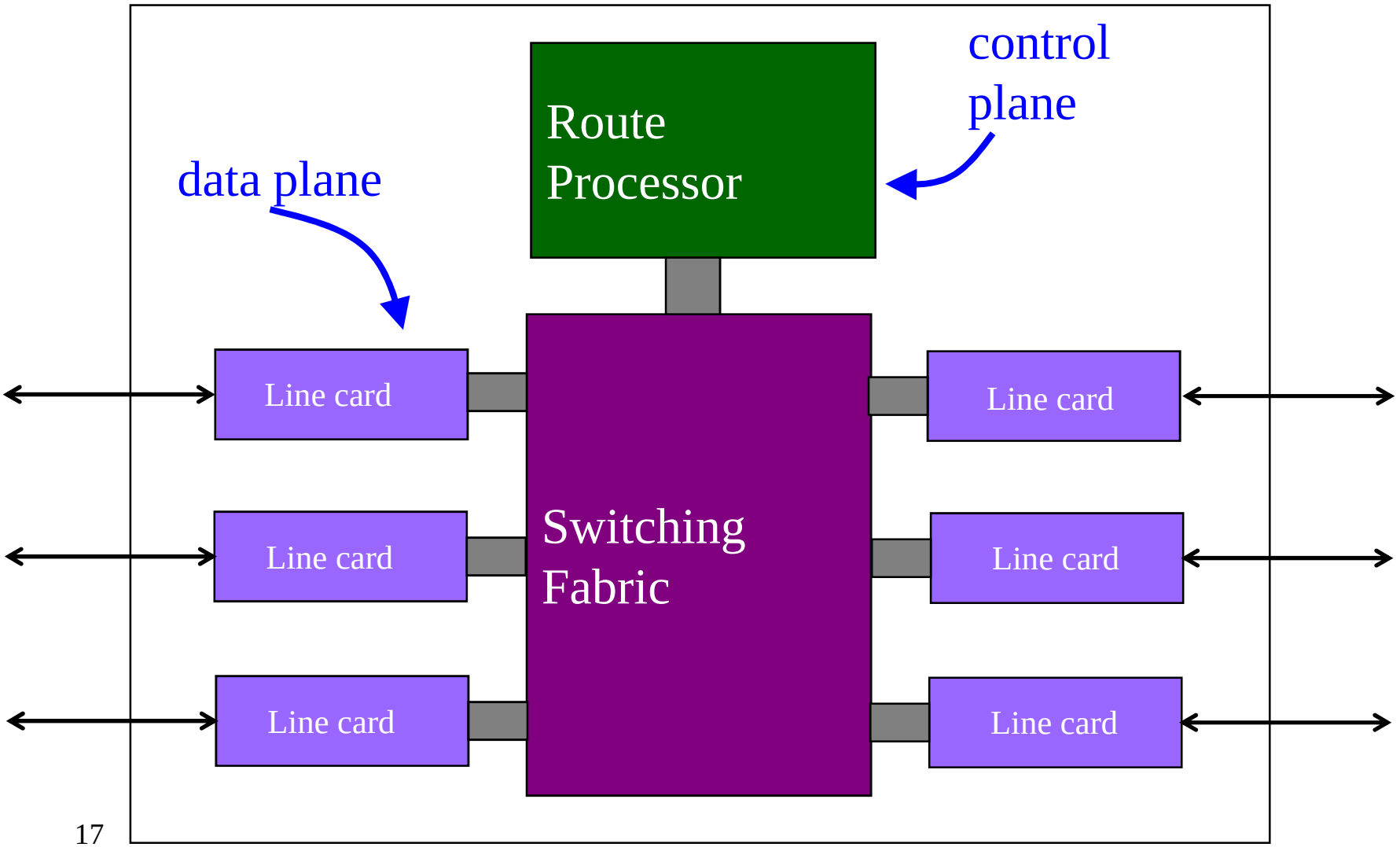
- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Individual router *creating* a forwarding table

- Forwarding: data plane
  - Directing a data packet to an outgoing link
  - Individual router *using* a forwarding table

How the control and data planes work together (logical view)

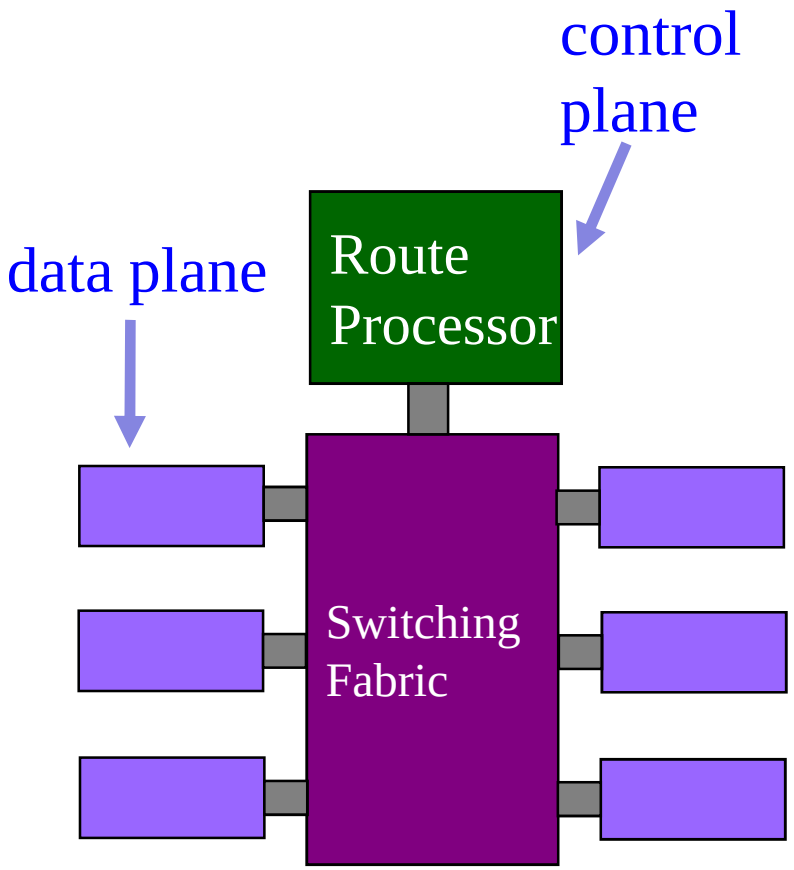


Physical layout of a high-end router



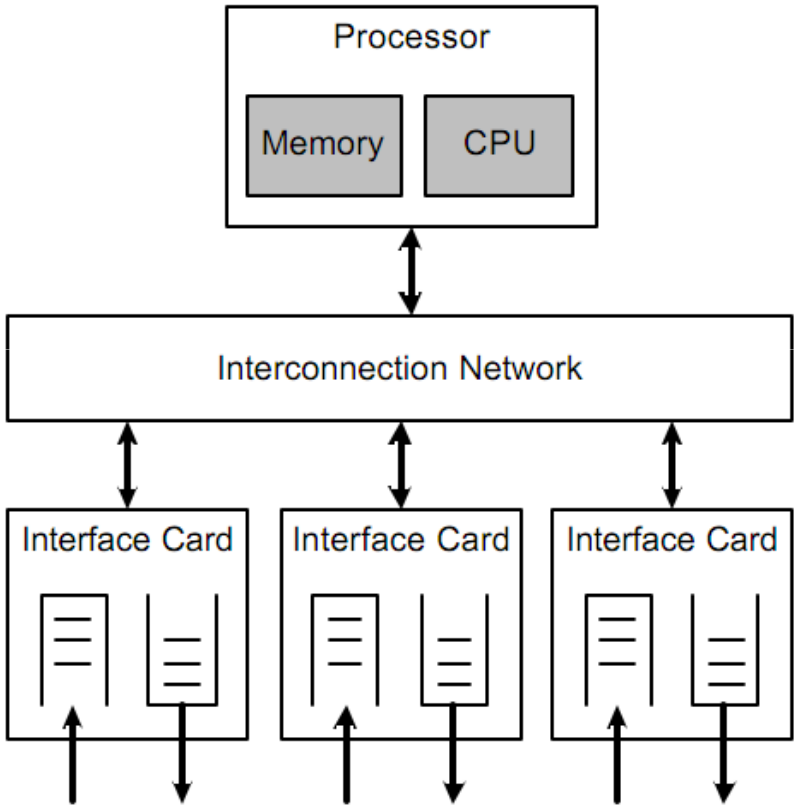
Routing vs. Forwarding

- Control plane's jobs include
  - Route calculation
  - Maintenance of routing table
  - Execution of routing protocols
- On commercial routers, handled by special-purpose processor called "route processor"
- IP forwarding is per-packet processing
  - On high-end commercial routers, IP forwarding is distributed
  - Most work is done by interface cards

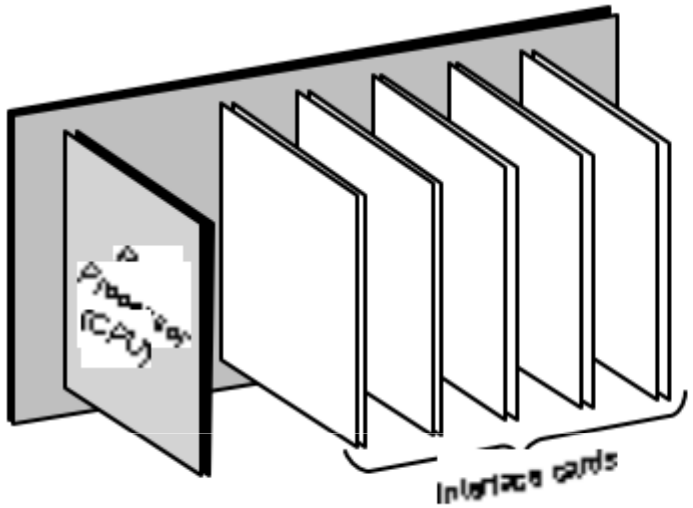


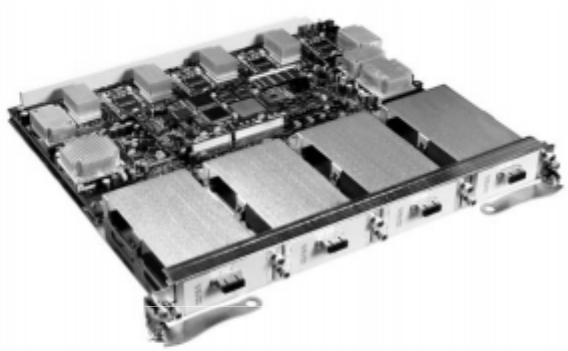
Router Components

- On a PC router:
  - Interconnection network is the PCI bus
  - Interface cards are the NICs (e.g., Ethernet cards)
  - All forwarding and routing is done on a commodity CPU
- On commercial routers:
  - Interconnection network and interface cards are sophisticated, special-purpose hardware
  - Packet forwarding oftend implemented in a custom ASIC
  - Only routing (control plane) is done on the commodity CPU (route processor)



Slotted Chassis





- Large routers are built as a slotted chassis
  - Interface cards are inserted in the slots
  - Route processor is also inserted as a slot
- This simplifies repairs and upgrades of components
  - E.g., “hot-swapping” of components

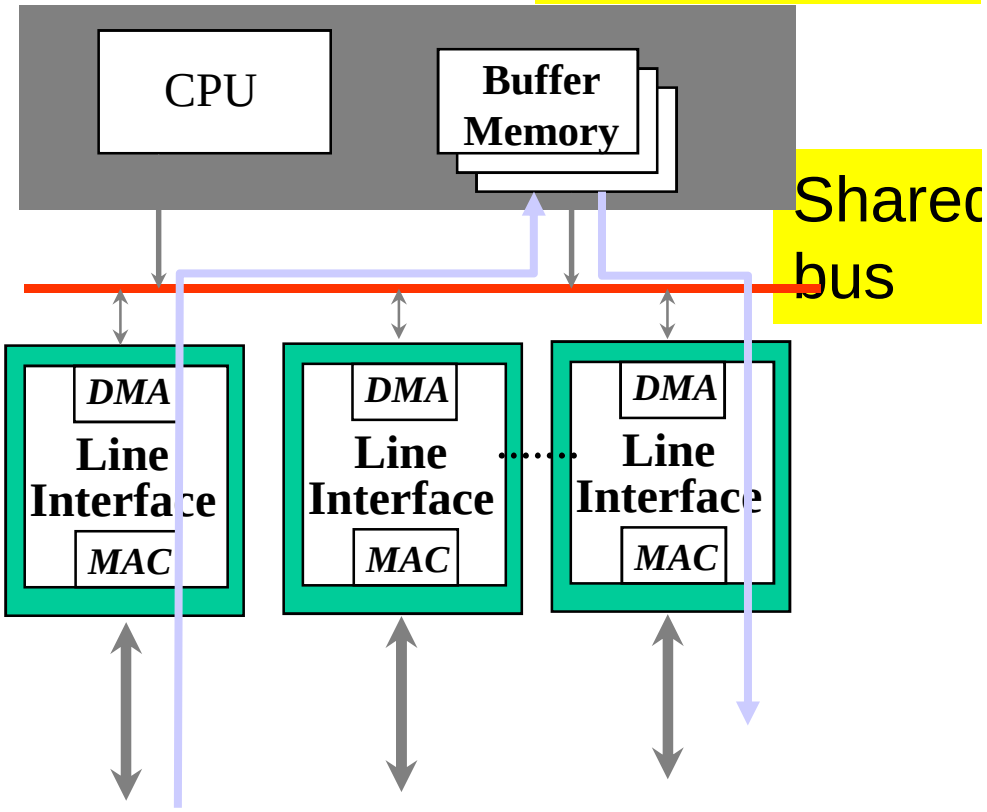
Evolution of router architectures

- Early routers were just general-purpose computers
- Today, high-performance routers resemble mini data centers
  - Exploit parallelism
  - Specialized hardware

- Until 1980s (1st generation): standard computer
- Early 1990s (2nd generation): delegate packet processing to interfaces
- Late 1990s (3rd generation): distributed architecture
- Today: distributed across multiple racks

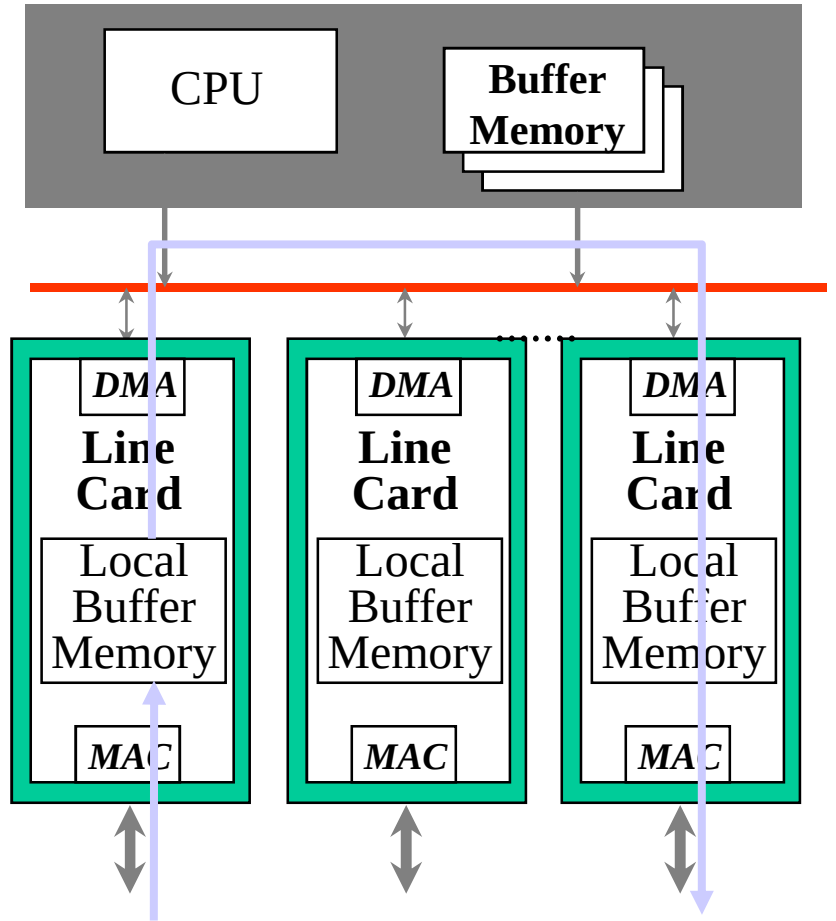
First generation routers

- This architecture is still used in low-end routers
- Arriving packets are copied to main memory via direct memory access (DMA)
- Interconnection network is a backplane (shared bus)
- All IP forwarding functions are performed by a commodity CPU
- Routing cache at processor can accelerate the routing table lookup
- Drawbacks:
  - Forwarding performance is limited by the CPU
  - Capacity of shared bus limits the number of interface cards that can be connected



Second generation routers

- Bypasses memory bus with direct transfer over bus between line cards
- Moves forwarding decisions local to card to reduce CPU utilization
- Trap to CPU for "slow" operations



Typically <5Gb/s aggregate capacity

Speeding up the common case with a "Fast path"

- IP packet forwarding is complex
  - But, vast majority of packets can be forwarded with simple algorithm
  - Main idea: put common-case forwarding in hardware, trap to software on exceptions
  - Example: BBN router had 85 instructions for fast-path code, which fits entirely in L1 cache
- Non-common cases handled by slow path:
  - Route cache misses
  - Errors (e.g., ICMP time exceeded)
  - IP options
  - Fragmented packets
  - Multicast packets

Improving upon second-generation routers

- Control plane must remember lots of information (BGP attributes, etc.)
  - But data plane only needs to know FIB
  - Smaller, fixed-length attributes
  - Idea: store FIB in hardware

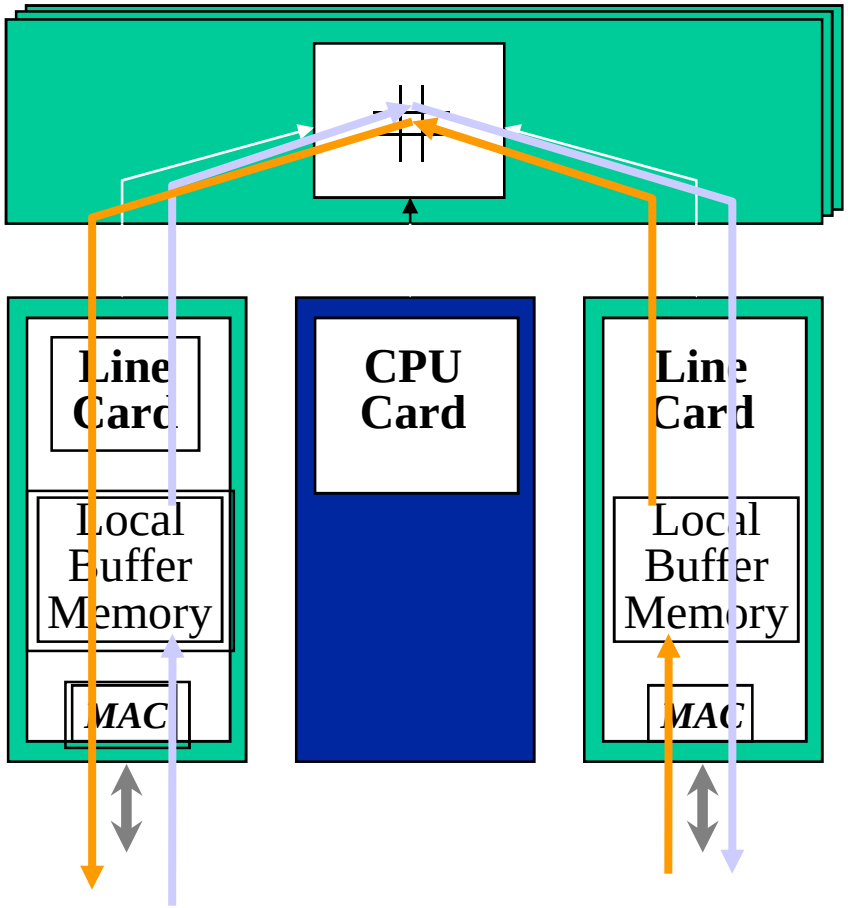
- Going over the bus adds delay
  - Idea: Cache FIB in line cards
  - Send directly over bus to outbound line card

Improving upon second-generation routers

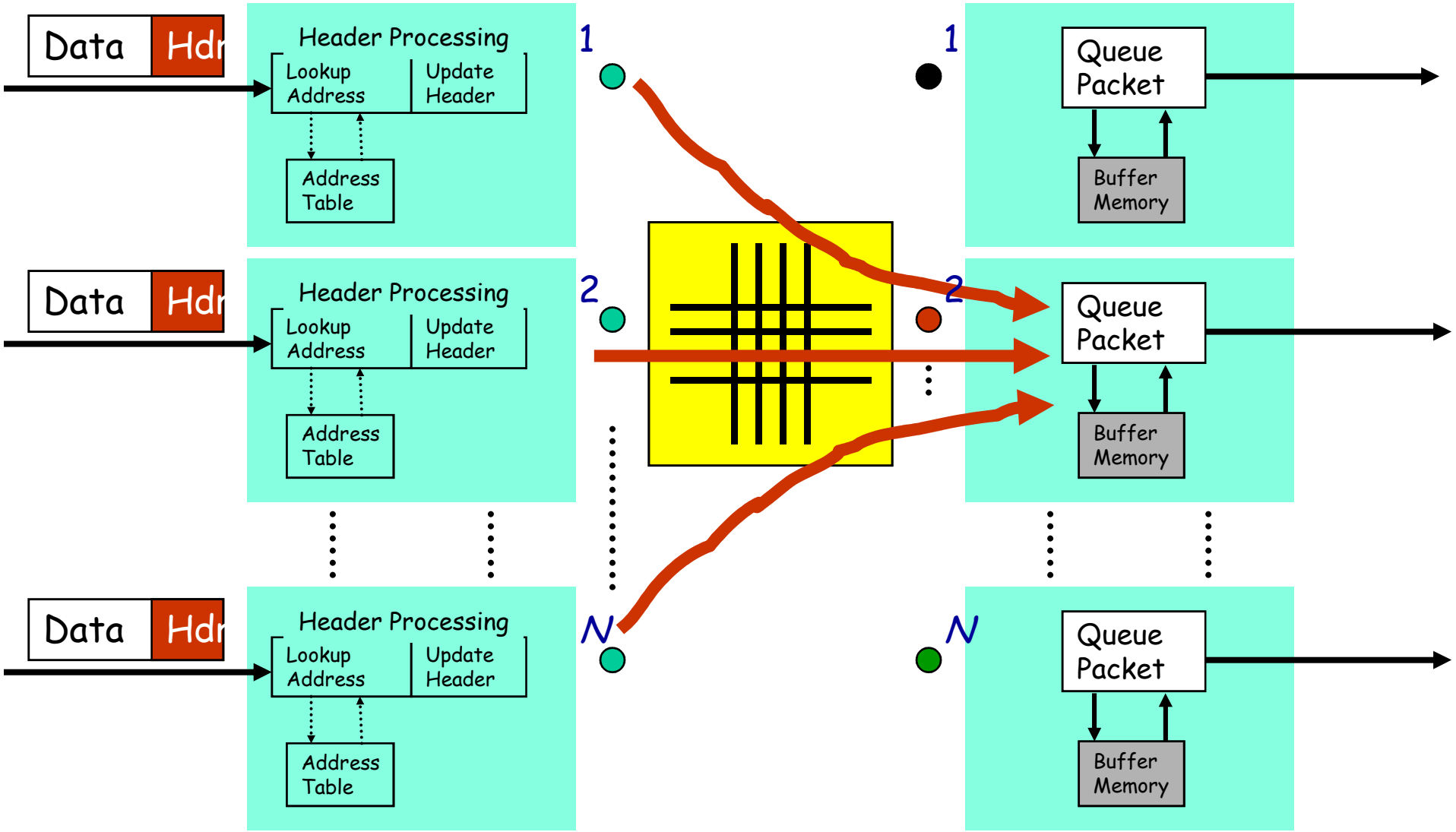
- Shared bus is a big bottleneck
  - E.g., *modern* PCI bus (PCIx16) is only 32Gbit/sec (in theory)
  - Almost-modern Cisco (XR 12416) is 320 Gbit/sec
  - Ow! How do we get there?
  - Idea: put a "network" inside the router
    - Switched backplane for larger cross-section bandwidths

Third generation routers

- Replace bus with interconnection network (e.g., a crossbar switch)
- Distributed architecture:
  - Line cards operate independently of one another
  - No centralized processing for IP forwarding
- These routers can be scaled to many hundreds of interface cards and capacity of > 1 Tbit/sec



Switch Fabric: From Input to Output



Crossbars

- N input ports, N output ports
  - One per line card, usually
- Every line card has its own forwarding table/classifier/etc --- removes CPU bottleneck
- Scheduler
  - Decides which input/output port pairs to connect in a given *time slot*
  - Often forward fixed-sized "cells" to avoid variable-length time slots
  - Crossbar constraint
    - If input i is connected to output j, no other input connected to j, no other output connected to i
    - Scheduling is a bipartite matching

Data Plane Details: Checksum

- Takes too much time to verify checksum
  - Increases forwarding time by 21%
- Take an optimistic approach: just incrementally update it
  - Safe operation: if checksum was correct it remains correct
  - If checksum bad, it will be anyway caught by end-host
- Note: IPv6 does not include a header checksum anyway!

Slashdot NEWS FOR NERDS. STUFF THAT MATTERS.

Stories Recent Popular Search

Technology: **Cisco Introduces a 322 Tbit/sec. Router**

Posted by kdawson on Tuesday March 09 2010, @02:45PM
from the one-loc-per-second dept.

CWmike writes

> "Today Cisco Systems introduced its next-generation Internet core router, the CRS-3, with about three times the capacity of its current platform. 'The Internet will scale faster than any of us anticipate,' Cisco's John Chambers said while announcing the product. At full scale, the CRS-3 has a capacity of 322Tbit/sec., roughly three times that of the CRS-1, introduced in 2004. It also has more than 12 times the capacity of its nearest competitor, Chambers said. The CRS-3 will help the Internet evolve from a messaging to an entertainment and media platform, with video emerging as the 'killer app,' Chambers said. Using a CRS-3, every person in China, which has a populatio 1.3 billion, could participate in a video phone call at the same time. (Or you cc nearly one Library of Congress per second through the device, or give everyone in San Fransisco a 1Gbps internet connection.) AT&T said it has been using the CRS-3 to test 100Gbit/sec. data links in tests on a commercial fiber route in Florida and Louisiana."

- Multi-chassis router
  - A single router that is a distributed collection of racks
  - Scales to 322 Tbps, can replace an entire PoP

Why multi-chassis routers?

- ~ 40 routers per PoP (easily) in today's Intra-PoP architectures
- Connections between these routers require the same expensive line cards as inter-PoP connections
  - Support forwarding tables, QoS, monitoring, configuration, MPLS
  - Line cards are dominant cost of router, and racks often limited to sixteen 40 Gbps line cards
- Each connection appears as an adjacency in the routing protocol
  - Increases IGP/iBGP control-plane overhead
  - Increases complexity of scaling techniques such as route reflectors and summarization

Multi-chassis routers to the rescue

- Multi-chassis design: each line-card chassis has some fabric interface cards
  - Do not use line-card slots: instead uses a separate, smaller connection
  - Do not need complex packet processing logic → much cheaper than line cards
- Multi-chassis router acts as one router to the outside world
  - Simplifies administration
  - Reduces number of iBGP adjacencies and IGP nodes/links without resorting to complex scaling techniques
- However, now the multi-chassis router becomes a distributed system → Interesting research topics
  - Needs rethinking of router software (distributed and parallel)
  - Needs high resilience (no external backup routers)

Matching Algorithms

Administrivia

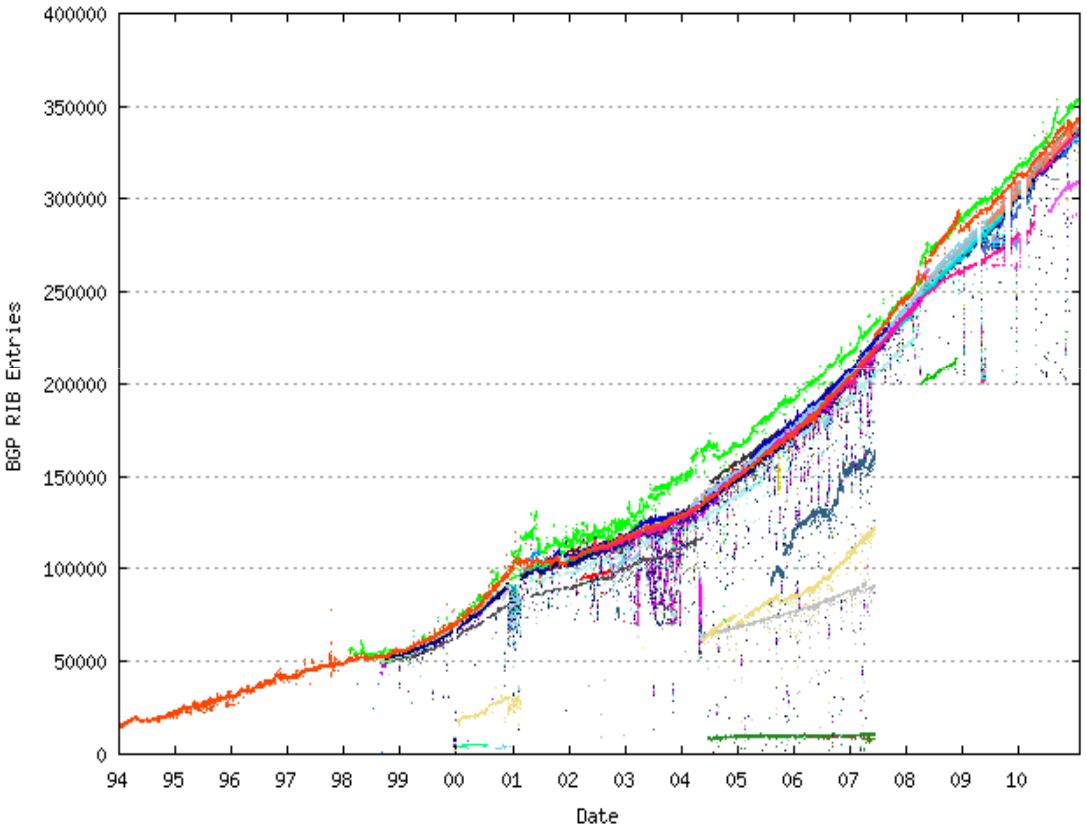
- Lecture topics+outlines due today
  - Next step: slide-based outline by 2/16
  - Put down the text for your slides (no graphics)
- Project topics due on 2/10
  - Send me a 1 paragraph description of your project, names of people in your group

What's so hard about IP packet forwarding?

- Back-of-the-envelope numbers
  - Line cards can be 40 Gbps today (OC-768)
    - Getting faster every year!
  - To handle minimum-sized packets (~40b)
    - 125 Mpps, or 8ns per packet
    - Can use parallelism, but need to be careful about reordering
- For each packet, you must
  - Do a routing lookup (where to send it)
  - Schedule the crossbar
  - Maybe buffer, maybe QoS, maybe ACLs,…

Routing lookups

- Routing tables: 200,000 to 1M entries
  - Router must be able to handle routing table loads 5-10 years hence
- How can we store routing state?
  - What kind of memory to use?
- How can we quickly lookup with increasingly large routing tables?



Memory technologies

Technology	Single chip density	$/MByte	Access speed	Watts/ chip
Dynamic RAM (DRAM) cheap, slow	64 MB	$0.50-$0.75	40-80ns	0.5-2W
Static RAM (SRAM) expensive, fast, a bit higher heat/power	4 MB	$5-$8	4-8ns	1-3W
Ternary Content Addressable Memory (TCAM) very expensive, very high heat/power, very fast (does parallel lookups in hardware)	1 MB	$200-$250	4-8ns	15-30W

- Vendors moved from DRAM (1980s) to SRAM (1990s) to TCAM (2000s)
- Vendors are now moving back to SRAM and parallel banks of DRAM due to power/heat

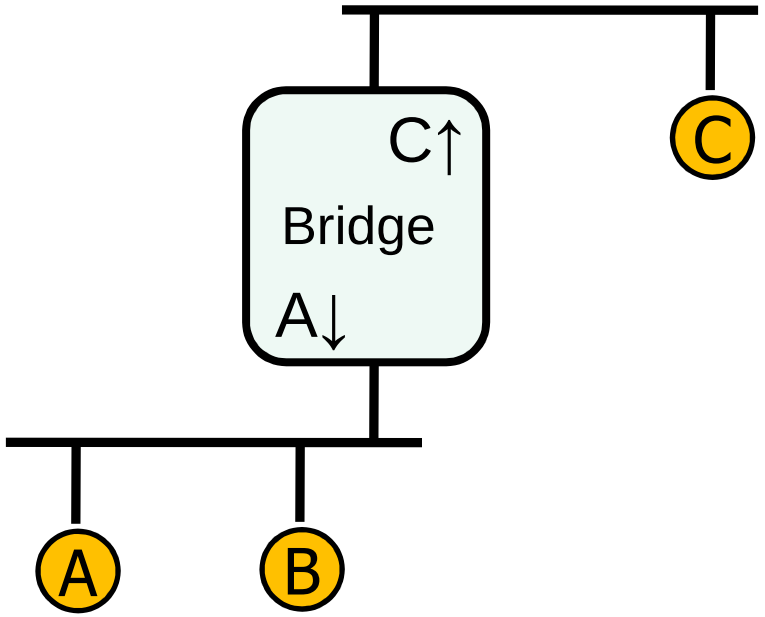
Fixed-Length Matching Algorithms

Ethernet Switch

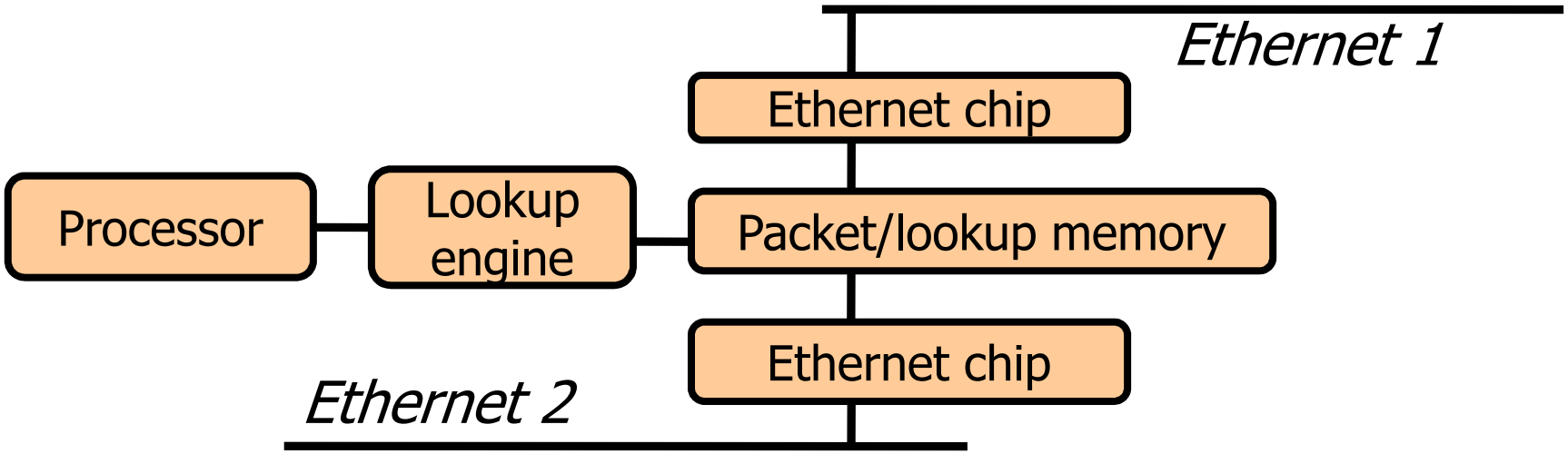
- Lookup frame DA in forwarding table.
  - If known, forward to correct port.
  - If unknown, broadcast to all ports.
- Learn SA of incoming frame.
- Forward frame to outgoing interface.
- Transmit frame onto link.
- How to do this quickly?
  - Need to determine next hop quickly
  - Would like to do so without reducing line rates

Why Ethernet needs wire-speed forwarding

- Scenario:
  - Bridge has a 500 packet buffer
  - Link rate: 1 packet/ms
  - Lookup rate: 0.5 packet/ms
  - A sends 1000 packets to B
  - A sends 10 packets to C
- What happens to C's packets?
  - What would happen if this Bridge was a Router?
- Need wirespeed forwarding

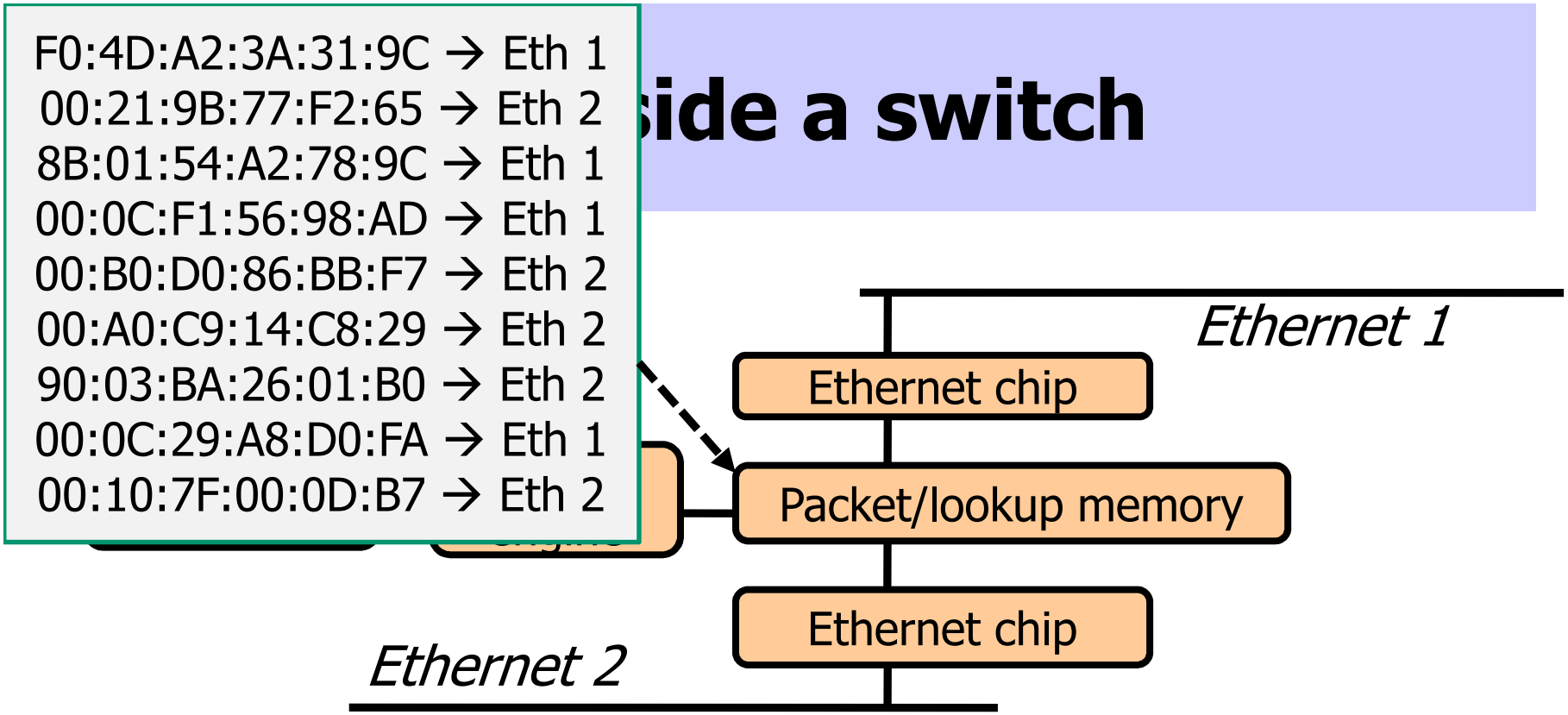


Inside a switch



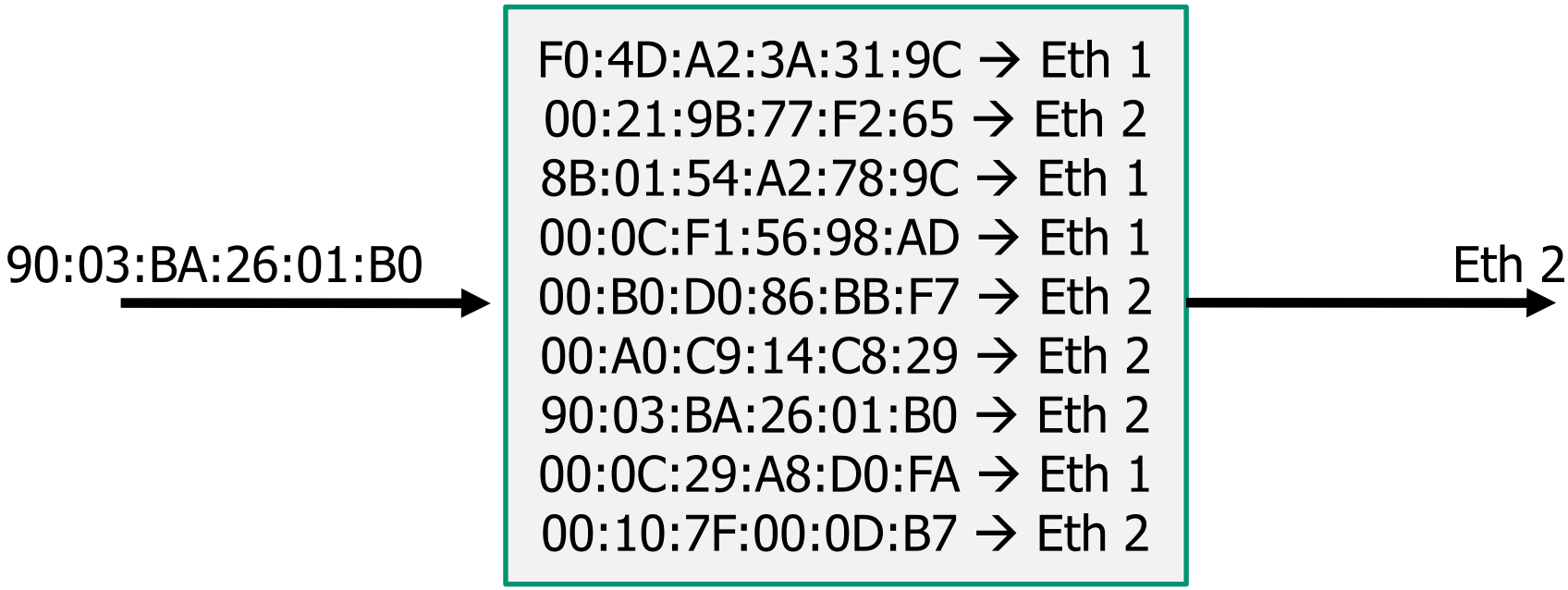
- Packet received from upper Ethernet
- Ethernet chip extracts source address S, stored in shared memory, in receive queue
  - Ethernet chips set in "promiscuous mode"
- Extracts destination address D, given to lookup engine

side a switch



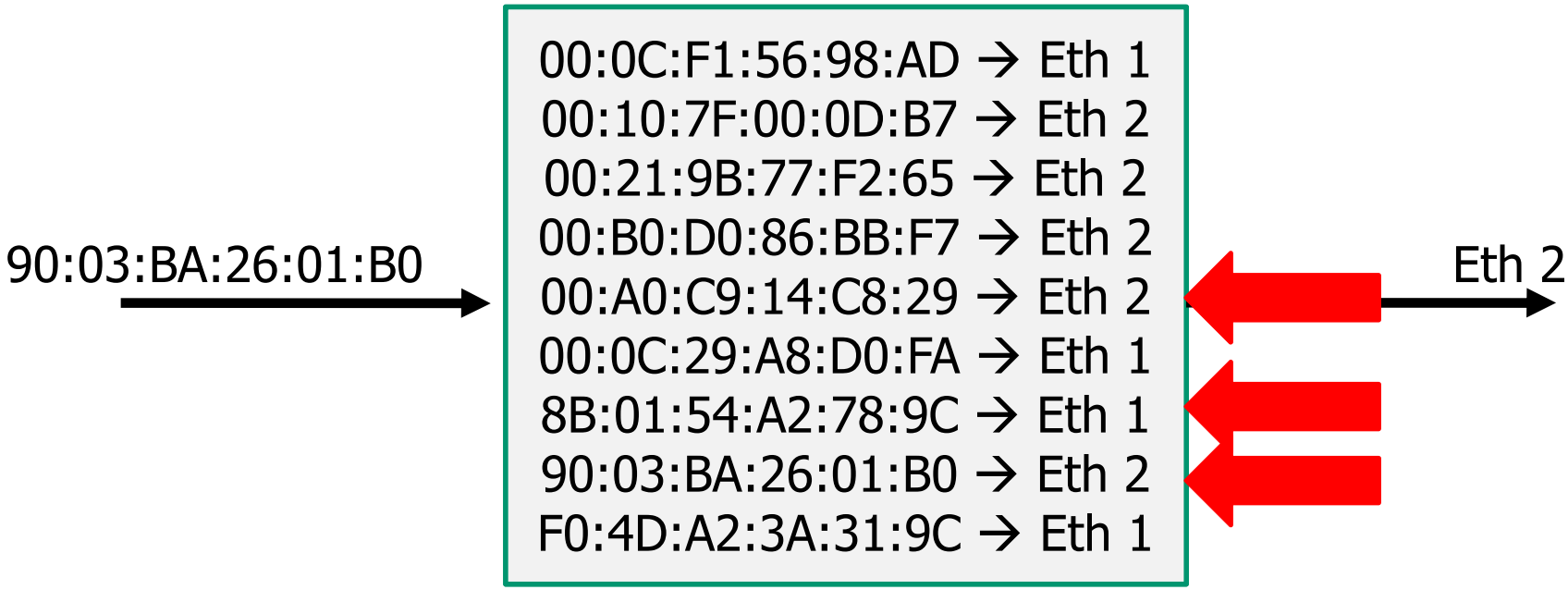
- Lookup engine looks up D in database stored in memory
  - If destination is on upper Ethernet: set packet buffer pointer to free queue
  - If destination is on lower Ethernet: set packet buffer pointer to transmit queue of the lower Ethernet
- How to do the lookup quickly?

Problem overview



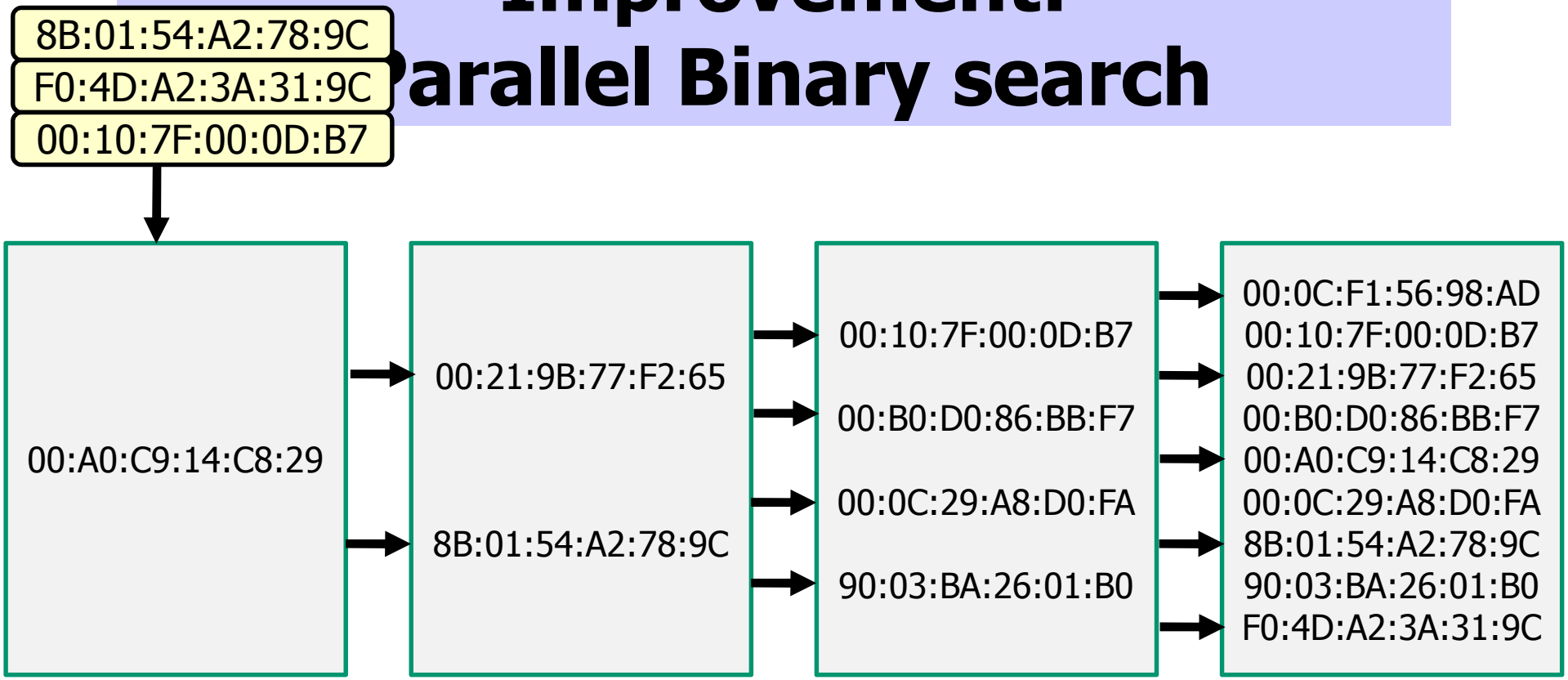
- Goal: given address, look up outbound interface
  - Do this quickly (few instructions/low circuit complexity)
- Linear search too low

Idea #1: binary search



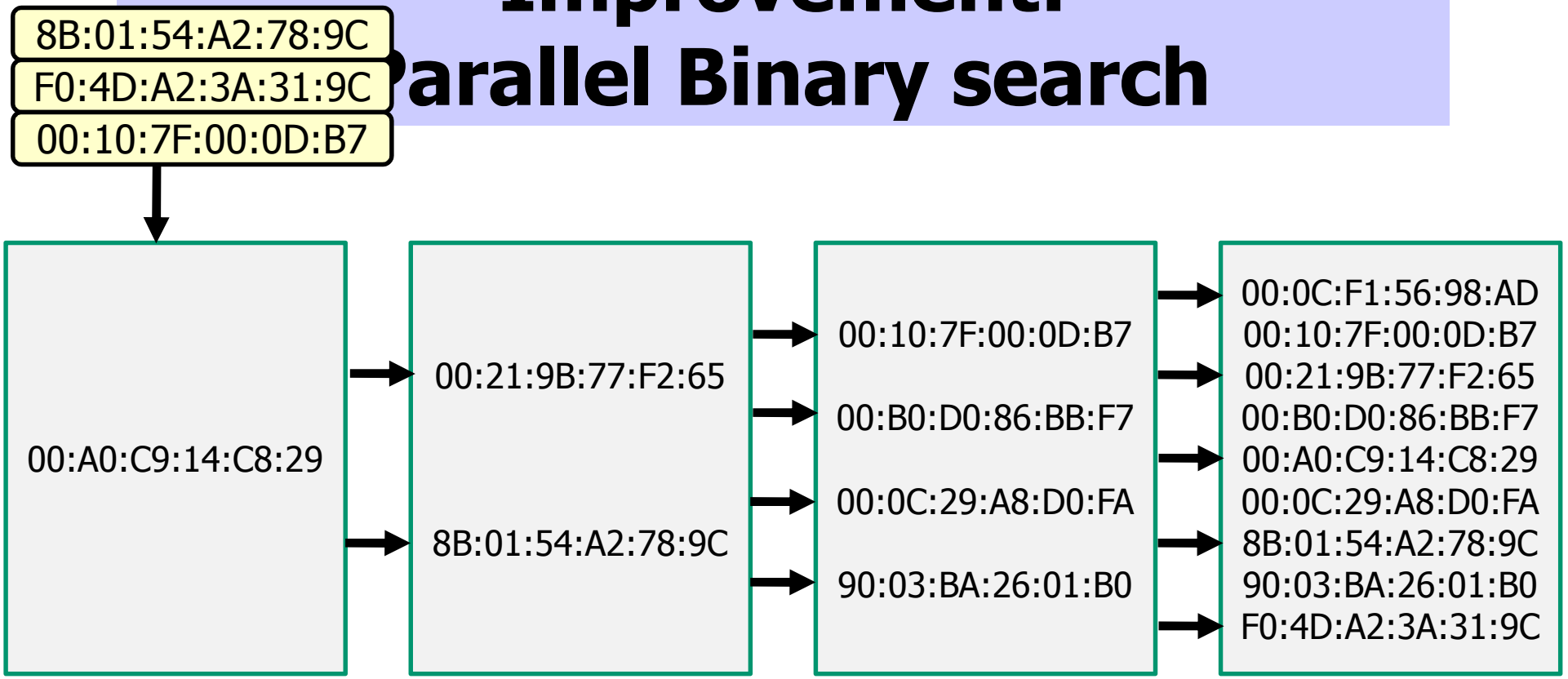
- Put all destinations in a list, sort them, binary search
- Problem: logarithmic time

Improvement: Parallel Binary search



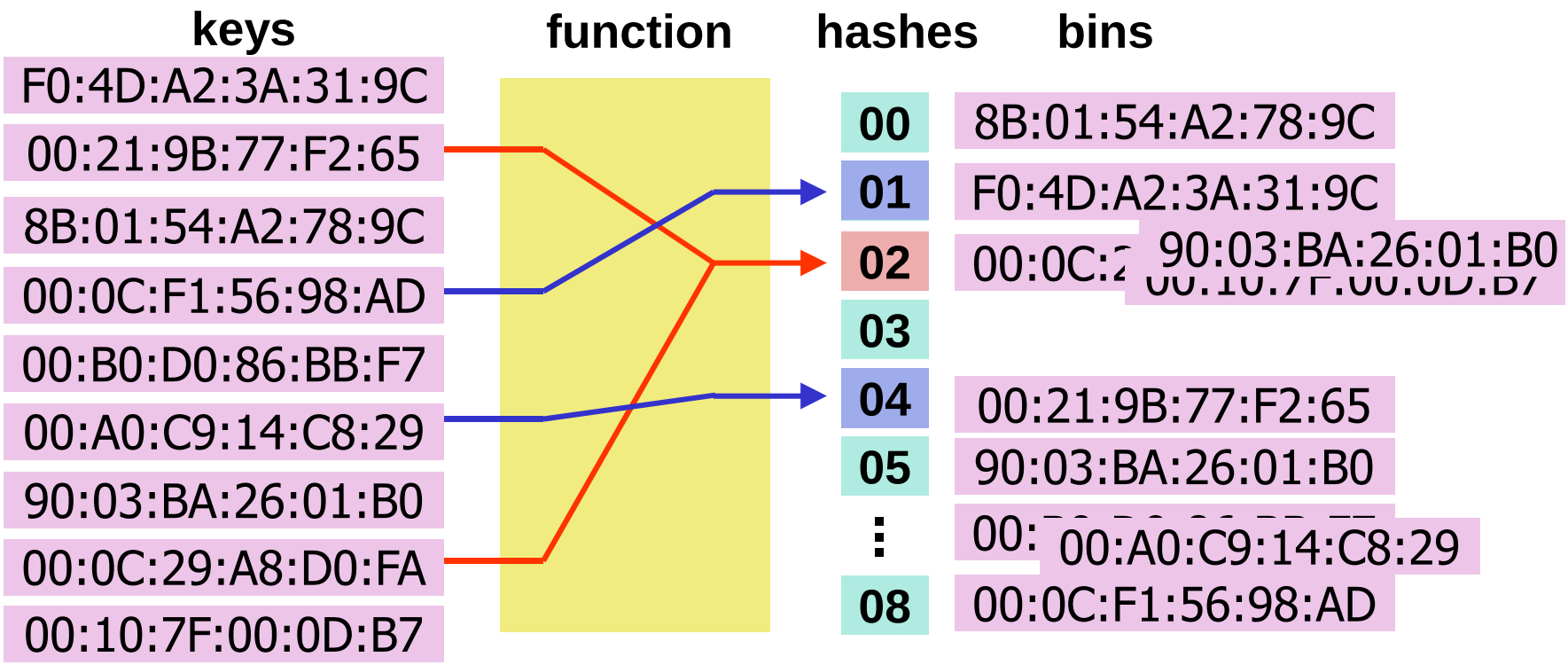
- Packets still have O(log n) delay, but can process O(log n) packets in parallel → O(1)

Improvement: Parallel Binary search



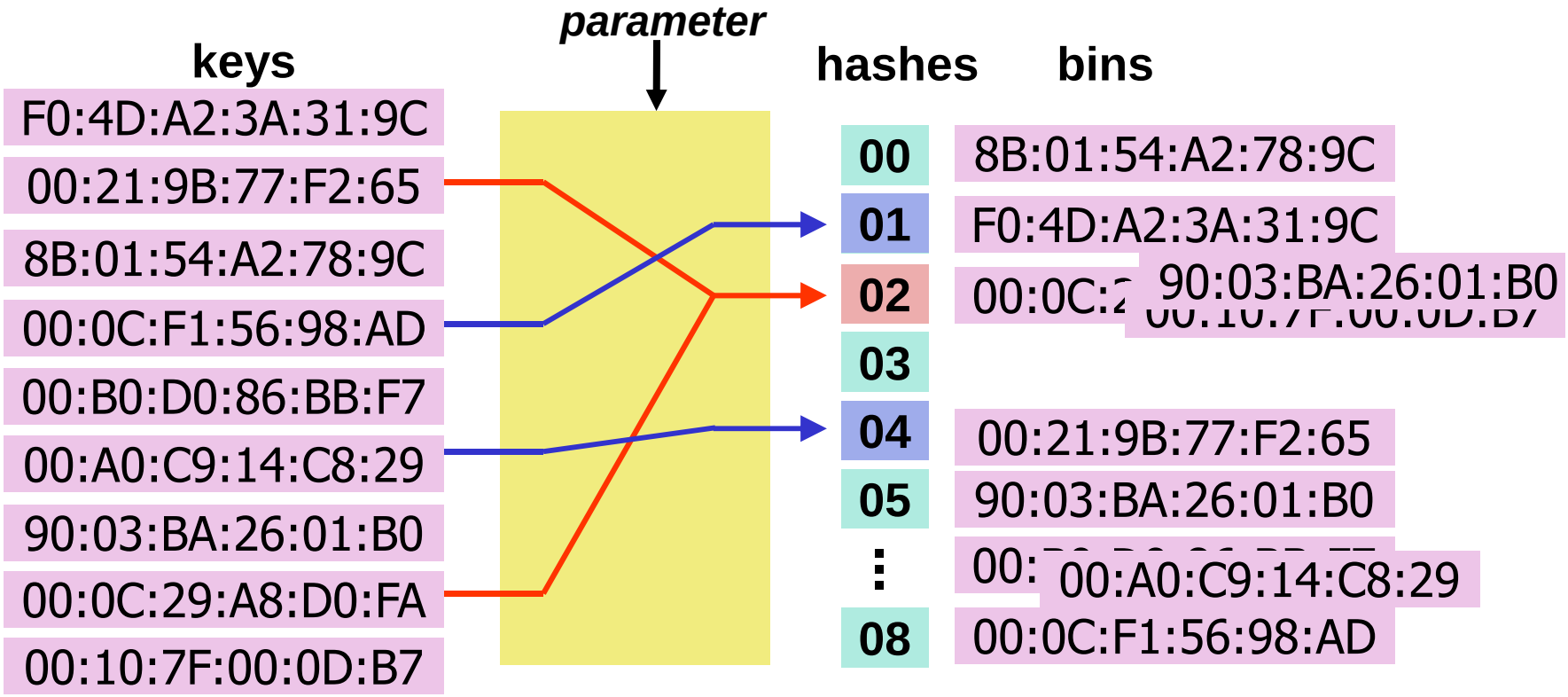
- Packets still have O(log n) delay, but can process O(log n) packets in parallel → O(1)

Idea #2: hashing



- Hash key=destination, value=interface pairs
- Lookup in O(1) with hash
- Problem: chaining (not really O(1))

Improvement: Perfect hashing



- Perfect hashing: find a hash function that maps perfectly with no collisions
- Gigaswitch approach
  - Use a parameterized hash function
  - Precompute hash function to bound worst case number of collisions

Variable-Length Matching Algorithms

Longest Prefix Match

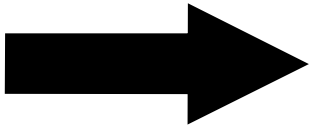
- Not just one entry that matches a destination
  - 128.174.252.0/24 and 128.174.0.0/16
  - Which one to use for 128.174.252.14?
  - By convention, Internet routers choose the longest (most-specific) match

- Need variable prefix match algorithms
  - Several methods

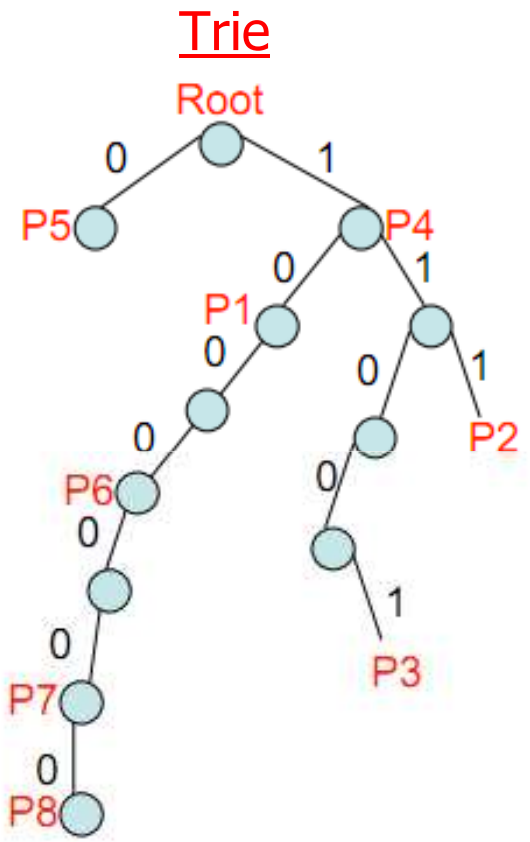
Method 1: Trie

Sample Database

- P1=10*
- P2=111*
- P3=11001*
- P4=1*
- P5=0*
- P6=1000*
- P7=100000*
- P8=1000000*

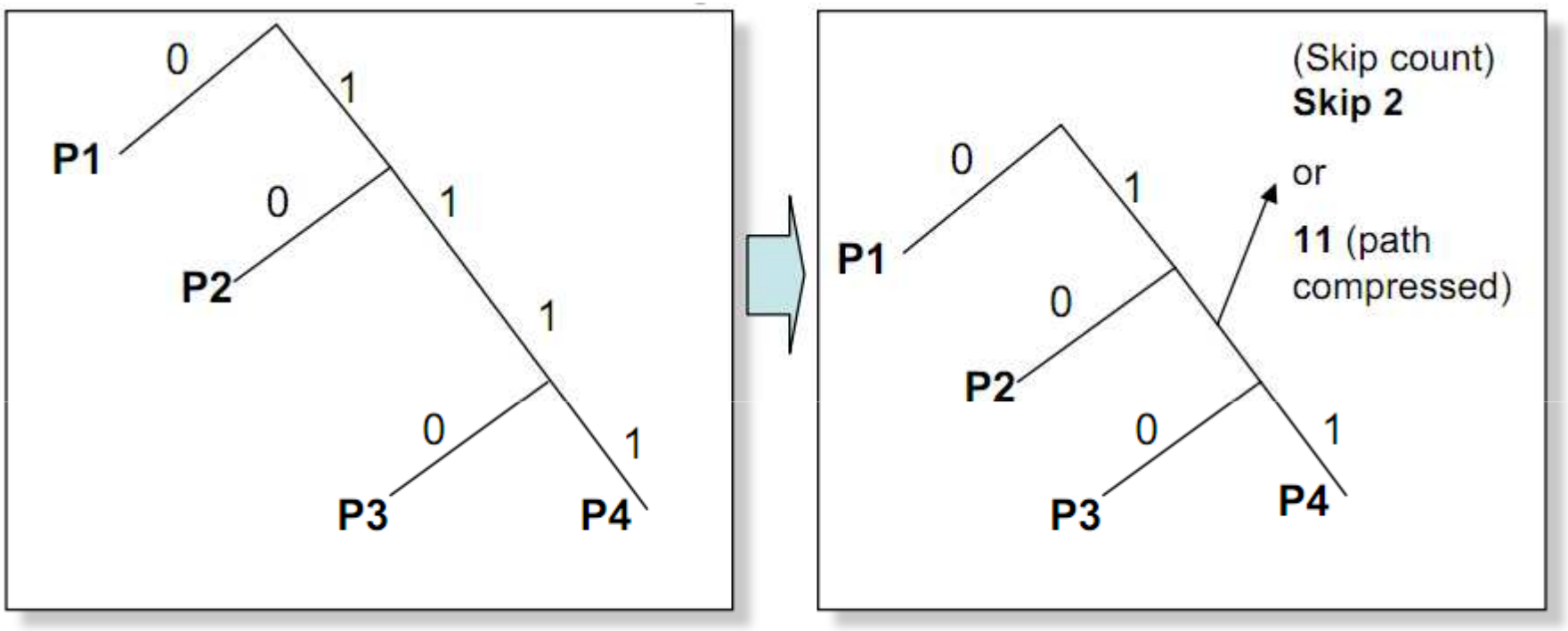


Trie



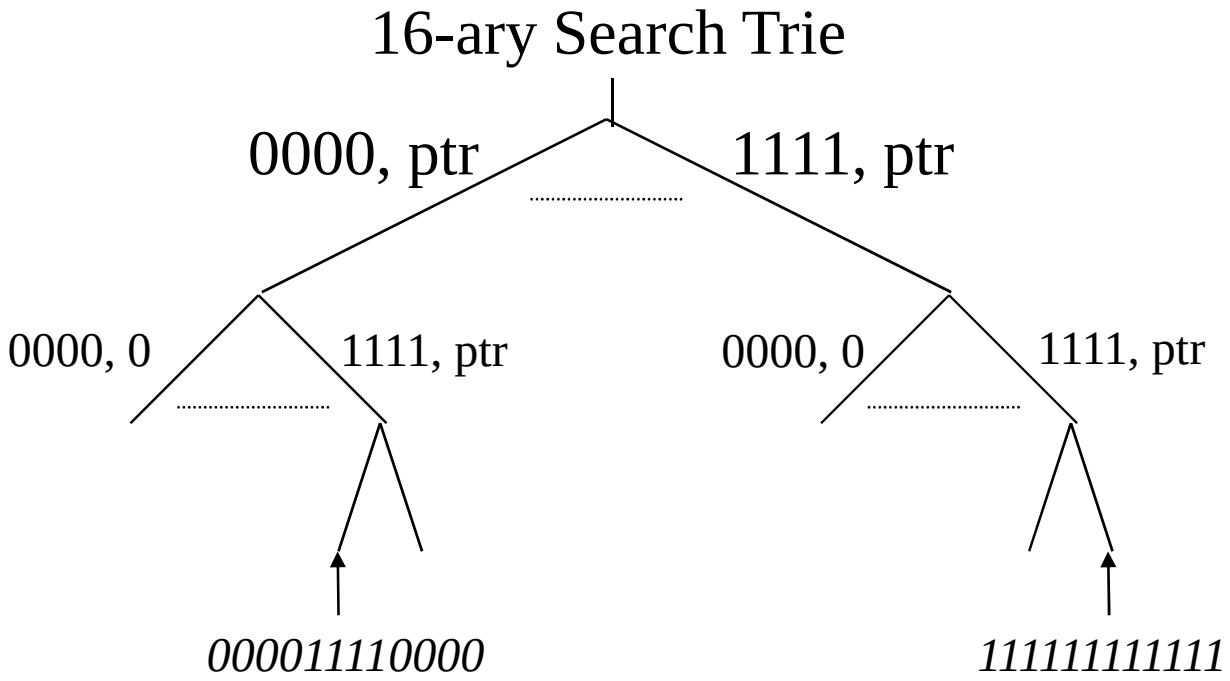
- Tree of (left ptr, right ptr) data structures
- May be stored in SRAM/DRAM
- Lookup performed by traversing sequence of pointers
- Lookup time O(log N) where N is # prefixes

Improvement 1: Skip Counts and Path Compression



- Removing one-way branches ensures # of trie nodes is at most twice # of prefixes
- Using a skip count requires exact match at end and backtracking on failure → path compression is simpler
- Main idea behind Patricia Tries

Improvement 2: Multi-way tree



- Doing multiple comparisons per cycle accelerates lookup
  - Can do this for free to the width of CPU word (modern CPUs process multiple bits per cycle)
- But increases wasted space (more unused pointers)

Improvement 2: Multi-way tree

$$E_w = D^{L-1}\left(1-\left(1-\frac{N}{D^L}\right)^D\right) + \sum_{i=1}^{L-1} D^i((1-D^{i-1})^N - (1-D^{1-i})^N)$$

$$E_n = 1 + D^L\left(1-\frac{N}{D^L}\right)^D + \sum_{i=1}^{L-1} D^i - D^{i-1}(1-D^{i-1})^N$$

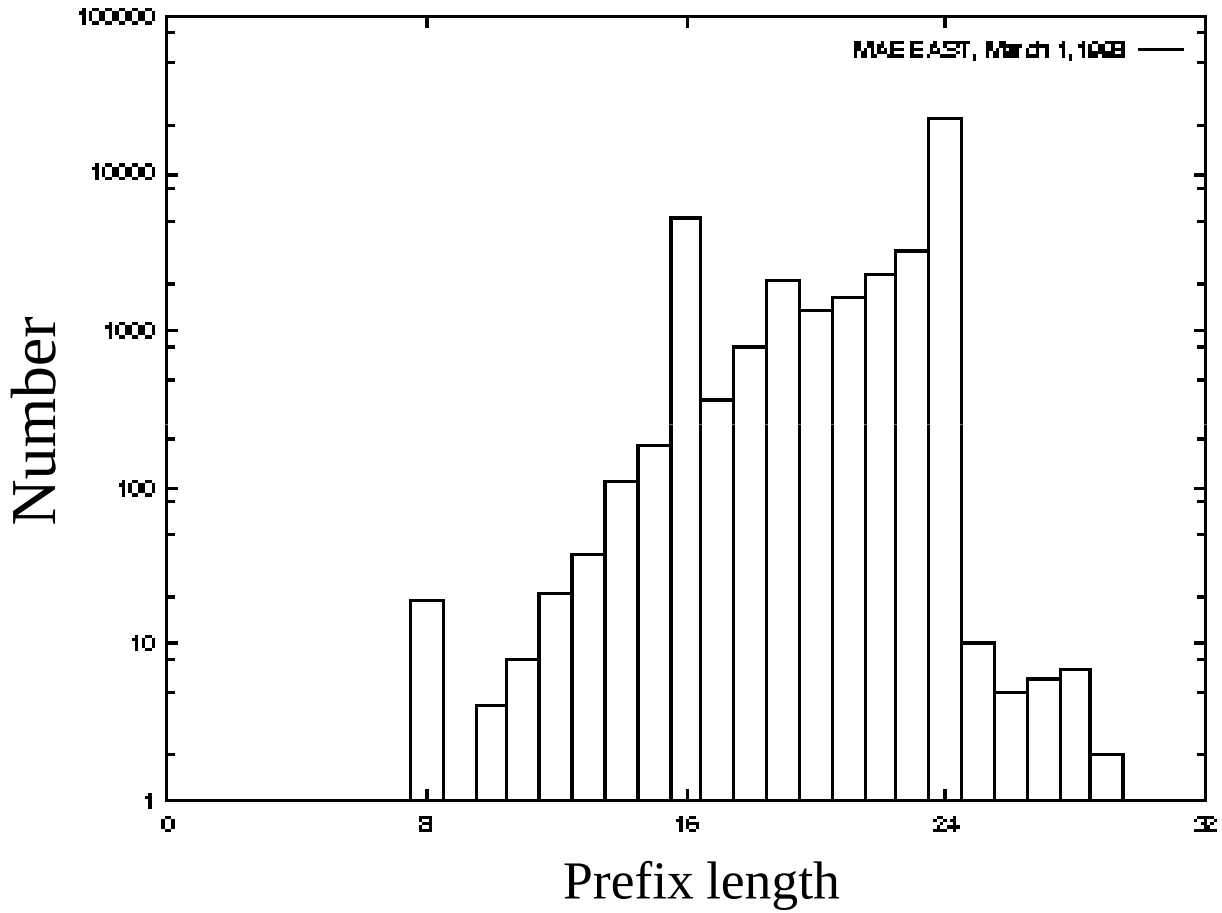
Where:

- D = Degree of tree
- L = Number of layers/references
- N = Number of entries in table
- E_n = Expected number of nodes
- E_w = Expected amount of wasted memory

Degree of Tree	*# Mem References*	*# Nodes (x10⁶)*	*Total Memory (Mbytes)*	*Fraction Wasted (%)*
2	48	1.09	4.3	49
4	24	0.53	4.3	73
8	16	0.35	5.6	86
16	12	0.25	8.3	93
64	8	0.17	21	98
256	6	0.12	64	99.5

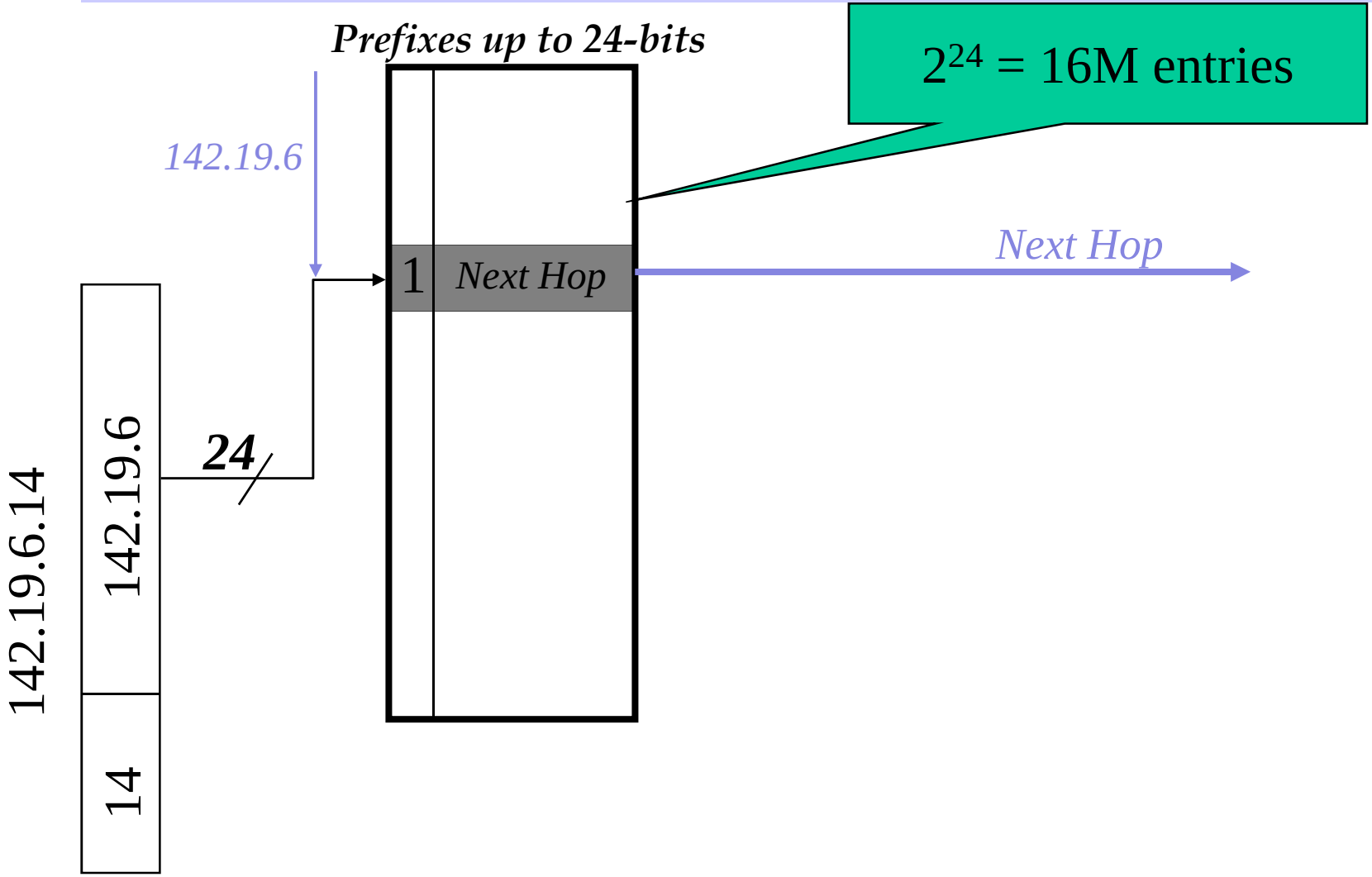
Table produced from 2^{15} randomly generated 48-bit addresses

Method 2: Lookups in Hardware

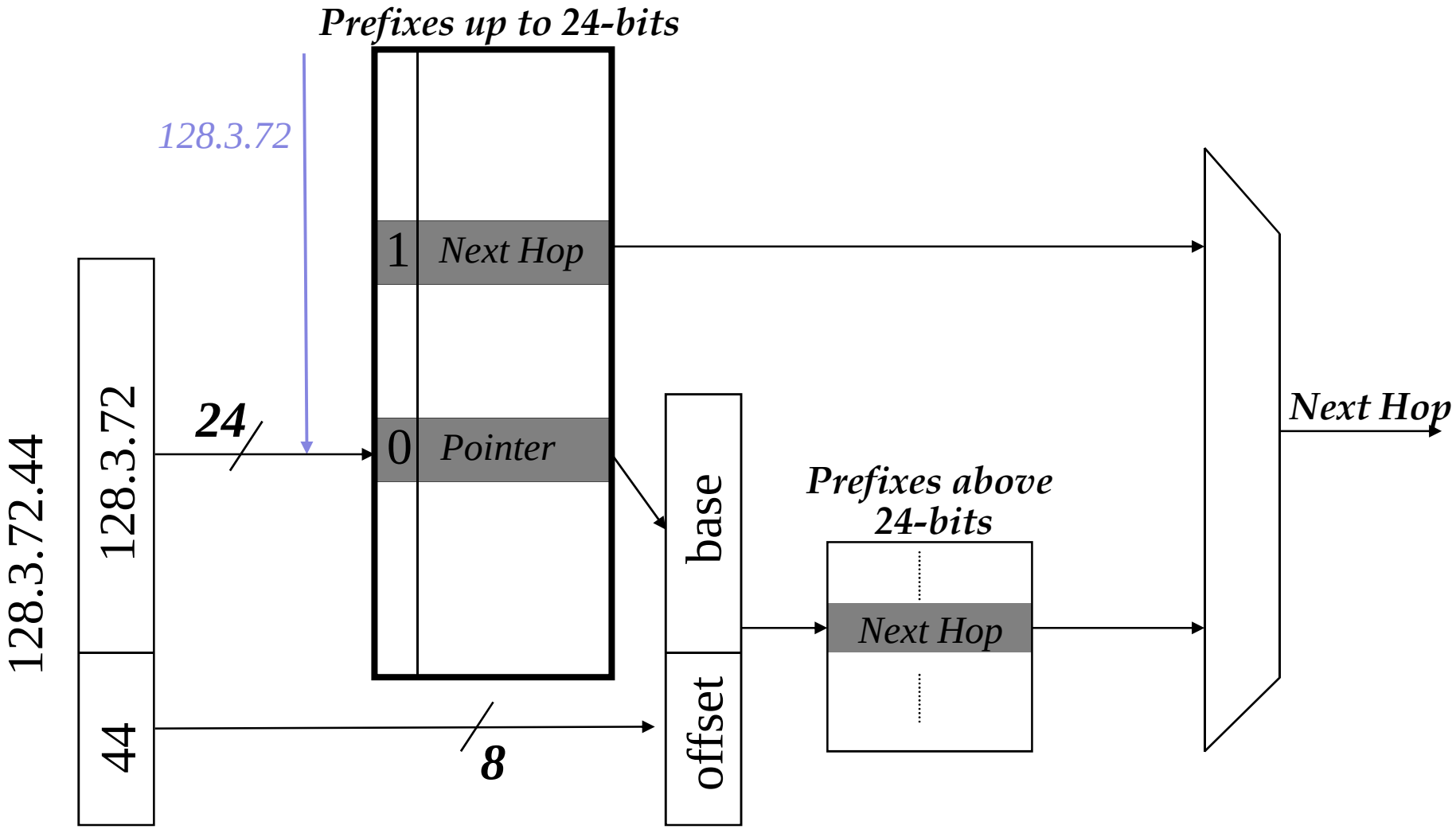


- Observation: most prefixes are /24 or shorter
- So, just store a big 2^24 table with next hop for each prefix
- Nonexistant prefixes → just leave that entry empty

Method 2: Lookups in Hardware



Method 2: Lookups in Hardware



Method 2: Lookups in Hardware

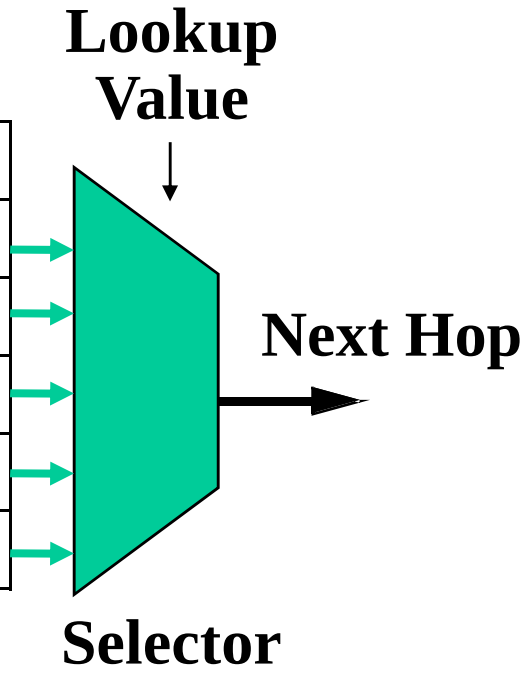
- Advantages
  - Very fast lookups
    - 20 Mpps with 50ns DRAM
  - Easy to implement in hardware

- Disadvantages
  - Large memory required
  - Performance depends on prefix length distribution

Method 3: Ternary CAMs

Associative Memory

Value	Mask	Next hop
10.0.0.0	255.0.0.0	IF 1
10.1.0.0	255.255.0.0	IF 3
10.1.1.0	255.255.255.0	IF 4
10.1.3.0	255.255.255.0	IF 2
10.1.3.1	255.255.255.255	IF 2



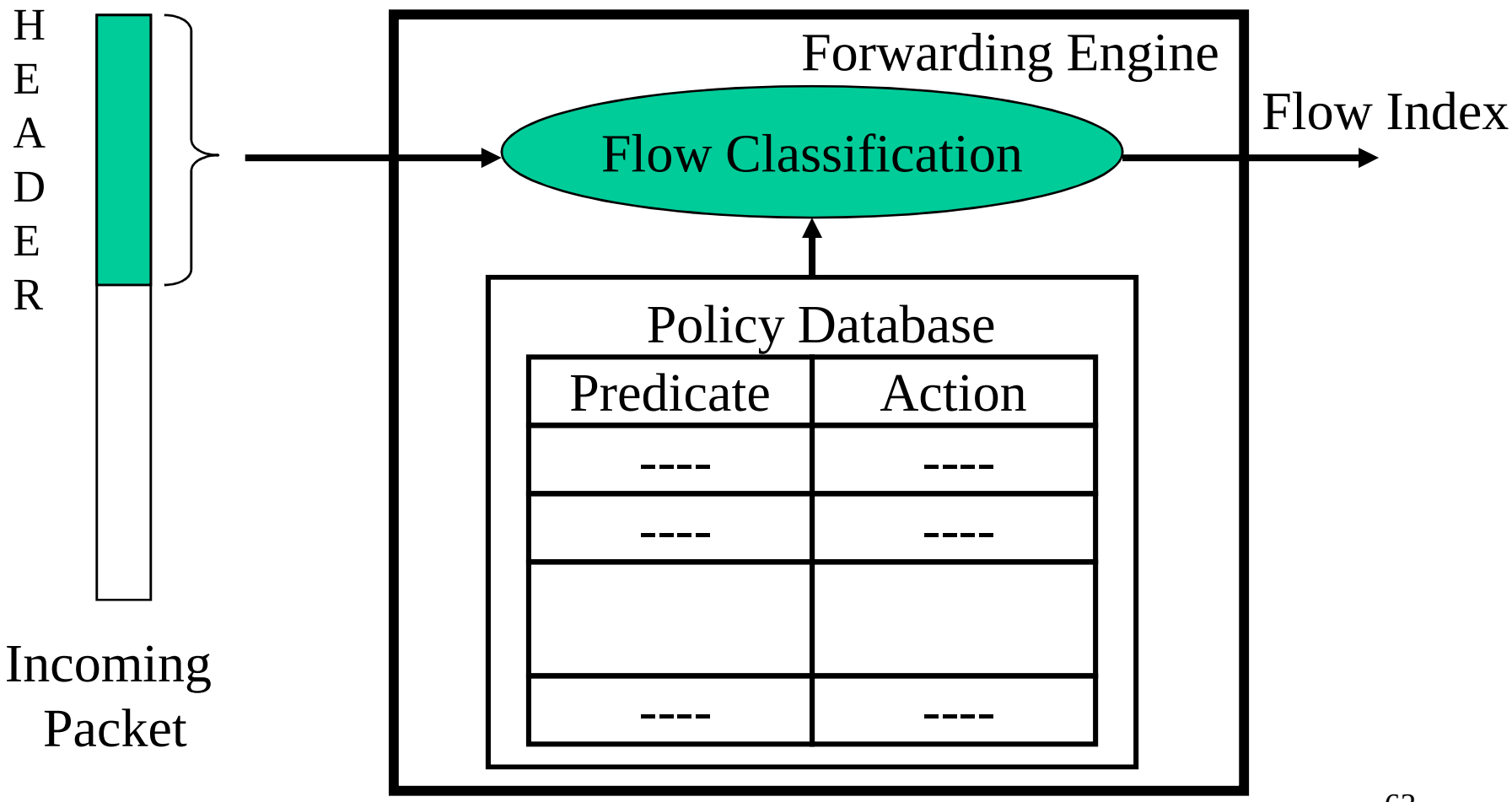
- "Content Addressable"
  - Hardware searches entire memory to find supplied value
  - Similar interface to hash table
- "Ternary": memory can be in three states
  - True, false, don't care
  - Hardware to treat don't care as wildcard match

Classification Algorithms

Providing Value-Added Services

- Differentiated services
  - Regard traffic from AS#33 as `platinum-grade'
- Access Control Lists
  - Deny udp host 194.72.72.33 194.72.6.64 0.0.0.15 eq snmp
- Committed Access Rate
  - Rate limit WWW traffic from sub-interface#739 to 10Mbps
- Policy-based Routing
  - Route all voice traffic through the ATM network
- Peering Arrangements
  - Restrict the total amount of traffic of precedence 7 from
  - MAC address N to 20 Mbps between 10 am and 5pm
- Accounting and Billing
  - Generate hourly reports of traffic from MAC address M
- → Need to address the Flow Classification problem

Flow Classification

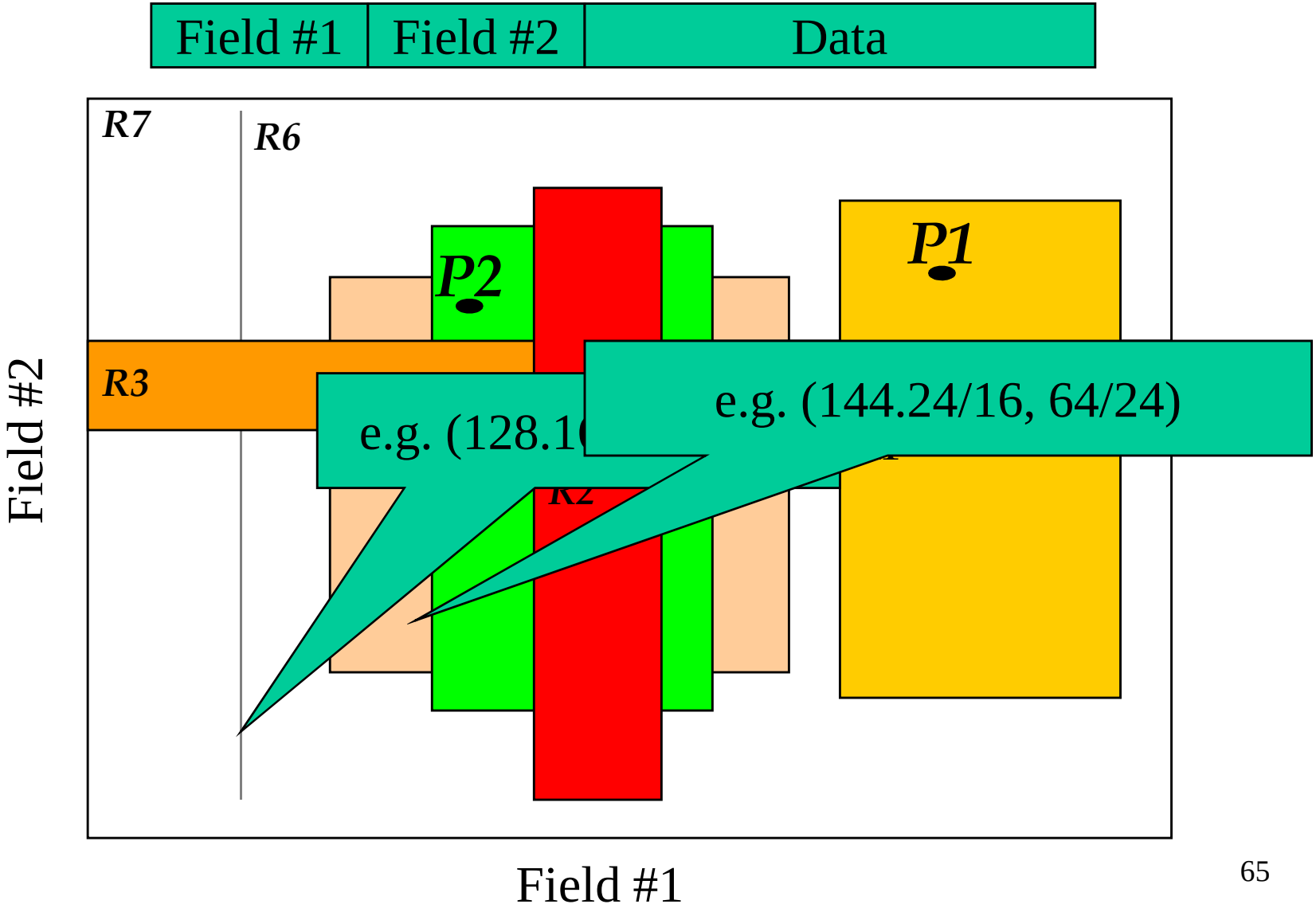


A Packet Classifier

	Field 1	Field 2	…	Field k	Action
Rule 1	152.163.190.69/21	152.163.80.11/32	…	Udp	A1
Rule 2	152.168.3.0/24	152.163.200.157/16	…	Tcp	A2
…	…	…	…	…	…
Rule N	152.168.3.0/16	152.163.80.11/32	…	**Any**	**An**

Given a classifier, find the action associated with the highest priority rule (here, the lowest numbered rule) matching an incoming packet.

Geometric Interpretation in 2D



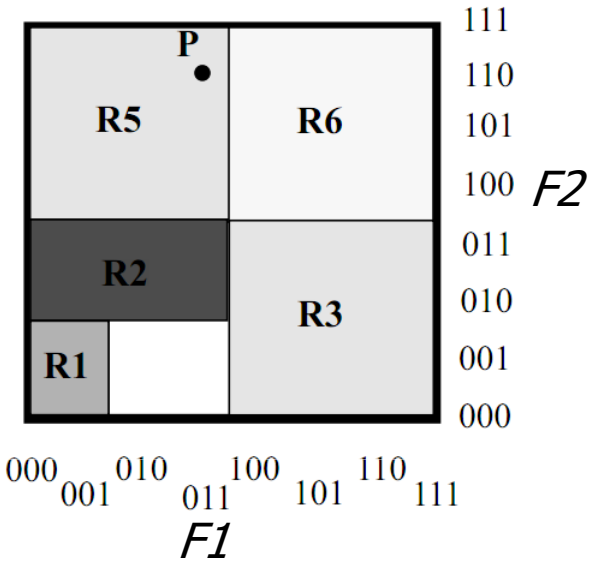
Approach #1: Linear search

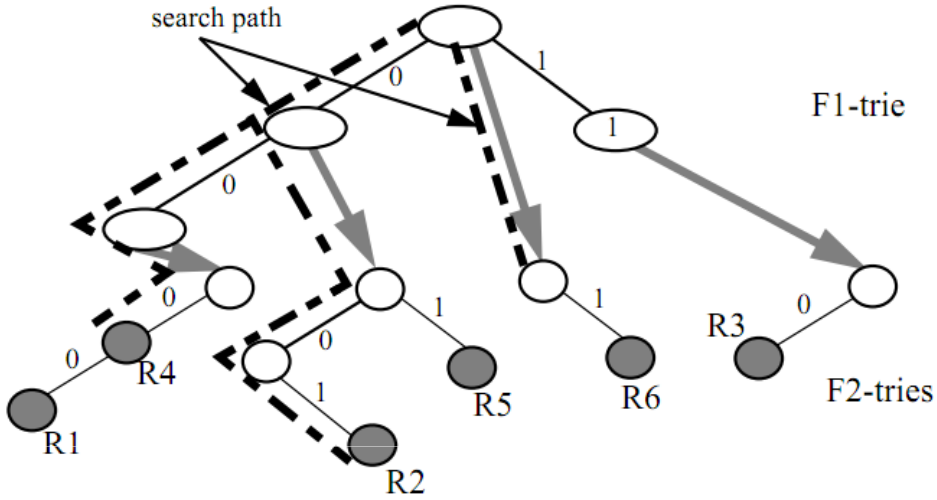
- Build linked list of all classification rules
  - Possibly sorted in order of decreasing priorities
- For each arriving packet, evaluate each rule until match is found
- Pros: simple and storage efficient
- Cons: classification time grows linearly with number of rules
  - Variant: build FSM of rules (pattern matching)

Approach #2: Ternary CAMs

- Similar to TCAM use in prefix matching
  - Need wider than 32-bit array, typically 128-256 bits
- Ranges expressed as don't cares below a particular bit
  - Done for each field
- Pros: O(1) lookup time, simple
- Cons: heat, power, cost, etc.
  - Power for a TCAM row increases proportionally to its width

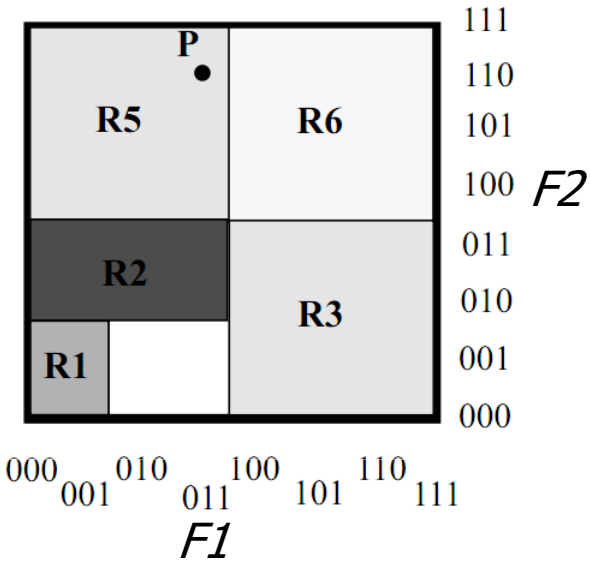
Approach #3: Hierarchical trie

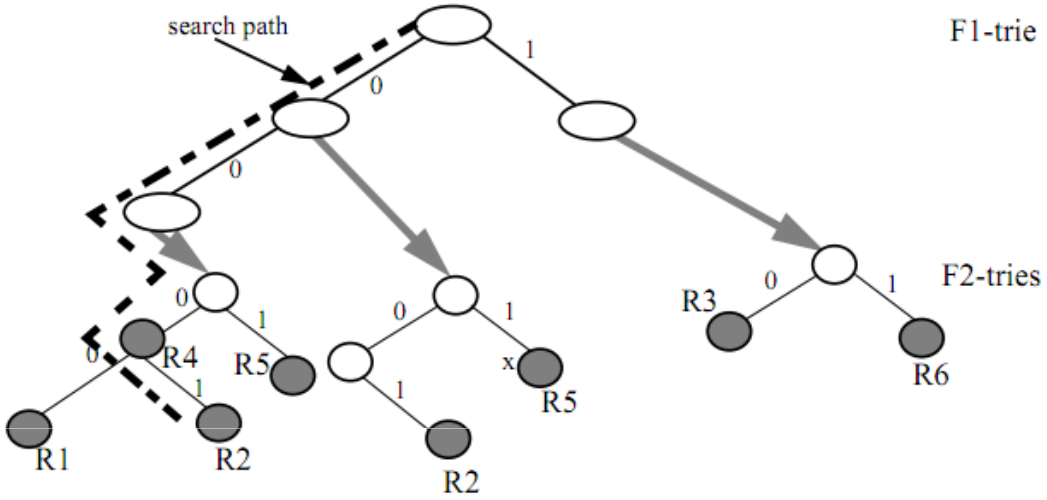




- Recursively build d-dimensional radix trie
  - Trie for first field, attach sub-tries to trie's leaves for sub-field, repeat
- For N-bit rules, d dimensions, W-bit wide dimensions:
  - Storage complexity: O(NdW)
  - Lookup complexity: O(W^d)

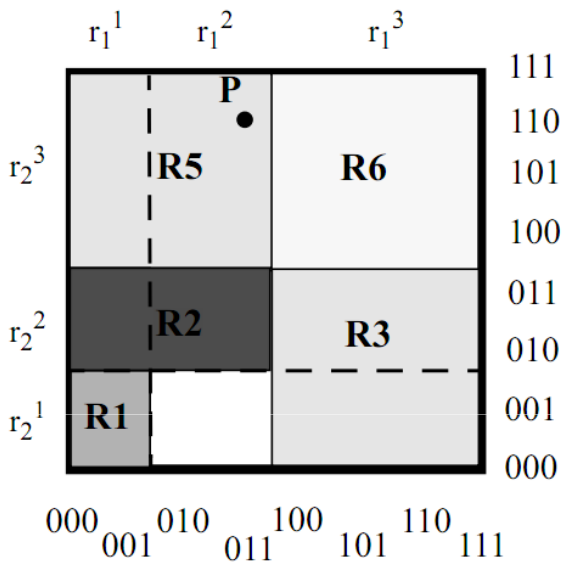
Approach #4: Set-pruning tries





- "Push" rules down the hierarchical trie
- Eliminates need for recursive lookups
- For N-bit rules, d dimensions, W-bit wide dimensions:
  - Storage complexity: O(dWN^d)
  - Lookup complexity: O(dW)

Approach #5: Crossproducting



Crossproduct Table

(r_1^1, r_2^1)	R1
(r_1^1, r_2^2)	R2
(r_1^1, r_2^3)	R5
(r_1^2, r_2^1)	—
(r_1^2, r_2^2)	R2
(r_1^2, r_2^2)	R5
(r_1^3, r_2^1)	R3
(r_1^3, r_2^2)	R3
(r_1^3, r_2^3)	R5

- Compute separate 1-dimensional range lookups for each dimension
- For N-bit rules, d dimensions, W-bit wide dimensions:
  - Storage complexity: O(N^d)
  - Lookup complexity: O(dW)

Other proposed schemes

Algorithm	Worst-case time complexity	Worst-case storage complexity
Linear Search	N	N
Hierarchical tries	W^d	NdW
Set-pruning tries	dW	$N^d dW$
Grid-of-tries	W^{d-1}	NdW
Crossproducting	dW	N^d
Bitmap-intersection	$(W + N/memwidth)\, d$	dN^2
Tuple space search	N	N
FIS-tree	$(l+1)\, W$	$l \times N^{1+1/l}$
Ternary CAM	1	N