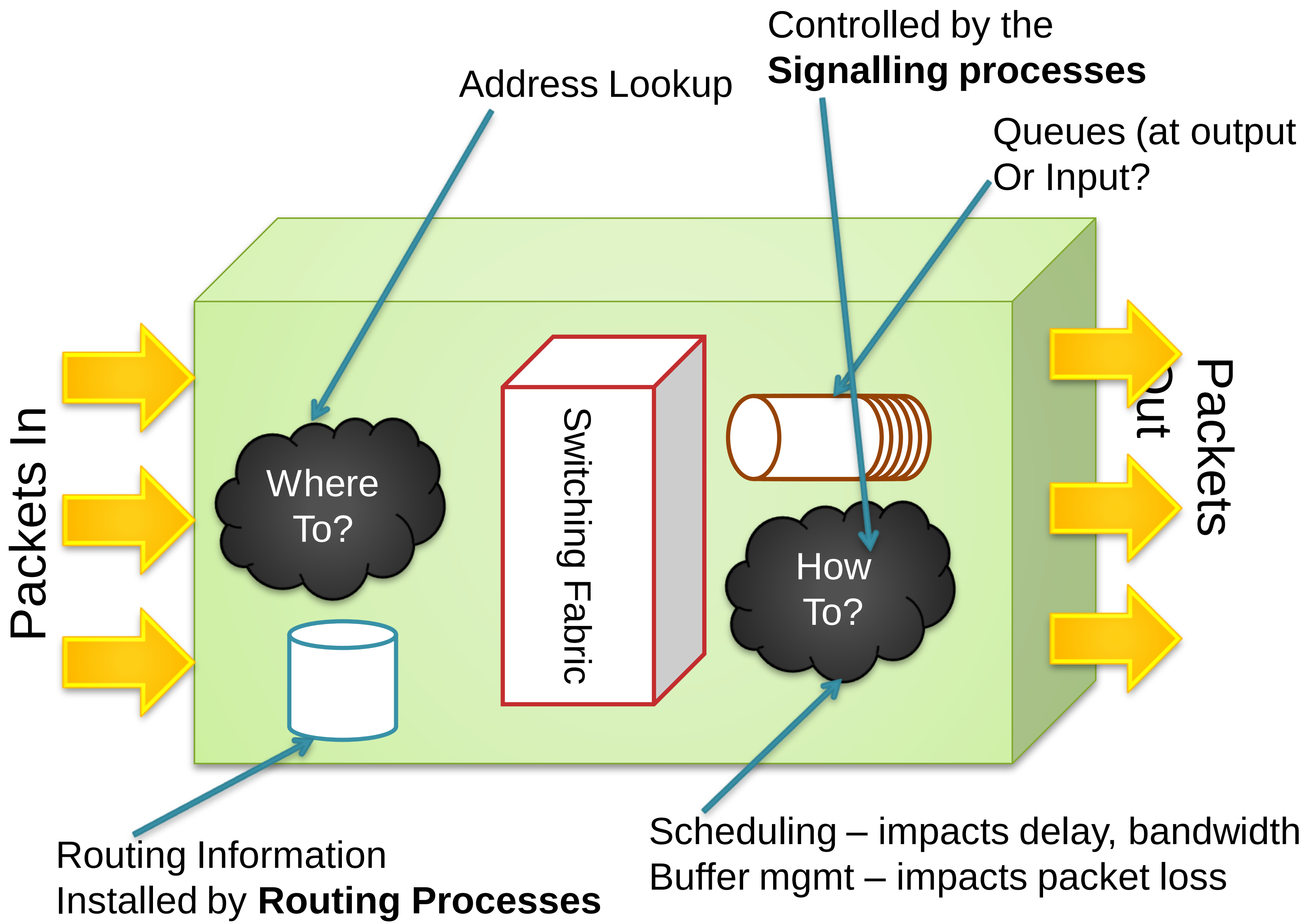
# Router Architectures

# Introduction



# Introduction…

- In the simplest scenario, a router handles two problems
  - Forwarding Problem
  - Routing Problem
- Forwarding Problem – for an incoming packet figure out the “outbound” interface
- Routing Problem – cooperate with other routers to figure out reachability to all possible network targets

# Introduction

- Forwarding Problem:
  - Route lookup is not a simple string match!
  - Longest prefix search – best match in among the routing table entries for the incoming packet
- Route lookup – several constraints
  - **Time:** find next-hop should be minimized
  - **Space:** route table should be represented compactly – so can fit in high speed memory

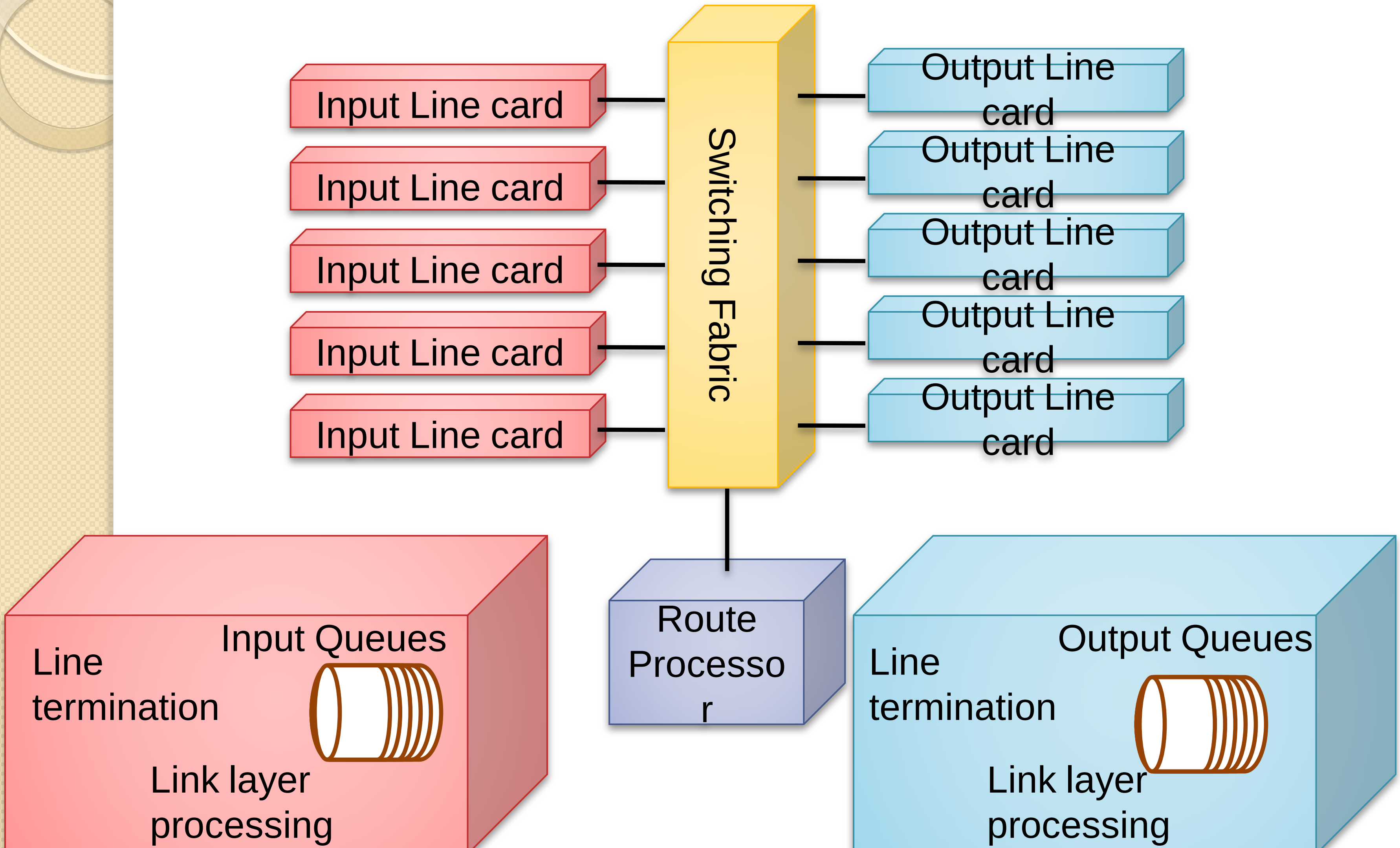
# Introduction

- Just solving the two problems – forwarding and routing is not sufficient to qualify as a modern router!
- Router is also a ***network resource allocator***!
  - Controls in a distributed manner the share of resource allocated to a traffic flow (e.g., web traffic can be a flow)

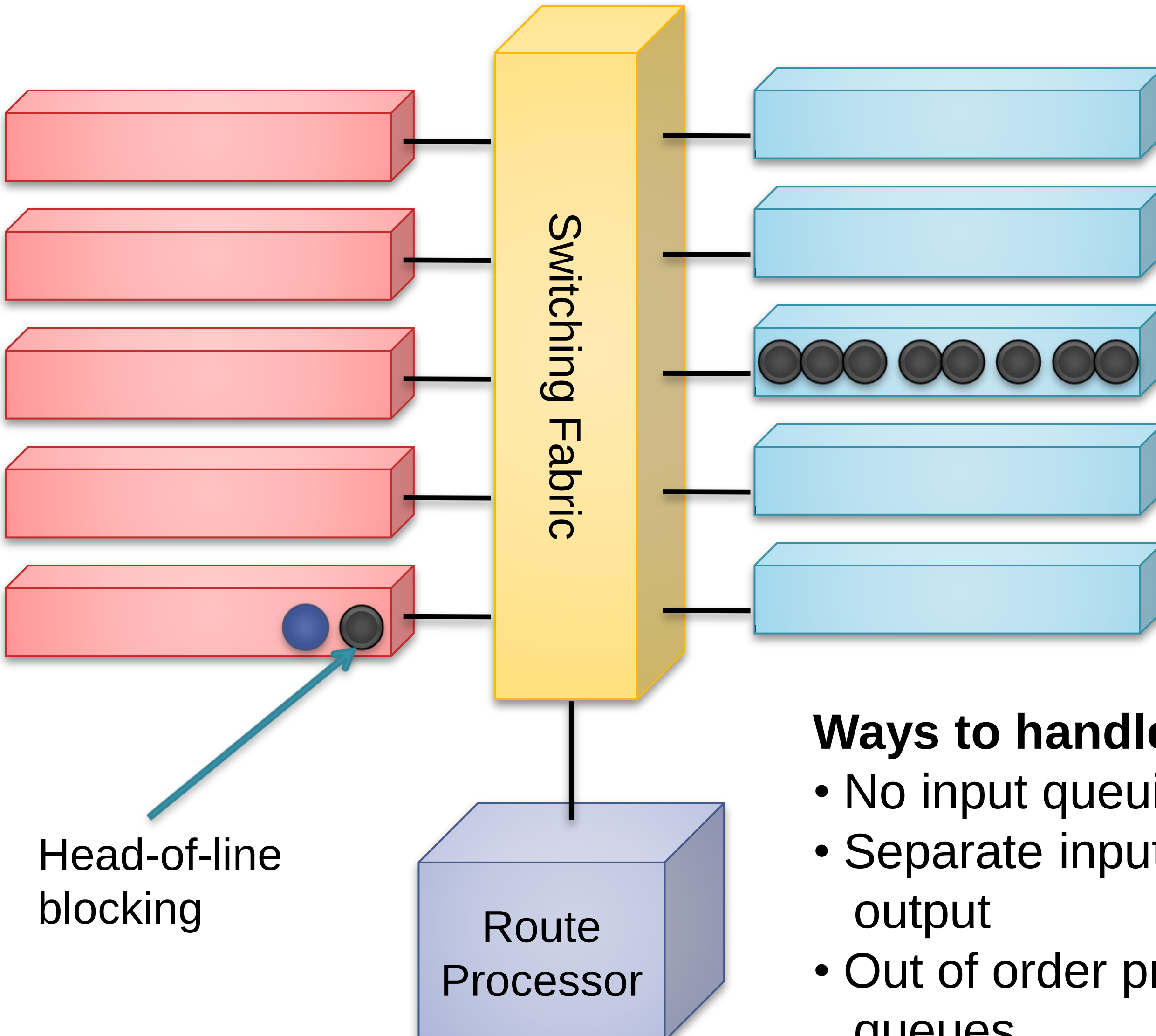
# Introduction…

- Recent advances have created multi-gigabit speed routers
  - Specialized hardware
  - Faster switching fabrics
  - Efficient and faster "lookup" algorithms

# Basic Router Architecture

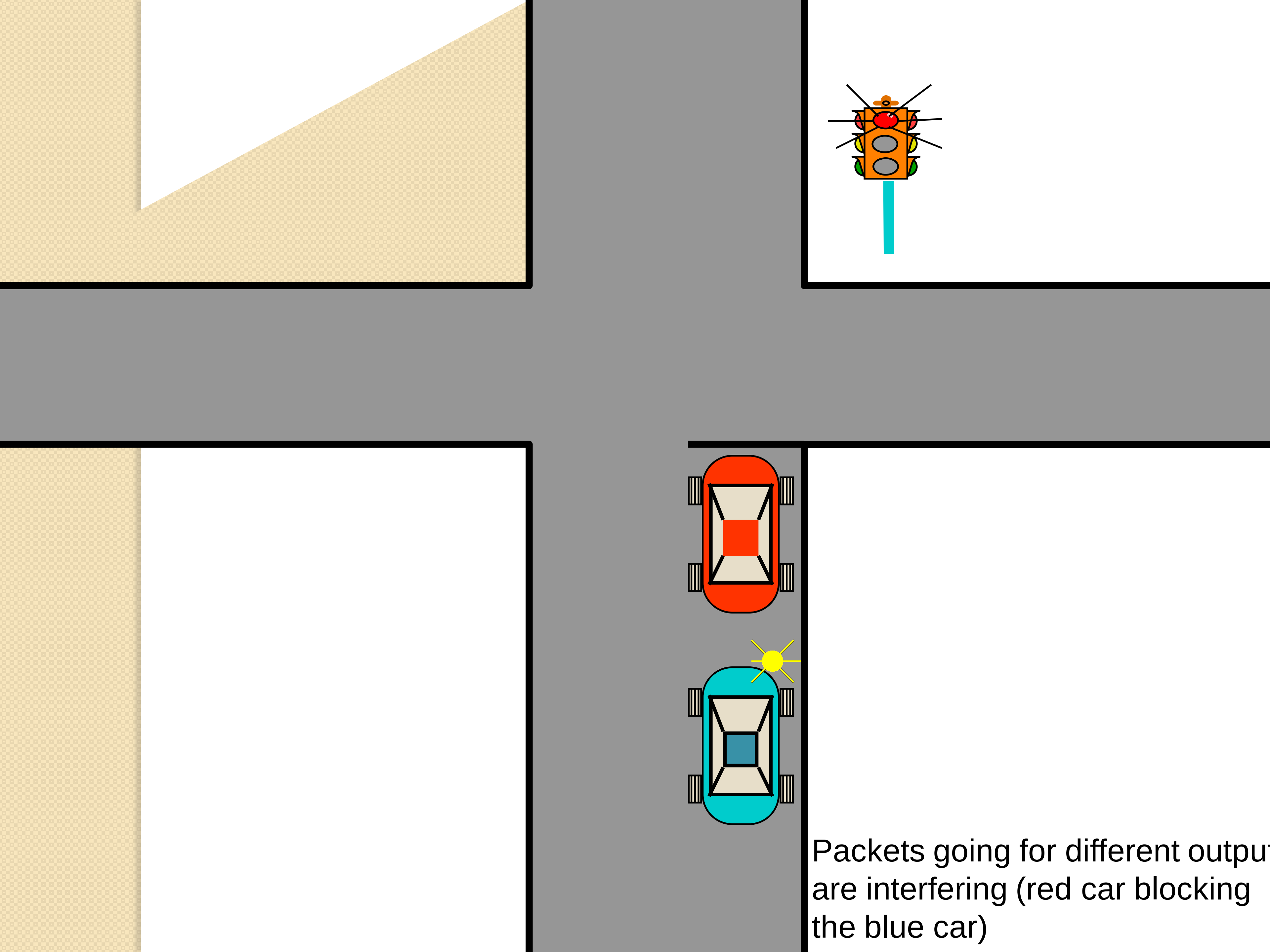


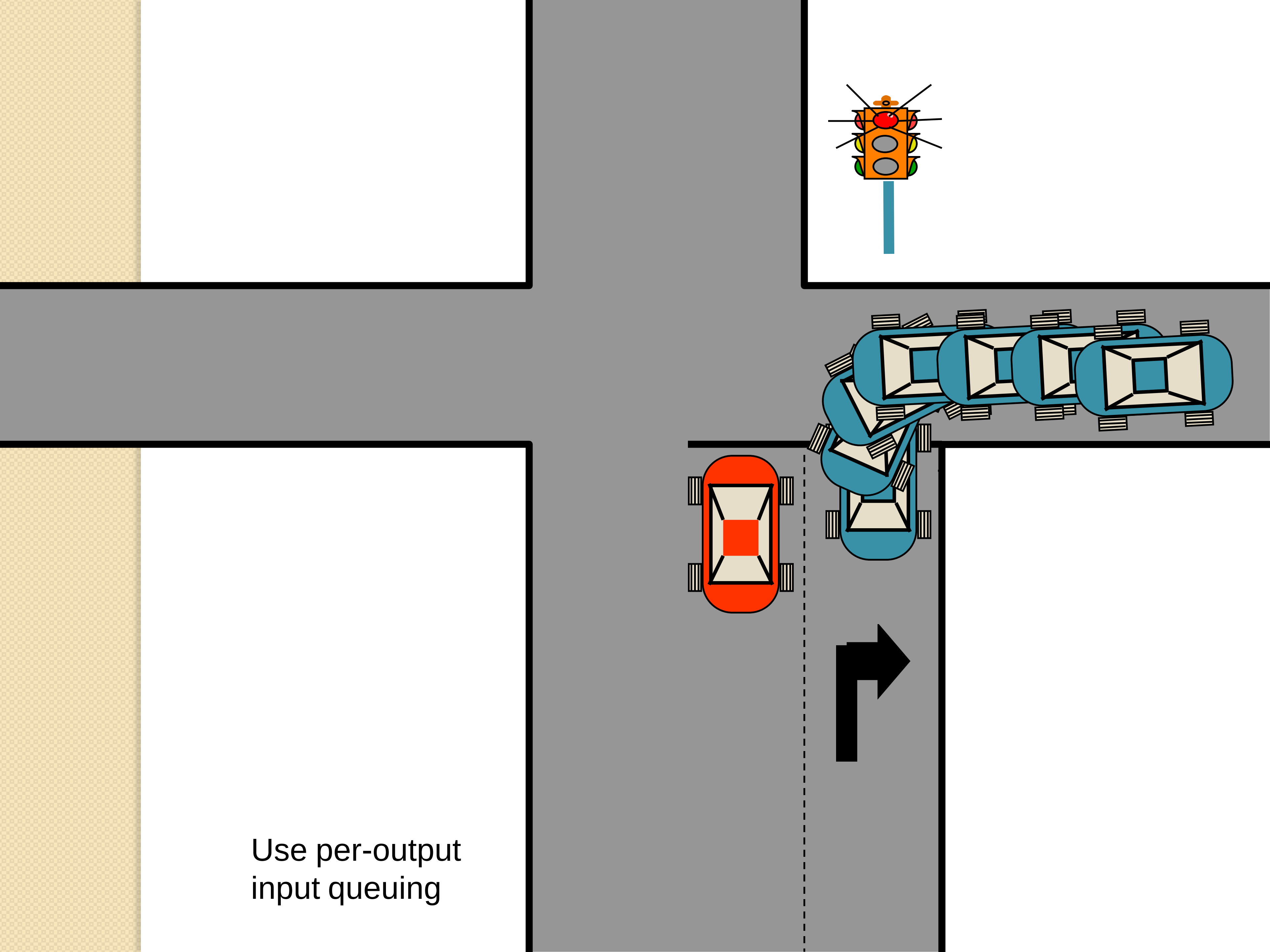
# Basic Router Architecture




**Ways to handle HOL blocking:**

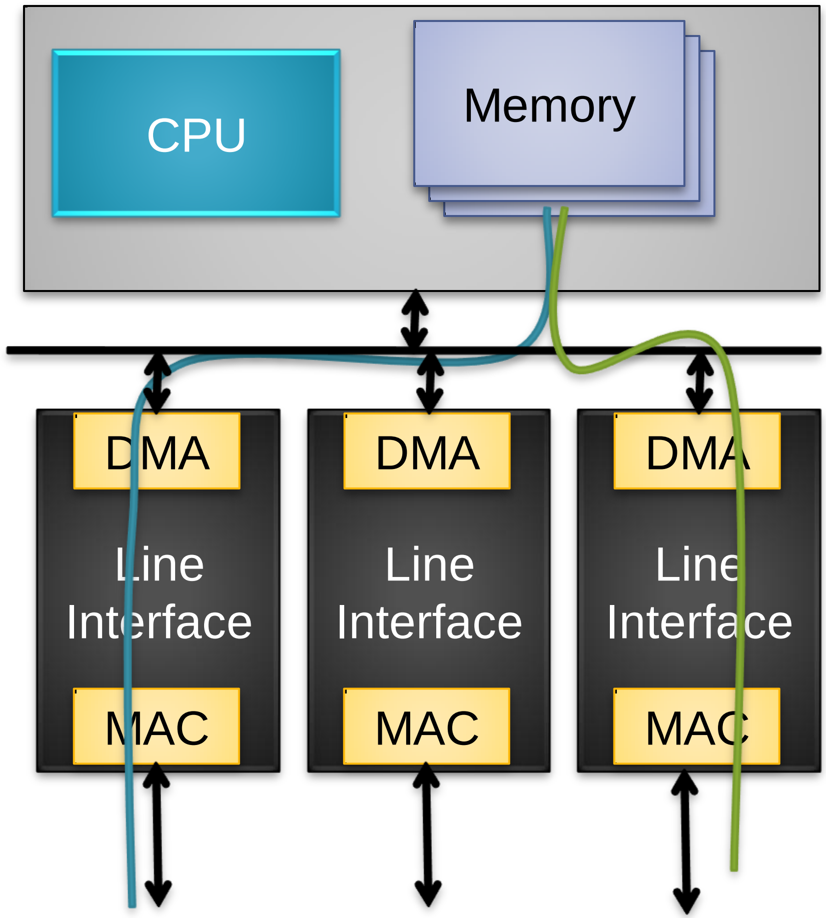
- No input queuing!
- Separate input queues for each output
- Out of order processing at input queues


Packets going for different outpu
are interfering (red car blocking
the blue car)


Use per-output
input queuing

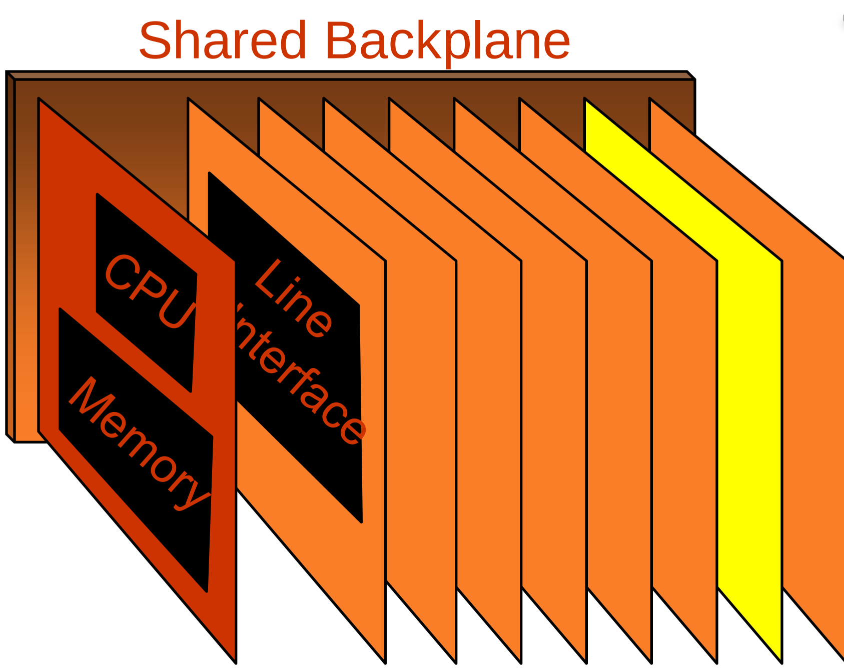
# More on Structure of IP Routers

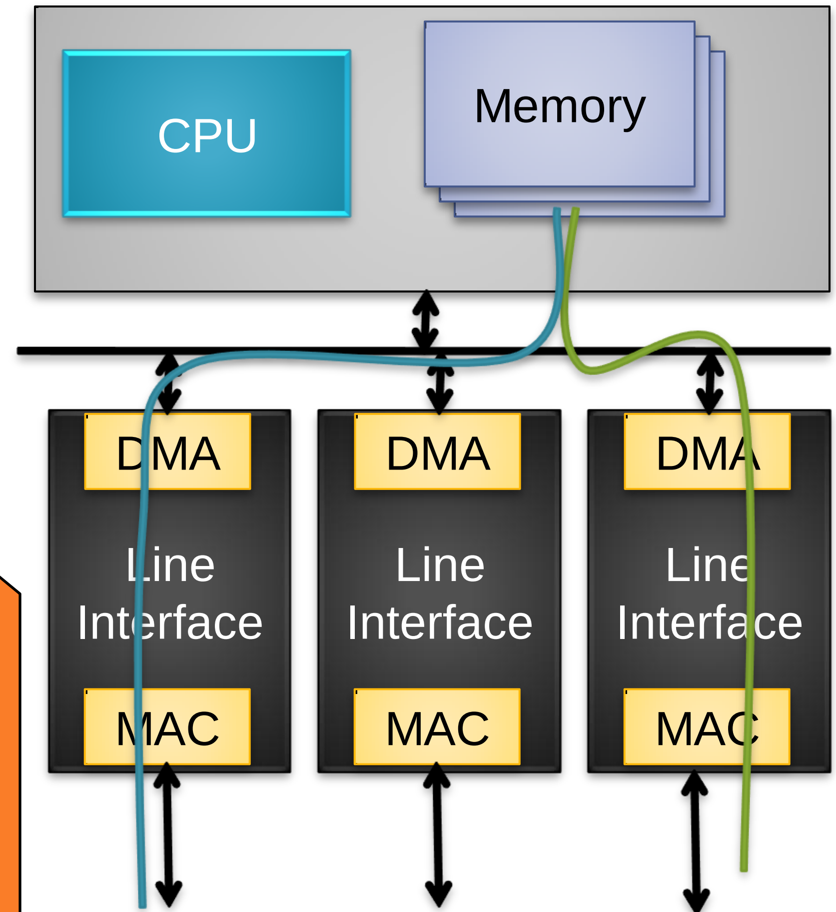
- Line Cards (Network Interfaces) – connects networks with different technologies
- Switching Backplane – shuffles the traffic from one interface to another – connects line cards & CPU
- Router Processor (CPU) – runs routing protocol, builds routing table, and overall management



# First Generation IP Routers

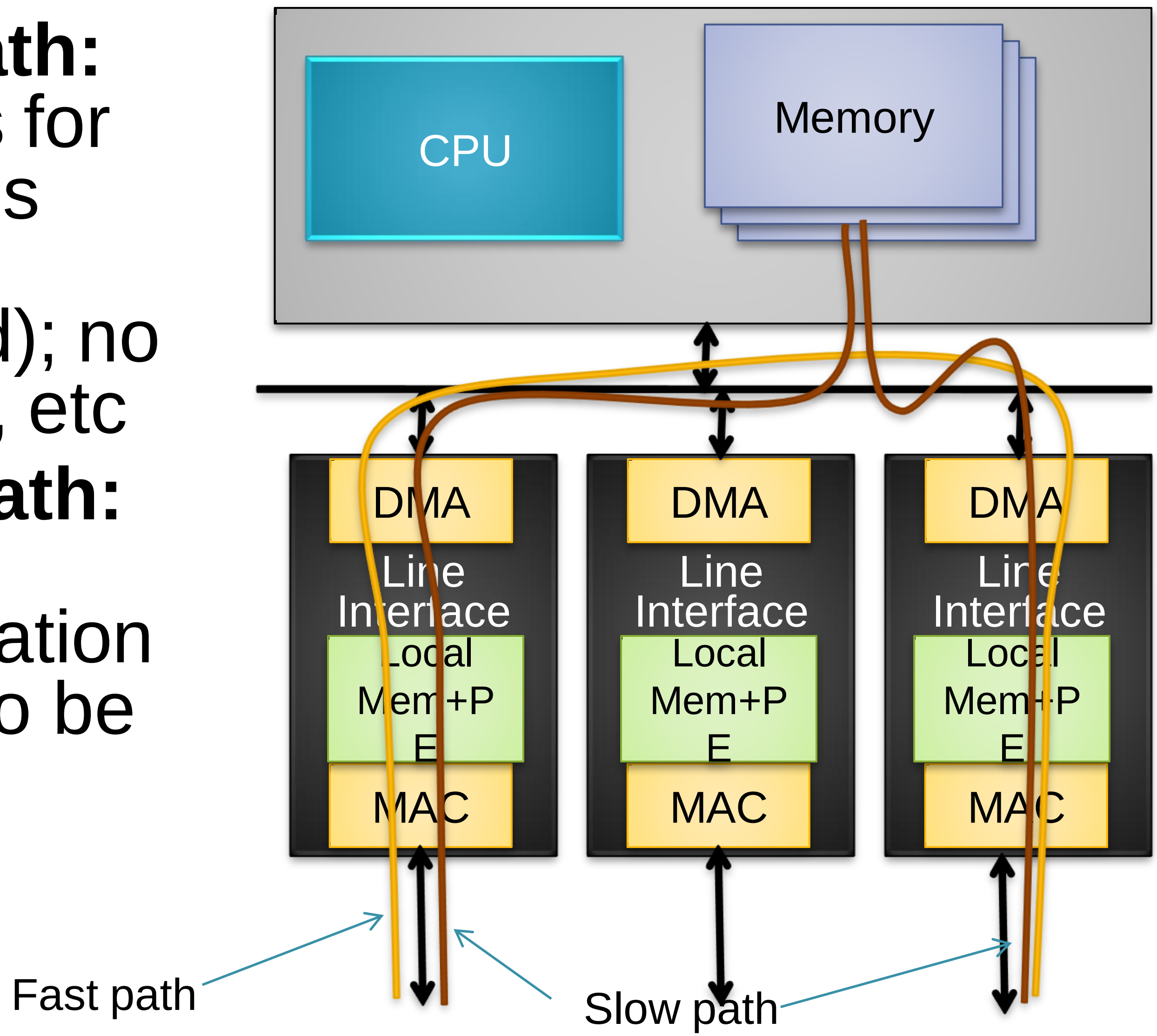
Very similar to what we get out of a Linux PC based router!



# Second Generation IP Routers

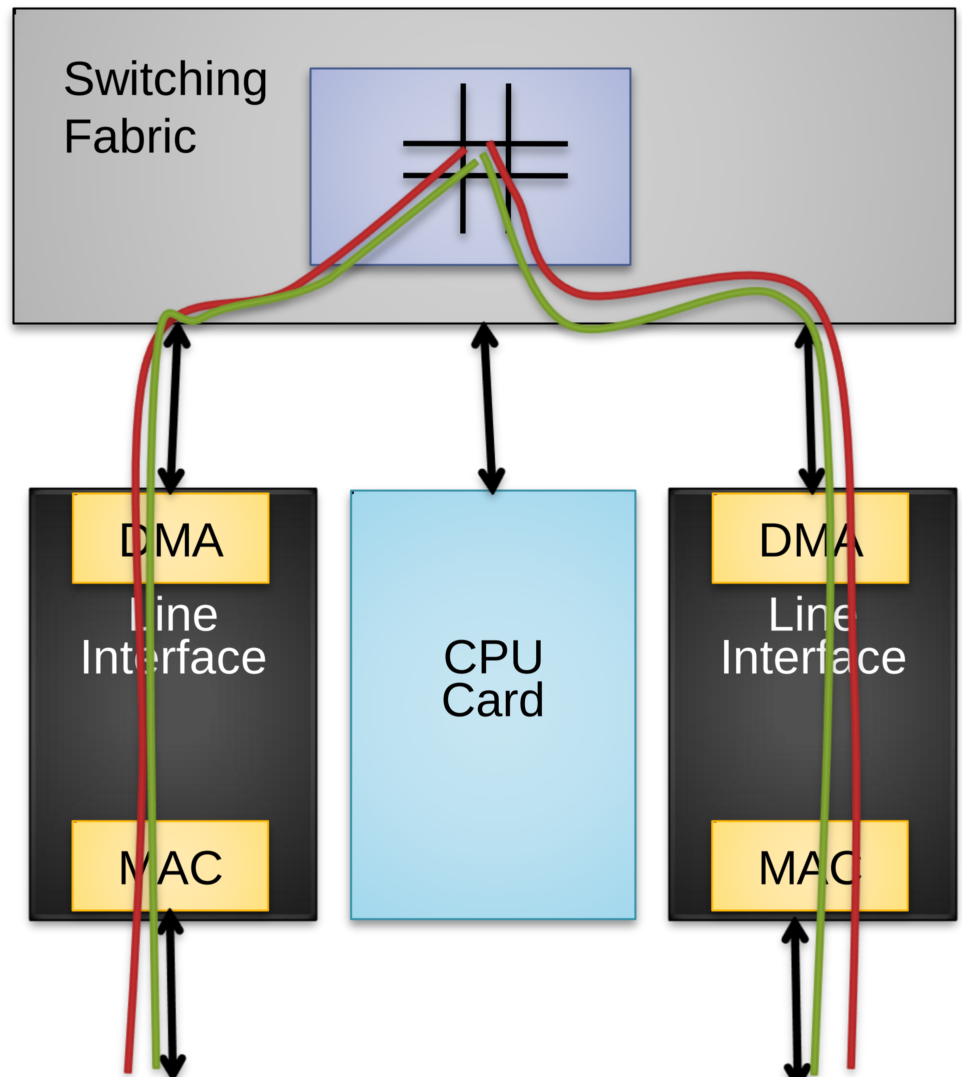
- **Fast path:** packets for lookup is known (cached); no options, etc
- **Slow path:** lookup computation needs to be done at CPU



# Third-Generation Switches/Routers

Switching Fabric provides the connectivity among the components

CPU/Router processor is only consulted for “slow” path traffic



# Basic Router Operations

- When a packet comes in:
  - validate the packet (link layer dependent processing)
  - queue at I/P side
  - lookup the next hop
  - switch across the connection fabric
  - queue at the O/P fabric
  - transmit
  - if there are functionality like QoS built into the router there may be added functionality and steps

# Next Hop Lookup

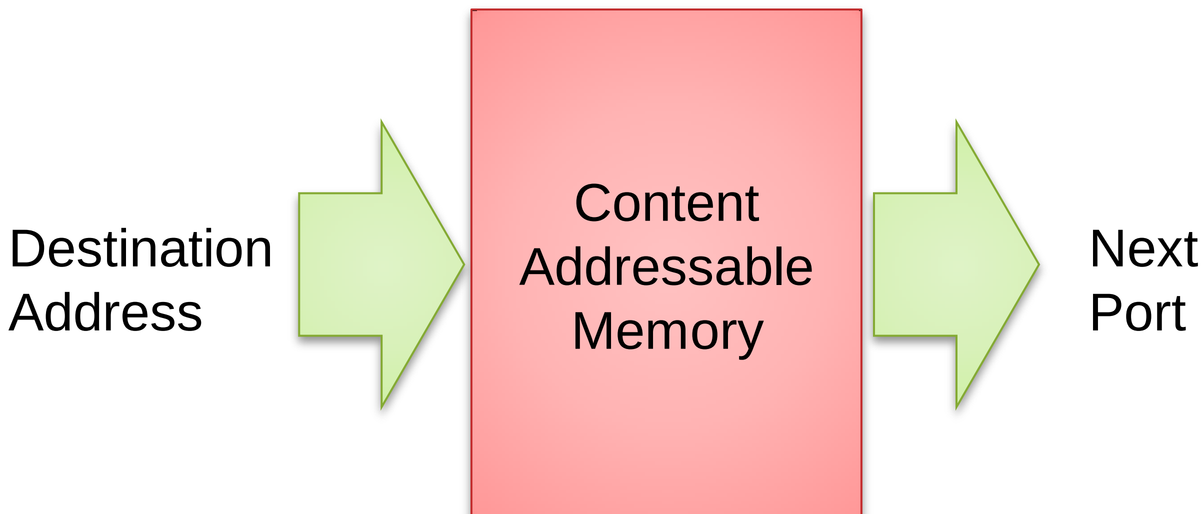
- Lookup basically does a "longest prefix match"
- It is necessary to design a very *fast* lookup algorithm
  - lookup database can be assumed to change very slowly – i.e., update frequency is much less compared to the read frequency

# Next Hop Lookup

- Approaches
  - CAMs
  - Trees, tries
  - hash tables
  - hybrid approaches

# Lookup Using CAMs

- CAMs allow retrieval of ***next_port*** (content) info. based on ***dest_addr*** (content)



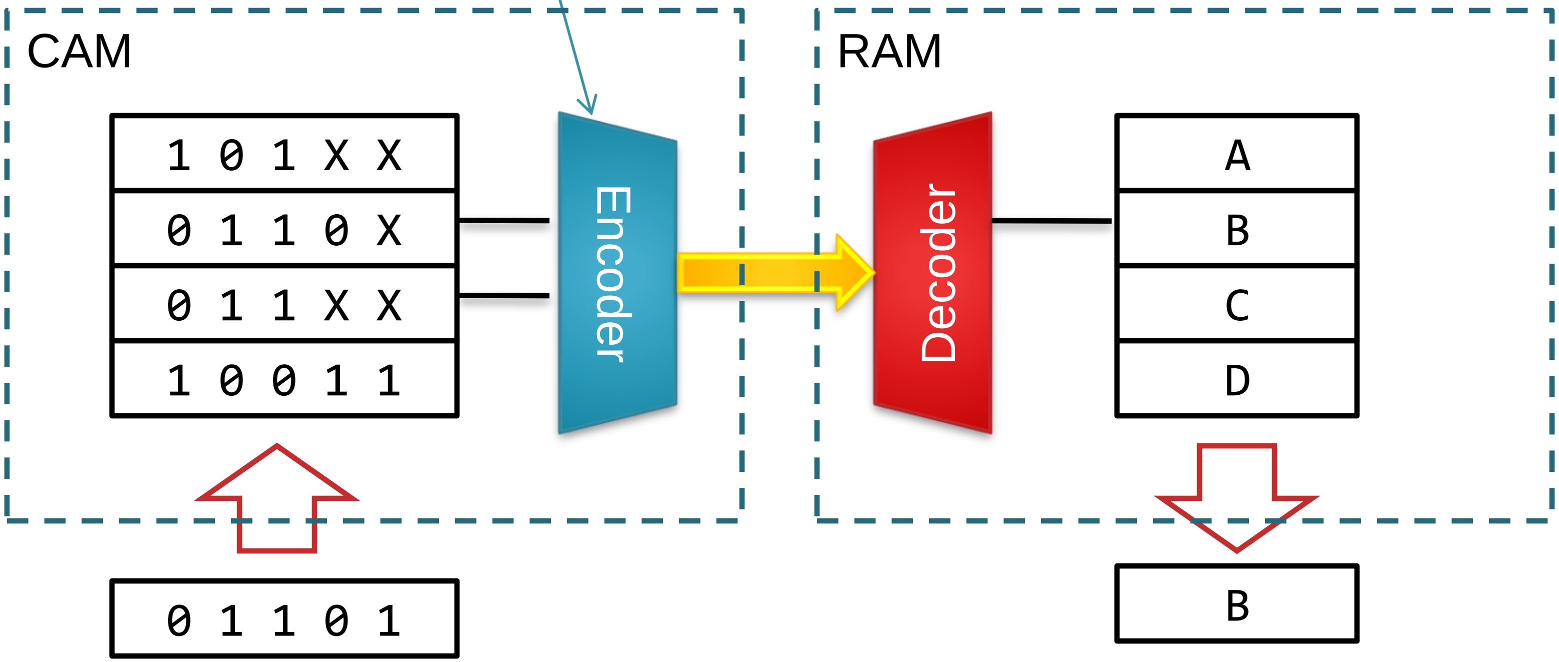
# Lookup Using CAMs

- Some limitations of CAMs
  - lower density than conventional (RAM) memory
  - slower access time
  - greater power consumption
  - best suited to exact matches, although can be used for longest prefix match

# CAM Example

| Entry | Address | Port Info. |
|---|---|---|
| 1 | 101XX | A |
| 2 | 0110X | B |
| 3 | 011XX | C |
| 4 | 10011 | D |

Priority encoder – selects 0110X over 011XX (provides longest prefix match)



# PATRICIA Trie

- PATRICIA (Practical Algorithms To Retrieve Information Coded in Alphanumeric) invented in 1968 by D. L. Morrison
- Tries commonly used data structures to organize routing table entries
- Search is based on binary representation of addresses
- Takes routing table entries and forms a radix tree structure consisting of address prefixes and positions in the address to test

# PATRICIA Trie

- Longest-match lookup consists of walking the trie until the end is reached or a match is discovered
- Patricia tries were used in previous generation routers
  - limited amount of memory is needed to store routing table (could fit into fast caches)
- Can result in large number of memory accesses -- particularly when backtracking takes place

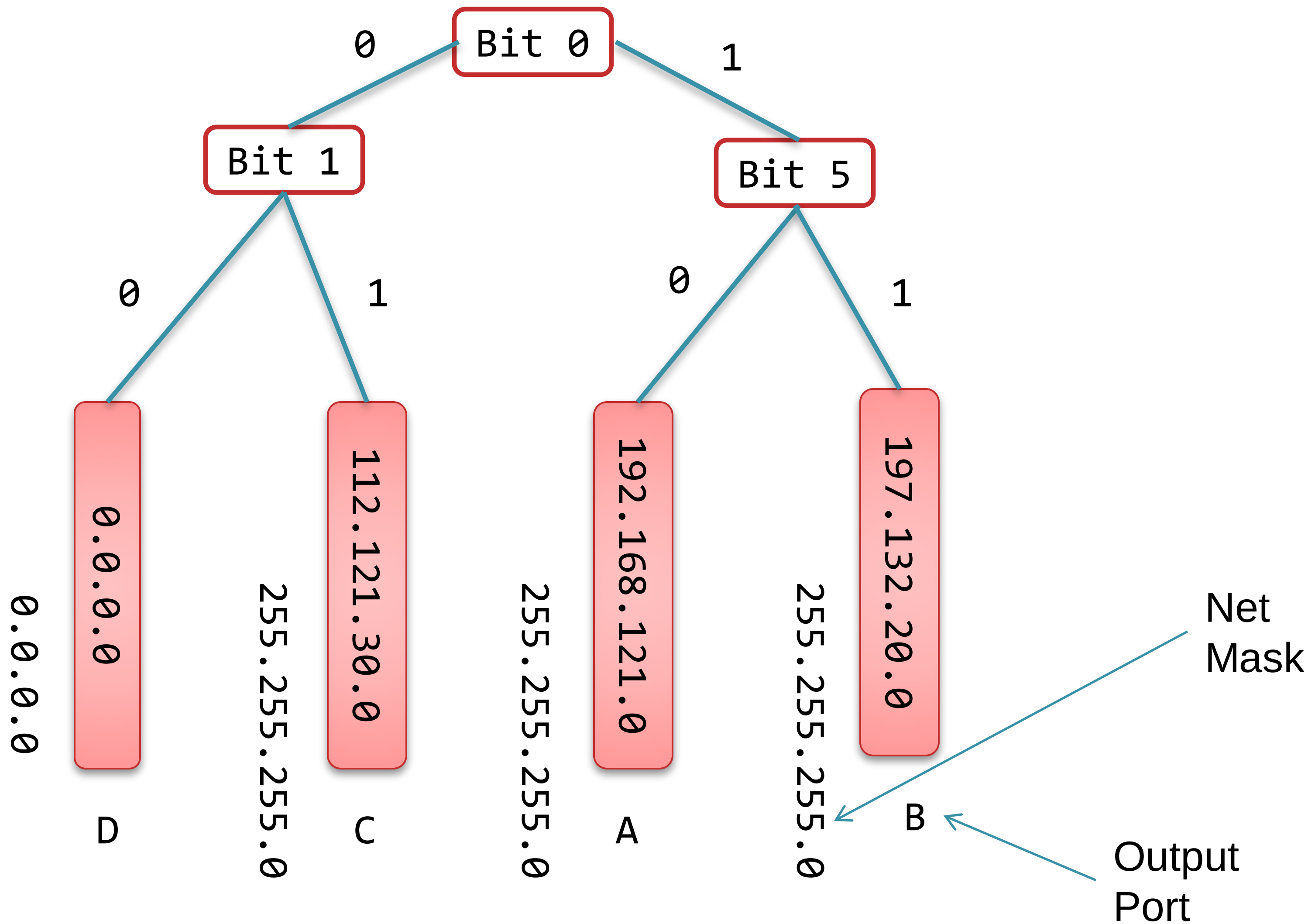
# Route Lookup Problem Again

Match the incoming destination to the "best" entry in the table below. If none match, take the default entry!

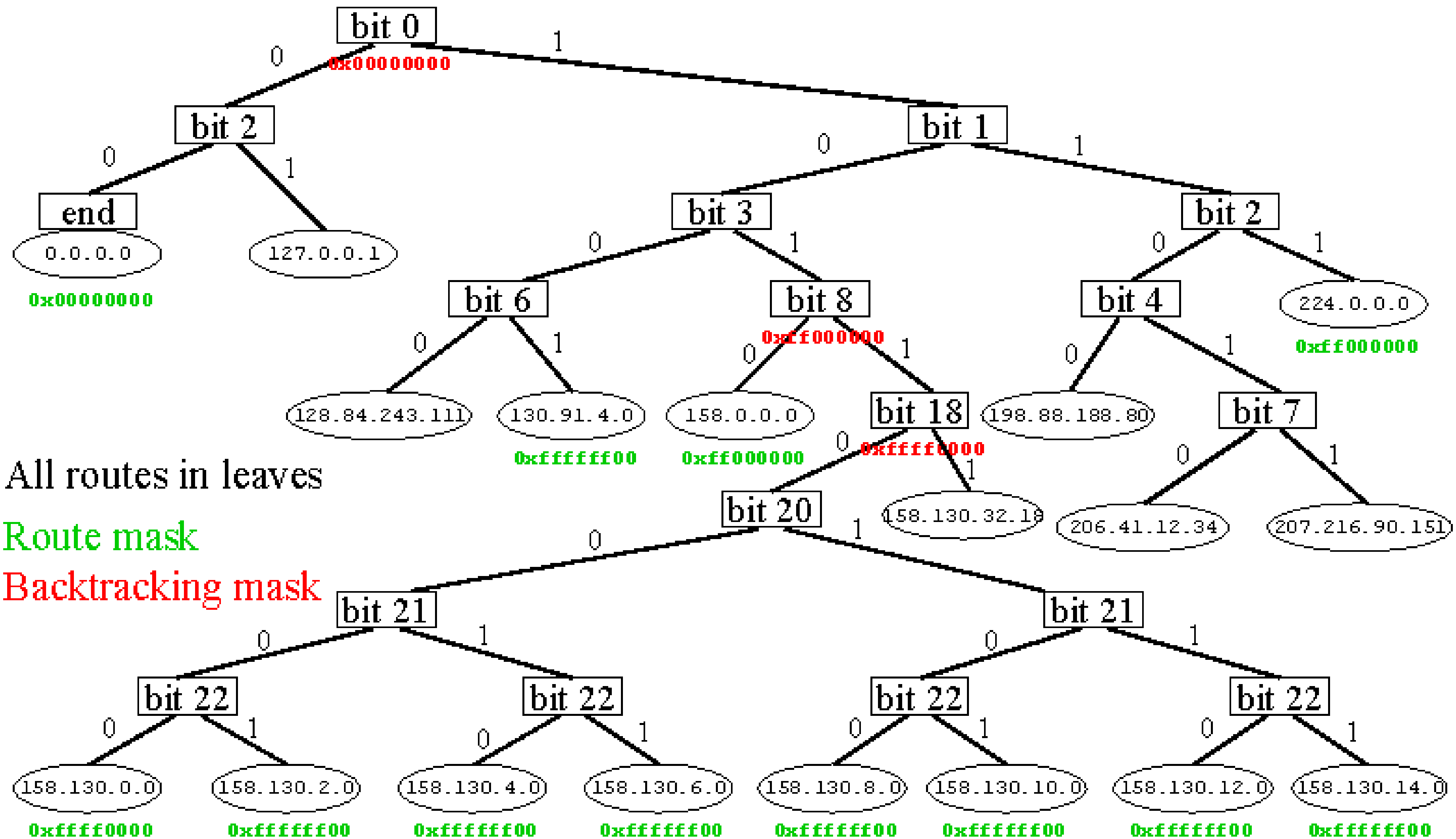
| Network | Mask | Output Port |
| --- | --- | --- |
| 192.168.121.0 | 255.255.255.0 | A |
| 197.132.20.0 | 255.255.255.0 | B |
| 112.121.30.0 | 255.255.255.0 | C |
| 0.0.0.0 | 0.0.0.0 | D |

Default route

# Trie Representation



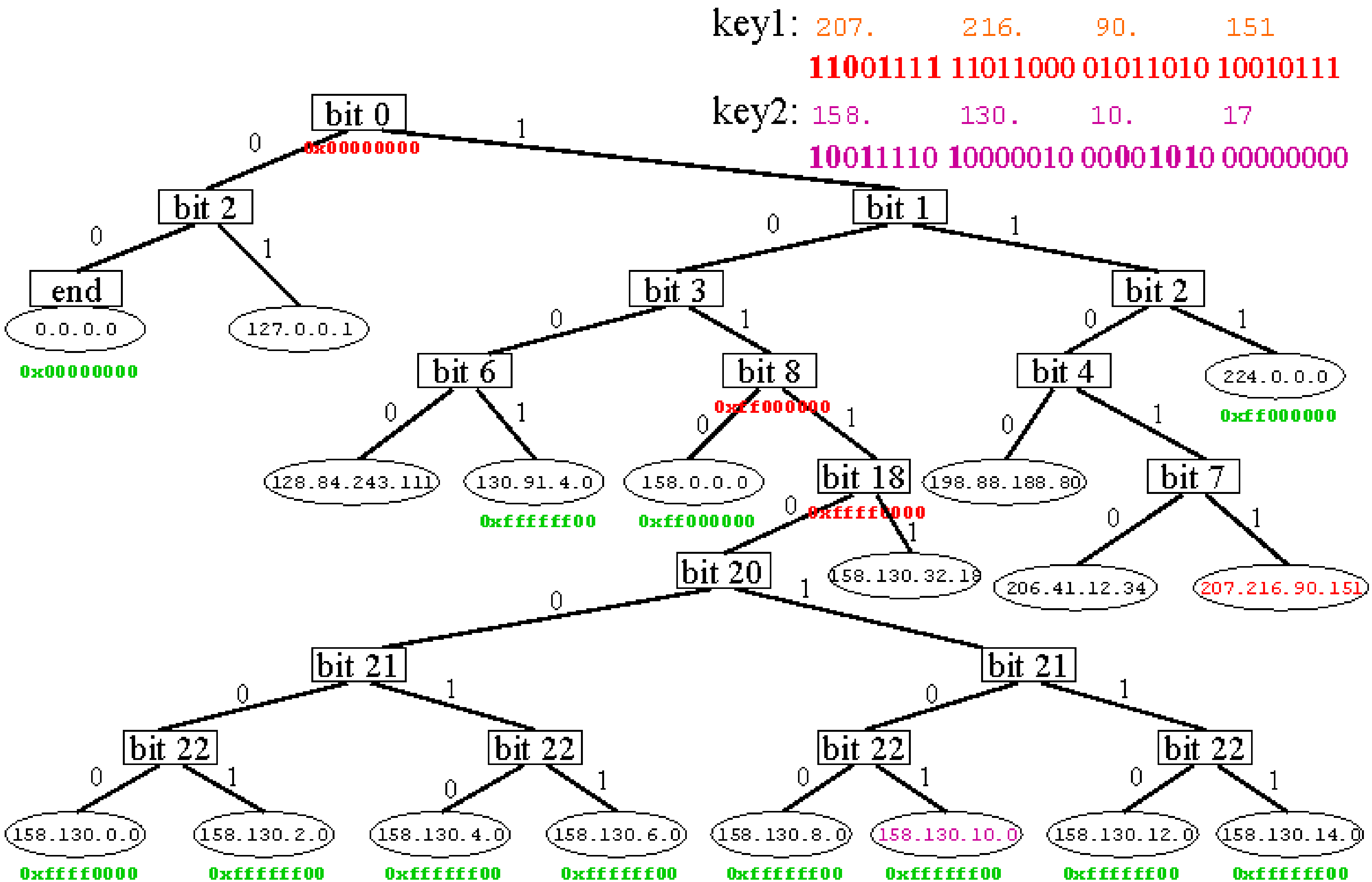
# PATRICIA Trie Representation



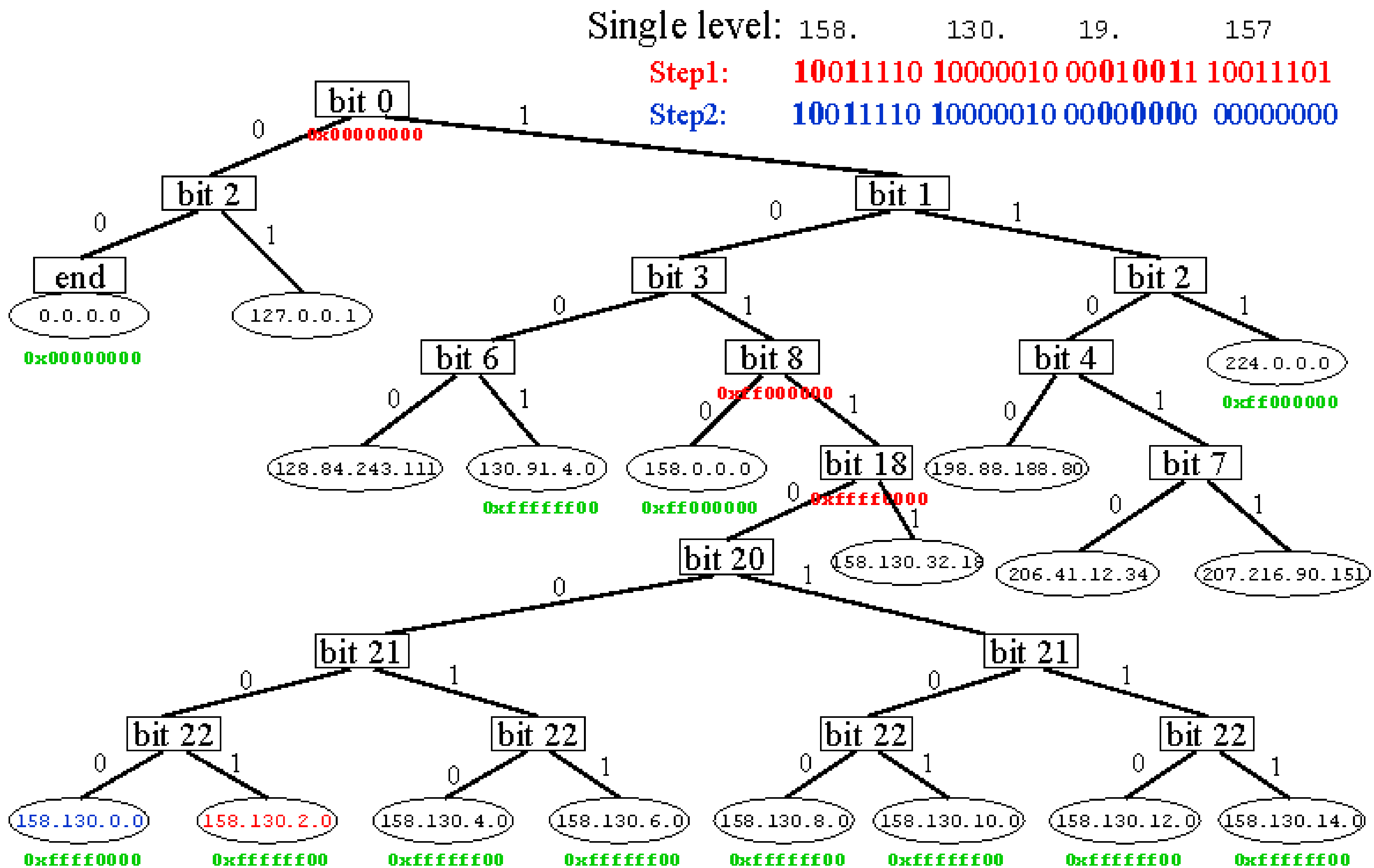
# Lookup Situations with PATRICIA Trie

- Direct (one pass) match
  - Host route
  - Network address
- Match after backtracking
  - Single pass
    - Host route
    - Network address
  - Multiple passes
    - Host route
    - Network address
- Default address

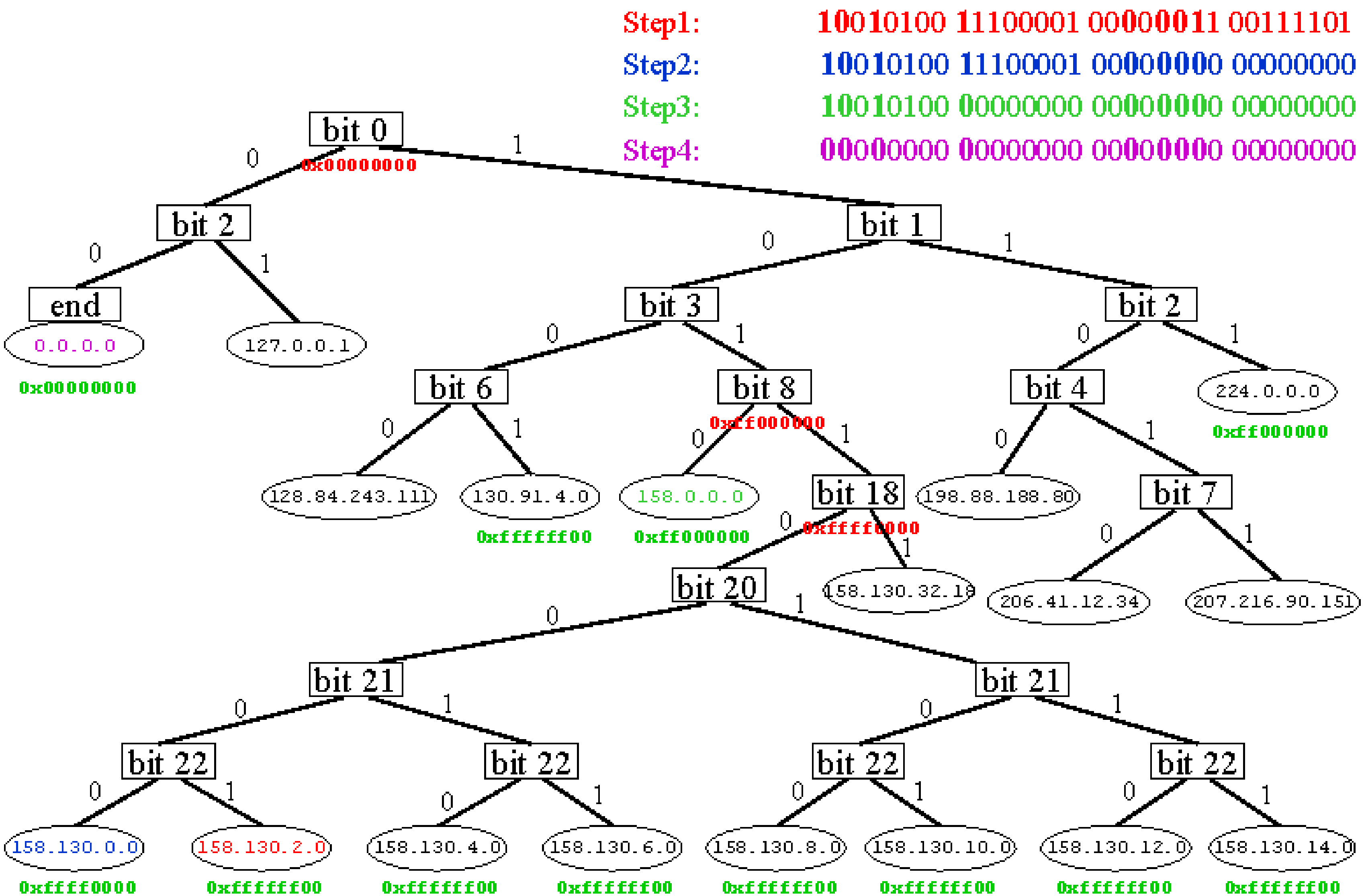
# Direct Match to Host & Net. Addr.



# Match Network Address after backtrack

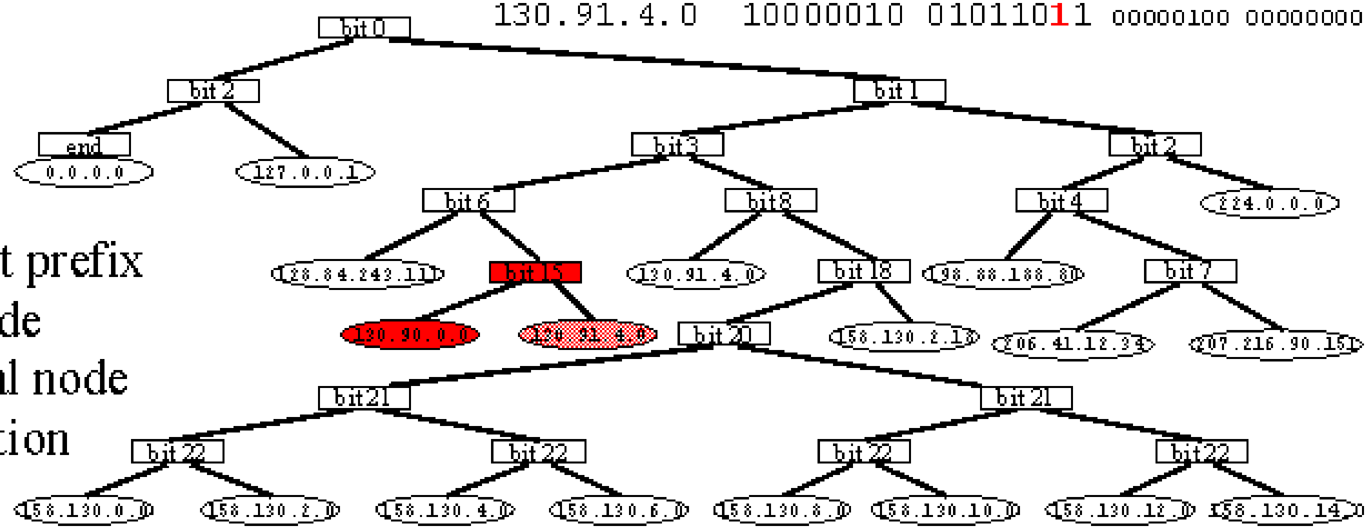


# Match Network Address after backtracking

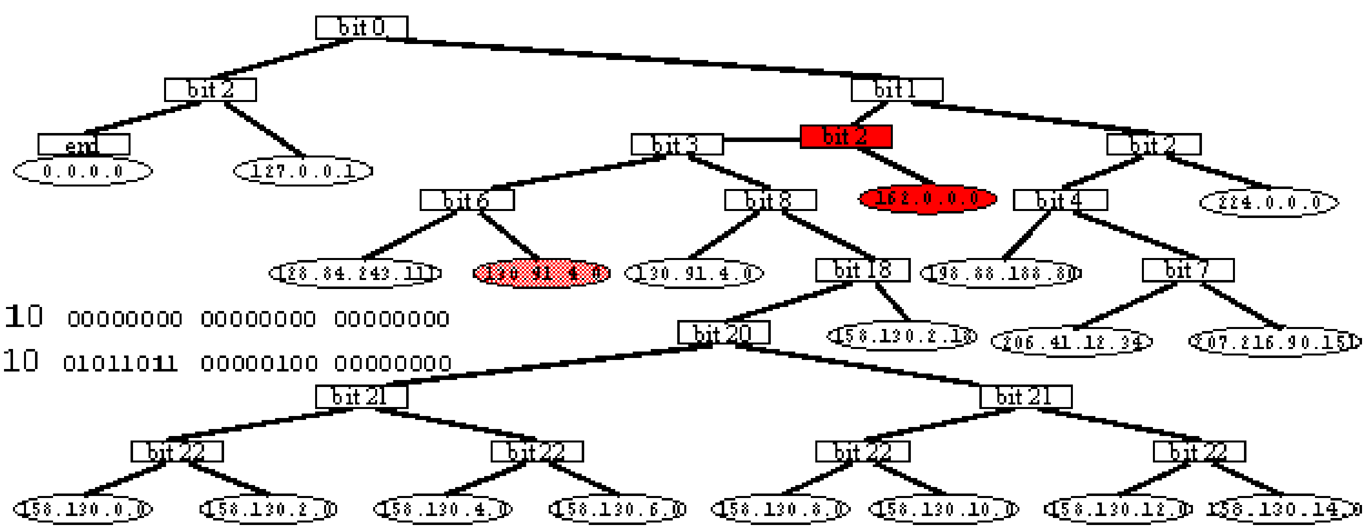


# Insertion in PATRICIA Trie

```
130.90.0.0   10000010 01011010 00000000 00000000
130.91.4.0   10000010 01011011 00000100 00000000
```



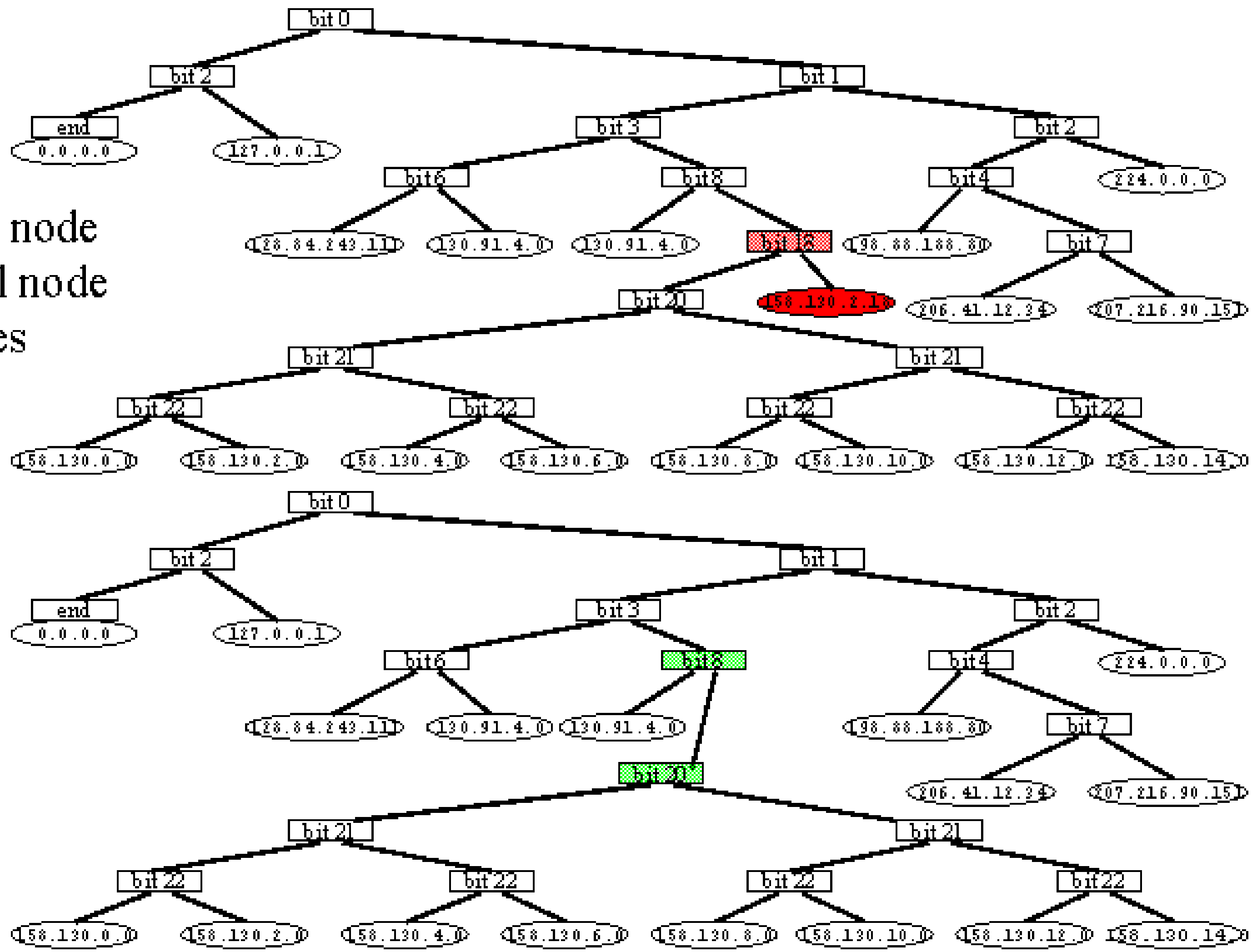

- Go down to leaf
- Determine longest prefix match with leaf node
- Insert new internal node at appropriate location and attach external node




```
162.0.0.0    10100010 00000000 00000000 00000000
130.91.4.0   10000010 01011011 00000100 00000000
```

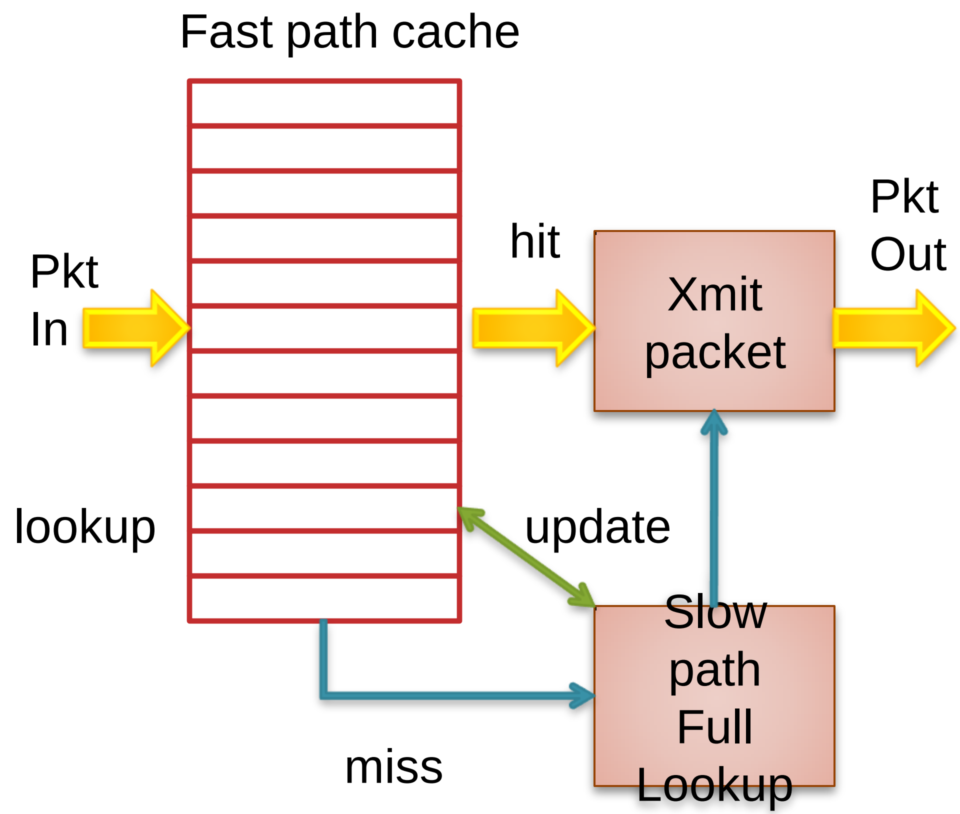
# Deletion in PATRICIA Trie

- Identify external node & parent internal node
- Delete both nodes
- Mend tree

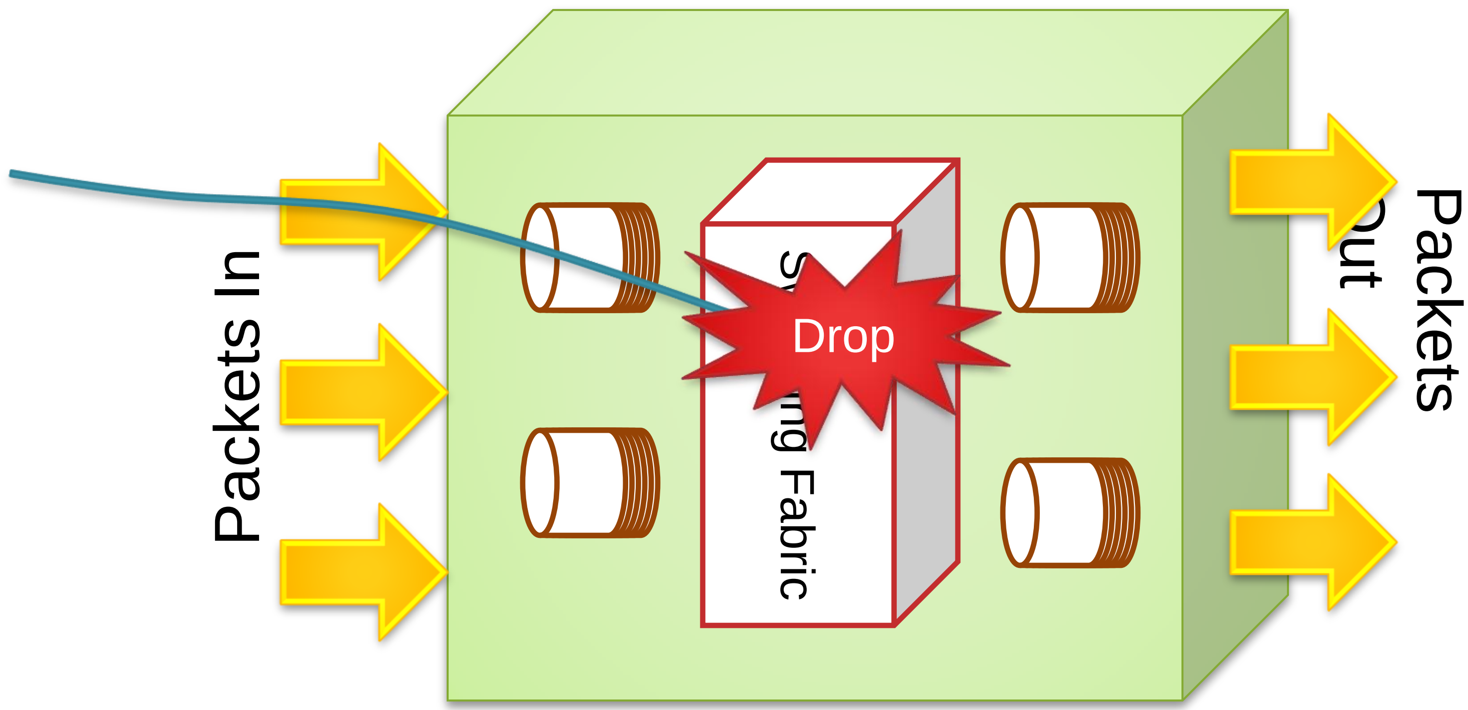


# Route Lookup Optimization

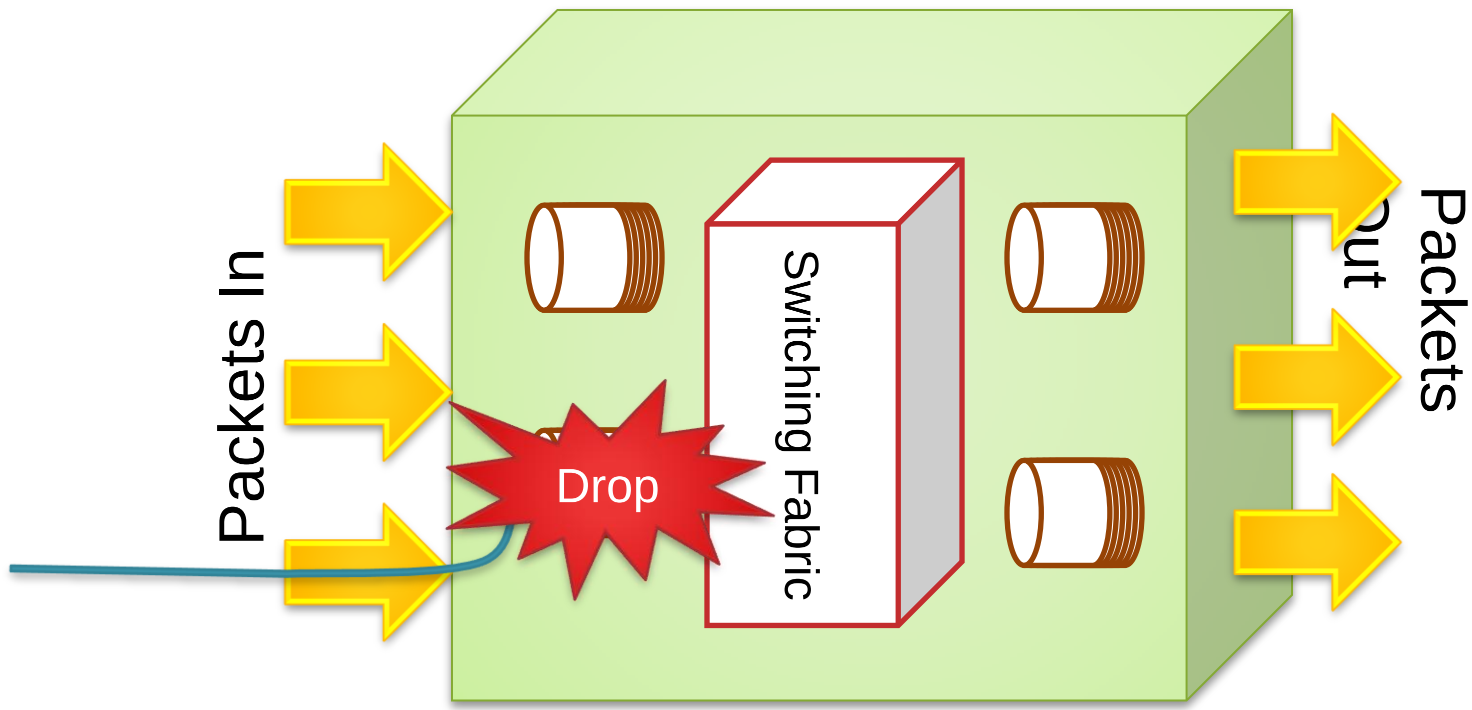
- Trie lookups costs ➔ cache active lookups
- "Slow path" invoked when cache miss
- Slow path updates fast path cache



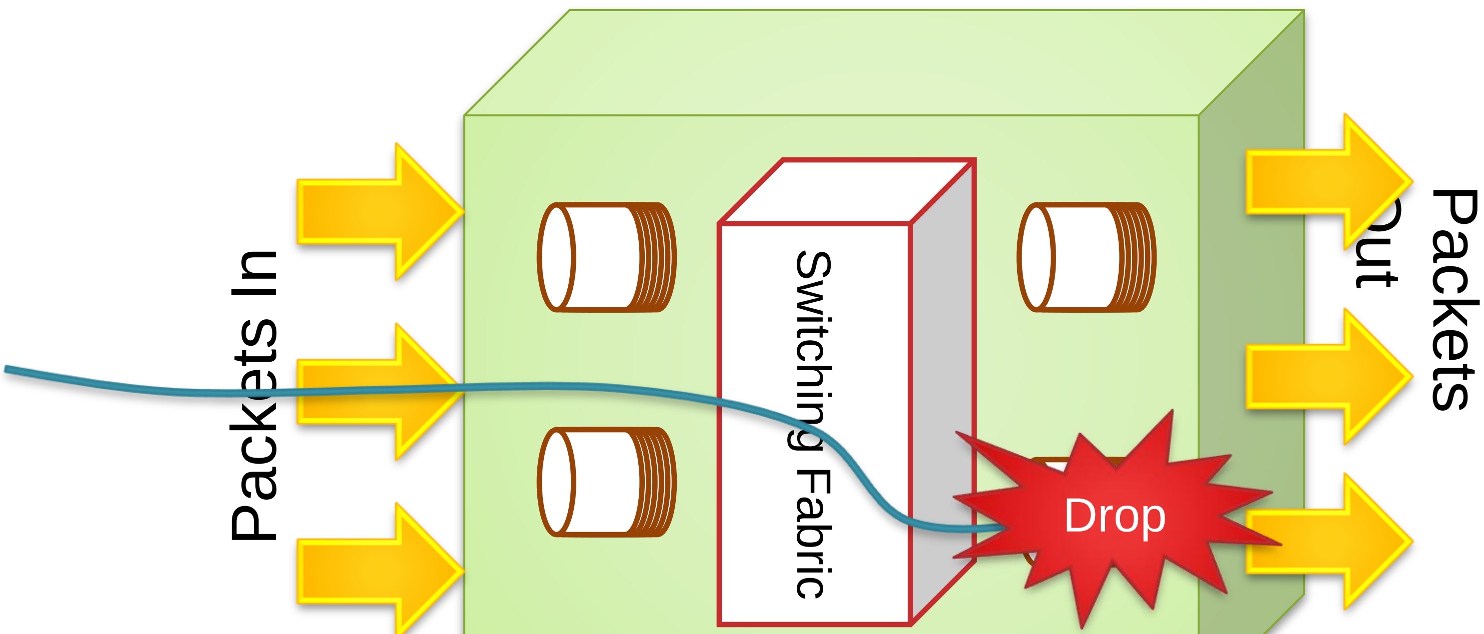
# Blocking Problems in Routers



# Blocking Problems in Routers



# Blocking Problems in Routers



# Blocking Problems…

- Input queue blocking.. major cause ***head-of-line (HOL) blocking***
- Contention within the switching fabric – aka - blocking
- Output queue blocking.. backlog from successive routers

# Dealing with blocking

- Overprovisioning
  - internal links much faster than inputs
- Buffers
  - at input or output
- Backpressure
  - if switch fabric doesn’t have buffers, prevent packet from entering until path is available
- Parallel switch fabrics
  - increases effective switching capacity

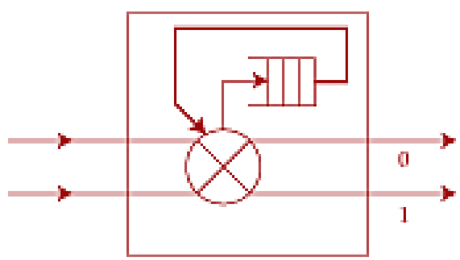
# Blocking in Switch Fabrics

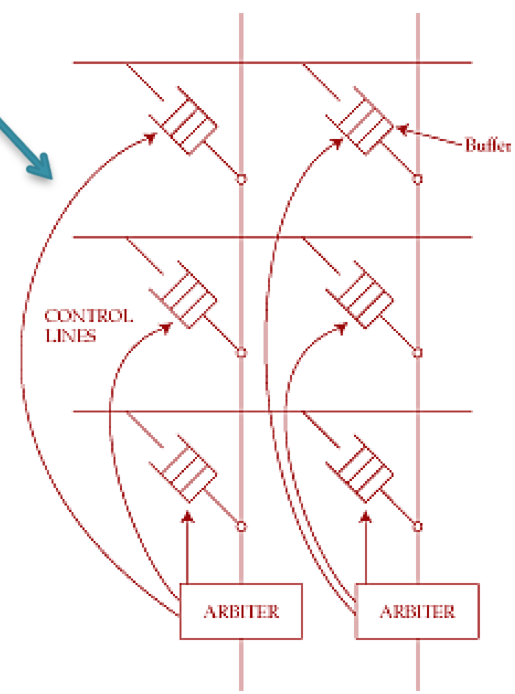
- Can have both internal (within fabric) and output blocking
- Internal - no path to output
- Output - trunk unavailable
- Unlike a circuit switch, cannot predict if packets will block (why?)
- If packet is blocked, must either buffer or drop it

# Buffering within Switch Fabric

- Buffering can be distributed within the switching fabric to deal with contention

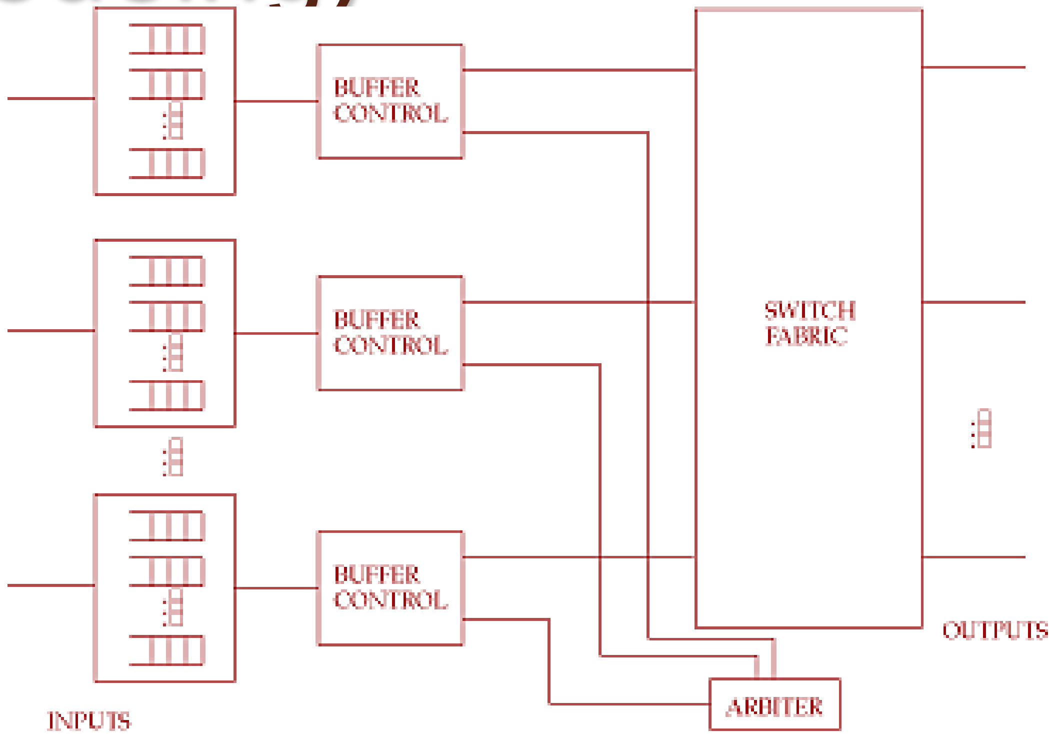
Buffered Cross-bar
(distribution of buffers)



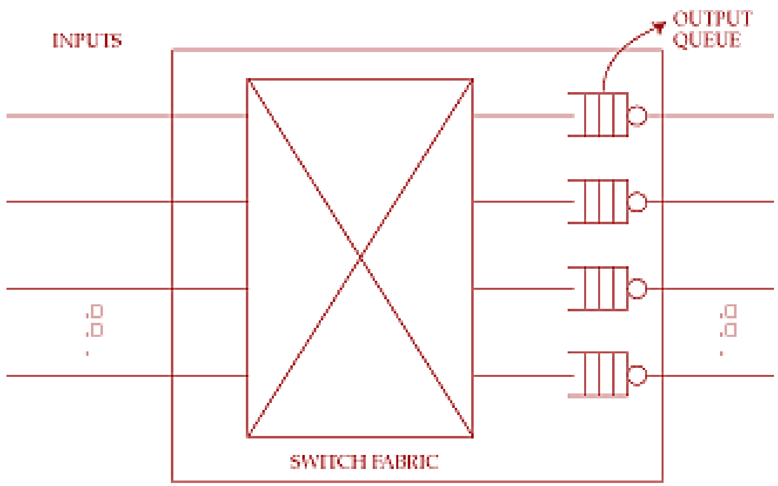

Buffering within a single
element of the Switch Fabric

# Input buffering (input queueing)




- No speedup in buffers or trunks (unlike output queued switch)
- Needs arbiter

# Output queueing




- Don’t suffer from head-of-line blocking
- But output buffers need to run much faster than trunk speed (why?)