

## Router Architectures

### Introduction

- ▶ To figure out what outbound interface the router should use
  - ❖ network address of destination with forwarding table (“routing table”)
- ▶ Routing table may be built **manually** or maintained **dynamically** by a routing protocol
- ▶ **Routing table** is used by a router to exchange information with other routers about network topology and reachability

## Introduction

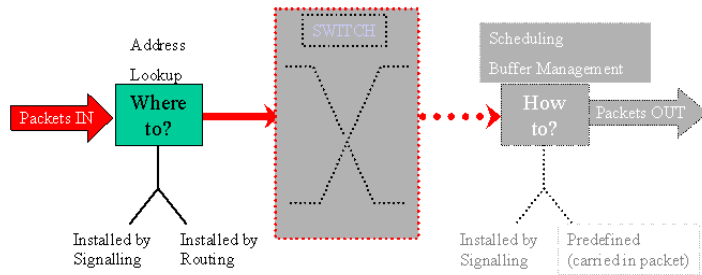
- ▶ Recent advances in routing architecture including
  - ❖ specialized hardware
  - ❖ switching fabrics
  - ❖ efficient and faster “lookup” algorithms
  - ❖ have created routers that are capable of routing at multi-gigabit speeds

### Introduction

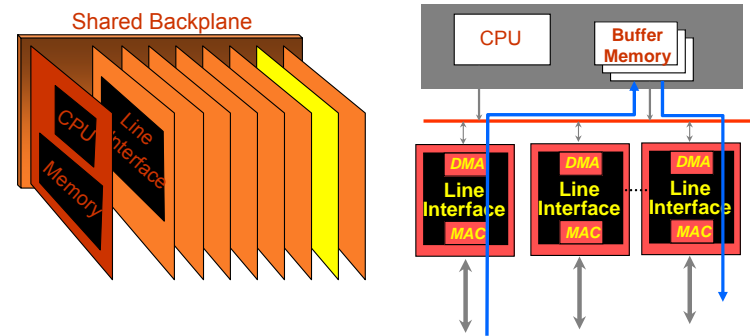
- ▶ Basic architecture of the router consists:
  - ❖ **line cards** -- enables router to attach to different networks that employ different datalink technologies
  - ❖ **router processor or CPU** -- runs the routing protocols -- builds and maintains the routing table and provides network management
  - ❖ **a backplane** -- serves as the interconnect between the input line cards, the CPU, and the output line cards

## Introduction

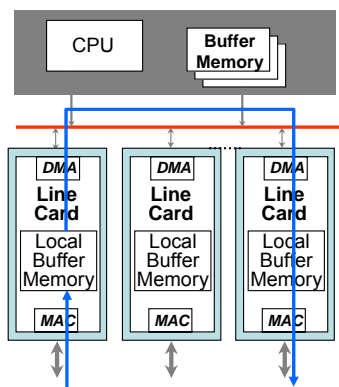
- ▶ Switching involves “lookup” -- retrieval of forwarding info. using packet header
- ▶ Should be able to keep up with incoming packets



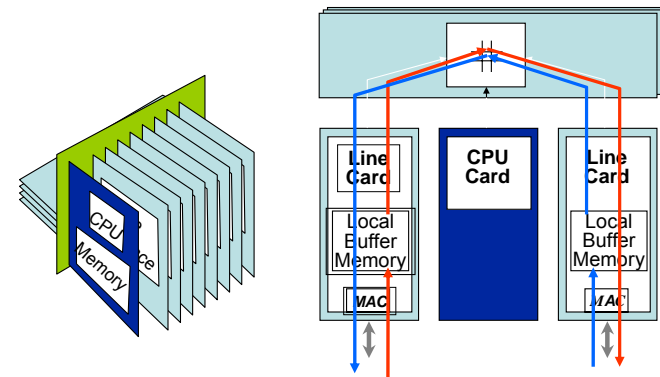
## First-Generation IP Routers



## Second-Generation IP Routers



## Third-Generation Switches/Routers



## 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 you to retrieve content of memory location based on (part of the) content
  - ❖ access using packet address
  - ❖ retrieve next hop information (and address)

## Lookup Using CAMs

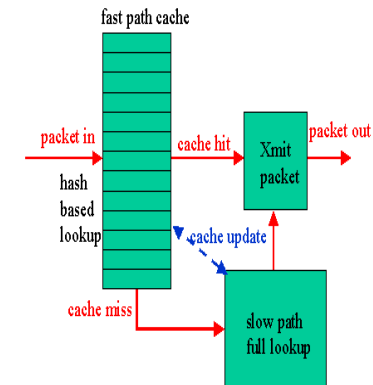
- ▶ Some limitations of CAMs
  - ❖ lower density than conventional (RAM) memory
  - ❖ slower access time
  - ❖ greater power consumption
  - ❖ best suited to exact matches, although can be used for longest prefix match

## Hashing Techniques

- ▶ Function  $f(K)$  maps key (address)  $K$  to table entry
- ▶ Goal for  $f$ 
  - ❖ easy to compute
  - ❖ minimum likelihood of two distinct keys colliding on same entry
  - ❖ robust to redundancy in key patterns, e.g., common network address
  - ❖ fast resolution of collisions

## Software Based Lookup:

- ▶ Active flows kept in fast path cache
- ▶ Exact match lookup (hash)
  - ❖ sequential search in case of collision (performance hit)
- ▶ Slow path invoked in case of cache miss
- ▶ Slow path updates cache
- ▶ Cache management to remove stale entries
- ▶ Coexistence of installed and learned entries



## Some properties of hashing

- ▶ Given a hash table of size  $M$  and  $N$  active addresses
  - ❖ what is the collision probability?
  - ❖ what is the probability of more than  $k$  addresses colliding on the same entry
- ▶ Hash based approach is highly sensitive to
  - ❖ number of simultaneously active flows (addresses)
  - ❖ duration of active flows (overhead of deletion/insertion)
  - ❖ actual distribution of addresses (non-random hashing)

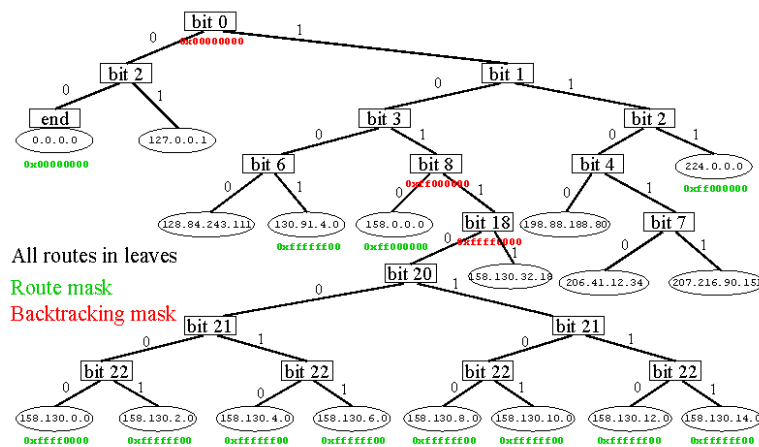
## PATRICIA Trie

- ▶ PATRICIA (Practical Algorithms To Retrieve Information Coded in Alphanumeric) invented in 1968 by D. L. Morrison
- ▶ 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 down the tree until the **end** is reached or a **match is discovered**
- ▶ Patricia tries were useful in previous generation routers
  - ❖ limited amount of memory is needed to store routing table
- ▶ Can result in large number of memory accesses -- particularly when backtracking takes place

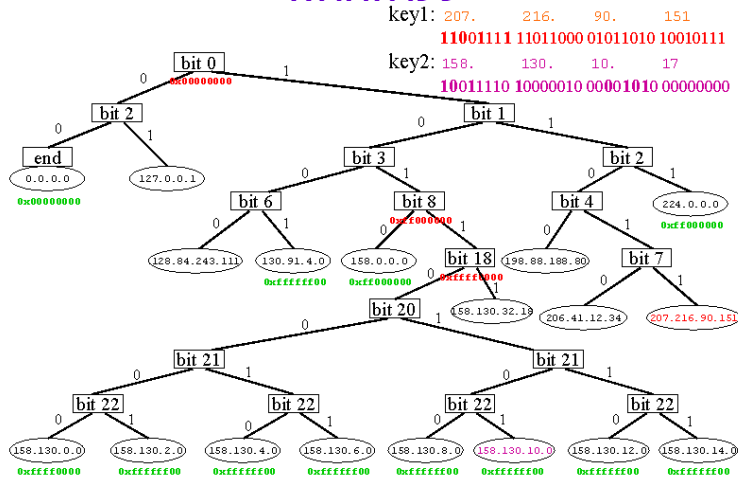
## PATRICIA Trie Representation



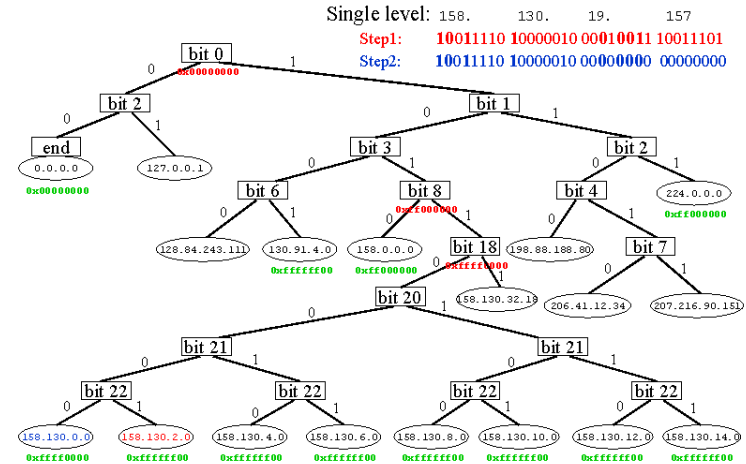
## Lookup Situations with PATRICIA Trie

- ▶ 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

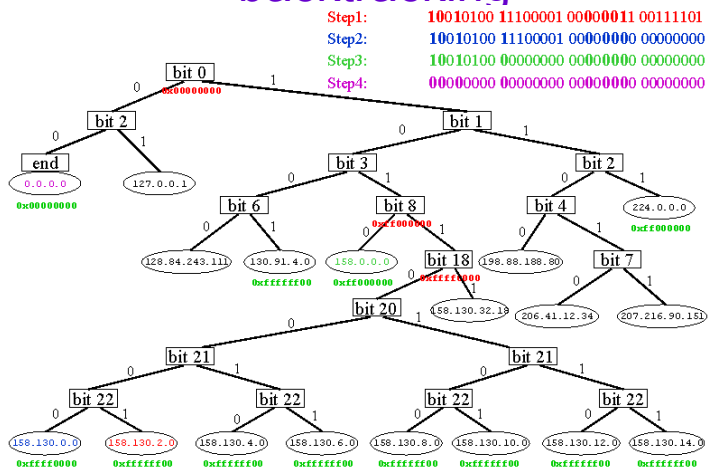
## Direct Match to Host & Network Address



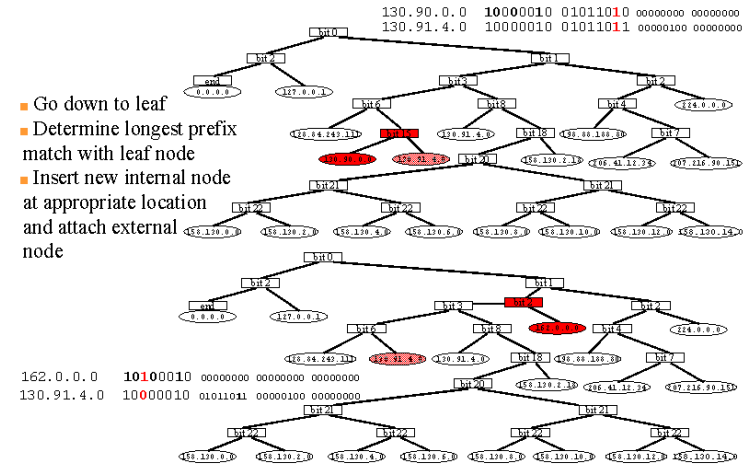
## Match Network Address after backtracking



## Match Network Address after backtracking

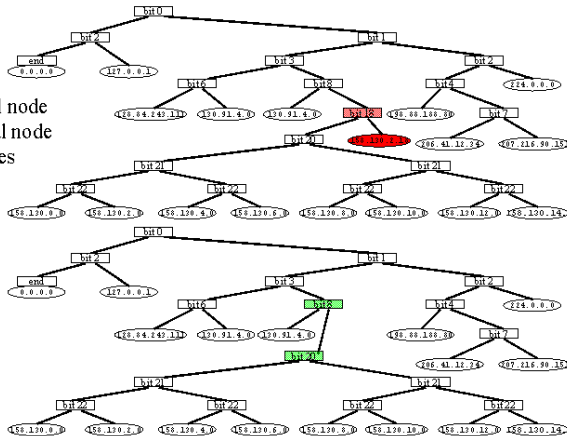


## Insertion in PATRICIA Trie



## Deletion in PATRICIA Trie

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



## Other switching element functions

- ▶ Participate in routing algorithms
  - ❖ to build routing tables
- ▶ Resolve contention for output trunks
  - ❖ scheduling
- ▶ Admission control
  - ❖ to guarantee resources to certain streams
- ▶ We'll discuss these later
- ▶ Here we focus on pure data movement

## Blocking in packet switches

- ▶ Can have both internal 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

## Dealing with blocking

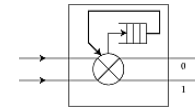
- ▶ Overprovisioning
  - ❖ internal links much faster than inputs
- ▶ Buffers
  - ❖ at input or output
- ▶ Backpressure
  - ❖ if switch fabric doesn't have buffers, prevent packet from entering until path is available
- ▶ Parallel switch fabrics
  - ❖ increases effective switching capacity

## Switch fabrics

- ▶ Transfer data from input to output, ignoring scheduling and buffering
- ▶ Usually consist of links and *switching elements*

## Switch fabric element

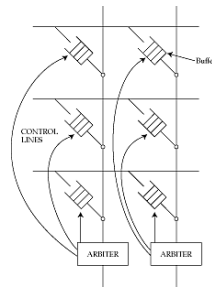
- ▶ Can build complicated fabrics from a simple element



- ▶ Routing rule: if 0, send packet to upper output, else to lower output
- ▶ If both packets to same output, buffer or drop

## Buffered crossbar

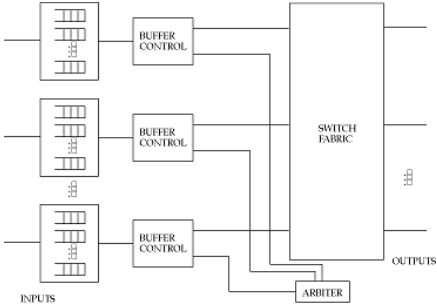
- ▶ What happens if packets at two inputs both want to go to same output?
- ▶ Can defer one at an input buffer
- ▶ Or, buffer crosspoints



## Buffering

- ▶ All packet switches need buffers to match input rate to service rate
  - ❖ or cause heavy packet losses
- ▶ Where should we place buffers?
  - ❖ input
  - ❖ in the fabric
  - ❖ output
  - ❖ shared

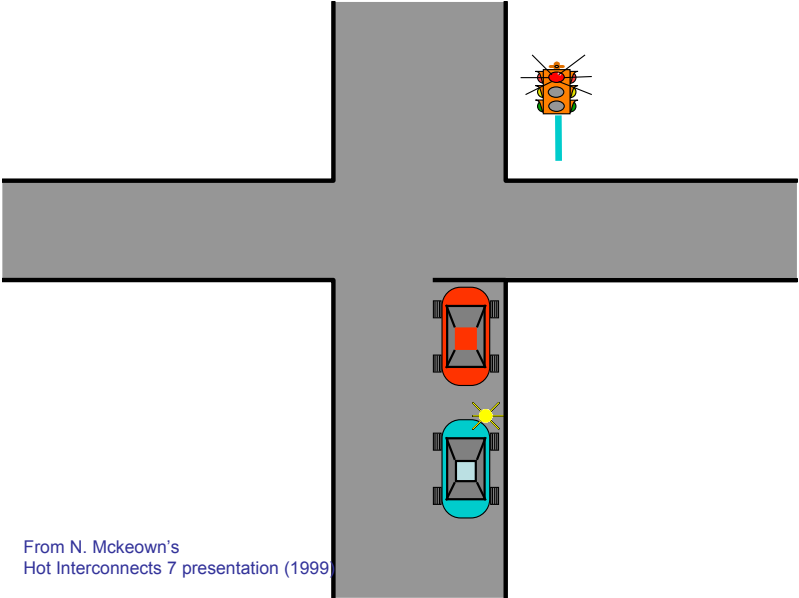
# Input buffering (input queueing)



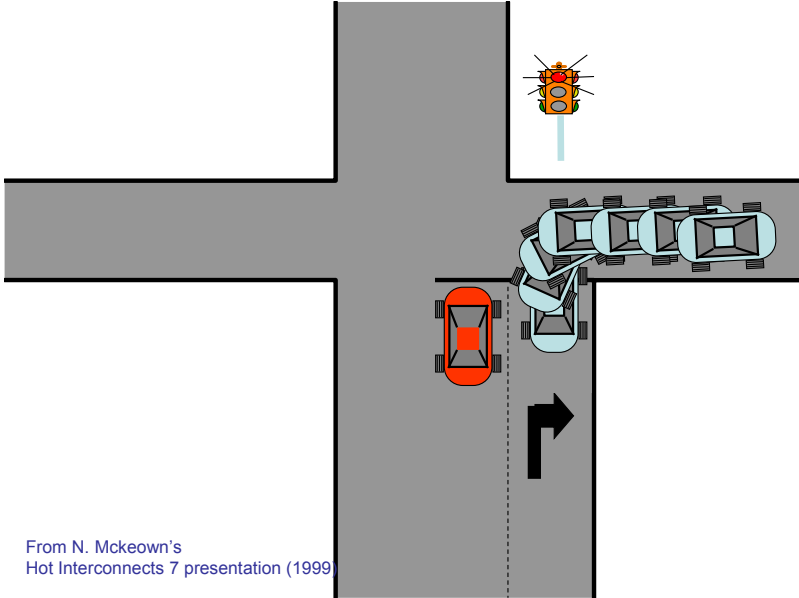
- ▶ No speedup in buffers or trunks (unlike output queued switch)
- ▶ Needs arbiter

# Input buffering (input queueing)

- ▶ Problem: *head of line blocking*
  - ❖ with randomly distributed packets, utilization at most 58.6%
  - ❖ worse with *hot spots*



From N. McKeown's Hot Interconnects 7 presentation (1999)



From N. McKeown's Hot Interconnects 7 presentation (1999)

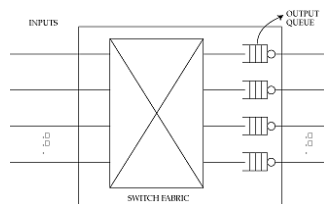
## Dealing with HOL blocking

- ▶ Per-output queues at inputs
- ▶ Arbiter must choose one of the input ports for each output port
- ▶ How to select?

## Dealing with HOL blocking

- ▶ Parallel Iterated Matching
  - ❖ inputs tell arbiter which outputs they are interested in
  - ❖ output selects one of the inputs
  - ❖ some inputs may get more than one *grant*, others may get none
  - ❖ if >1 grant, input picks one at random, and tells output
  - ❖ losing inputs and outputs try again
- ▶ Used in DEC Autonet 2 switch

## Output queueing

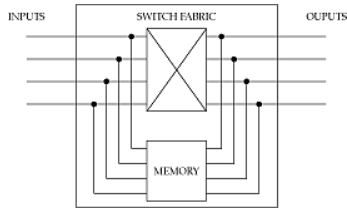


- ▶ Don't suffer from head-of-line blocking
- ▶ But output buffers need to run much faster than trunk speed (why?)

## Output queueing

- ▶ Can reduce some of the cost by using the *knockout* principle
  - ❖ unlikely that all N inputs will have packets for the same output
  - ❖ drop extra packets, fairly distributing losses among inputs

## Shared memory



- ▶ Route only the header to output port
- ▶ Bottleneck is time taken to read and write multiplexed memory
- ▶ Doesn't scale to large switches
- ▶ But can form an element in a multistage switch