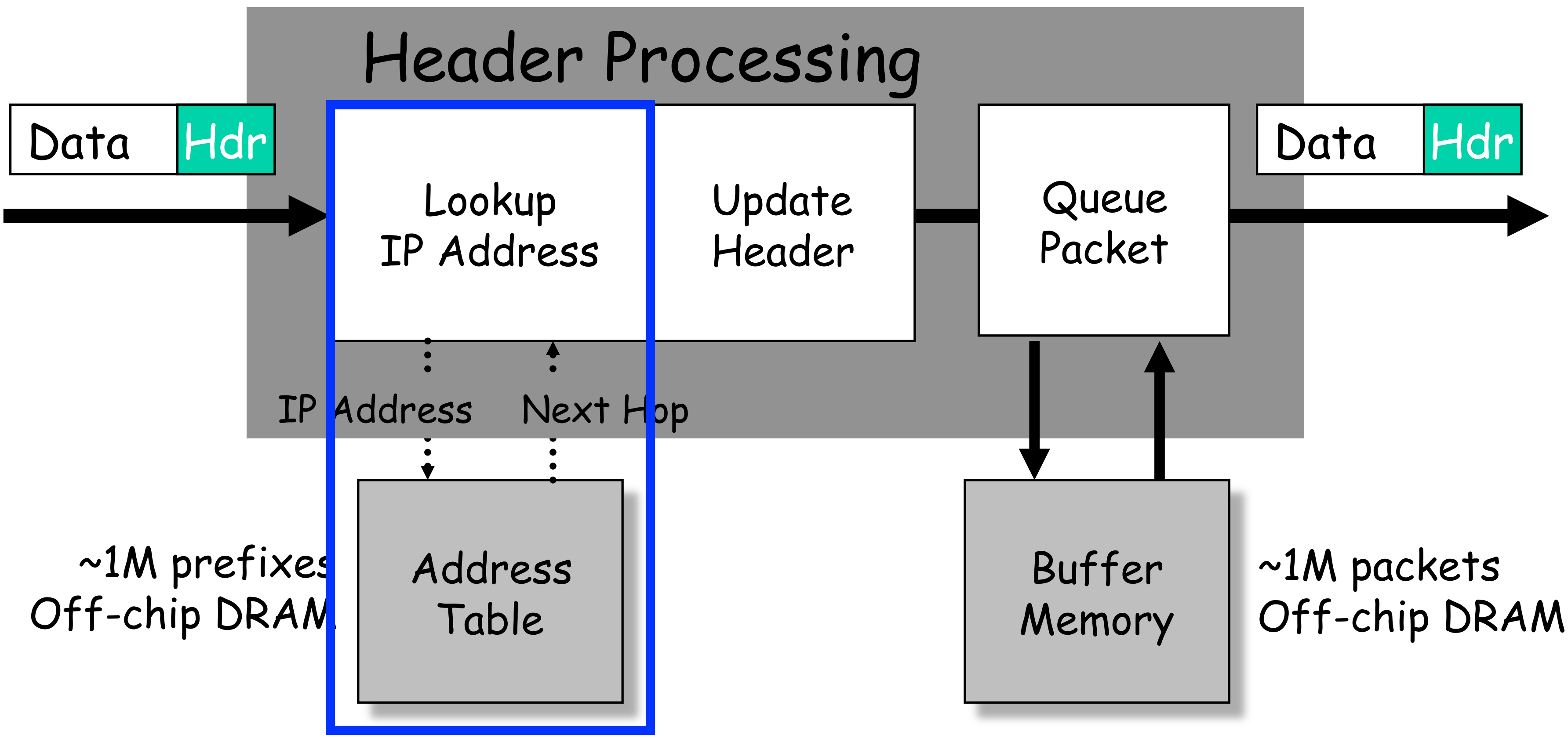
Last Lecture: Network Layer

1. *Design goals and issues*
2. *Basic Routing Algorithms & Protocols*
3. *Addressing, Fragmentation and reassembly*
4. *Internet Routing Protocols and Inter-networking*
5. *Router design*
   1. *Short History + Router Architectures*
   2. *Switching fabrics* ✔
   3. *Address lookup problem*
6. *Congestion Control, Quality of Service*
7. *More on the Internet's Network Layer*

Last Lecture: Network Layer

1. *Design goals and issues*
2. *Basic Routing Algorithms & Protocols*
3. *Addressing, Fragmentation and reassembly*
4. *Internet Routing Protocols and Inter-networking*
5. *Router design*
    1. *Short History + Router Architectures*
    2. *Switching fabrics*
    3. *Address lookup problem* ✔
6. *Congestion Control, Quality of Service*
7. *More on the Internet's Network Layer*

Reminder on Router Architecture



Basic Requirements

1) Fast lookup **and** fast update

2) Scalable (speed & table size)

3) Inexpensive (fit in memory, e.g.)

Lookups Must Be Fast

Year	Aggregate Line-rate	Arriving rate of 40B POS packets (Million pkts/sec)
1997	622 Mb/s	1.56
1999	2.5 Gb/s	6.25
2001	10 Gb/s	25
2003	40 Gb/s	100
2006	80 Gb/s	200

1. Lookup mechanism must be simple and easy to implement
2. Memory access time is the bottleneck

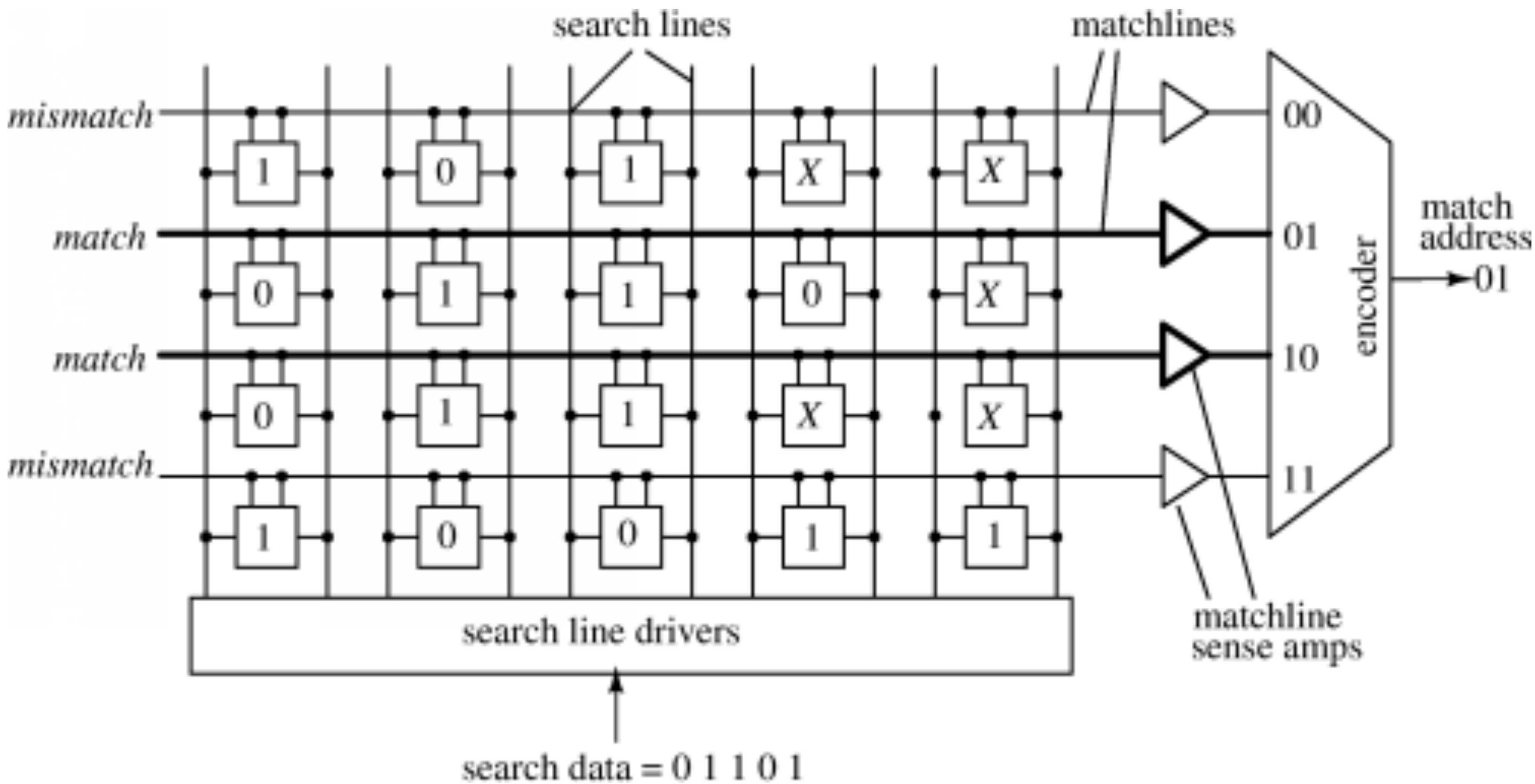
   200Mpps × 2 lookups/pkt = 400 Mlookups/sec → 2.5ns per lookup

Memory Technologies (2006)

Technology	Max single chip density	$/chip ($/MByte)	Access speed	Watts/ chip
Networking DRAM	64 MB	$30-$50 ($0.50-$0.75)	40-80ns	0.5-2W
SRAM	8 MB	$50-$60 ($5-$8)	3-4ns	2-3W
TCAM	2 MB	$200-$250 ($100-$125)	4-8ns	15-30W

Note: rough estimates only. Manufacturer & technology & market dependent

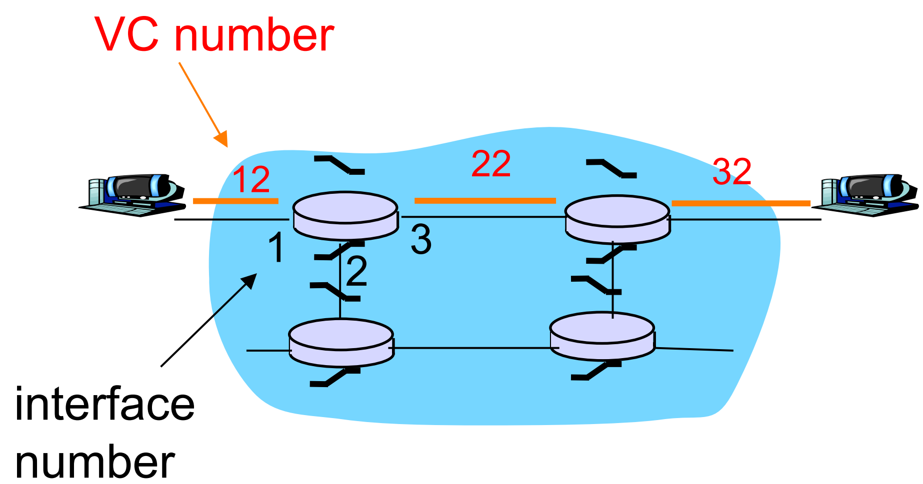
(Ternary) Content Addressable Memory



Lookup Problem: Protocol Dependent

Networking Protocol	Lookup Mechanism	Techniques we will study
MPLS, ATM (virtual circuits) Ethernet	*1. Exact match search*	– Direct lookup – Associative lookup – Hashing – Binary/Multi-way Search Trie/ Tree
IPv4, IPv6 (datagram)	*2. Longest-prefix match search*	-Radix trie and variants -Compressed trie -Binary search on prefix intervals

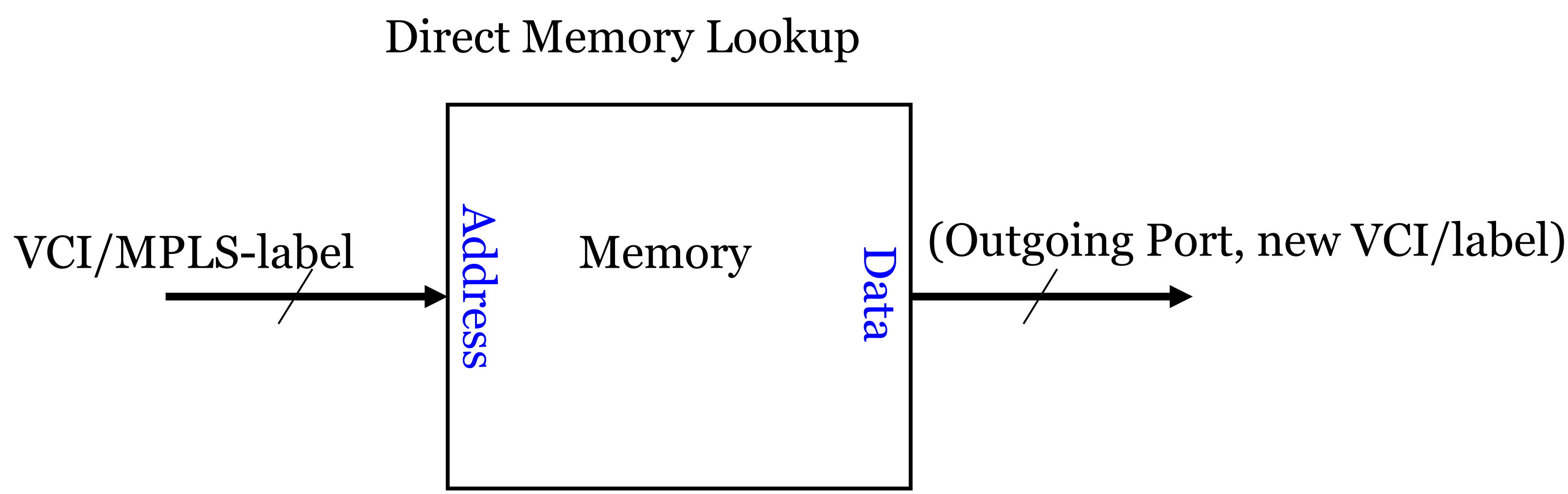
1. Exact Match – Virtual Circuit Reminder



Forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Exact Matches in ATM/MPLS



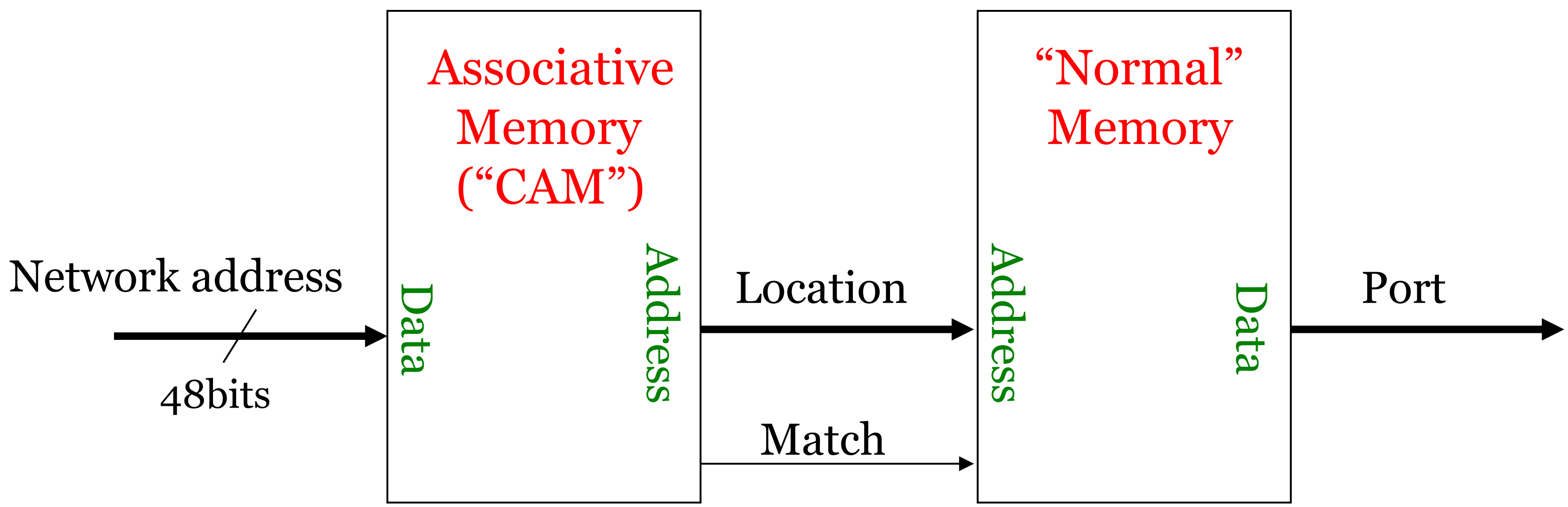
- VCI/Label space is 24 bits
  - Maximum 16M addresses.
  - With 64b data, this is 1Gb of memory.
- VCI/Label space is private to one link
- Therefore, table size can be "negotiated"
- Alternately, use a level of indirection

Exact Matches in Ethernet Switches

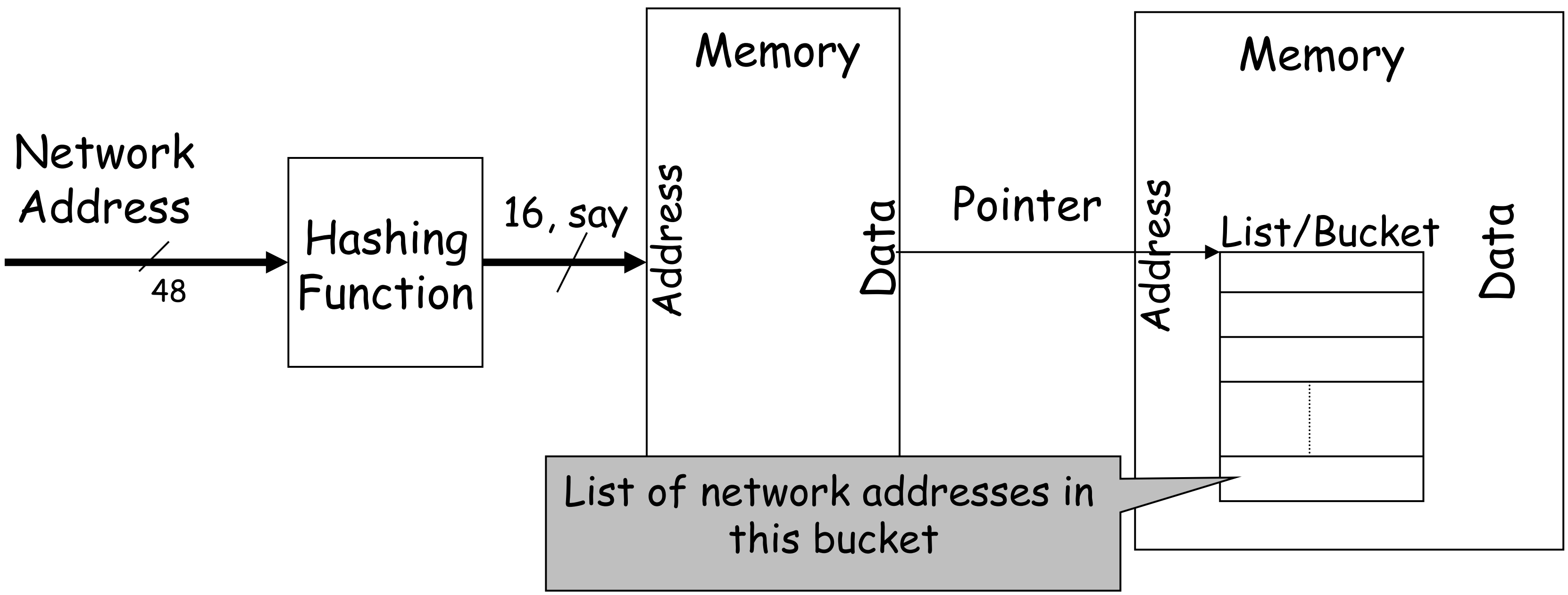
- Layer-2 addresses are usually 48-bits long,
- The address is global, not just local to the link,
  - The range/size of the address is not "negotiable" (like it is with ATM/MPLS)
- $2^{48} \approx$ 262,144 Gig, *too large*, cannot hold all addresses in table and use direct lookup.

Ethernet Switches: Associative Lookups

- Associative memory (aka Content Addressable Memory, CAM) compares all entries in parallel against incoming data in one clock-cycle



Exact Matches: Hashing



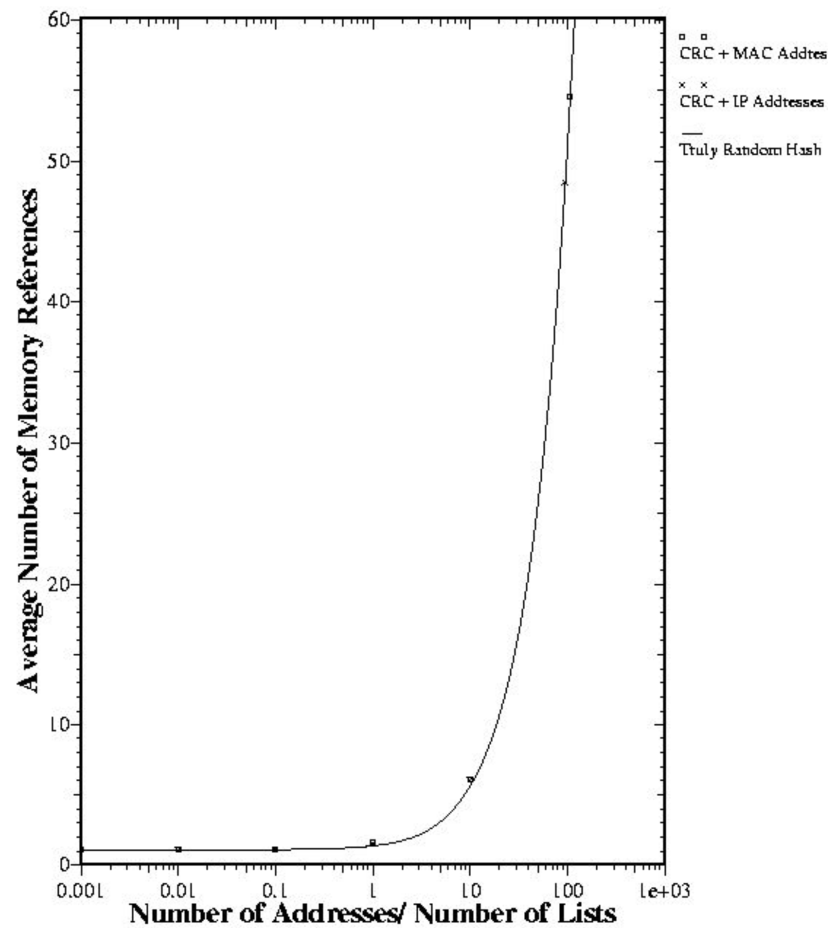
- Use a pseudo-random hash function (relatively insensitive to actual function)
- Bucket linearly searched (or could be binary search, etc.)
- Leads to unpredictable number of memory references

Performance of Lookup With Hashing

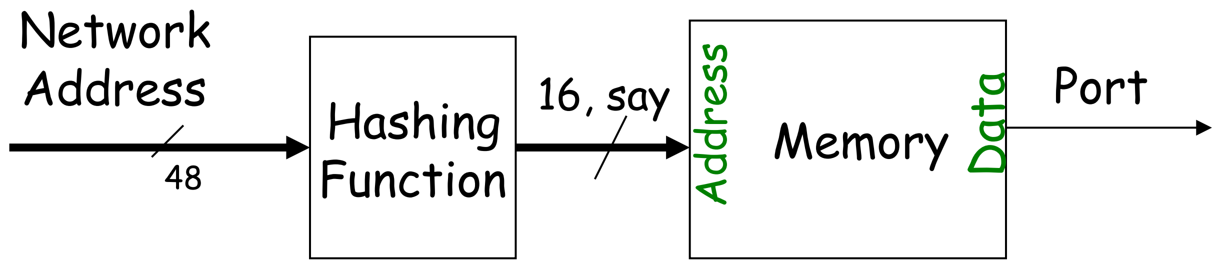
Expected number of memory references:

$$ER = \frac{1}{2}(\text{Expected length of list} \mid \text{list is not empty})$$

$$= \frac{1}{2}\left(1 + \frac{\alpha}{1 - (1 - \frac{1}{N})^M}\right)$$



“Perfect” Hash Function



There always exists a “perfect” hash function (since # of hosts connected to the switch is ≤ 2^{16})

With a perfect hash function, memory lookup always takes O(1) memory references.

Problem:

- Finding perfect hash functions (particularly a minimal perfect hash) is complex.
- Updates make such a hash function yet more complex
- Advanced techniques: multiple hash functions, *bloom filters*…

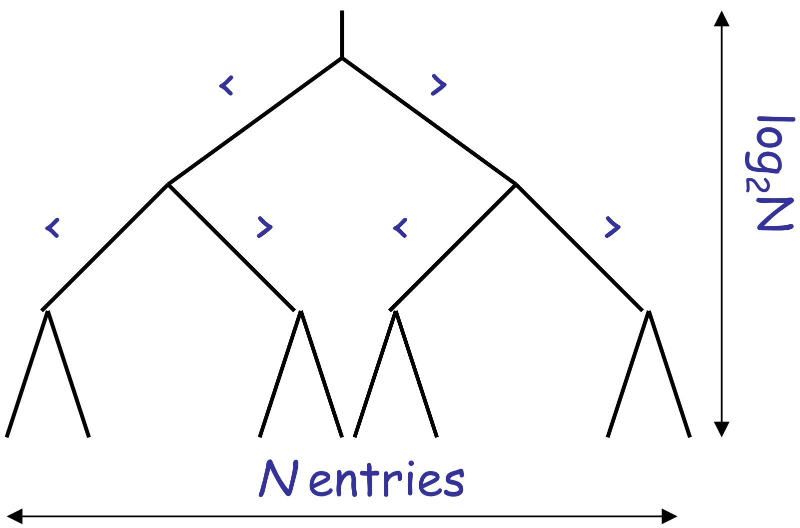
Hashing: Pretty Good Choice for Exact Match

- *Advantages*:
  - Simple to implement
  - Expected lookup time is small
  - Updates are fast (except with perfect hash functions)
- *Disadvantages*
  - Relatively inefficient use of memory
  - Non-deterministic lookup time (in rare cases)

- $\Rightarrow$ Attractive for software-based switches. However, hardware platforms are moving to other techniques (but they can do well with a more sophisticated form of hashing)

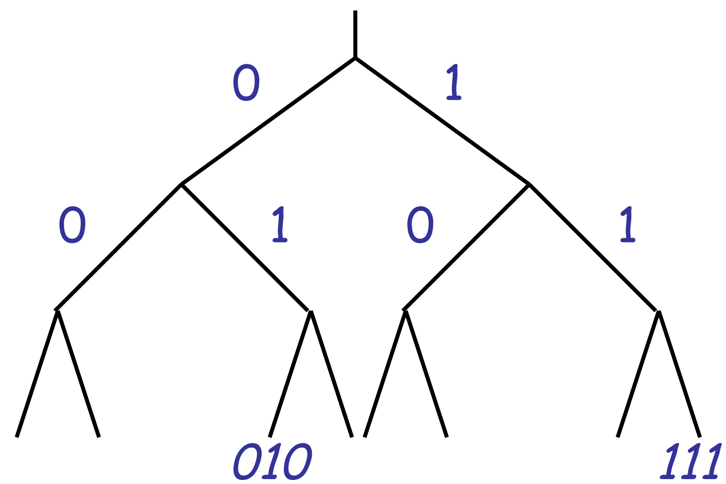
Trees and Tries

Binary Search Tree



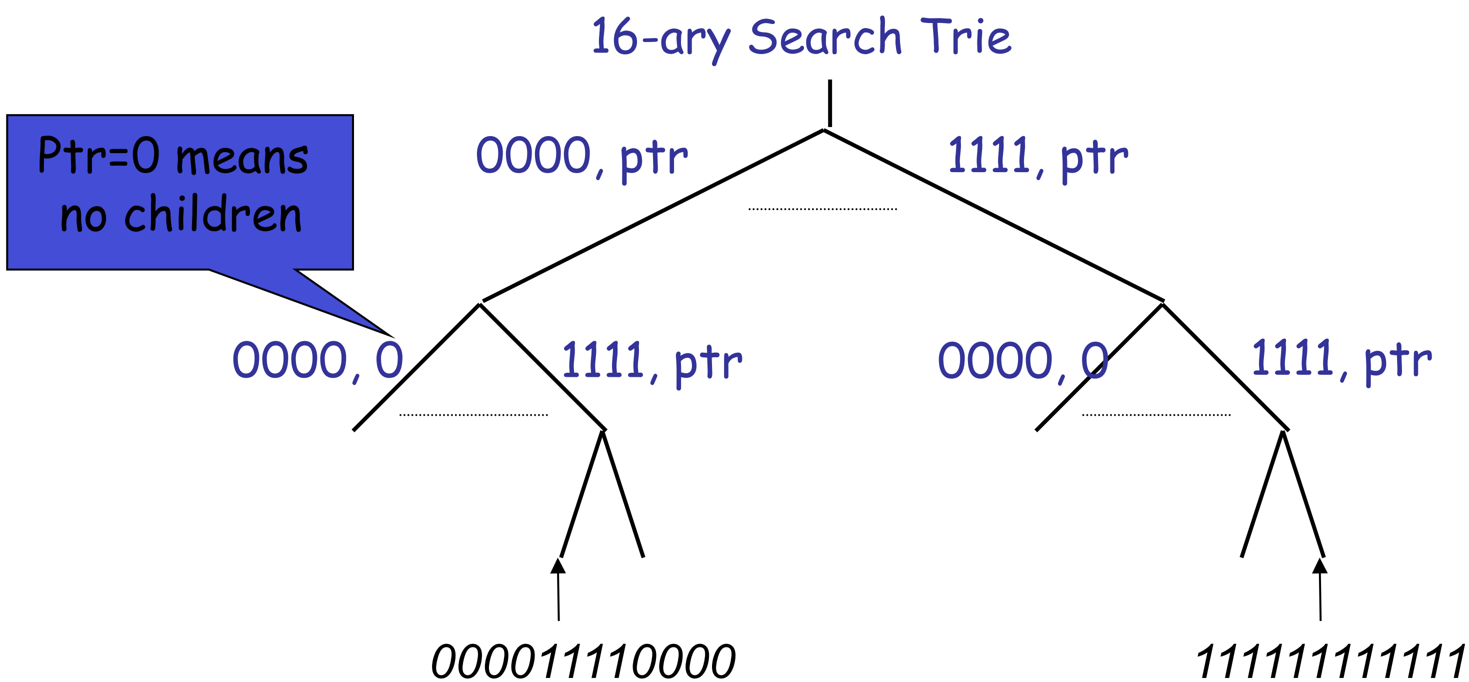
Lookup time dependent on table size, but independent of address length, storage is O(N), assuming comparisons can be done in O(1)-time

Binary Search Trie



Lookup time bounded and independent of table size, storage is O(NW)

Multiway Tries



Question: Why don't we just make it a 2^{48}-ary trie?

Multiway Tries

As degree increases, more and more pointers are "0"

Degree of Tree	# Mem Ref erences	# Nodes ($x10^6$)	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

More on Tries

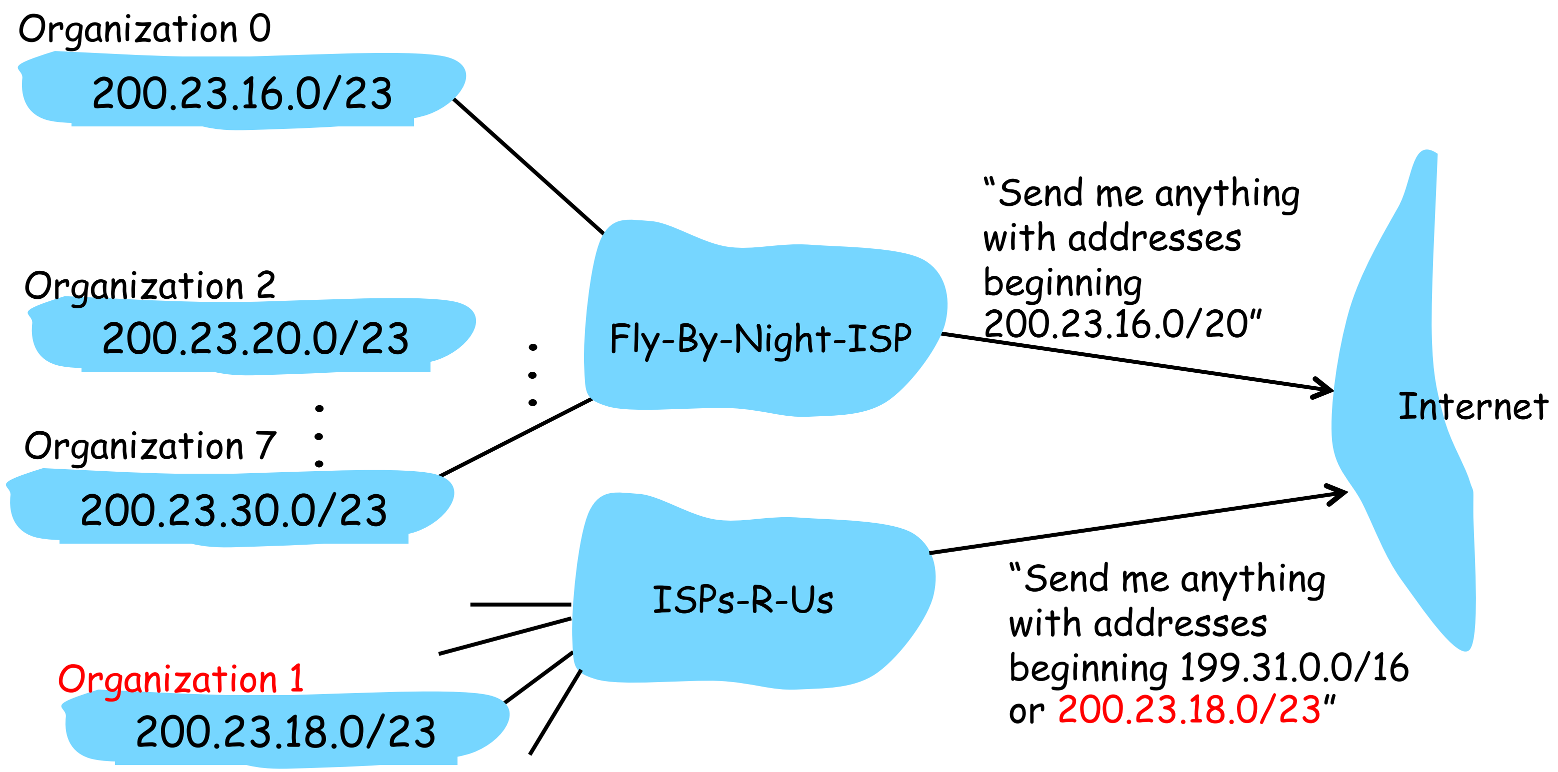
- *Advantages*:
  - Bounded lookup time
  - Simple to implement and update

- *Disadvantages:*
  - Inefficient use of memory and/or requires large number of memory references

More sophisticated algorithms compress 'sparse' nodes.

2. Longest Prefix Match -- Reminder

ISPs-R-Us has a more specific route to Organization 1



Requires routers to do *longest prefix match*, per packet, every few nanosecond

Example

```
address: 11001111 01011100 00000000 10000111
mask:    11111111 11111111 11111111 11111111

address: 11001111 01011100 00000000 00000000
mask:    11111111 11111111 00000000 00000000

address: 11001111 01011100 00000000 00000000
mask:    11111111 11111111 11100000 00000000
```

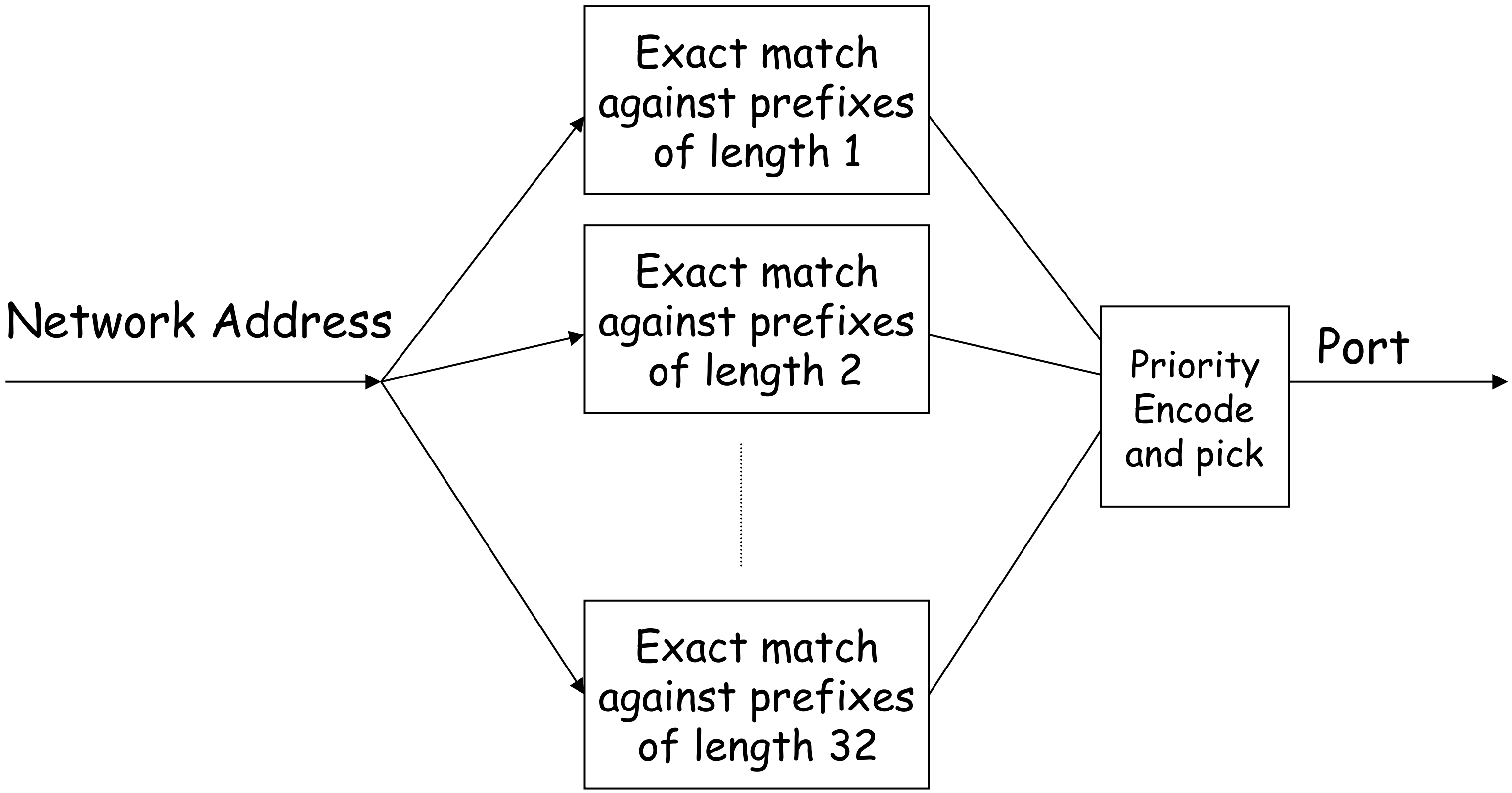
Packet's Destination Address:

11001111 01011100 00000000 10010111

Longest Prefix Matching (LPM) Is Hard!

- Arriving packet does not carry "network prefix"
  - Why?
- Hence, one needs to search among the space of all prefix lengths; as well as the space of all prefixes of a given length
- There are many proposed solutions, we'll only talk about a few

Use … 32 Exact Matching Algorithms

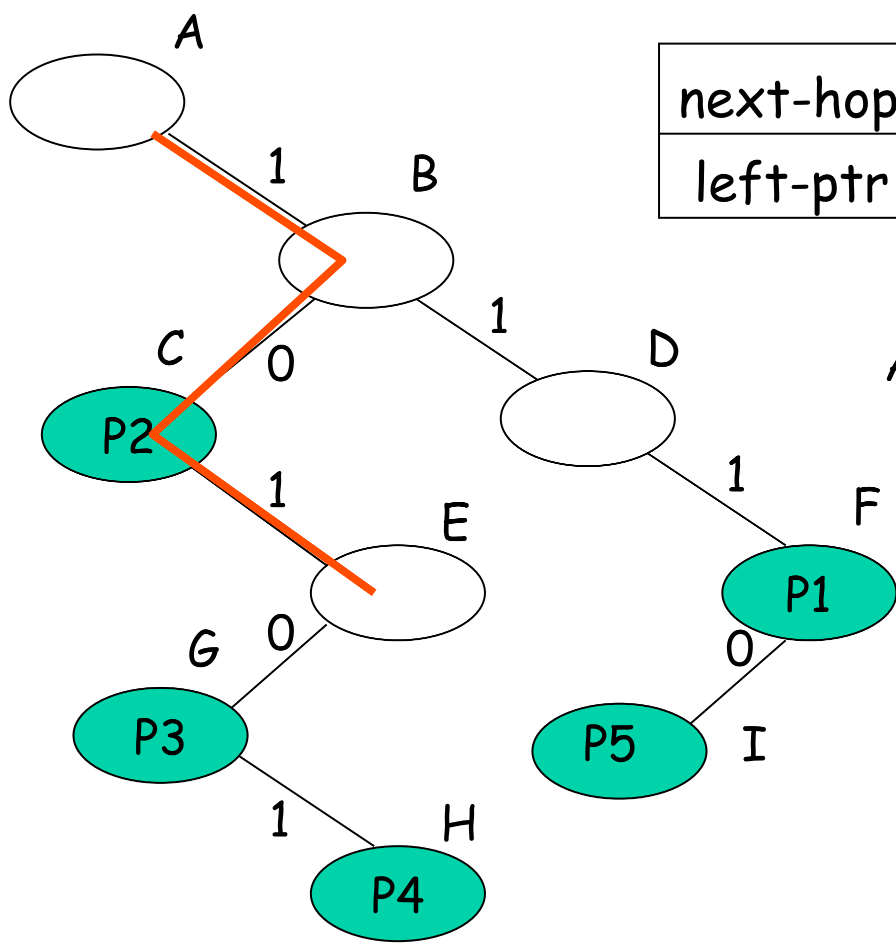


Objectives

- Speed (= number of memory accesses)
- Storage requirements (= amount of memory)
- Low update time (support >10K updates/s)
- Scalability
  - With length of prefix: IPv4 unicast (32b), Ethernet (48b), IPv4 multicast (64b), IPv6 unicast (128b)
  - With size of routing table: (sweetspot for today's designs = 1 million)
- Flexibility in implementation
- Low preprocessing time

Radix Trie (Recap)

P1	111*	H1
P2	10*	H2
P3	1010*	H3
P4	10101	H4



Trie node

next-hop-ptr (if prefix)	
left-ptr	right-ptr

Add P5=1110*

Lookup 10111

Pros and Cons

- W-bit prefixes: O(W) lookup, O(NW) storage and O(W) update complexity

Advantages

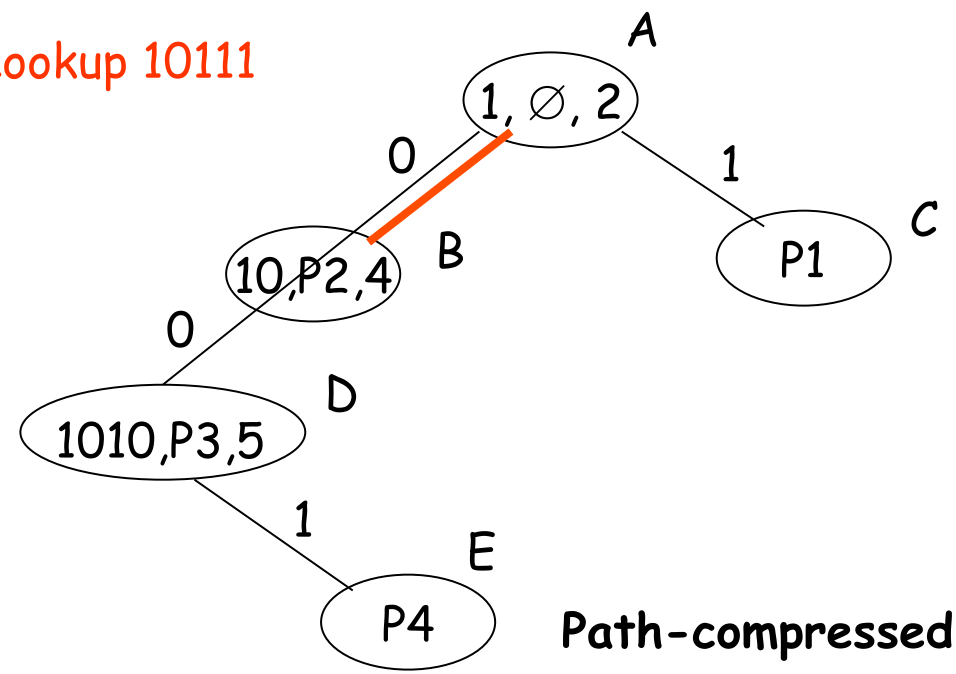
- Simplicity
- Extensible to wider fields

Disadvantages

- Worst case lookup slow
- Wastage of storage space in *chains*

Patricia Tries: Compress the Un-branched

Lookup 10111



P1	111*	H1
P2	10*	H2
P3	1010*	H3
P4	10101	H4

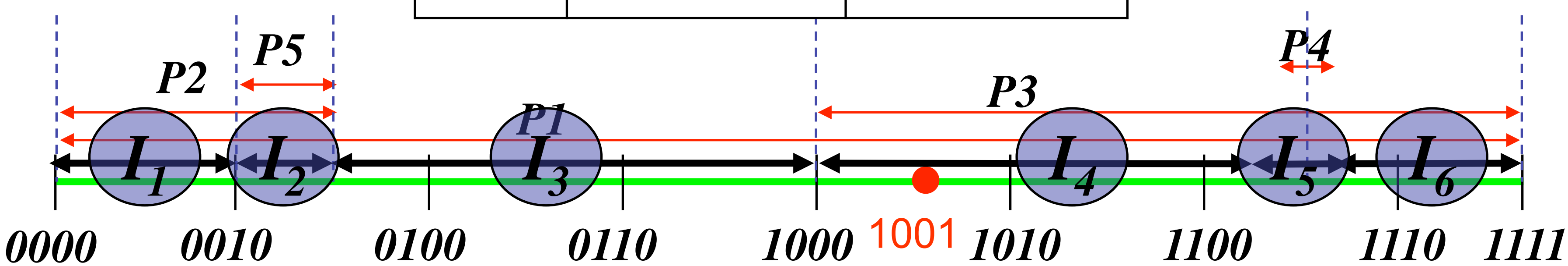
Path-compressed tree node structure

variable-length bitstring	next-hop (if prefix present)	bit-position
left-ptr	right-ptr	

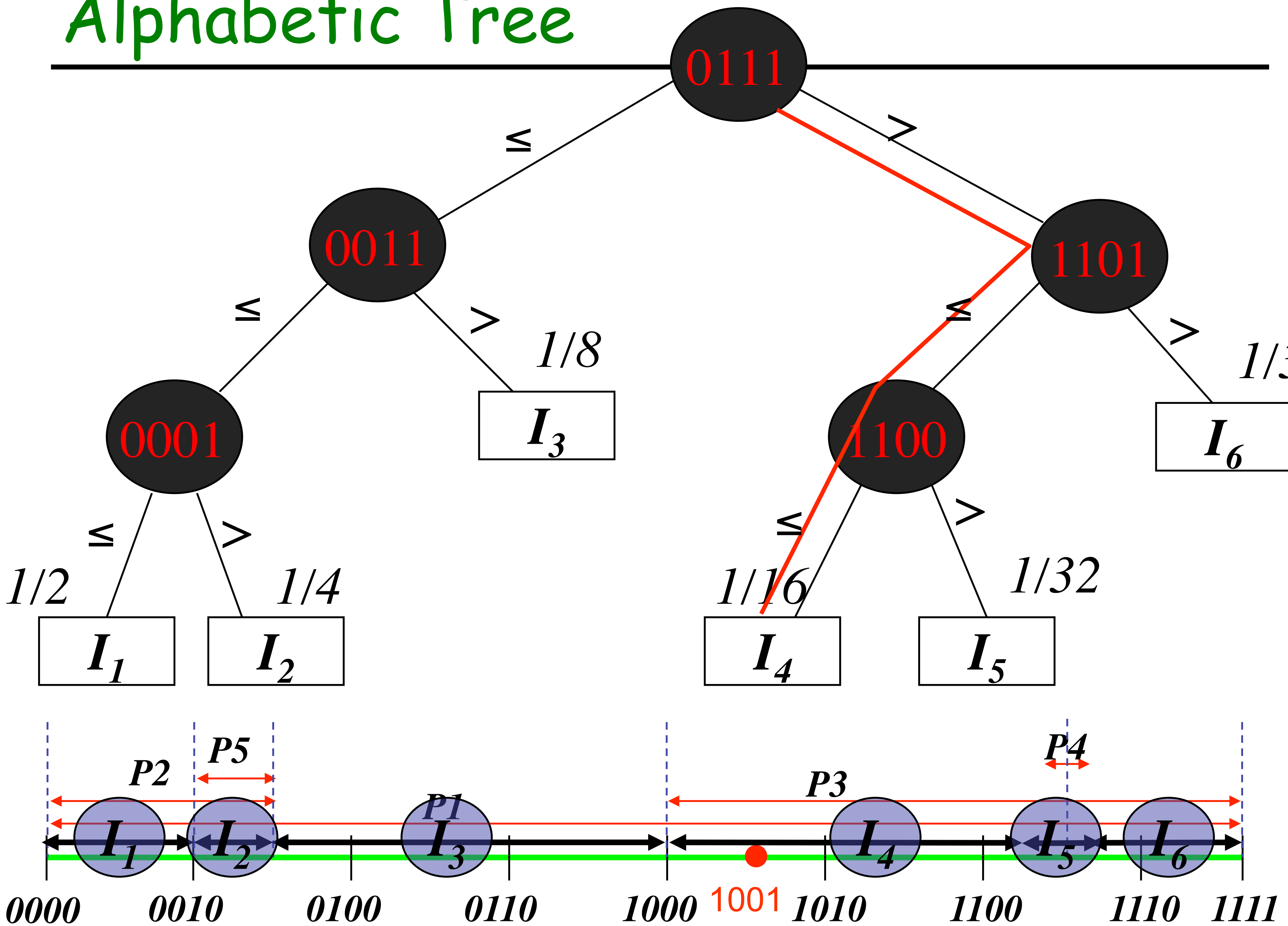
PATRICIA: Practical Algorithm To Retrieve Information Coded as Alphanumeric

Binary Search on Prefix Intervals [Lampson98]

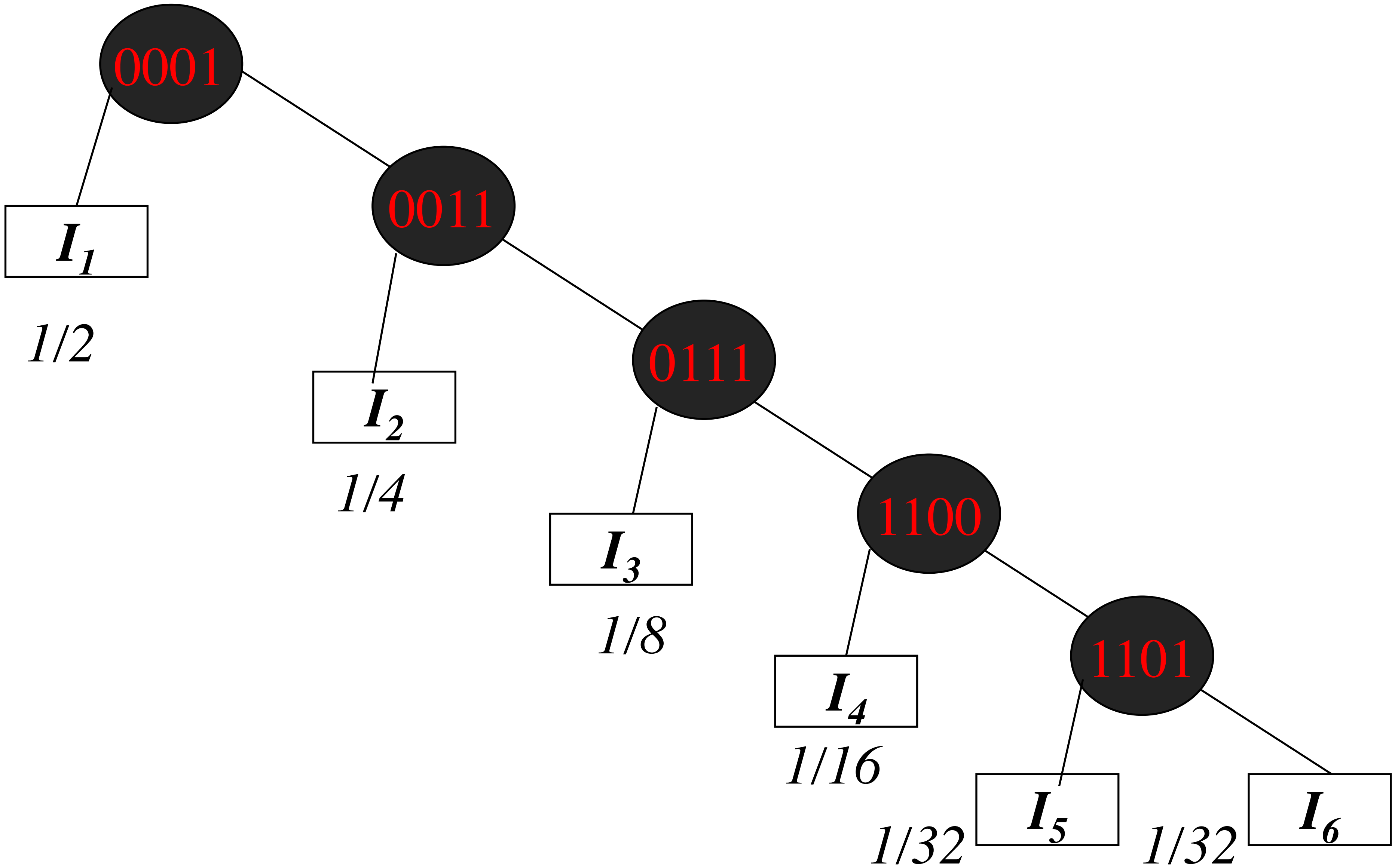
	Prefix	Interval
P1	/0	0000…1111
P2	00/2	0000…0011
P3	1/1	1000…1111
P4	1101/4	1101…1101
P5	001/3	0010…0011



Alphabetic Tree



Another Alphabetic Tree



Multiway Search on Intervals

- W-bit N prefixes: O(logN) lookup, O(N) storage

Advantages

- Storage is linear
- Can be 'balanced'
- Lookup time independent of W

Disadvantages

- But, lookup time is dependent on N
- Incremental updates more complex than tries
- Each node is big in size: requires higher memory bandwidth

Rough Comparison

Algorithm	Build	Search	Memory
(pure) Binary Search	O(WN logN)	O(Wlog N)	O(NW)
Trie	O(NW)	O(W)	O(NW)
Binary prefix search	O(NlogW)	O(logW)	O(N logW)

But, those are not the only practical concerns. Update time, e.g., is important!