University of Massachusetts Amherst

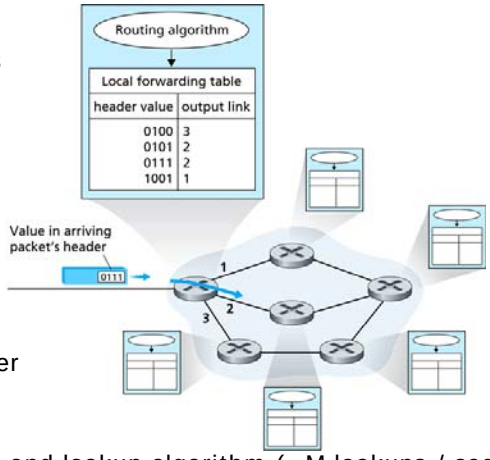
ECE697AA – Lecture 20

Routers: Prefix Lookup Algorithms

Tilman Wolf
Department of Electrical and Computer Engineering
11/14/08

Forwarding Tables

- Routing protocols involve a lot of information
  - Path choices, policies
- Input port of router needs condensed version
  - Given an IP destination address, find output port
  - "Forwarding Table"
- Forwarding table entries
  - IP subnet prefix (aggregated CIDR)
  - Output port of router
  - Address of next-hop router (unless point-to-point)
- Performance is important
  - Need good data structure and lookup algorithm (~M lookups / sec)
  - Capability for updates (~100s updates / sec)

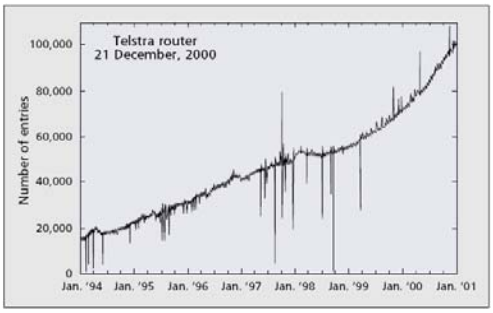


Forwarding Tables

- Example forwarding table:

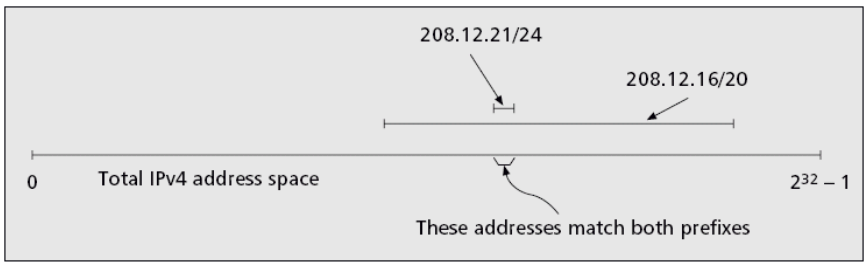
Destination address prefix	Next-hop	Output interface
24.40.32/20	192.41.177.148	2
130.86/16	192.41.177.181	6
208.12.16/20	192.41.177.241	4
208.12.21/24	192.41.177.196	1
167.24.103/24	192.41.177.3	4

- Typical table size:
  - O(100k) entries



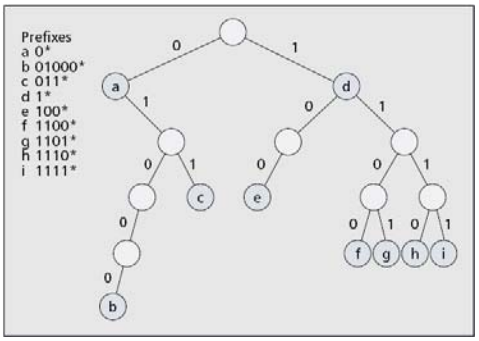
Prefix Matching

- CIDR allows any size prefix
  - Overlapping prefixes possible
  - Must find best matching prefix (BMP)
  - Same as longest matching prefix



Binary Trees

- Classical solution
- Tree structure
  - Node at level l reflects all prefixes of length l
  - Path in tree corresponds to prefix value
    - » Left child represents “0”
    - » Right child represents “1”
  - Leaf nodes and some internal nodes are prefixes
- Search algorithm
  - Follow prefix through tree
  - Last found prefix is longest prefix match

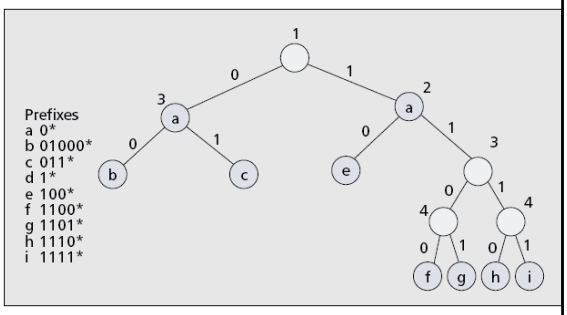


Binary Trees

- Update
  - Adding prefix:
    - » Search for new prefix
    - » When end of tree then add nodes
  - Removing prefix:
    - » Unmark node as prefix
    - » Remove unused nodes toward root
- Shortcomings of binary tree
  - Easy to “fall of tree” which requires backtracking
  - One memory lookup per prefix bit

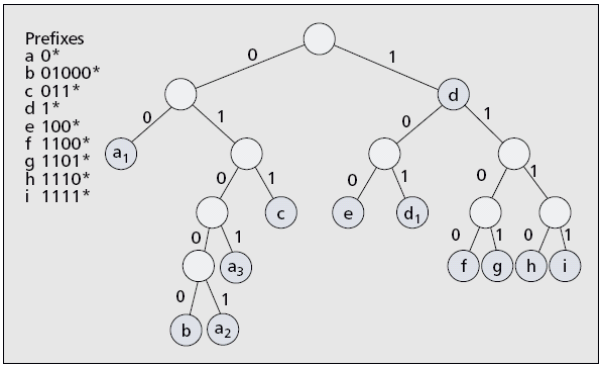
Path Compressed Tree

- Removal of one-way branching nodes
  - Shorter search
- Prefixes end up in "other" nodes
  - Requires additional information on node
- Still need to maintain BMP
  - Leaf node of tree might not match
- Implementation:
  - PATRICIA trees
  - Variant used in BSD
    - » Search of leaf without comparison
    - » Backtracking towards root with comparison
  - Works well for sparse trees



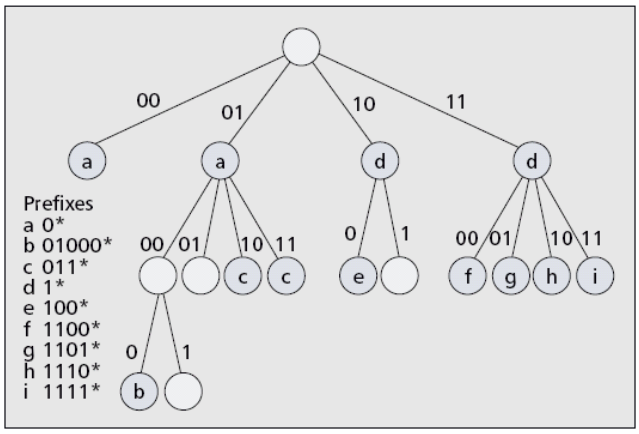
Prefix Transformations

- Disjoint prefixes
  - Don't allow overlapping prefixes
  - If overlap occurs, partition prefixes
- Prefix expansion
  - Increase length of prefix
  - Replace by all possible longer prefixes



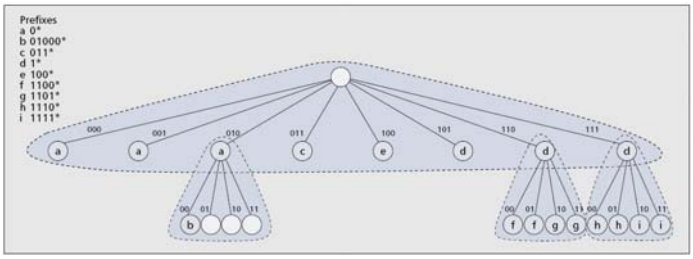
Multibit Tries

- Trie is tree with nodes that have exactly 2^n children
  - n bits encoded in each step ("n-bit stride")
  - Requires prefix expansion
- Example with variable stride:
- What are the tradeoffs in choosing the stride length?

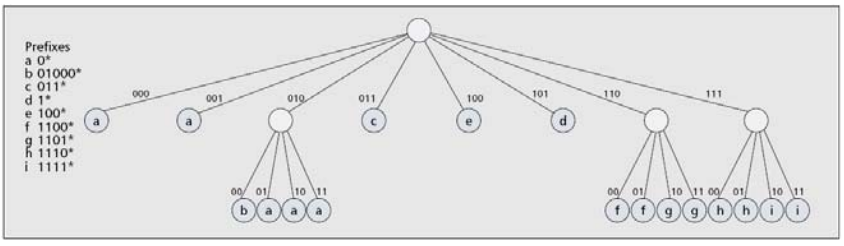


Multibit Tries

- Updates
  - With internal nodes
    - » Only local subtree changes

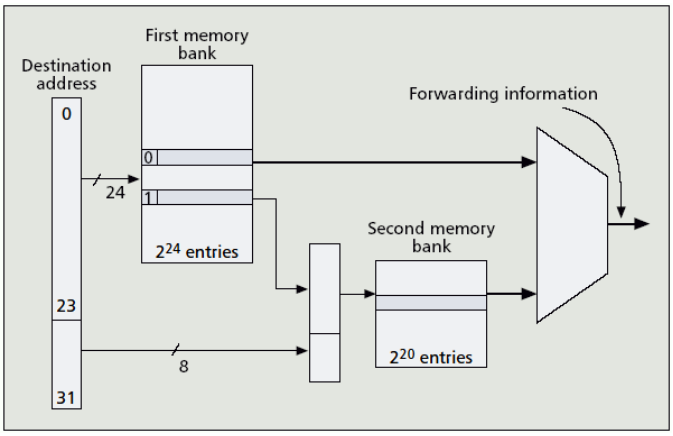


  - Disjoint-prefix multibit trie ("leaf pushing")
    - » Subtrees on all levels can change



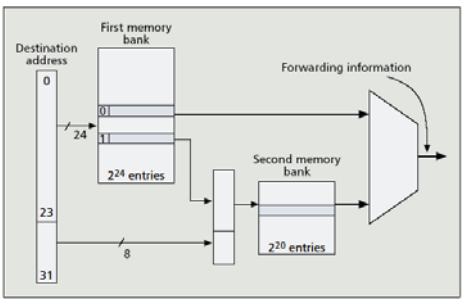
Multibit Tries in Hardware

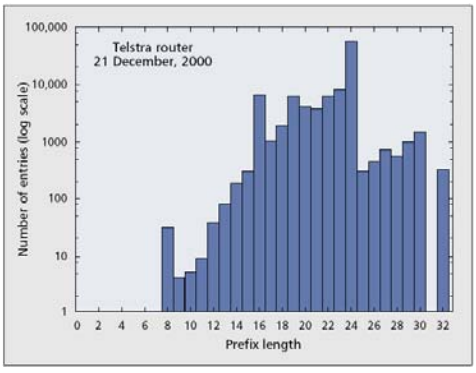
- Hardware implementation of trie
  - Table for each stride
  - Leaf pushing to enforce stride
- Example:
  - 2 strides
    - » 24 bit
    - » 8 bit
  - Memory requirement
    - » 16 bit entries
    - » 32MB in 1st bank
    - » Less in 2nd bank



Multibit Tries in Hardware

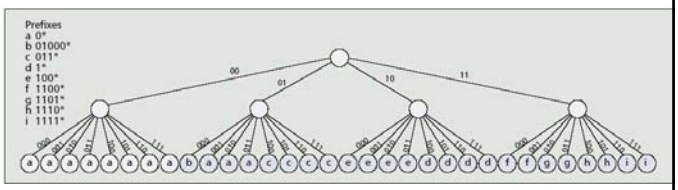
- How well does this work?
  - Depends on prefix-distribution
- Prefix length distribution
  - Most prefixes are 24 bits
  - Few prefixes longer

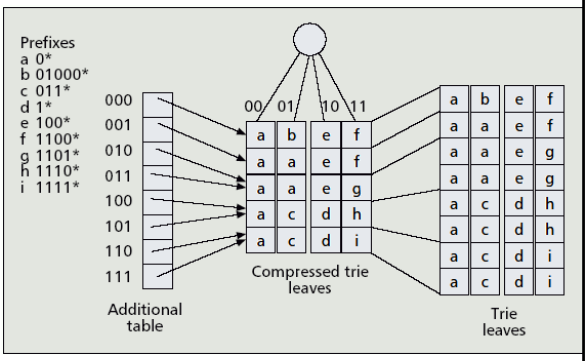




Multibit Trie Compression

- Trie can be represented in a table
  - Expand all prefixes
  - Arrange in 2-D
- Compression
  - Many rows in table are identical
  - Store only one instance of row and maintain pointes
- Small memory requirement
  - Updates difficult



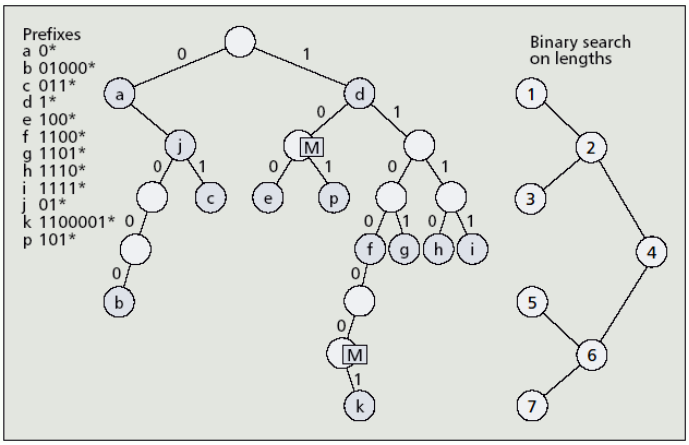


Binary Search on Prefix Lengths

- Lookup simple if length of matching prefix is known
  - We can't know length of longest matching prefix
- Search on prefix length
  - Linear search inefficient (32 steps)
  - Binary search better (5 steps)
- Need additional information tree
  - Each search step needs to provide information
  - "Markers" are added to tree
- Search algorithm
  - Half search space in each step
  - If prefix or marker is found, search for longer prefix

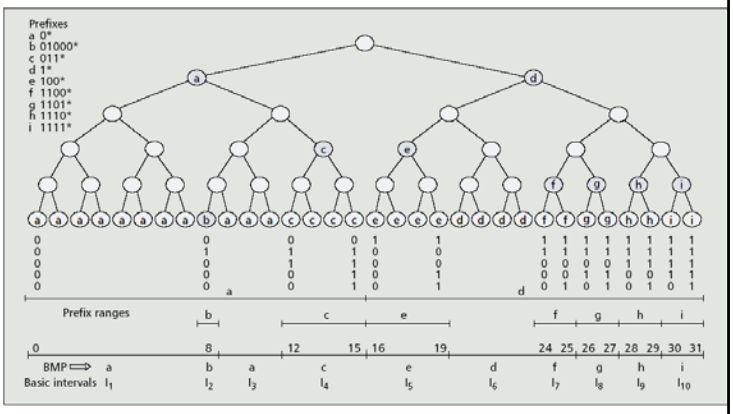
Binary Search on Prefix Lengths

- Search for 11000010:
  - 1st step: search for length 4 (1100) results in f
  - 2nd step: search for length 6 (110000) results in marker
  - 3rd step: search for length 7 (1100001) results in k



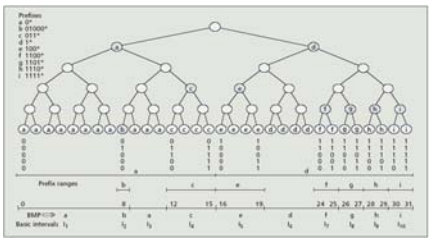
Prefix Range Search

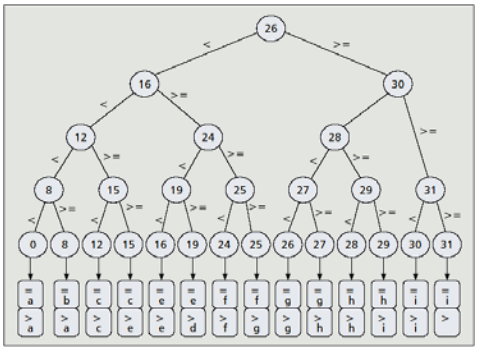
- If all prefixes are same length, search that space
  - 32 bit prefix expansion for all prefixes
- Challenges
  - Overlapping prefix ranges
  - Large number of intervals

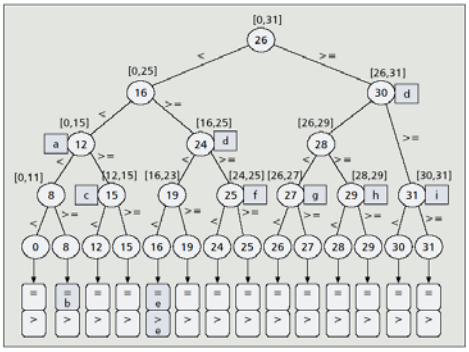


Prefix Range Search

- Search data structure
  - Search tree for intervals



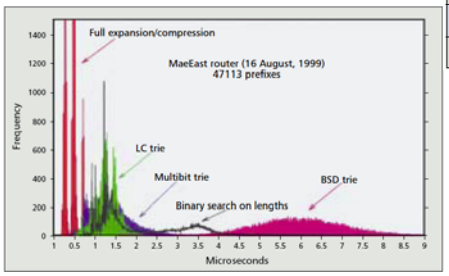




Summary

- Many different algorithms
- Important performance tradeoffs
  - Speed (# of memory references)
  - Space (size of data structure)

Scheme	Worst case lookup	Update	Memory
Binary trie	$O(W)$	$O(W)$	$O(NW)$
Path-compressed tries	$O(W)$	$O(W)$	$O(N)$
k-stride multibit trie	$O(W/k)$	$O(W/k + 2^k)$	$O(2^kNW/k)$
LC trie	$O(W/k)$	–	$O(2^kNW/k)$
Lulea trie	$O(W/k)$	–	$O(2^kNW/k)$
Full expansion/compression	3	–	$O(2^k + N^2)$
Binary search on prefix lengths	$O(\log_2 W)$	$O(N\log_2 W)$	$O(\log_2 W)$
Binary range search	$O(\log_2 N)$	$O(N)$	$O(N)$
Multiway range search	$O(\log_2 N)$	$O(N)$	$O(N)$
Multiway range trees	$O(\log_2 N)$	$O(k\log_k N)$	$O(Nk\log_k N)$



Homework

- Read
  - Pankaj Gupta and Nick McKeown, “Algorithms for packet classification,” *IEEE Network*, vol. 15, no. 2, pp. 24–32, Mar. 2001.
- SPARK
  - Assessment quiz