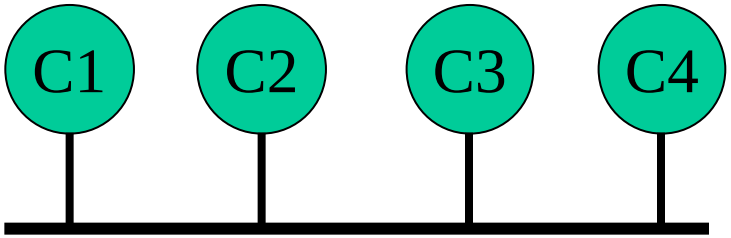
Longest Prefix Matching: Applying Algorithms to Networking

Ketan Mayer-Patel

October 11, 2005

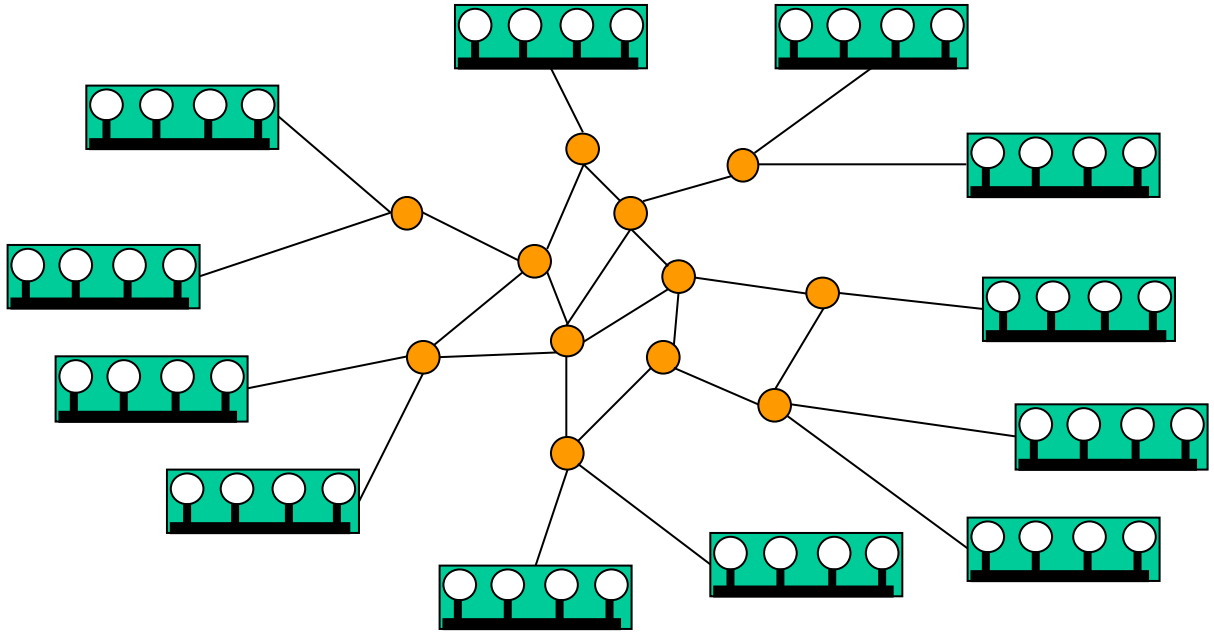
What's a network?

- A collection of computers connected together.



What's an internetwork?

- A collection of networks connected together.



Routing and Forwarding

- So, how does a message get from computer A, to computer B?
  - Think of a letter.
  - What do you need to get a letter from point A to point B?
    - Address
  - What makes up an address?
    - Street and number
    - City, State, and ZipCode.

Routing and Forwarding

- We can think of the letter being delivered in two parts:
  - First, we get it to the right zipcode.
  - Then, we get it to the right house.
- This happens for computer networks as well.
  - First we get it to the right network.
  - Then we get it to the right machine.

IP Addresses

- A computer's address on the Internet:
  - 32 bits long
    - Typically written in "dot notation"
    - 152.2.131.156
  - It has two parts:
    - Network address
    - Host address
  - Number of bits used for each part is variable.

IP Addresses

- To understand an IP address completely, need to know:
  - 32-bit address
  - How many bits are for the network.
- Example: 152.2.131.156
- 10011000000000101000001110011100

Network Number Host Number

Forwarding

- In the middle of the Internet, only the network part of an address matters.
- Message comes in some incoming link.
- Need to decide on which outgoing link to send the message on.
- This is called hop-by-hop forwarding.

Routing Table

- To do forwarding:
  - Build a table for all network numbers.
    - Table associated net number with outgoing link.
  - When message arrives, look up network number, find associated link, send message.
- Building the routing table is a different problem.

The Problem with Scale

- What do we care about?
  - Speed.
    - Want to make forwarding look up as fast as possible.
  - Space
    - The number of network number is quite large.
    - If we have to have a line in the table for every possible network number, then each router would need a large amount of memory.
    - Also affects speed.

Longest Prefix Routing

- Insight: we can compress a lookup table by dealing with ranges instead of individual network numbers.

Longest Prefix Example

000	A
001	A
010	A
011	B
100	A
101	B
110	B
111	B

00*	A, 2
010	A, 3
011	B, 3
100	A, 3
101	B, 3
11*	B, 2

0*	A, 1
011	B, 3
1*	B, 1
100	A, 3

*	A, 0
011	B, 3
1*	B, 1
100	A, 3

Problem Recap

- Given:
  - A set of prefixes.
  - Length of each prefix.
  - An address to match.
- Want:
  - Longest matching prefix.

Linear Brute Search

- Examine each entry.
- Remember the longest match.
- Time for adding an entry?
  - O(1)
- Time for deleting an entry?
  - O(n)
- Avg. time for lookup?
  - O(n/2) ~ O(n)
- Worst case lookup?
  - O(n)
- Space needed?
  - O(n)

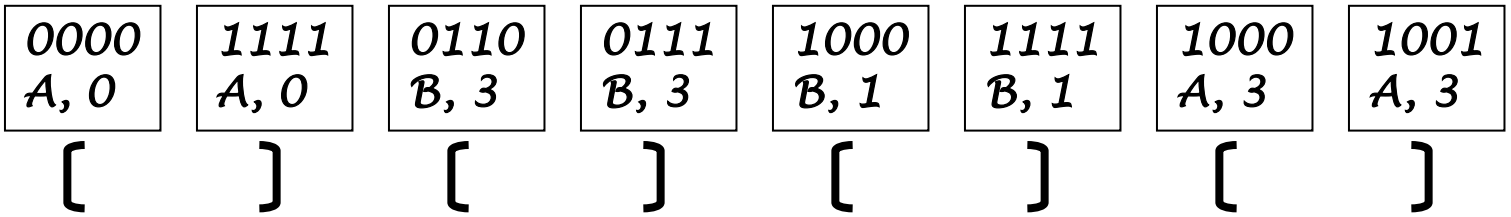
Sorted Ranges

- From each table entry, generate two “markers”.
  - Each marker is one bit longer than longest.
  - Left marker created by filling in the rest with 0.
  - Right marker created by filling in the rest with 1.
  - Associate prefix length and prefix with markers.
  - Sort markers (break ties with length).

Sorted Range Example

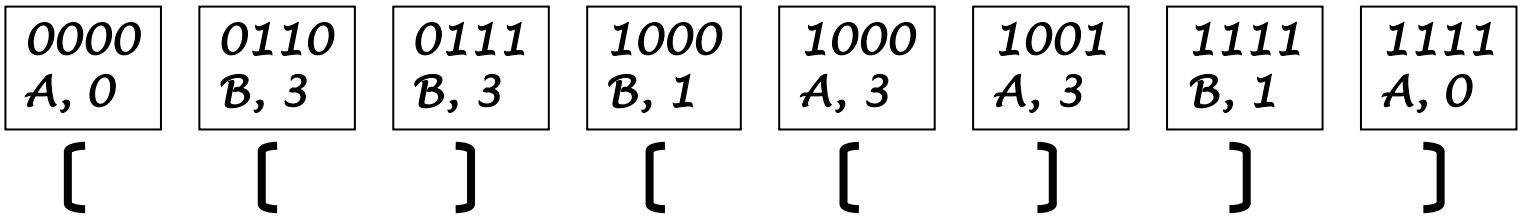
*	A, 0
011	B, 3
1*	B, 1
100	A, 3

Markers have 4 bits.



Sorted Range Example

- Once sorted we can use binary search to do lookup.



Sorted Range Performance

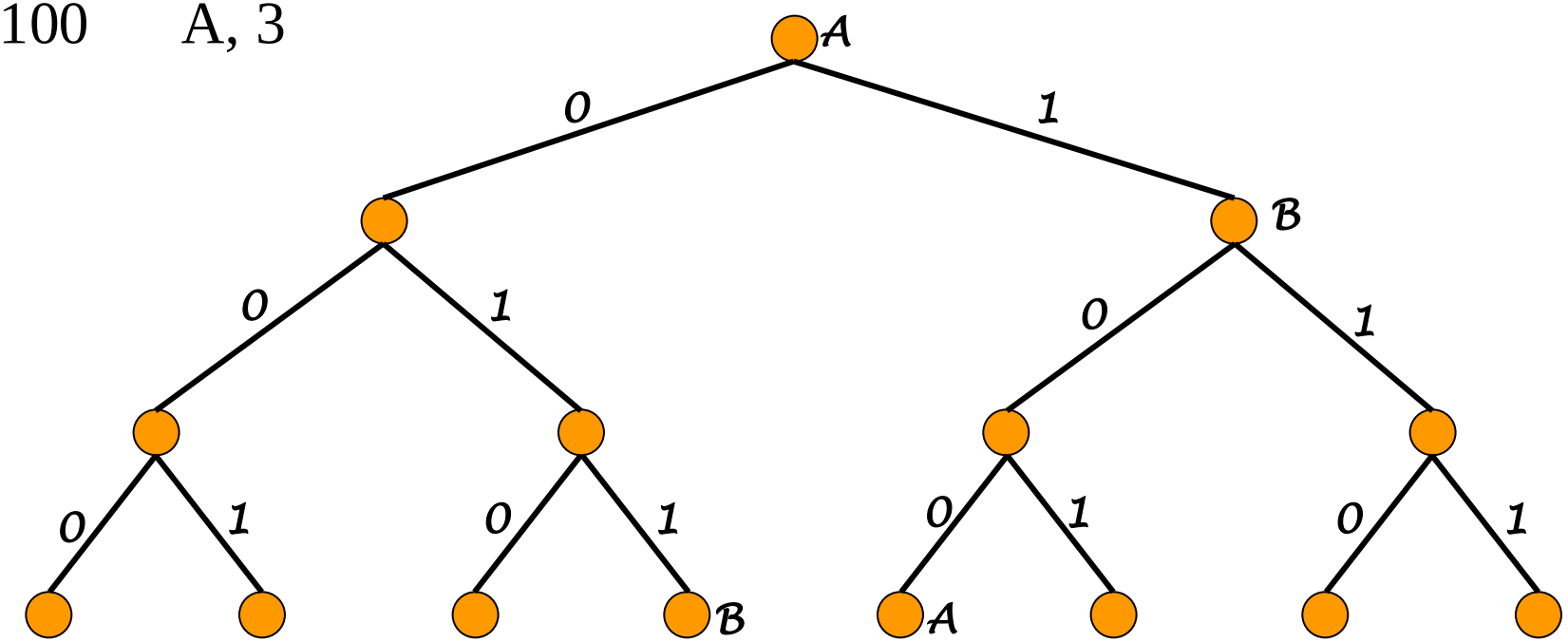
- Time for adding an entry?
  - O(n log n)
- Time for deleting an entry?
  - O(n log n)
- Time for lookup?
  - O(log n)
- Space needed?
  - O(2n)

Radix Tree

- Popular in actual routers.
- Build a binary tree.
- Each level of the tree is indexed by next bit.
- Only build as much of the tree as needed for all of the entries.
- Associate link with each node that matches a prefix in the tree.

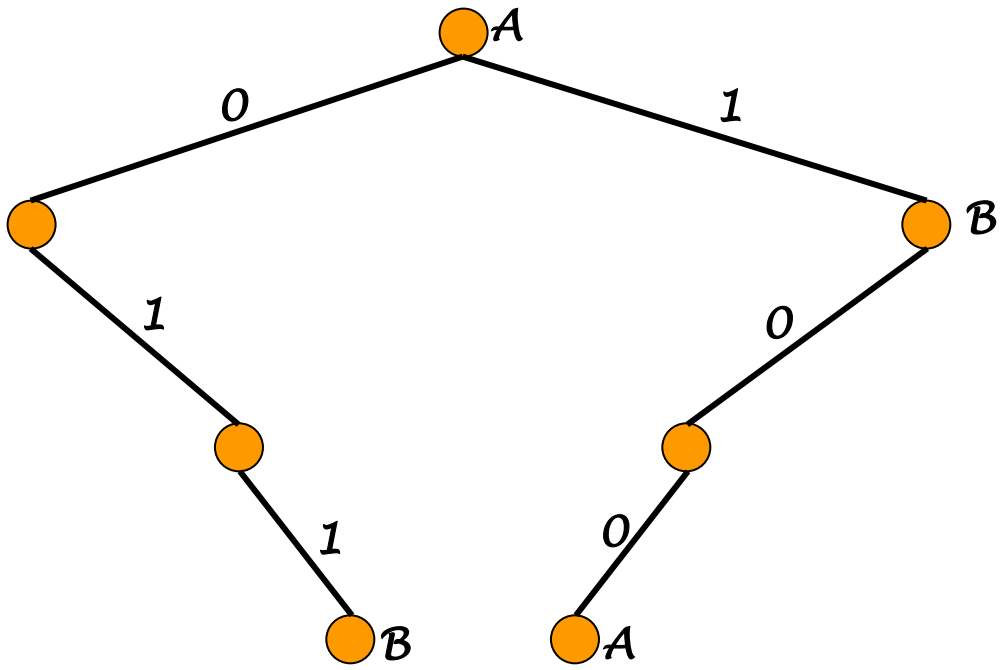
Radix Tree Example

*	A, 0
011	B, 3
1*	B, 1
100	A, 3



Radix Tree Example

*	A, 0
011	B, 3
1*	B, 1
100	A, 3



Radix Tree Performance

- Time for adding an entry?
  - O(entry length) = O(log e)
- Time for deleting an entry?
  - O(log e)
- Time for lookup?
  - O(log e)
- Space needed?
  - $O(2^{\log n+1})$ => Worst case, usually much better.

One more question.

- Still have the problem of building the prefix table to begin with.
- Which of these data structures and algorithms can help us with that?
  - Radix tree.