cs 591B1: ALGORITHMIC ASPECTS OF COMPUTER NETWORKS **Spring 2002**

Lecture 10 — April 8, 2002

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Weichao Ma*

In today's lecture we discussed the basics of *Routing with a Clue* paper. As the authors argue, adding a 5-bit "clue" in the IP header will speedup the IP-lookup in a factor of 10. This paper along with a few others we will be studying are examples of deterministic papers, in which people not only care about asymptotic performance, but are also interested in improvements even of a constant factor in the practical experiments.

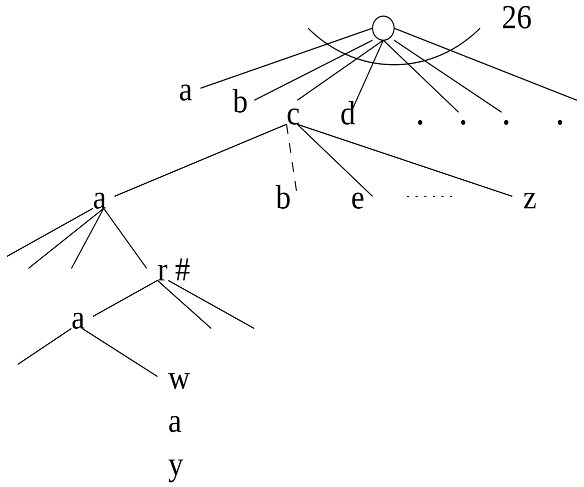
10.1 IP Lookup Basics

In current internet, each router maintains a routing table which consists of IP prefixes and corresponding next hops. When a packet arrives, the IP router looks up in its routing table for the **longest** prefix that matches the destination address of the packet. Since the size of the routing table can be huge, how to do fast IP lookup is important in order to achieve scalability.

A naive approach is to do sequential search while putting the popular entries at the top. However, linear search as well as binary search and hashing schemes are all relatively expensive. Prior to 1996, the state of art was to use **Trie** to store the entries of the routing table.

10.1.1 An Example of Trie

A *trie* (from re**trie**val), is a multi-way tree structure useful for storing strings over an alphabet. One example is to use trie to store dictionaries. The idea is that all strings sharing a common stem or prefix hang off a common node. Since the strings are words over {a..z}, a node has at most 26 children. The elements in a string can be recovered in a scan from the root to the leaf that ends a string. All strings in the trie can be recovered by a depth-first scan of the tree.



Note:

- 1 memory access is needed for each symbol in a word.
- Not all vertices in the tree represent the end of words, those that do are specifically marked so. ("#" is used to mark such vertices in the figure above)
- Any unmarked vertex in the tree that has no marked decendants is removed from the tree. For example, in English, no words start with prefix "cb", therefore, the branch "b" hang off from "c" is removed.

- In a common implementation of the trie data structure, called *Patricia*, all of the internal unmarkd vertices that have only one child, are collapsed together to save memory access. For example, in the "caraway", the part "way" is collapsed because no other words have prefix "caraw".

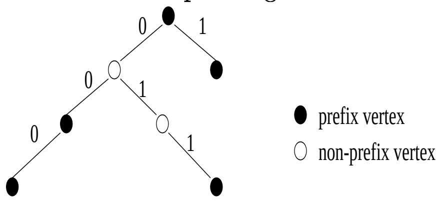
10.1.2 Trie used in routing table

Trie can be also used to store IP prefixes in the rouing table. The root of tree represents the empty string. Each edge going to the left from a vertex represents a 0 and an edge going to the right a 1. The binary string associated with a vertex in the tree is the sequence of bits on the edges along the path from the root of the tree to that vertex.

Routing table

Prefix	NexHop
*	B
00*	C
000*	B
011*	D
1*	A

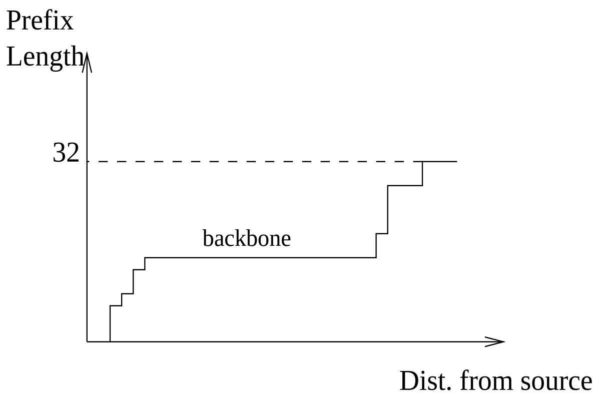
Coresponding Trie

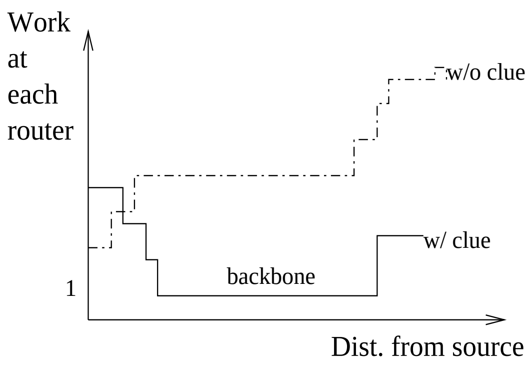


Given a IP destination, the router can traverse the Trie until there is no branch. e.g. Given 00111100, the router will stop traversing the tree when reached the second solid node in the left branch, and return C as the next hop. To avoid backtracking, the router needs to remember the last solid node it encounters. e.g. Given 010010, the router needs to backtrack when reaches at the node 01. If it remembers the last solid node encountered is empty string, then next hop B will returned immediately, since * is the longest prefix in this case. Note that *Patricia* implementation is especially important in storing prefixes, because the entries in routing tables normally share significant portion of common parts. For example, if a routing table has 2 entries: 128.197.9.* and 128.197.7.*, then clearly 128.197 needs to be compressed to save memory access.

10.2 Distributed IP Lookups

One fundamental observation about routing table is that neighboring routers have very similar routing tables and thus in many cases the best matching prefix (BMP) found in one router is either also the BMP that is found in the next router or very close to it. Therefore, instead of searching the similar structures repeatedly, paper [1] suggests to piggyback a 5-bit clue on an IP packet to indicate the BMP computed at the previous router. Then the longest prefix match computation will be distributed among several routers along the IP packet path. This Distributed IP lookups scheme also divides the cost of processing a header among the routers along the packet path. Each router starts the IP lookup where its predecessor stopped. The gain of the scheme is clearly shown in the figure below, which is a speculative graph of the length of the packet best matching prefix along a path from the source to the destination and its derivative which depicts the expected amount of work.

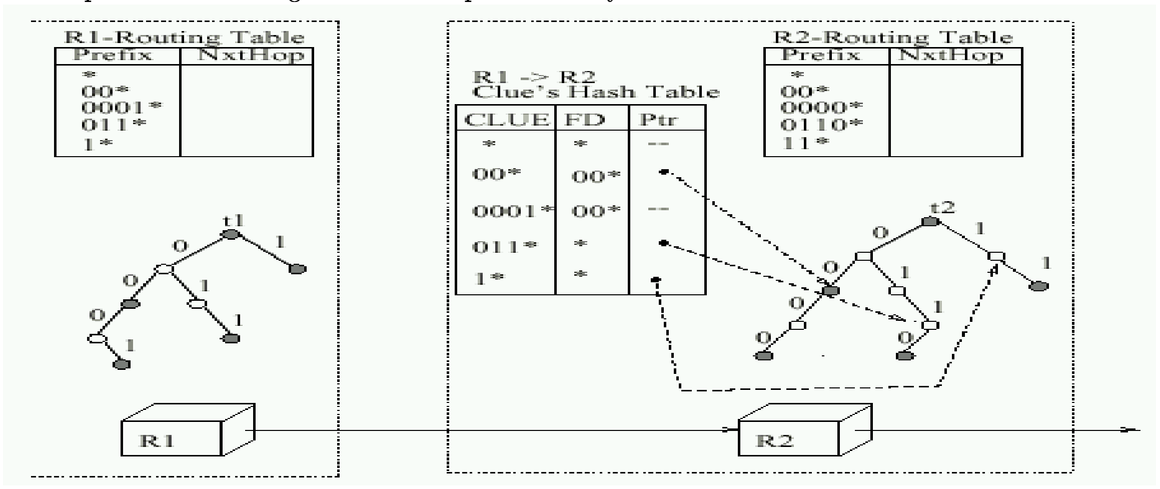




10.2.1 How to use a clue

The word *clue* stands for the longest prefix match that router R1 found for packet destination address and send to R2. Along with the destination address, the clue can be conveyed in 5 bits, which represents the number of bits used as a prefix. For example, for the address 125.7.19.123 and 5 bits of value of 16 represent the clue 125.7.

Each router maintains a hash table of all the clues it may receive from its neighbors (see the figure below). The information provided by a clue is essentially one of the two types: either that the clue directly implies the longest prefix match of the packet destination at this router, or that a search for a longer prefix should be performed starting in a location pointed at by that clue.



10.2.2 Potential problem

One problem is that clues may not map to any marked node in Trie. In that case, router can start the search over again just as in the case without the clue. Note that this situation is very rare, because routing tables usually get more specific nearer the destination, not less.

Bibliography

[1] Y. Afek, A. Bremler-Barr and S. Har-Peled. "*Routing with a Clue*," Transactions on Networking, 9(6), pp. 693-705, Dec. 2001. A prelimunary version appeared in ACM SIGCOMM '99.