Binary Search for Prefix Lookup

Yang Chen, Yajun Shi, Zeyang Ma, Xinyu Wang

Abstract

This report describes our implementation of binary search prefix lookup program. Our program takes a list of IP address and performs binary search to match each prefix to its designated next-hop address. The program is able to perform prefix lookup more efficiently than brute-force prefix matching.

1. Introduction

Prefix matching has always been a great challenge in the field of computer networking. Typically, routers are responsible for matching the destination of an incoming package to its corresponding prefix to determine its next-hop address. Due to large package flow and speed requirement for routers. Efficient algorithms are needed for routing in a real-world situation. One simple solution is to build a prefix table and perform a linear search matching for each IP address. However, such method fails to handle large incoming flow of packages. Binary prefix lookup matching has been introduced to solve the problem for decades. We implement a C++ program to perform binary search on prefix matching. The report is organized as follows:
Section 2 describes the overall structure and functionality of the algorithm.
Section 3 describes the data we use for building lookup table and the data for testing.
Section 4 evaluates the performance and limitation of the algorithm.

2. Structure and Functionality

Our program first tries to build a lookup table with a given list of IP addresses and their corresponding next-hop destination. We convert each IP address into an integer form for convenience. For example, an IP address 192.168.100.100 is converted to 192168100100. Then, each IP address is converted to two IP addresses which indicate its low range and upper range according to its prefix length. For example, 192.168.100.100/24 is transformed to 192.168.100.0 and 192.168.100.255. if we have another IP address which lies in this range, we know that the destination of this incoming package matches the prefix. The lookup table we build is of the form that each row has an IP address, and two next-hop addresses. If the destination of an incoming package is greater than the IP address in a row, we set the next-hop to be one of the next-hop addresses. If it equals the IP in this

					>	=
1)	A	0	0	0	1	1
2)	A	B	0	0	2	2
3)	A	B	C	0	3	3
4)	A	B	C	D	3	4
	A	B	C	F	2	3
	A	B	F	F	1	2
	A	F	F	F	-	1

Figure 1 From CS216 Lecture: prefixlookups George, Varghese

row, we set the next hop to be other address in this row. An example table is shown above. After all IP addresses are converted to ranges, we sort the range to build a final lookup table.

For each IP address which we are trying to match, we start its comparison at the mid of our table. If the IP equals the IP of current row, set its next hop to equal-hop column in the table. If not, continue binary search with new boundary. The program goes through each input IP address and stops until all destinations are found.

3. Data

We use a randomly generated BGP table to build our lookup table. The data is given by Prof. Varghese George. The dataset contains millions of IP addresses and their corresponding destinations. The test data we use is one million randomly generated IP address.

4. Performance and Evaluation

In the training bgptable, there are 723726 prefixes, each has its corresponding nexthop. The cost of time to build a lookup table is 2.05845 seconds. Performing binary search on one million IP addresses takes 0.33357 seconds.

The size of the final lookup table is 1216892, where each node contain 3 element: 32 bits ip address, 32 bits nexthop address for greater ip and 32 bits nexthop address for equal ip, thus the memory cost of storing the lookup table is 1216892x32x32 = 1246097408 bits.

We observe two problems with this implementation. First, binary search for prefix matching is not fast enough for wire-speed networking. A router with binary search for prefix matching could not handle huge flow of networking packages due to its speed. Packages could be dropped. Second, insertion is extremely slow. If we want to insert another IP address and its destination into our lookup table, it takes more than 2 seconds to reorganize the lookup table. Improvements are needed to further expedite the lookup process and building a lookup table.

5. Reference

George Varghese. 2004. Network Algorithmics,: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.