Study of IP Address Lookup Algorithms

Subhashini Venugopalan
Department of Computer Science and Engineering, IIT Madras

Abstract

The following is a report of the work done in implementing three longest prefix matching algorithms for IP packet address look-up: (i)Binary Trie (ii) Multi-bit stride based trie (3 levels, 8-bit strides) and (iii) Linear length search based on Hash table. Section 1 explains the input file details and the usage. Section 2 explains the output format. In section 3, we show the results of the trials. Our observations and conlusions are presented in sections 4 and 5 respectively. Implementation resources that were used are mentioned in section 6. The implementations are based on ideas presented in [1].

1 Inputs

1.1 File format of the two input files

1.1.1 PrefixFile: Containing Prefix addresses

Each line of the file contains two fields: An IP address field (of 24 bits) in dotted decimal format and a field specifying the length of the address to be considered. Eg:

17.0.0 8

20.180.0 14

139.192.0 11

1.1.2 IP Address file: Containing input IP addresses for look-up

This file contains the IP addresses that need to be looked up in 24-bit dotted decimal format.

17.130.40

20.182.37

139.182.9

1.2 Usage

```
$./bmp.o -p [Prefixfile] -i [IpAddrFile] -o [OPfile] -W 24 -a [B|M|H]
```
Options:

- `-p <prefix file name> default file name is ‘prefixfile’`
- `-i <input ip address file> considers default as ‘inputfile’`
- `-o <output file> by default it writes output to ‘outputfile’`
- `-a <algorithm type> B for Binary Trie, M for multi-stride trie, H for linear search on hash table.`

1.3 Additional

The README file contains exact details of execution.

2 Outputs

The output file contains the outputs in the following format:

Output Format

Input IP Address	**Best Matched Prefix**	**Time* (μs)**	**All Matched Prefixes**
Dotted decimal IP address	Longest prefix match.	(1667 clock ticks)	Prefixes separated by tabs

*Time is actually measured in clock tics within the program. It was converted to microsec (within the program itself) based on the processor it was running on.

3 Results

3.1 Comparison Table

The following table presents the results of the trial runs on random, valid input prefixes and IP addresses.

Comparison of the performance of the three algorithms

No. IP Addresses	**No. of Prefixes**	**Average Look-Up time* ($\sim \mu$s)**		
		Binary Trie	**Multi-bit Stride**	**Hash Table**
51	36	1.2562	0.4922	7.2845
101	66	1.1654	0.7744	7.3802
201	137	1.1917	0.8573	7.4564
989	688	1.3354	0.8840	8.5465
4744	3299	1.3498	0.8241	8.9957
46689	40008	1.4682	0.7070	10.2214
73570	66686	1.4572	0.6801	10.7506
105265	99754	1.4587	0.6679	10.3427
193421	199815	1.4887	0.6706	10.3771
275708	300213	1.5004	0.6604	10.0808

*Time is actually measured in clock tics, it was converted to microsec within the program itself (1667tics = 1μs) based on the processor it was running on.

3.2 Notes

3.2.1 Verification of Results

The results were verified by comparing the best matched prefix using all the three algorithms. i.e. The best matched prefix had to be the same for a given input IP address irrespective of the algorithm that was chosen. Random brute force checks were also made to check for inconsistency in all three algorithms.

3.2.2 Time computation details

The actual time was computed in clock tics using 'rdtsc'. This allows us to get the time stamp counter values. Difference between the time stamp counter at two different times gives us the number of CPU cycles or clock tics that have taken place in that duration. The formula for conversion to time in seconds is:

$$\text{Number of clock tics} \times \text{CPU frequency} = \text{time in seconds}$$

3.2.3 Sources of error

1. The time is only an approximate value in microseconds. However, the clock tics would be a more consistent metric across different CPUs and would actually be a better estimate for comparison than considering time.
2. Hash Table matches are checked from length=1 to 24. Hence, the time taken is significantly larger than starting with length 8 directly. This was primarily done to present all matched prefixes for an IP.

3.3 Graph

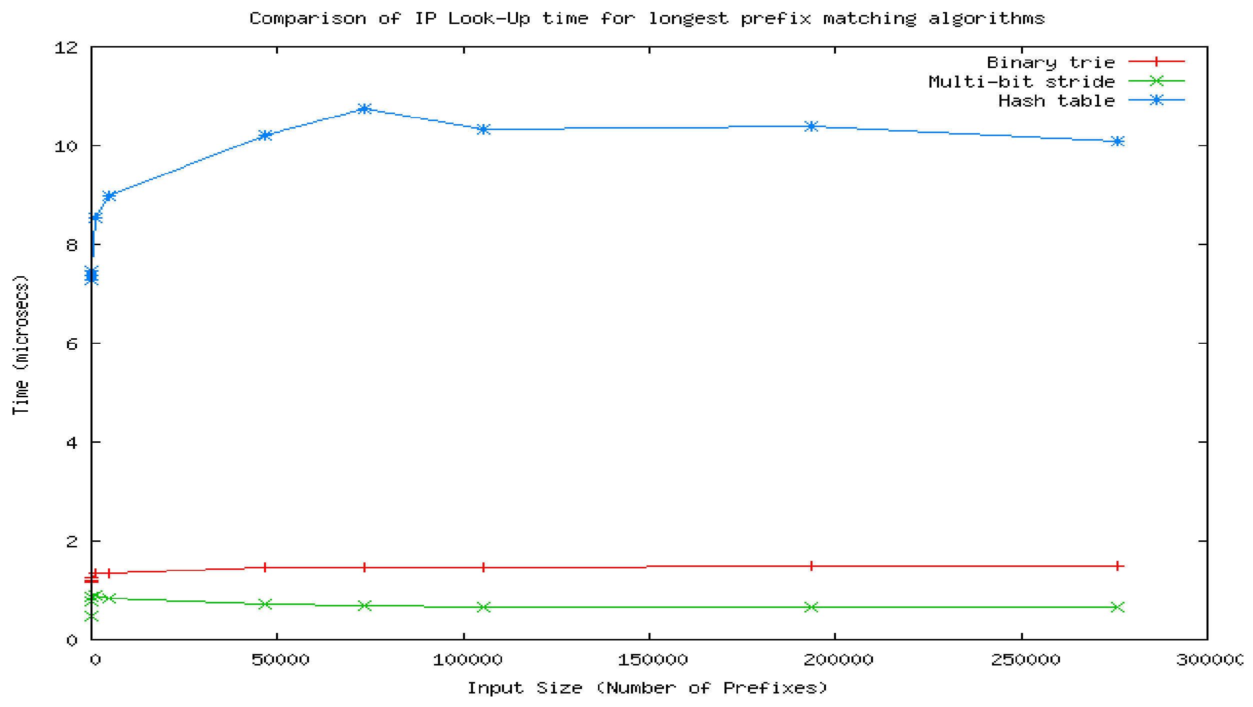

Figure 1: Graph showing the comparison of the look-up approaches.

4 Observations

1. Multi-bit stride is the fastest of the three algorithms. It effectively takes less than $1\mu s$ to give the best prefix-match even when the number of prefix addresses are as large as 300,000.
2. Binary Trie also performs consistently well with small and large number of prefix values. It takes about $1.5\mu s$ on an average, and is much better than Hash Tables.
3. Hash Table takes the longest time of the three algorithms. It takes close to $6\mu s$ when the prefixes are fewer and takes close to $10\mu s$ for each look-up when the prefix file is extremely large. [But we are looking for all matches of the input so the time is considerably increased].
4. Comparing the three algorithm with respect to memory, the multi-bit stride consumes the most memory. In the trial runs it required significantly larger amount of memory than the other two for a prefix file of size 300,000. Although, theoretically the hash table's memory requirement is less, it occupies more space than the binary trie due to the constraints of perfect hashing. Of the three structures, the binary trie consumes the least memory, then comes the hash table and finally the multi-bit stride trie.

5 Conclusion

1. Using multi-bit strides is the fastest of the three IP look-up approaches.
2. Binary Tries appear to be the most suitable structure for IP look-up both in terms of memory and time. Although it takes slightly longer than multi-bit stride, it is a lot more memory efficient than multi-bit stride based tries.
3. The performance of hash tables is comaparitively poor, it takes much longer than the trie approach.

6 Implementation Sources

1. For hashing, the STL template hash-map was used to create the hash tables and the default hash function was considered for hashing the keys.
2. The trials were done on a 2GB, 2.6GHz, 64-bit CPU in the systems. But the operating frequency at the time of trial was 1667 Mhz.

7 References

1. M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, “Survey and taxonomy of IP address lookup algorithms,” *IEEE Network Mag.*, vol. 15, no. 2, pp. 8-23, Mar.-Apr. 2001