Scalable Best Matching Prefix Lookups

Marcel Waldvogel George Varghese Jonathan Turner Bernhard Plattner
Washington University in St. Louis and ETH Zürich

Abstract

All global routing protocols use hierarchies to allow scaling to a world wide community while keeping the routing database size manageable. Databases of variable length prefixes are a powerful tool for providing this in a flexible manner, but require a Longest Prefix Matching algorithm. In this paper, we report a fundamentally new solution that is both algorithmically interesting and practical.

Our scheme is based on doing binary search on hash tables organized by prefix lengths, and scales very well as address and routing table sizes increase: independent of the table size, it requires a worst case time of $\log_2(address\ bits)$ hash lookups. With the current Internet Protocol, which uses 32 bit addresses, at most 5 hash lookups are needed; for the upcoming 128 bit addresses of the next generation Internet Protocol (IPv6), 7 lookups suffice. Several refinements, including specializing the Binary Search with every match, considerably reduce the average number of hash search steps to less than 2.

1 The Lookup Problem

Messages on the Internet are ferried by a system of automated post offices, called routers, that are interconnected by communication links. For each message received from any of its input links, the router decides which of its outgoing links it will forward this message to based on the destination address encoded in the packet.

Because of the flexible hierarchies needed to avoid database size explosion, this forwarding decision cannot be done by a simple exact match but by Longest Prefix Matching. To keep up with faster links, each lookup should take a few hundred nanoseconds.

Internet addresses are 32 bit strings, and each prefix length can be vary from 1 to 32 bits. Most current implementations are based on bitwise branching tries. Thus, each possible length is tested sequentially, requiring up to the number of address bits memory accesses; this is too slow for high-speed routers.

2 Binary Search on Prefix Lengths

Instead of searching for prefixes one length at a time, the new scheme divides the problem in half at each stage. Instead of asking if the longest prefix is at length 1, length 2, etc. (the naive way), the new scheme does binary search on the *set* of possible prefix lengths.

Suppose we could start by asking the question "Given our prefix database and the current message's destination address, does the longest prefix matching this address have a length in the range from $17 \ldots 32$". If the answer is no, we know the answer must be in the range $1 \ldots 16$; so we can cut down the range further by asking whether the longest prefix length is in the range $9 \ldots 16$. If we continue in the same way, we will do binary search on prefix lengths and find the correct length prefix in time that is logarithmic in the address length.

However, there are some pitfalls to be avoided. They can be demonstrated using a small database with U.S. destinations organized in Country.State.City format, containing the prefixes USA.* (which is stored in the length 1 table); USA.CA.* (length 2), and USA.MO.SL (length 3).

First, suppose we get a packet destined for USA.CA.LA. We don't know which length the correct prefix has, so we start with the middle (length 2) database, extract the first 2 symbols and get a match with USA.CA.*. Thus, we can immediately discard the length 1 table (as we have already found a longer match) but we certainly need to search the length 3 table as it may have a longer match. This gives us a simple rule: we start by looking for a match (using say hashing) in the database corresponding to the middle length of our current set. If we get a match, we try lengths longer than the probe; if we do not, we try shorter prefixes, halving the range of prefix lengths with each step.

To make this rule work in all cases (e.g. when searching for USA.MO.SL) we need to introduce a signpost at length 2. These are pseudo-prefixes, pointing the search routine towards the longer prefixes. Without this signpost (USA.MO.*), no match would be found at length 2, and shorter prefixes would be searched, missing the USA.MO.SL entry.

Second, consider a search for the destination address USA.MO.KC. The new signpost in the length 2 table will lead us to length 3, where we won't get a match. The real longest prefix of USA.MO.KC is USA.* in the length 1 table. This problem can be solved without inefficient backtracking by storing a reference to the correct prefix with the signpost. Still, the final search procedure is simple and fast (extremely small constant factors).

3 Conclusions

This scheme has been described in a paper that was presented at the ACM Annual Networking conference (SIGCOMM) in Cannes, France. More details, further refinements (of which the most significant is a way of specializing the binary search with every search step by compactly encoding search trees), and references to prior work can be found in [1].

Internet routers soon will have to look up 128 bit addresses instead of 32 bit ones, as the Internet prepares to retrofit to serve a global community of users and user devices, where even your toaster and Nike shoes might have Internet addresses. Our binary search is especially attractive because of its scalability when dealing with next generation 128 bit addresses. A number of companies have finalized licensing agreements to use this scheme in products that should be introduced in 1998.

References

[1] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. High Speed Scalable IP Lookups. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997. Also available from http://www.tik.ee.ethz.ch/~mwa/