A Search Algorithm for Bloom Filter Based Packet Forwarding

Maksim Yegorov
Prof. Minseok Kwon, Advisor

Objective
Longest prefix matching has long been the bottleneck of the Bloom filter-based solutions for packet forwarding implemented in software. We propose a search algorithm to match a destination IP address against a compact representation of the FIB table in CPU cache on general purpose hardware with an average performance target of $O(\log n)$ for an $n$-bit address.

Context
To match an incoming packet to an outgoing interface link, the router searches for an approximate match between the destination address and the prefixes stored in its forwarding table. Because of the sheer throughput demand, the lookup has traditionally been performed in hardware. A software solution for programmable routers that can compete with hardware on speed would be welcome.

Bloom Filter Approach
The Bloom filter (BF) data structure was originally used by Dharmapurikar et al. for parallel look up implemented in hardware. By contrast, the software implementations have traditionally used linear search to find the longest matching prefix.

Experiments
We compare the relative performance of the linear and guided search schemes on core router BGP tables from the University of Oregon Route Views Project.

<table>
<thead>
<tr>
<th>traffic</th>
<th>metric</th>
<th>linear</th>
<th>guided</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>bit lookup</td>
<td>48.3</td>
<td>22.1</td>
</tr>
<tr>
<td></td>
<td>hashing</td>
<td>20.0</td>
<td>11.2</td>
</tr>
<tr>
<td>by prefix address space</td>
<td>bit lookup</td>
<td>48.0</td>
<td>22.3</td>
</tr>
<tr>
<td></td>
<td>hashing</td>
<td>17.4</td>
<td>10.3</td>
</tr>
<tr>
<td>by prefix frequency</td>
<td>bit lookup</td>
<td>33.3</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>hashing</td>
<td>10.3</td>
<td>8.6</td>
</tr>
</tbody>
</table>

Summary
1. We propose a search algorithm for finding the longest matching prefix in a Bloom filter with $O(\log n)$ average case performance;
2. We suggest a scheme to eliminate expensive off-chip hash table lookups by encoding the next hop information in a guided Bloom filter that can fit in L3 cache and lends itself to the logarithmic time search.

References

Guided Search Algorithm
For each packet, the guided search scales the full height of the binary search tree until it reaches a leaf, then decodes the best matching prefix from the most recent hash match, and finally verifies the match using remaining hash functions on the bmp itself. Occasionally, it will default to linear search over the lower prefix lengths.

Insertion algorithm for guided search