Abstract—In networking systems, IP lookup plays crucial role in forwarding incoming IP packets. IP lookup is performed by checking longest prefix match of destination address against prefixes stored in router’s routing table. To reduce the time required to correctly forward an incoming packet, one needs to improve performance of IP lookup mechanism. Although Bloom filter[1] and cuckoo filter[2] are used in contemporary networking systems to reduce time required for IP lookup, they do not leverage information about IP prefix distribution in routing table. We introduce an enhanced cuckoo filter called length aware cuckoo filter which uses double insertion for unpopular items to reduce false positive rate.

Keywords— IP lookup, Bloom filters, Cuckoo filter

I. INTRODUCTION

Routing systems dynamically add/delete IP prefixes to/from routing table. These IP prefixes are used by IP lookup mechanism to forward incoming IP packets. Bloom filters [1] have been previously used to speed up IP lookup. Bloom filter is space optimized data structure used for set membership test. Disadvantage of Bloom filter is it does not support dynamic deletion of items. Some variants of Bloom filter address this limitation but at cost of extra space.

Cuckoo filter [2] outperforms Bloom filter and it’s variants by supporting dynamic deletion of item without need to reconstruct the filter. False positive is commonly associated with Bloom and cuckoo filter. A false positive is an error in filter when a nonexistent item searched for is mistakenly found. Performance of Bloom and cuckoo filter is based on their false positive rate.

This paper introduces a new filter called length aware cuckoo filter (LACF) which aims to reduce false positive rate. LACF is best suited in situations where distribution of data set is known in advance. In routing table, prefixes can be classified as popular or unpopular based on their length. Popularity of a prefix denotes the likelihood of it’s occurrence in routing table. Unlike Bloom and cuckoo filter, LACF leverages such classification and inserts unpopular prefixes twice into filter called as double insertion. Set membership test in LACF for unpopular IP prefix needs at least two lookups which helps reduce false positive rate for unpopular prefixes while for popular prefixes it remains the same. [3] has previously demonstrated decrease in false positive rate for Bloom filter when number of hash function used depended upon popularity of IP prefix length. LACF uses more number of hash functions for unpopular prefixes compared to popular ones.

This paper is divided into following sections. Section II talks about Bloom filter and cuckoo filter in detail. Section III focuses on detailed implementation of LACF. Section IV explains integration of LACF in Linux kernel. Sections V demonstrates experiments performed on LACF and showcases results obtained. Section VI gives final remarks on LACF and it’s performance.

II. RELATED WORK

A. BLOOM FILTER

Bloom filter is space optimized data structure used for set membership test. It is represented as m-bit array initialized to 0. An item is inserted in filter by setting k different positions to 1. These positions are generated by using k different hash functions. Output of each hash function produces a position in filter and corresponding bit is set to 1.

Lookup for an item is performed by checking bits at positions generated by same hash functions used for insertion. If bit at each position to be searched for is set to 1, then lookup is successful else unsuccessful. Due to constraints on space allocated to filter, output of hash functions for multiple items may produce same positions and set bits to 1 which introduces false positives.

B. CUCKOO FILTER

Cuckoo filter[2] supports dynamic deletion of items as well as outperforms standard Bloom filter. Instead of setting bits in filter to 1, cuckoo filter uses 8-bit fingerprint generated from given item to be inserted. Two different positions are calculated within filter by using different hash functions. If either position is empty then fingerprint is inserted in empty one otherwise a position out of two is selected and the existing fingerprint is kicked out. The new fingerprint is then...
successfully inserted. Same procedure is followed to find a free position within filter for fingerprint that got kicked out. If no such position can be found then filter is said to be full.

Lookup in cuckoo filter is performed by generating two positions same as insertion. If fingerprint is present in either of these positions then lookup is successful else unsuccessful. Cuckoo filter allows multiple insertion of similar fingerprints which helps support dynamic deletion of items without need to worry about number of similar fingerprints inserted.

Fig. 2. Cuckoo Filter

\[ a_1 = h_1(X) \]
\[ a_2 = a_1 \text{xor} h_2(fp) \]

fp represents 8-bit fingerprint of an item X. \( a_1 \) and \( a_2 \) are positions generated by using hash functions \( h_1 \) and \( h_2 \).

III. LENGTH AWARE CUCKOO FILTER

Length aware cuckoo filter (LACF) is enhanced cuckoo filter aims to reduce false positive rate. LACF is tailored for applications where the distribution of dataset is known in advance. LACF divides dataset into popular and unpopular items. It leverages information about popularity of items and inserts unpopular item twice in filter. By doing so, LACF decreases false positive rate associated with set membership test. As opposed to cuckoo filter, more number of hash functions are used in case of a unpopular item.

Lookup for a popular item is performed in similar fashion as cuckoo filter. For an unpopular item, lookup is performed for two positions. Due to increased number of lookups in case of unpopular item, LACF guarantees to yield lower false positive rate than cuckoo filter. The double insertion is shown in Fig. 3 using hash function sets \((h_1, h_2)\) and \((h_{1b}, h_{2b})\).

For unpopular item, positions \(a_{1b}\) and \(a_{1b}\) are calculated for double insertion.

A. IP PREFIX DISTRIBUTION

IPv4 BGP table analysis\[4\] indicates that around 55% of the IP prefixes are of length 24 and remaining 45% are from 13 to 23. IP lookup for incoming IP packet is done by longest prefix match (LPM) mechanism. In routing systems, LPM can become bottleneck and slow down lookup mechanism significantly. As distribution of IP prefixes is known in advance, it makes LACF a perfect candidate for IP lookup. When LACF is used with routing subsystem, IP prefixes which are unpopular are looked up twice to reduce false positive rate. This results in faster lookup.

Fig. 3. Length Aware Cuckoo Filter

Average prefix distribution

Fig. 4. IP Prefix Distribution

IV. LINUX KERNEL INTEGRATION

Routing subsystem in Linux kernel uses level compressed (LC) trie to store IP prefixes and corresponding forwarding information base (FIB). LC trie is also called as FIB table. Linux kernel has two routing tables, FIB LOCAL and FIB MAIN. IP prefixes are inserted into them depending configuration of route as local, link or global. FIB LOCAL stores routes configured as local and FIB MAIN stores routes configured as either link or global.

To forward an incoming IP packet, LC trie performs longest prefix matching by traversing the tree. An incoming IP packet is forwarded to default gateway if LPM is unsuccessful. Although LC trie is highly space optimized data structure, the drawback is it needs to perform backtracking in case of lookup failure. Performing backtracking is time consuming operation. To reduce time required for IP lookup in LC trie, LACF is integrated with existing routing subsystem.

A. ROUTE CONFIGURATION

When a route is successfully configured in either FIB LOCAL or FIB MAIN, prefix is inserted into LACF. Default gateway is first route entry configured into FIB MAIN. LACF keeps track of whether default gateway information has been configured or not. Although routing table is divided into two FIB tables, only one filter is used.
B. IP LOOKUP

LACF performs longest prefix match on each incoming IP packet by reducing length of IP address by one bit every time. If LPM is successful within LACF, the request is forwarded to LC trie which handles subsequent forwarding mechanism. Unsuccessful lookup is handled based on whether default gateway has been previously configured. First lookup in FIB MAIN table is for default gateway entry and corresponding FIB information is stored in a global variable. Error code -EAGAIN is returned if lookup was unsuccessful and default gateway entry was not previously configured. Otherwise, global variable containing FIB information of default gateway is returned. By doing so, we eliminate need for traversal and subsequent backtracking in LC trie which reduces time required for forwarding incoming IP packet. Fig. 5 demonstrates flow for both configuring new route as well as lookup for incoming IP packet.

C. ROUTE DELETION

Process of route deletion is straightforward. When routing subsystem successfully deletes a route and corresponding FIB information, fingerprint corresponding to that prefix is also removed from LACF.

V. RESULTS

Fig. 6, Fig. 7 and Fig. 8 shows comparison of standalone LACF and cuckoo filter for different distribution of popular and unpopular IP prefixes. Fig. 9 shows LACF and cuckoo filter comparison when they are integrated in Linux kernel. To measure their performances in Linux kernel, ping command was used to test how fast IP lookup was performed. 50% filter is occupied of popular prefixes and remaining 50% is occupied by unpopular prefixes which are doubly inserted.

VI. CONCLUSION

Results obtained from standalone implementation and Linux kernel integration of LACF show LACF has lower false positive rate than cuckoo filter. This proves LACF makes a perfect candidate for faster IP lookup.