Use of Cuckoo Filters with FD.io VPP for Software IPv6 Routing Lookup

Minseok Kwon  
Rochester Institute of Technology  
jmk@cs.rit.edu

Shailesh Vajpayee, Pragash Vijayaragavan, Arjun Dhuliya  
Rochester Institute of Technology  
{srv6224,pxv3620,amd5300}@rit.edu

John Marshall  
Cisco Systems, Inc.  
jwm@cisco.com

ABSTRACT
The filter technologies, e.g., Bloom filters, have been used for IP lookup for their compactness and efficiency. We investigate the performance of cuckoo filters with Cisco’s VPP (Vector Packet Processing) for IP lookup. We also introduce a variant called a length-aware cuckoo filter that treats incoming IP addresses discriminatively, and study its performance with VPP. As proof-of-concept, we implement cuckoo filters with VPP, and test them on both functions and performance with focus on the ip6-input node in VPP.

CCS CONCEPTS
• Networks → Network algorithms; Data path algorithms; Packet classification;

KEYWORDS
IP lookup, cuckoo filters, packet forwarding, software routers

ACM Reference format:
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION
For IP lookup, hashing with network prefixes as keys has widely been used at network routers due to its extremely short search time. With classless addressing, the challenge lies with finding the longest matching prefix. Tries [4] and filters, e.g., Bloom filters [3], have been adopted to tackle these challenges. Recently, tries using tree bitmap [2] have gained attention as they reduce search time with simple bitwise operations, whereas filters require hashing, which includes several bitwise operations, for multiple prefixes. The filters are, however, tiny enough to fit in on-chip memory with small memory footprint, and they allow parallel prefix matching with easy table updates. Another related technique is Cisco’s VPP, Vector Packet Processing [8], which processes a group of packets together instead of individually to utilize CPU instruction cache maximally to speed up the lookup. With the emergence of these new techniques, one yet unanswered question is how the lookup algorithms behave with VPP, more interestingly, in programmable routers. We focus on the filter techniques and their performance with VPP. A cuckoo filter (CF) [7] provides set-membership testing based on cuckoo hashing with low space requirement and false positive rates [7]. We also propose a modified cuckoo filter named a length-aware cuckoo filter, LACF in short, that uses network prefix length and its popularity to insert prefixes discriminatively [9]. The LACF can further reduce false positive rates, enabling faster IP lookup without additional storage requirement. We implement both the CF and LACF using the VPP library that runs on Intel’s DPDK (Data Plane Development Kit) adopted by FD.io (Fast data - Input/Output) [8]. Our system is 100% software-based, flexible, and programmable, making it easy to change the forwarding logic, but is still fast and high-throughput. We are in the process of testing and evaluating their performance on a simple network topology where one Cisco UCS server that generates packets is connected to another that serves as a router via 80Gbps Ethernet. Each server has Intel Xeon E5-2640 v4 @ 2.40GHz, 10-core with Hyper-Threading, 64GB RAM, 1TB SSD, 40GE NICs, and Intel DKDP-installed.

This is the first attempt to integrate cuckoo filters with VPP. We will compare cuckoo filters plus VPP with trie-based technologies, e.g., Poptrie [2], for IP lookup with respect to lookup and update time, and memory usage. We also plan to investigate use of VPP with P4 [6] for truly high-throughput software routers with user programmability.

2 ARCHITECTURE AND IMPLEMENTATION

Figure 1 gives a high-level view of our lookup engine with the filter and VPP technologies. A vector of packets are first passed to the packet processing graph that identifies which packets they are, e.g., IPv6 or MPLS, and then processes the packets accordingly. If a packet turns out to be IPv6 (since our focus is on IPv6), the packet is sent to the filter, either CF or LACF, for extracting and testing.
its prefixes. The output of the filter is a few prefixes that appear to be valid prefixes, but need to be checked against the actual routing table, because they may be false positives. Hence, these prefixes are searched in the routing table from long to short, and then the packet is forwarded to the next hop that is discovered from the entry with the longest matching prefix.

A cuckoo filter [7] (CF) is a highly efficient data structure that provides set-membership testing. An item is inserted into an empty slot out of the two places computed by two hash functions. If both are occupied, however, cuckoo hashing relocates one of the items to its alternate location, and inserts the new item in the slot that is now available after the relocation. If the alternate location is already occupied, the current item is displaced and re-inserted, and the process continues until an empty slot is found. We propose to develop an LACF [9] whose idea is to insert an unpopular element twice, and to ensure that the element is present at both places when searching, while inserting a popular one still only once. This reduces the false positive rate since most of the prefixes tested in lookup are unpopular ones and more hashes are used for those prefixes, and the low false positive rate facilitates to reduce lookup time even further. This, however, should not affect the rate for the other prefix lengths because only unpopular items are inserted redundantly, and its fraction is small.

VPP [8] helps elevate the performance of network switches and routers to extreme limits by processing an entire vector of packets. The vector processing helps utilize the instruction cache fully as the packets in the vector are processed together, as its cost is amortized over leading to more stable and high throughput even when the number of packets in the vector fluctuates. We integrate our CF and LACF implementations with VPP 17.04. The route entries need to be read and inserted into the filter from the FIB (Forwarding Information Base) in the VPP implementation. The filter is updated as the FIB is updated for addition, deletion and updates of entries. Instead of bihash used in the original VPP implementation, we use CRC32 with 16 bit fingerprints, which enhance the lookup speed of cuckoo filters dramatically with code optimization.

3 EVALUATION

We set up a testbed to measure the performance of cuckoo filters and length-aware cuckoo filters when they run with VPP. The testbed uses two Cisco UCS C240 M4 Rack Servers connected through 80Gbps network links. Each server has Intel Xeon 2.4GHz E5-2640 v4 processor with 10 cores (hyper-threaded), and 25MB (L3), 256KB (L2) cache, and 64GB DDR memory. This implies that it may even be possible to have the filter in L2 cache if the filter is sufficiently small. Also, each server is equipped with two Intel XL80 dual-port 40G QSFP plus network cards for communication links. As shown in Figure 2, one server serves as a packet generator that sends packets to the other server that processes and forwards the received packets. Those packets are transmitted only between the two servers, and do not travel outside of the testbed. The two servers are also connected to the regular network through which we can configure and control their settings and behaviors. For the packet generation, we use MoonGen [5], a scriptable high-speed packet generator that uses DPDK for packet processing. MoonGen can send 64KB packets at up to 10Gbps on a single core.

Figure 2: Illustration of the testing environment

We are in the process of measuring lookup rates to evaluate the performance. For now, we perform lookups by reading traffic from a file that contains randomly generated IPv6 addresses for preliminary testing. The routing table has about 800,000 route entries with a distribution of prefix lengths as shown in Figure 3(a). The routes with prefix lengths /120, /110, /64, /48, and /32 are significantly more popular than the other prefix lengths. In Figure 3(b), we present the lookup rates measured for VPP and VPP with CF as preliminary results. The x-axis represents the total number of lookups performed in the experiment changing from 20,000 to 600,000, and the y-axis denotes the total time in second for all the lookups. The lookup time for VPP with CF decreases in all the cases compared to VPP, e.g., approximately 31% decrease for 600,000 lookups. In VPP with CF, a significant portion of lookups are performed in the filter, which resides in cache due to its compact size and frequent access, resulting in lower lookup time.

Figure 3: Comparison of the lookup rates for VPP and VPP with CF.

Later, we will use the traffic generator to generate both streaming traffic with a close-to-constant bit rate and bursty one with the peak rate close to the maximum link bandwidth (80Gbps). While table lookup is critical to show the effectiveness of packet forwarding, table update performance is also important, especially when the table is updated frequently due to dynamic network conditions. Besides the table lookup and update performance, we will measure 1) memory footprint, 2) cache performance, 3) CPU cycles, 4) instruction counts, and 5) false positive rates.

ACKNOWLEDGMENTS

The work is supported by the Cisco Research Center University Funding Program in Cisco Systems, Inc. The authors are grateful to Dave Barach and Neale Ranns for their help on VPP.
REFERENCES