Towards Practical Use of Bloom Filter based IP Lookup in Operational Network

Tong Yang[1,2], Gaogang Xie[1]
[1]Institute of Computing Technology,
Chinese Academy of Sciences (CAS), China.
[2]DNSLAB, China Internet Network Information Center,
Beijing 100190.
yangtongemail@gmail.com, xie@ict.ac.cn.

Ruian Duan
College of Computing,
Georgia Institute of Technology, USA.
ruian@gatech.edu.

Xianda Sun
David R. Cheriton School of Computer Science,
University of Waterloo, Canada
x6sun@uwaterloo.ca

Kavé Salamatian
University of Savoie, France.
kave.salamatian@univ-savoie.fr

Abstract [1]—**Bloom Filter is a widely used data structure in computer science. It enables memory efficient and fast set membership queries. Bloom filter-based solutions have been proposed in the past decade for lookup in forwarding tables of backbone routers [2]. However, the main shortcomings of using Bloom Filters for lookup lie in the absence of support for deletion operations that are needed to update the forwarding tables. Counting Bloom Filter supporting deletion has therefore to be used, increasing significantly the memory requirement. Moreover, Counting Bloom Filter suffers from both false positive and false negative. In this paper, we propose to solve the issue with deletion of Bloom Filters by using a Withdrawal To annOuncement (WTO) mapping that replaces withdrawal with announcements, transforming deletions into additions or record changes. Experimental evaluation show that the proposed techniques improve largely the performance of Bloom Filter used for forwarding lookup and open way for the use of Bloom Filters in real operational settings.**

I. Introduction

Bloom Filter (BF) data structures have been applied to a large set of applications in computer science [5][6][7]. Bloom Filters are used for fast and memory efficient set membership queries. In the past decade, applications of these structures to networking problems have been proposed.

In order to increase lookup speed and reduce its cost and power consumption, Dharmapurikar *et al.* [2] proposed the Prefix Bloom Filter (PBF) structure that uses on-chip Bloom Filter to represent the trie[2] used to find the longest matched prefix. The evaluation shows that in average about 1.003 off-chip memory accesses is needed for any single lookup, faster than TCAM in average but with larger worst case complexity. PBF only uses SRAM and achieved therefore lower cost and lower power consumption. However, any lookup solution has to deal with updates that are frequent in the current operational network. Unfortunately, Bloom Filter cannot support deletion operations that are needed to do updates. Therefore, they have to be replaced with Counting Bloom Filter (CBF) that uses a k bits counter to replace each bit of Bloom Filter array, *i.e.*, supporting deletion operations CBFes entails k times more memory. This can prevent CBFs to be stored in on-chip FPGA memory. Moreover, in addition to false positive that is common in BFs, CBF can suffer from false negatives happening when a counter overflows [4]. False negative results in wrong lookups that are not acceptable for ISPs.

Nonetheless, while deletion operations are problematic for Bloom Filters, insertion operations are natural. We propose in this paper the Withdrawal To annOuncement (WTO) mapping that transforms withdrawal messages to announcement messages, that have the same effect on the forwarding behaviour of the routing table. The technique is motivated by the fact that when a prefix is withdrawn in a forwarding or routing table, it always has a shorter less specific prefix that has a default next-hop. The idea of WTO mapping is therefore to transform a prefix deletion message into a prefix insertion (or change) with the next-hop set to the next-hop of the closest ancestor prefix node. We present the details of WTO mapping in Section IV. As most of update and withdrawals happen in the leaves, WTO mapping achieves excellent performances.

II. WTO Algorithm

First we give some conclusions of Bloom Filter and Counting Bloom Filter:

- Bloom Filter supports insertion operations and membership query, but cannot support deletion operations.
- Bloom Filter has false positive, but no false negative.
- Counting Bloom Filter uses a counter in place of a bit of Bloom Filter, hence can support deletion operations.
- Counting Bloom Filter has both false positive and false negative.

In order to support incremental update, PBF adopts Counting Bloom Filter, then two problems arise:

[1] This work is supported by NSFC (61202489), and the National Science & Technology Pillar Program No.2012BAH01B03, and the Instrument Developing Project of CAS under Grant No.YZ201229.

[2] Trie is a tree-like data structure allowing the organization of prefixes on a digital basis by using the bits of prefixes to direct the branching, an excellent survey of trie-based lookup solutions are provided in [1].

- If a counter in the Counting Bloom Filter costs *x* bits, then *x* times memory is needed as compared with Bloom Filter. Therefore, Counting Bloom Filter is probably too large to be held in on-chip memory.
- Counting Bloom Filter has false negative. When false negative occurs, PBF algorithm probably returns a mistaken next-hop.

Actually, the above two problems can be solved if there is no withdrawal message, but we cannot just ignore the withdrawal messages.

Deletions are needed when withdrawal messages happen. A withdrawal message means that the announced prefix and its next-hop should be deleted from routing table, *e.g.*, let's assume that the prefix1011*:4 has to withdraw. Now, if a packet with destination address IP 10110* arrives, to which egress should it be forwarded? In practice, there are always shorter prefixes matching the IP, like 101*, 10* and *, in the routing table. So in this example, the withdrawal message just changes the longest matched prefix from 1011* to 101*. Therefore, in place of deleting the prefix 1011*, one can just change its next-hop to the next-hop of prefix 101*. By doing this the forwarding behavior of the routing table will change. Nonetheless, we have just transformed the withdrawal messages into an announcement messages and suppress the need for a deletion operation in Bloom Filters. This example illustrates the rationale of Withdrawal To annOuncement (WTO) mapping algorithm. WTO algorithm aims at changing withdrawal message into announcement one while keeping the forwarding behavior of routers unchanged.

By eliminating the need for deletion, WTO algorithm enables to simply use a Bloom Filter instead of the Counting Bloom Filter.

The WTO mapping algorithm works as follows. It seeks a way to convert withdrawal messages to announcement messages by changing the next-hop of the prefix to be deleted to its nearest ancestor's next-hop in the trie. For instance, as shown in Figure 1, node A, B, C, D and R are 4 prefix nodes in a trie and the circles represent that the egress port of these nodes is 1, while the rectangles represent port 2. When a withdrawal message: withdraw 101*, arrives instead of removing node D and updating the Bloom Filter, WTO algorithm changes the next-hop of node D to port 1 (the next-hop of node C), suppressing the need to apply an operation on the Bloom Filter.

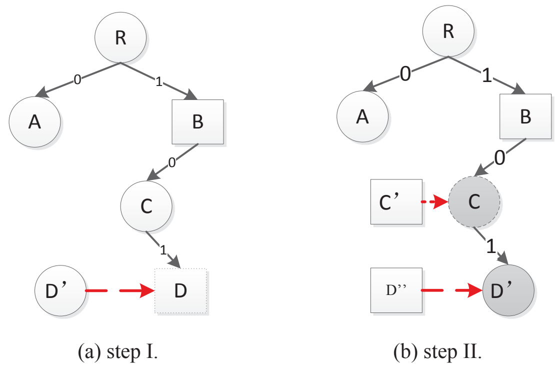

Figure 1. The scheme of WTO algorithm.

It is noteworthy that WTO algorithm may cause domino effect. In the example shown in Figure 1, when deleting node C after node D being deleted, WTO algorithm needs to change both the egress port of node C and D to port 2. To address this potential problem, a simple solution consists in checking the sub-trie rooted at the updated node. However, this entails a longer time and larger memory. To accelerate WTO mapping, we assign a flag to each node. When during deletion operation, the updated node is not really deleted but changed, just like the node D in step I of Figure 1, the flag of the nearest ancestor node (node C in Figure 1) should be set to true. During next deletion, WTO algorithm finds that the flag of a node is true, it continues to traverse the sub-trie to change the corresponding nodes. This happens in the example of Figure 1, when node C is deleted after node D.

However, getting the nearest ancestor and its flag needs a pointer in each node to point back to its parent node. To avoid this back pointer, we record the flag of the nearest ancestor node in the next node during the traversal process needed to find the updated node. This further reduces the additional memory accesses resulting from back pointers.

The above technique reduces strongly in practice the needed sub-trie traversals.

Nonetheless, when the domino effect happens, it needs more time to apply an update than the common situations. The rate of domino effect depends on the update messages. In other words, the performance of WTO algorithm depends on the characteristics of update messages, in particular the withdrawal update messages. According to our previous experimental results in [12], the update messages happen mainly in *Leaves nodes*. This suggests that even when the domino effect happens, only a few levels of the prefixes are affected.

In addition to this, we have also carried out large-scale experiments and find that withdrawal messages are much fewer than announcement messages (see Figure 3 in Section III). This phenomenon indicates that WTO algorithm is applicable for real routers. As explained above, the WTO mapping can fix the of the major problems that lead authors of [2] to use CBFs in place of BF, controlling therefore the memory increase. Moreover, the increase in the number of memory accesses induced by domino effect of the WTO algorithm is not too large to result in performance loss (see Figure 6 in Section III).

III. EXPERIMENTAL RESULT

In this section, we will validate using empirical experiment the proposed techniques to implement Bloom Filter based IP lookup.

A. *Experimental Settings*

1) *Data Set*

The data set is taken from RIPE NCC [11] at www.ripe.net, which collects routing updates from peers. In order to objectively evaluate the performance of WTO algorithm, we extracted the RIB on 2012/6/1 at 8:00 AM from 10 backbone routers, and all corresponding update messages happening during a full day are downloaded and parsed.

2) *Computer Configuration*

Our experiments have been conducted on a windows XP sp3 machine with Pentium (R) Dual-Core CPU 5500@2.80GHz and 4G memory.

B. *Experiments on WTO Algorithm*

The *x*-axis of Figure 2~7 represents the update time of update messages. For instance, '201210231945' means the time

2012-10-10 23:19:45. The labels, rrc00, rrc01, *etc.* are the router ID defined by RIPE Network Coordination Centre [11].

As the WTO algorithm is executed for any withdrawal messages, so we study the number of withdrawal messages. We show in Figure 2 the number of withdrawal messages for the 10 routing tables. The maximum value is around 927496 for one day (for rrc00). This number of withdrawal means in average 927496/24/3600≈10.73 updates per second. Moreover this number is small compared with the number of total updates: 16956829 that means in average 196.26 update messages per second.

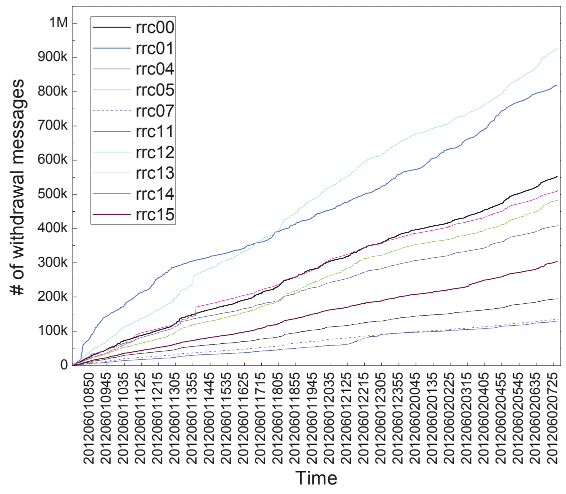

Figure 2. The number of updates in one day over 12 routers.

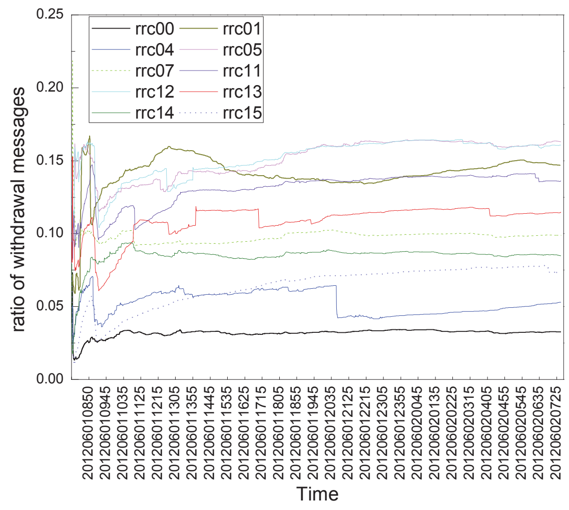

Figure 3. The ratio of the number of withdrawal update messages to that of total update messages.

As mentioned in Section I, the relative rarity of '*withdrawal messages*' makes WTO algorithm work well. To validate this, we plot the ratio of the number of withdrawal messages to the total update messages in Figure 3. It can be observed that the ratio ranges from 0.03 to 0.16 with a mean of 0.1. This suggests that only 1/10 update messages are withdrawal. That means WTO is performed in average 1/10 of time.

As mentioned in Section III, WTO algorithm may cause domino effect. When the flag of updating node or the nearest ancestor node is true, WTO algorithm must traverse the sub-trie rooted at the updating node, then additional memory accesses are needed. Actually, because update messages are generally happening in '*Leaves*' [12], additional memory accesses are very low. We plot in Figure 4 the number of additional memory accesses,. Results show that the maximum number of additional memory accesses is 3620 over one day. Given a common DRAM working at 333MHz, 3620 additional memory accesses only need 3620/333000000=10.9us that is negligible.

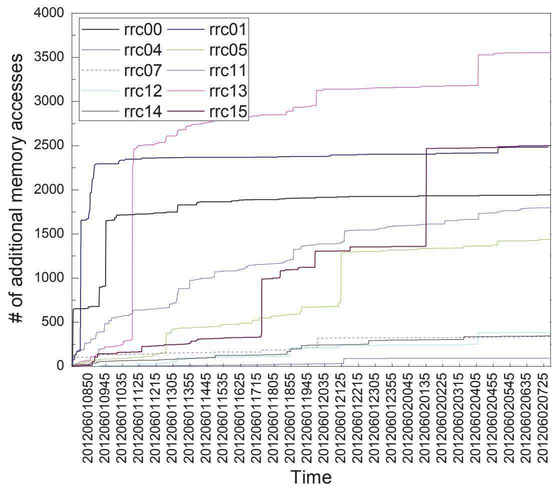

Figure 4. The number of additional memory accesses using WTO algorithm for 10 routing tables over one day.

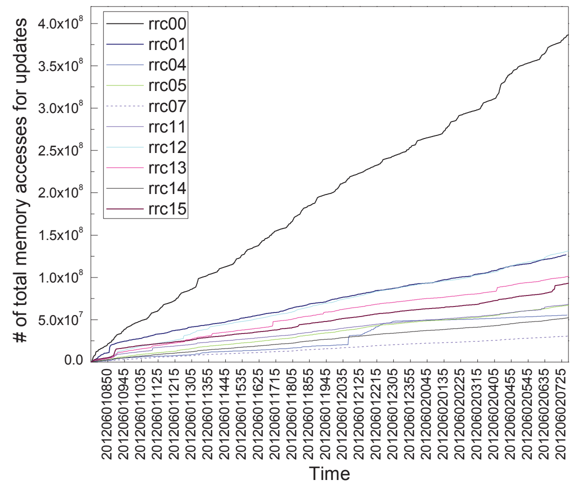

Figure 5. The number of memory accesses for update messages over one day.

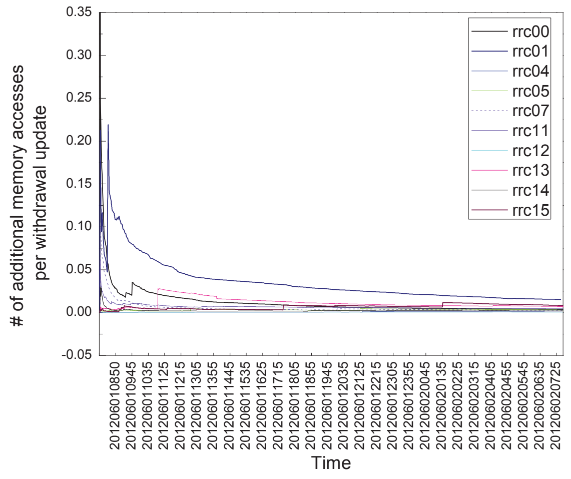

Figure 6. The number of additional memory accesses in average for each withdrawal message.

To make a comparison, we plot the total memory accesses for updates including announcement and withdrawal messages over one day in Figure 5. The number of memory access number is about 0.4 billion, 110497 times of the additional 3620 memory accesses. That's to say, the negative effect of time overhead brought by WTO algorithm is only around 10^{-6} of the original update time.

We show in Figure 6, the number of additional memory accesses for each withdrawal after using WTO algorithm. It can be observed that in average only 0.001 additional memory accesses are needed.

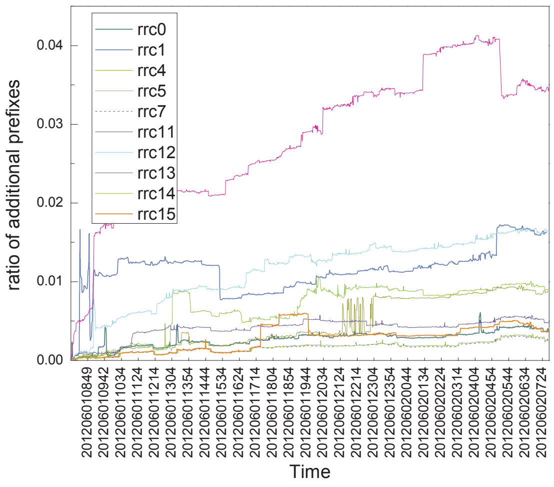

Figure 7. The ratio of the number of additional prefixes to that of total prefixes on 10 routing tables over one day.

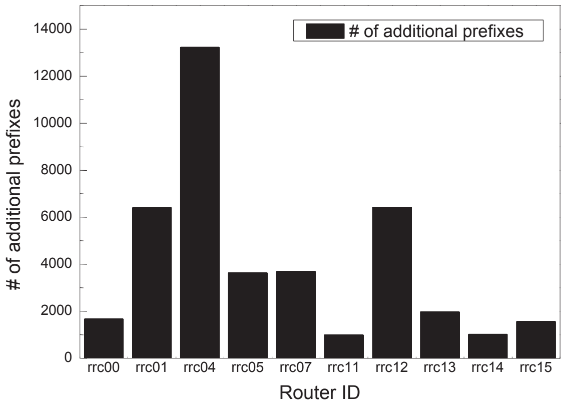

Figure 8. The number of additional prefixes on 10 routers over one day.

The other negative effect of WTO algorithm is the introduction of *additional prefixes*. As WTO algorithm changes the withdrawal messages to announcement messages, there will be more prefixes after using WTO algorithm. The number of additional prefixes is an important metrics for WTO algorithm, as too many additional prefixes means larger routing table size, that might make WTO algorithm not practical.

According to Figure 2, there are 927496 update messages at most. One might think that the number of *additional prefixes* will be 927496. However, in practice the number is much smaller as only a small fraction of prefixes are frequently updated. We show in Figure 7 and 8, the number of additional prefixes observed after applying WTO.

The ratio of the number of additional prefixes to the total prefixes is shown in Figure 7. It shows that the ratio ranges from 0.002 to 0.04 with a mean of 0.01. This means that WTO algorithm results in average in only 0.01 additional prefixes produced for each withdrawal message.

As shown in Figure 8, the number of additional prefixes ranges from 979 to 13215 with a mean of 4050. This is much smaller than the number withdrawal messages (927496). If the router has enough memory, WTO algorithm can always work well. If the memory becomes insufficient, we can periodically perform a refresh, when the router is idle.

IV. Conclusion

In order to solve the issue raised by the usage of Bloom Filter based techniques for IP lookup, we propose WTO algorithm to solve the update problem for Longest Prefix Matching. We carried out experiments to evaluate the performance of WTO algorithm, and results show that they can overcome the shortcomings of Bloom filter-based solutions at the cost of negligible overhead.

References

[1] Ruiz-Sánchez M Á, Biersack E W, Dabbous W. Survey and taxonomy of IP address lookup algorithms. Network, IEEE, 2001, 15(2): 8-23.

[2] Sarang Dharmapurikar, Praveen Krishnamurthy David E. Taylor. Longest Prefix Matching Using Bloom Filters. In ACM SIGCOMM, 2003.

[3] Bloom B H. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422-426.

[4] Deke Guo, Yunhao Liu, Xiangyang Li, and Panlong Yang. False Negative Problem of Counting Bloom Filter. Knowledge and Data Engineering, IEEE Transactions on, 2010, 22(5): 651-664.

[5] Haoyu Song, Fang Hao, Murali Kodialam, T.V. Lakshman. IPv6 Lookups using Distributed and Load Balanced Bloom Filters for 100Gbps Core Router Line Cards. In Proc. IEEE INFOCOM 2009: 2518-2526.

[6] Song H, Dharmapurikar S, Turner J, et al. Fast hash table lookup using extended bloom filter: an aid to network processing. ACM SIGCOMM Computer Communication Review. ACM, 2005, 35(4): 181-192.

[7] Yan Qiao, Tao Li, Shigang Chen. One Memory Access Bloom Filters and Their Generalization. INFOCOM, 2011 Proceedings IEEE. IEEE, 2011: 1745-1753.

[8] Yu M, Fabrikant A, Rexford J. BUFFALO: bloom filter forwarding architecture for large organizations. Proceedings of the 5th international conference on Emerging networking experiments and technologies. ACM, 2009: 313-324.

[9] Dan Li, Henggang Cui, Yan Hu, Yong Xia, Xin Wang. Scalable data center multicast using multi-class bloom filter. Network Protocols (ICNP), 2011 19th IEEE International Conference on. IEEE, 2011: 266-275.

[10] FPGA Data Sheet.
http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[11] RIPE Network Coordination Centre.
http://www.ripe.net/data-tools/stats/ris/ris-raw-data.

[12] Tong Yang, Zhian Mi, Ruian Duan, Xiaoyu Guo, Jianyuan Lu, Shenjiang Zhang, Xianda Sun and Bin Liu. An Ultra-fast Universal Incremental Update Algorithm for Trie-based Routing Lookup. Network Protocols (ICNP), 2012 20th IEEE International Conference on. IEEE, 2012: 1-10.

[13] A. Broder and M. Mitzenmacher. Network applications of Bloom filters: A survey. Internet Mathematics, vol. 1, no. 4, pp. 485–509, 2005.

[14] AS6447 BGP Routing Table Analysis Report.
http://bgp.potaroo.net/as6447/.