Virtual Routing Tables Polymerization for Lookup and Update

Tong Yang, Shenjiang Zhang, Xianda Sun, Huichen Dai Ruian Duan, Jianyuan Lu, Zhian Mi and Bin Liu
Department of Computer Science and Technology, Tsinghua University, Beijing, China

***Abstract*—Virtual router research has drawn increasing attention in recent years, and the most challenging issues of virtual routers are compression, lookup, and incremental update of 10~200 routing tables. In this paper, we propose a set of solutions to achieve that storage, lookup time, and update time don't expand to 10~200 times, but reduce to 1~2 times.**

I. INTRODUCTION

Network virtualization has attracted extensive concern over recent years, as it provides new possibilities for the evolution path to the Future Internet by building a practical testbed which can simulate the deployment and protocol of a new global Internet architecture. Virtual router is one of the most important research fields in network virtualization because of the essential role of routers in the Internet.

Existing researches of implementations of virtual routers can be mainly classified into three categories: software-based solutions [1], FPGA-based solutions [2], and hybrid-structure-based solutions [3]. They are all in pursuit of the instance number and the performance of virtual routers, and the key factors [5][6][7][8] determining the performance are storage structure, lookup algorithm, and update mechanism of virtual routing tables.

Routing lookup and compression are well-studied research fields, and various solutions have been proposed, but they can scarcely be applied to 10~200 virtual routing tables directly. To operate 10~200 routing tables, some advocate separate solutions [2][3], which can isolate the effects from each other. However, the hardware support takes on a linear growth with the increase of the number of virtual routers. In contrast, merged solutions [4][5][6][7][8] consolidate all individual routing tables into a shared trie to carry out intensive lookup, which can significantly reduce the hardware cost and power consumption, especially when the number of virtual routers increases. Existing merged solutions adopt trie structure or SRAM pipeline: trie-based solutions need multiple memory accesses, hence cannot achieve fast lookup, while SRAM pipeline cannot hold multiple routing tables in on-chip memory, accordingly can hardly work well for virtual routers.

In real routers, TCAM-based solutions are the de facto standard, owing to its simplicity and high lookup speed. Therefore, we strive to hold the multiple virtual routing tables in one TCAM to achieve fast lookup, while aiming for fast incremental update. The main shortcomings of TCAM-based solutions are the high power consumption and high cost. To minimize the hardware cost, power consumption, and routing lookup delay, 1) we hold 10~200 routing tables in one TCAM chip to perform fast lookup; 2) we propose an update algorithm with complexity of *O(1)*; 3) we achieve that the power consumption, lookup delay become *1/h* of the original, where *h* is the forwarding times in the virtual routers.

Authors' Email: yangtongemail@gmail.com, {zsj09, sunxd08, dhc10, dra08, lu-jy11}@mails.tsinghua.edu.cn, mzaort@gmail.com, liub@tsinghua.edu.cn
Supported by NSFC (61073171, 60873250), Tsinghua University Initiative Scientific Research Program (20121080068), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20100002110051).

As far as we know, this is the first effort to store all virtual routing tables into a single TCAM; this is the first effort to achieve that one packet is only looked-up once from the source router to the destination router in a virtual network; this is also the first effort to achieve that one outside update message which would have been processed in every virtual router is updated only once.

II. VIRTUAL ROUTING TABLES POLYMERIZATION

Suppose there are *n* virtual routers in a virtual network, and one packet needs *h* hops in average to pass through this network. At this moment, the requirements for routing table storage and routing updates become *n* times as much as the original, while the delay of routing lookup gets to *h* times as long as the original. Actually, there is only one node physically, thus intuitively the requirements for storage, lookup, and update can be reduced to *1* or *2* times. In this section, we propose Virtual Routing Tables Polymerization (VRTP) algorithm to polymerize the requirements for routing table storage, lookup and incremental update.

i) With regard to the storage structure, VRTP adopts **Trie Outline** algorithm to draw an outline of all routing tables in a virtual network.

Step I: As shown in Figure 1, every routing table is stored in a trie. We overlie all routing tables, and get an outline trie, which is then extended into a '*healthy trie*'. A *healthy trie* must satisfy two constraints: 1) all nodes excluding the root node must have a sibling node; 2) only leaf nodes have next-hops.

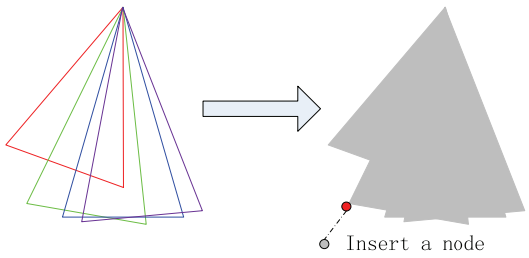

Fig. 1. The basic principle of trie outline algorithm.

Step II: Every trie is pushed into the outline's leaf nodes. As a result, the inner nodes of outline trie have no next-hop, but the leaf nodes have a next-hop array, accordingly the next-hop information can be represented by a huge matrix.

ii) With regard to lookup, based on this outline trie, we propose an approach to store the routing table in one TCAM for the first time, while existing algorithms don't adopt TCAM, because one TCAM can only hold one or two backbone routing tables, and multiple TCAMs call for expensive cost and high power consumption. The naive TCAM approach is to put the prefixes of all routing tables in TCAM, thereby multiple TCAMs are needed, while the goal of VRTP is to adopt only one TCAM. The scheme of VRTP is as follows: as mentioned above, every leaf node of the outline trie has a next-hop array, consequently the associate SRAM of TCAM should be large enough to store a large number of next-hop array. If the SRAM cannot hold the next-hop array, VRTP puts another independ-

ent SRAM to store the next-hop array, and the associate SRAM of TCAM merely stores an address which points to the corresponding next-hop array. Experimental results show that 10 routing tables can be held in a 700K TCAM. In addition, because there are only leaf nodes in outline trie, thus the prefixes in TCAM can be random, while traditional TCAM-based solutions must put the prefixes in order of prefix length because of prefix overlap[2]. This also benefits the parallel lookup of TCAM a lot.

Furthermore, we propose an approach to achieve that every packet which is forwarded *h* times in the virtual network merely be looked-up once. This is convenient to implement owing to the **Trie Outline** algorithm: given an incoming packet, the lookup result will get a next-hop array, which indicates all the next-hops that every virtual router should forward the packet to. In fact, only the virtual routers in the forwarding path need to lookup this packet. In this way, the lookup count of every packet can keep one with the increase of the virtual routers.

iii) With regard to incremental update, update messages can be divided into inner nodes update and leaf nodes update, while inner nodes update can be changed into leaf nodes update. This provides a good opportunity for the application of our previously proposed Blind Spot (BS) algorithm [9]. The *core idea* of BS algorithm is to eliminate domino effect by setting those updating nodes which would have produced domino effect as Blind Spot, so as to operate fast update. The update complexity of BS algorithm is *O(lookup+h)*, where the lookup complexity of TCAM is *O(1)*, and the average depth of Blind Zone (*h*) is around 2 according to the experimental results. Therefore, the update complexity of VRTP is *O(3)*, i.e., *O(1)*. Although BS algorithm can avoid the prefix movements in TCAM, but the update time is still *n* times as long as one real routing table update using BS algorithm. In order to reduce *n* to 1, we classify the routing update messages into two categories: the update messages from outside and inside virtual network. Actually, most update messages come from outside network, and all virtual routers must process them. For one outside update message, rather than update *n* virtual routing table one by one, VRTP update it once in the outline trie: 1) locate the update node, one inner node update can be translated into several leaf nodes update; 2) the following update procedure is to update the corresponding next-hop array of leaf nodes, which is convenient and fast. In this way, the update times can be reduced from *n* to 1.

III. EVALUATION

We have downloaded and parsed 10 routing tables and one-day update messages from www.ripe.net to evaluate the number of prefixes and the update performance of outline trie.

As shown in Figure 2, the x-axis represents the router's ID, every bar stands for one router's prefix number. The green line means the solid node (leaf node) number of the outline trie, while the red curve indicates the number of all nodes in the outline trie. It can be observed that the number of solid nodes which will be stored in TCAM keeps a little growth with the increase of virtual routers. This is because the routing tables of the same moment in backbone routers have the similarities of number and outline, and the similarities will be more remarkable in the routers of one virtual network. Every rectangle-like area in Figure 3 is the update result of the marked router, and the x-axis of Figure 3 means the time of update messages. Results show that there are only around 2247 blind spots (0.31% of the overall prefix number) on 10 routers for one-day updates.

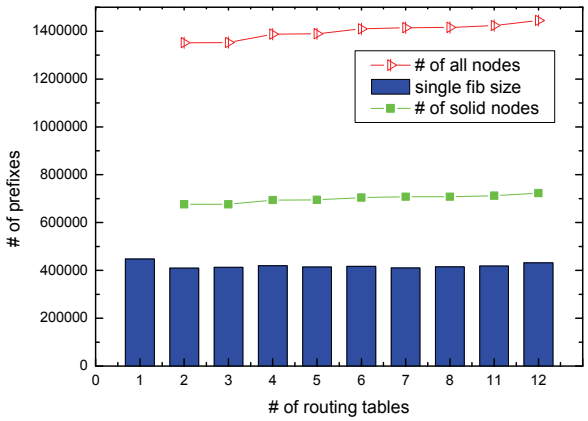

Fig. 2. the increase of the prefix number of the outline trie for 10 routing tables over one day.

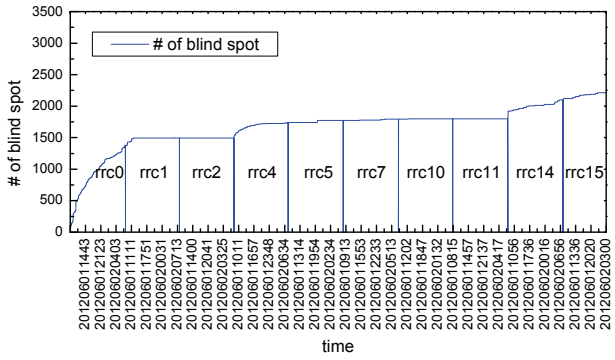

Fig. 3. the increase of BS count on 10 routers for one-day update

REFERENCES

[1] Mihai Dobrescu, Norbert Egi, Katerina Argyraki, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Maziar Manesh, Sylvia Ratnasamy. RouteBricks: Exploiting Parallelism To Scale Software Routers. In Proc. SOSP 2009.

[2] M. B. Anwer, M. Motiwala, M. Tariq, and N. Feamster. SwitchBlade: A platform for rapid deployment of network protocols on programmable hardware. In Proc. ACM SIGCOMM, 2010.

[3] Deepak Unnikrishnan, Ramakrishna Vadlamani, Yong Liao, Abhishek Dwaraki, Jérémie Crenne, Lixin Gao, and Russell Tessier. Scalable Network Virtualization Using FPGAs. In Proc. FPGA 2010.

[4] J. Fu and J. Rexford. Efficient IP address lookup with a shared forwarding table for multiple virtual routers. In Proc. CoNEXT 2008.

[5] H. Song, M. Kodialam, F. Hao, and T. V. Lakshman, "Building scalable virtual routers with trie braiding," In Proc. INFOCOM 2010.

[6] Thilan Ganegedara, Hoang Le, Viktor K. Prasanna. Towards On-the-fly Incremental Updates for Virtualized Routers on FPGA. In Proc. FPL 2011.

[7] Hoang Le, Thilan Ganegedara and Viktor K. Prasanna. Memory-Efficient and Scalable Virtual Routers Using FPGA. In Proc FGPA 2011.

[8] O˜ guzhan Erdem. Hoang Le, Viktor K. Prasanna. Cuneyt F. Bazlamac. Hybrid Data Structure for IP Lookup in Virtual Routers Using FPGAs. In Proc. FCCM 2011.

[9] T. Yang, Z. Mi, R. Duan, X. Guo, J. Lu, S. Zhang, X. Sun, B. Liu. An Ultra-fast Universal Incremental Update Algorithm for Trie-based Routing Lookup. Accepted by Proc. ICNP 2012.

[2] Prefixes overlap refers to that some prefixes are a part of others. The prefix of an inner node must be a part of leaf node prefix. In this case, an incoming IP could match multiple nodes, thus the prefixes of the matched nodes should be put orderly to select the longest matched prefixes by priority encoder. Due to the elimination of prefix overlap by Trie Outline algorithm, the priority encoder is no longer needed.