第６部

特集6 高速PCルータ研究会

浅井 大史

The High-Speed PC Router Research Group was initiated on April, 2014. This research group aims to develop a high-performance router using commercial off-the-shelf hardware. The interests of this research group span software implementation of high-performance network functions including Operating Systems, Network Algorithms, and FPGA implementations, but the activities are not limited to these topics; e.g., storage system. Research group meetings are basically held every month. Half of its active members are the members of WIDE Project although it is not an official activity of WIDE Project. One of the contributions of this research group in 2015, which was delivered with the strong collaboration of WIDE Project, is a fast IP routing table lookup algorithm, named Poptrie. The paper on Poptrie was presented in ACM SIGCOMM 2015 [36]. In this report, we summarize the experiments of Poptrie with the data collected in the network operated by WIDE Project. Please refer to the original paper for the details of Poptrie and the results of evaluation.

Poptrie is extended from a multiway trie, and it compresses its data structure into a small memory footprint (e.g., 2.4 MiB for a global tier-1's full route) in order to reduce CPU cache misses in the lookup procedure. We have shown that Poptrie achieves significantly good performance; it runs 4-578% faster than the other state-of-the-art technologies, and is advantageous for longer prefixes such as IGP routes, which have been more challenging in terms of

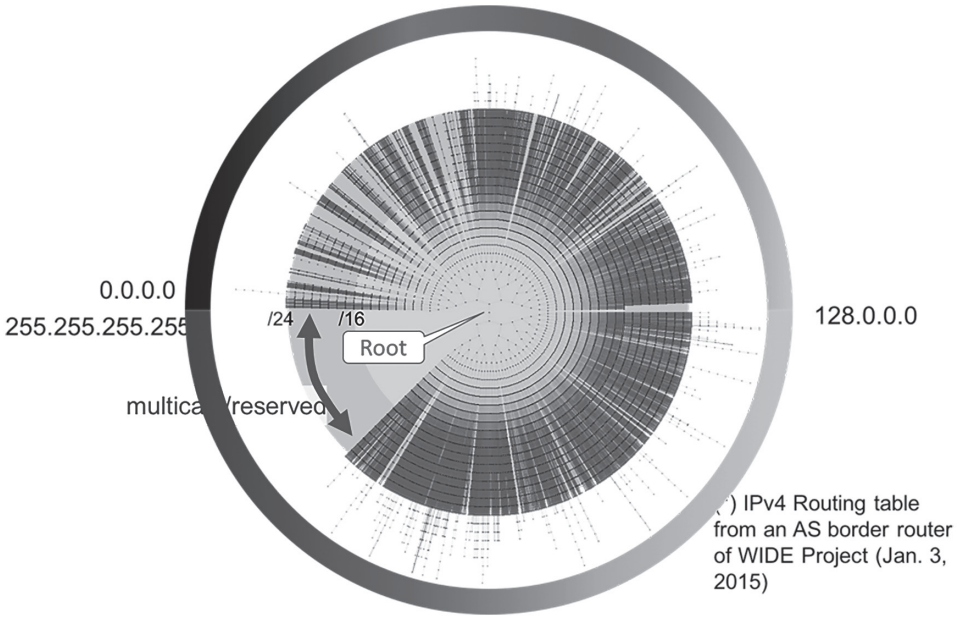

Figure 1　A Visualized Data Structure of Radix Tree

performance. Poptrie relies on two key ideas to achieve high lookup rate: 1) reducing the number of instructions executed to search down a trie data structure by reducing the lookup depth of a trie, and 2) increasing CPU cache efficiency by compressing the data structure within CPU cache size. To visually demonstrate these two key ideas of Poptrie compared to the conventional radix tree, these data structures for an IPv4 routing table at an AS border router of WIDE Project are illustrated in Figure 1 and 2.

In these figures, address space and prefix length are represented in circumferential and radial directions, respectively. As illustrated in these figures, the search depth of the conventional radix tree is up to 32 (for IPv4), and thus it requires a number of memory access to search down the tree. Moreover, it has many pointers to descendant nodes, and consequently, consumes a large memory footprint. On contrary, the maximum depth of Poptrie is reduced to 4. It also reduces the number of pointers to achieve a small memory footprint. Our extension to a multiway trie requires to add a procedure to count the number of 0s or 1s bits. Fortunately, an instruction of this procedure is implemented in recent x86-64 processors. With other performance optimization techniques described in Ref. [36], Poptrie achieves very good performance.

We evaluated the performance of Poptrie on various routing tables including future-envisioned synthetic datasets as well as real Internet routing tables. One important experiment to demonstrate the advantage of Poptrie was conducted using the routing table of a backbone router of WIDE Project and a traffic trace captured at the transit link from the backbone router (i.e., MAWI WG Traffic Archive [37]). Since the performance of IP routing table lookup in software is substantially affected by the traffic pattern, the evaluation with a set of IP routing table and traffic trace on a running environment is relatively important. Here, we summarize the dataset we used for the experiment. The routing table of the backbone router was obtained at January 3, 2015, and contains 516,100 entries of IPv4 prefixes, including IGP routes as well as BGP ones, with 32 distinct next-hop addresses. We used a MAWI traffic trace, captured on December 16, 2014, for 15

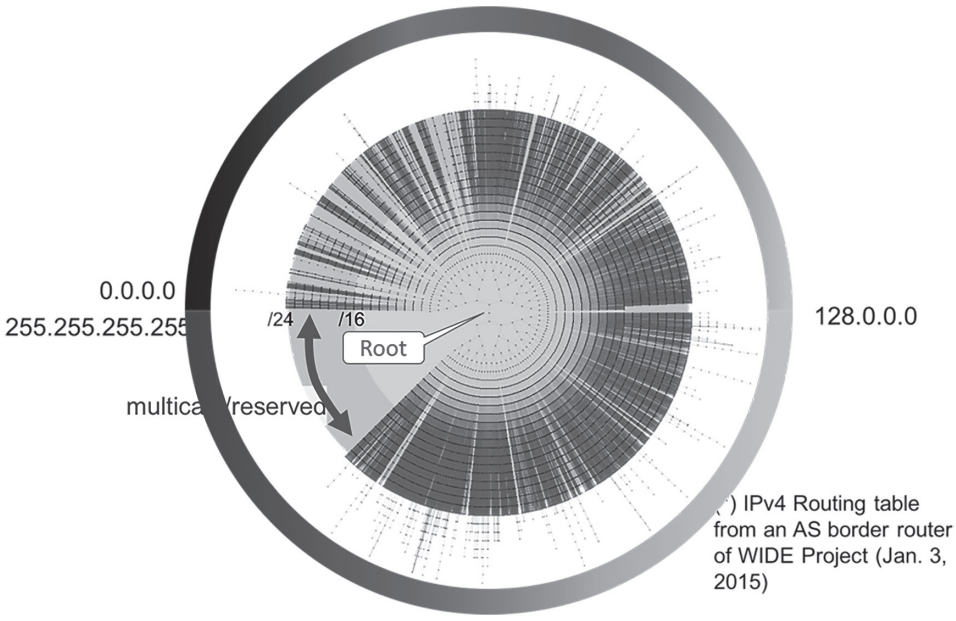

Figure 2 A Visualized Data Structure of Poptrie

minutes. An IP address that had probed the entire IPv4 address space with a huge amount of ICMP packets were excluded (USC ANT project: http://www.isi.edu/ant/address/). These packets accounted for 24.4% of the total IPv4 packets in this trace. After the exclusion, we obtained 97,126,495 IPv4 packets with 644,790 distinct destination IPv4 addresses, and used them for the following experiment.

We conducted the performance evaluation on Poptrie (with two parameters, $Poptrie_{16}$ and $Poptrie_{18}$) and three state-of-the-art algorithms; Tree BitMap [38], SAIL [39], and DXR (D16R and D18R) [40]. We measured the average lookup rate on a computer with Intel® Core i7-4770K CPU and four 8 GiB DDR3-1866 RAMs. Note that we have used only one core from the CPU to evaluate the serialized lookup performance. As a reference experiment, we measured the average lookup rate for random-destination traffic, which is more challenging in terms of CPU cache pollution. Figure 3 presents the average lookup rate for random-destination traffic on the WIDE backbone core router. From this figure, we confirm that $Poptrie_{18}$ outperforms the other state-of-the-art algorithms. The average lookup rate for MAWI traffic trace, i.e., real traffic trace, on the WIDE backbone core router is shown in Figure 4. We also confirm that $Poptrie_{18}$ performs the best among the evaluated algorithms. By comparing Figure 3 and 4, the lookup rates of Poptrie and DXR for MAWI traffic trace were degraded compared to those for random-destination trace. This is because a larger number of packets goes to IGP routes that are generally more specific than BGP routes in the MAWI traffic trace. 32.5% of the packets in MAWI traffic trace on the WIDE backbone core router have the search depth in the radix tree of more than 18, while for the whole IPv4 address space only 22.1% have that of more than 18. These addresses cannot be looked up in the first stage of the algorithm of $Poptrie_{18}$ and D18R. Moreover, 21.8% of the packets of MAWI traffic trace have the search depth in the radix tree of more than 24, while only 1.66% of the whole IPv4 address space have that of more than 24. SAIL performs better in the lookup rate for MAWI traffic trace than for random-destination traffic. This is because SAIL could take advantage of the CPU cache due to the locality of the destination IP addresses in the real traffic trace, i.e., the sequences of packets with the identical destination IP address.

In summary, we demonstrated that Poptrie outperformed the other state-of-the-art algorithms. With real datasets from WIDE Project, Poptrie is proven to be applicable to the real Internet routing tables and traffic patterns.

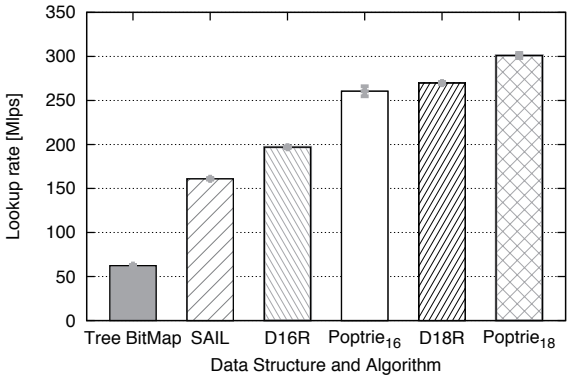

Figure 3 The average lookup rate for random-destination traffic on a WIDE backbone core router

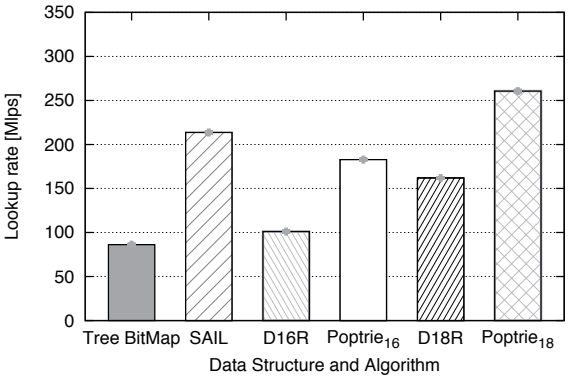

Figure 4 The average lookup rate for MAWI traffic trace on a WIDE backbone core router