Dear Sir/Madam,

In response to the reviewers' comments for our submission TC-2016-10-0714, we have revised the manuscript as suggested. For reviewer's convenience, we reply to the comments for the previous version one by one in the attachment of this letter.

Thanks a lot!

Yours Sincerely,

Yi Wu

Department of Computer Science
Sun Yat-sen University, Guangzhou, China
Email: wu.yi.621@gmail.com

18 Mar 2017

Response to Review Comments

Reviewer #1:

Comments:

The authors have detailed a scalable multi-block routing architecture for pipelining any prefix trie-based routing algorithm that links prefix nodes by pointers. The scheme uses multiple memory blocks for trie storage and buffer for queuing packets, resolving any temporary memory access contentions. Furthermore, the scheduling of queuing packets to access memory blocks for routing is modeled as a bipartite graph matching problem. The authors propose that the scheme may be considered a universal scalable solution for IP lookup using prefix trie with nodes connected by pointers. A qualitative comparison is presented in terms of memory requirement, power consumption, throughput and update efficiency. While the qualitive consideration in section 6 is highly relevant, it seems rather limited given the number of prior works compared with the present approach, both in CP and LP category [2, 5, 8, 9]. A more detailed benefit of the proposed design, preferrably with example(s) scenarios over existing trie-based schemes is needed to further highlight the relevance and novelty of the present work. Additionally, font size in Fig. 2 - Fig. 8 needs to be increased for convenient reading.

Response:

We adapt the proposed routing architecture along with the trie-based index in use by separately organizing the short and the long prefixes in a routing table using different data structures. In brief, each short prefix is inserted into a quick table while all the long prefixes are organized as one or multiple prefix tries using a specific trie-based routing algorithm. For each packet, it first searches for the longest matching prefix among short prefixes in the quick table and then traverses one of the prefix trie to find a possible solution among the long prefixes. Because the nodes in the prefix tries are scattered into a pipeline consisting of multiple memory blocks, more than one packets may request multiple nodes stored in the same memory block. A routing buffer is employed to cache packets for resolving potential memory access contentions.

A series of simulation results are conducted to evaluate the proposed pipeline system in terms of memory requirements and system's throughput. The experimental results indicate that our scheme has a more balanced storage demand in comparison with linear and circular pipelines. Besides, both our and the circular pipelines have a high lookup throughput under synthetic IPv4 traffic loads, while the latter has an advantage over the former by roughly 10% under IPv6 traffic loads.

All the figures have been adjusted for convenient reading.

Reviewer #2:

Comment 1:

The references are too old. Although the prefix matching problem is a very old problem in router design (since 1993), however, still many research papers are being published in this area every year and some of them apply pipelining schemes. Therefore, the authors should cite the new published researches about IP lookup problem, Trie based IP Lookup problems and especially the ones using pipelined architectures.

Response:

Recent works on designing high-performance pipelined routing architectures using trie-based data structures are cited in this revision, e.g., [10, 11, 30] in the introduction.

Comments 2 & 4:

Almost all of the results showed in figures, are about variations of Q; the average number of queuing packets in the routing buffer; based on the variations of other parameters. However, three main parameters of prefix matching problems are: search speed, updated speed (is it incremental or not?) and storage requirements. In this article, the improvements of these three parameters are not evaluated. Some figures should be included in the article for showing exactly the results of search speed, update speed and storage requirements of the proposed pipelined architecture compared to the basic algorithms of BT, PT, 6-FST and 2-MPT without pipeline.

The proposed pipeline structure is not compared with any existing pipelined IP Lookup structure. In order to evaluate the performance of this structure, it "must" be compared with one of the existing pipelined prefix tries.

Finally, I believe that this research could be resubmitted after Major revisions listed above.

Response:

We modify the proposed routing architecture and the trie-based index in use. A series of computer software simulations are conducted to investigate the performance of our scheme in terms of memory requirements and system's throughput with the index built by various trie-based routing algorithms, where the results for our scheme are compared with the counterparts for the linear and circular pipelined routing schemes to make a deep study. As observed from the experiments, our scheme achieves a more balanced storage demand over the pipeline stages than the other two before and after the execution of incremental updates. Besides, the lookup throughput of the proposed random scheme is similar to that of the circular pipeline under IPv4 traffic loads, while the latter has an advantage over the former by roughly 10% under IPv6 traffic loads.

We point out that, the memory accesses to the pipeline introduced by incremental updates can be scheduled together with those for route lookups in our routing scheme by using an off-chip preprocessor to group the involved write operations into one or multiple “write batch”, where the write operations in each batch target for different memory blocks and thus can be done at the same time. The pipeline system establishes a write request for each batch and schedules the requests together with the existing searching requests. To keep the consistent state of each prefix trie, each write request has a higher priority than all the searching ones. Because the frequency of updates is lower than that of lookups by multiple orders of magnitude, it is reasonable to expect that the workload introduced by the former will bring no significant negative impacts on the system's lookup throughput.

Comment 3:
Also, all results of this paper are about "the average" Q. The question is: Why didn't the authors consider the "worst case" Q? Queuing delay is one of the most important parameters in router design. Therefore, its "worst case" seems to be required for consideration in router design.

Response:
It is assumed that the routing buffer for queuing packets is of an infinite capacity. We investigate in experiments the maximum length of the queuing buffer as a function of the normalized workload under the packet arriving processes following a Bernoulli distribution, where a burst (including one or multiple packets) arrive at the routing buffer with a certain probability at each time slot. Our experiments indicate that almost no changes occur to the system’s maximum throughput under traffic loads with different burst sizes.

As a big change in this revision, we would emphasize that the experiments have been redesigned in order to meet the requirements raised by the review comments for the previous revision. This is quite a heavy job and we hope that it has helped improve the quality of this manuscript.

Thanks a lot!