Packet Classification Using a Bloom Filter in a Leaf-Pushing Area-based Quad-Trie

Hyesook Lim
Department of Electronics Engineering
Ewha Womans University
52, Ewhayeodae-gil, Seodaemun-gu
Seoul 120-750, Korea
hlim@ewha.ac.kr *

Ha Young Byun
Department of Electronics Engineering
Ewha Womans University
52, Ewhayeodae-gil, Seodaemun-gu
Seoul 120-750, Korea
hayoung77@ewhain.net

ABSTRACT

Packet classification is one of the most essential functions that Internet routers should perform at wire-speed for every incoming packet. An area-based quad-trie (AQT) for packet classification has an issue in search performance since many rule nodes can be encountered in a search procedure. A leaf-pushing AQT improves the search performance of the AQT by making a single rule node exist in each search path. This paper proposes a new algorithm to improve the search performance of the leaf-pushing AQT further. The proposed algorithm builds a leaf-pushing AQT using a Bloom filter and a hash table stored in on-chip memories. The level of a rule node and a pointer to a rule database are identified by sequentially querying the Bloom filter and by accessing the hash table, respectively.

Categories and Subject Descriptors

C.2.6 [**Internetworking**]: Routers

Keywords

packet classification; Bloom filter; area-based quad-trie; leaf-pushing;

1. INTRODUCTION

Internet routers nowadays are required to intelligently provide the different quality of services for various application programs. Packet classification is an essential pre-requisite to provide such services [1]-[3]. For a given set of rules composed of multiple header fields, packet classification determines the highest priority rule matching each input packet. A rule database is stored in an external memory because of its size as the number of rules increases. Since an access to an external memory takes 10-20 times longer than that to an on-chip memory [2], the packet classification performance directly relates to the number of external memory accesses.

To provide efficient packet classification, it is required to extract a small number of possible matching rules using an on-chip memory and compare with those rules in an external memory. This paper proposes to use a Bloom filter and a hash table in extracting possible matching rules for each input packet. As a space-efficient probabilistic data structure used to test whether an element is a member of a set, Bloom filters have been popularly applied to various network applications [3].

*This work was supported by the NRF grant funded by the Korea government MEST (2014R1A2A1A11051762) and by the MKE under the ITRC support program supervised by the NIPA (2012-H0301-12-4004).

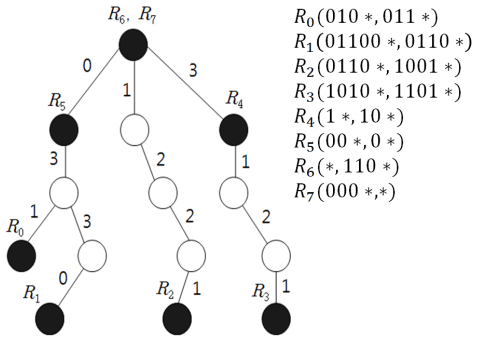

Figure 1: Area-based quad-trie (AQT)

For an arbitrary example set of rules, Figure 1 shows an area-based quad-trie (AQT) [4], which is constructed using the codewords of source and destination prefix fields. For a given rule, each bit of the source and the destination prefix is combined together to generate a codeword, and the codeword is used in locating a node storing the rule. For example, rule R_0 (010*, 011*) has the codeword of 031, and the rule is stored in the corresponding node. If the lengths of two fields are different, the codeword length is determined by the shorter length. The AQT does not provide high-speed search since a search path can have many rule nodes.

A leaf-pushing AQT pushes down every internal rule node into leaves in order to solve the search performance issue in the AQT. Figure 2 shows the leaf-pushing AQT [5] corresponding to the AQT of Fig. 1. For the example case of an internal rule node R_4 (1*, 10*), since the destination prefix is longer, the source prefix of R_4 should be extended. The source prefix can be extended to two bits, 10* and 11*. Combining the extended source prefix 10* with the destination prefix 10*, we have a codeword of 30*. Similarly, combining the extended source prefix 11* with the destination prefix 10*, we have a codeword of 32*. Hence rule R_4 is pushed down to two nodes, 30* and 32*, which are leaf nodes. Hence, rule R_4 is not necessarily extended further.

2. PROPOSED ALGORITHMS

Figure 3 shows the proposed architecture. Every node in the leaf-pushing AQT trie is programmed into a Bloom filter. Every rule node and incomplete internal nodes (internal

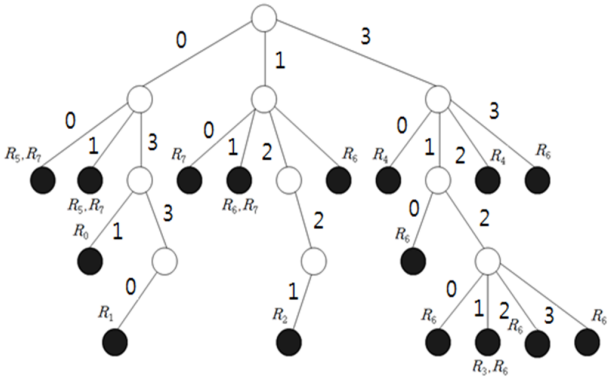

Figure 2: Leaf-Pushing AQT

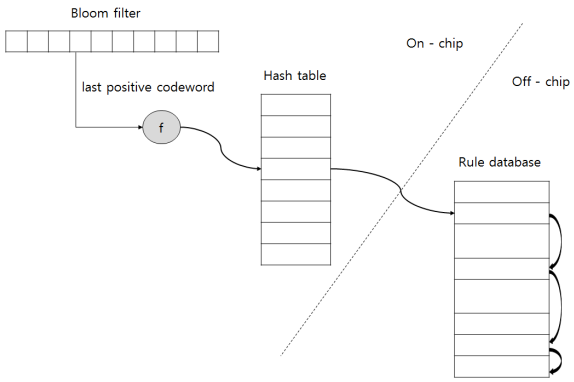

Figure 3: Proposed Architecture

nodes with less than 4 children) are stored in the hash table. Rules in a rule node are stored in a rule database using a linked list in the order of decreasing priority so that the highest priority rule is compared earlier. For a given input, the Bloom filter is sequentially queried while increasing the level until a negative result is produced. The Bloom filter negative means that there is no node in the current and longer levels of the trie, and hence the Bloom filter querying is stopped. The level with the last positive, which is the level of the negative minus 1, will be the level of a rule node (if the positive is true). A hash table is accessed using the value of the identified rule node to get a pointer pointing the highest priority rule of the node to access the rule database. In case of a false positive (no matching entry in the hash table), the proposed algorithm tracks one level back until a matching entry is found, but the false positive rate can be controlled sufficiently small by increasing the Bloom filter size. Since internal nodes are only accessed by back-tracking in case of false positives in our proposed architecture, internal nodes with 4 children do not have to be stored in the hash table.

3. PERFORMANCE EVALUATION

Simulations using C++ have been performed for rule sets created by Classbench [6]. Three different types of rule sets, access control list (acl), firewall (fw), and Internet protocol chain (ipc), are generated with sizes of approximately 10000 and 100000 rules each. Rule sets are named using the set type followed by the size. A 64-bit cyclic redundancy check generator was used to obtain hash indices for the Bloom filter and the hash table [3].

Table 1 shows the memory requirement for the Bloom filter (M_b), the hash table (M_h), and the rule database (M_r) for the number of rules (N), the number of stored rules (N_s), and the number of nodes (T) in the leaf-pushing AQT. As shown, N_s is much larger than N because of rule replication caused in leaf-pushing process. The size of a Bloom filter $m = 16T'$, where $T' = 2^{\lceil log_2 T \rceil}$, and the number of hash indices $k = 16ln2$ in programming the Bloom filter. Table 2 shows the search performance of the proposed algorithm in terms of the average numbers (A) and the worst-case numbers (W) of BF queries (A_b, W_b), hash table accesses (A_h, W_h), and the rule comparisons (A_r, W_r), respectively. It is shown that the number of rule comparisons highly depends on set types and trie characteristics, while a single hash table access is required in average to perform a classification in our proposed algorithm.

Table 1: Memory Requirement

	N	N_s	T	M_b (KB)	M_h (KB)	M_r (MB)
acl10K	9735	120950	82299	256	1312.0	2.6
acl100K	95340	139329	1697	4	20.8	3.1
fw10K	4351	39473	477	1	5.1	0.9
fw100K	82473	500751	973	2	10.5	11.1
ipc10K	8112	56508	2581	8	20.3	1.2
ipc100K	94370	251870	2629	8	20.8	5.6

Table 2: Search Performance

	A_b	W_b	A_h	W_h	A_r	W_r
acl10K	29.2	31	1	2	1.7	28
acl100K	25.9	31	1	4	165.0	776
fw10K	10.4	32	1	1	172.8	666
fw100K	11.5	31	1	1	352.2	2983
ipc10K	17.4	31	1	2	37.6	175
ipc100K	21.7	31	1	1	85.3	459

4. REFERENCES

[1] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," *ACM Computing Surveys*, vol. 37, no. 3, pp. 238–275, Sept. 2005.

[2] H. Lim, N. Lee, G. Jin, J. Lee, Y. Choi, and C. Yim, "Boundary Cutting for Packet Classification," *IEEE/ACM Trans. Networking*, vol. 22, no. 2, pp. 443–456, Apr. 2014.

[3] A. G. Alagu Priya and H. Lim, "Hierarchical packet classification using a Bloom filter and rule-priority tries," *Computer Communications*, vol. 33, no. 10, pp. 1215–1226, Jun. 2010.

[4] M. M. Buddhikot, S. Suri, and M. Waldvogel, "Space decomposition techniques for fast layer-4 switching," *Prot. for High Speed Networks*, pp. 25–41, Aug. 1999.

[5] J. Lee, J. Mun, and H. Lim, "Leaf-Pushing Area-based Quad-Trie for Packet Classification," *submitted to Sigcomm2015.*

[6] D. E. Taylor, J. S. Turner, "ClassBench: A Packet Classification Benchmark," *IEEE/ACM Trans. Networking*, vol. 15, no. 3, pp. 499–511, Jun. 2007.