Computer Communications 103 (2017) 116–129

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Utilizing 2-D leaf-pushing for packet classification

Jungwon Lee, Hayoung Byun, Ju Hyoung Mun, Hyesook Lim*

Department of Electronics Engineering, Ewha Womans University, Seoul 03760, Republic of Korea

A R T I C L E I N F O

Article history:
Received 9 December 2015
Revised 26 January 2017
Accepted 11 February 2017
Available online 14 February 2017

Keywords:
Internet
Router
Packet classification
Trie
Area-based quad-trie
Leaf pushing
Bloom filter
Decision tree

A B S T R A C T

Packet classification is one of the most challenging functionalities performed by routers at wire-speed for every incoming packet. For search spaces composed of multiple rules represented geometrically, various space decomposition algorithms have been studied to provide effective search methods. While an area-based quad-trie (AQT) provides a simple and intuitive way of mapping the geometrical search space into a two-dimensional (2-D) trie structure, it does not provide high-speed classification performance because the mapping is incomplete. This paper proposes the application of leaf-pushing into the 2-D trie to improve the classification performance of the AQT. The leaf-pushing AQT provides a more effective method of searching rules covering each input packet in the decomposed space. We also discuss an efficient implementation technique for our algorithm using a Bloom filter and a hash table. Simulation results show that our proposed leaf-pushing AQT improves the packet classification performance up to 37 times for sets with up to 100,000 rules compared with the AQT. To be compared with other space decomposition algorithms, a refined structure of the leaf-pushing AQT is also proposed. Simulation results show that the proposed refined structure provides an effective space decomposition method as well as the balance between memory requirements and classification speed, while most of other space decomposition algorithms show a trade-off between them.

1. Introduction

As the Internet has grown, various applications have been developed, and the amount of traffic carried by the Internet has increased rapidly in recent years. The Internet needs to provide different priorities or quality of services (QoS) to different applications, users, or data flows. Packet classification is an essential pre-requisite for Internet routers to provide such services [1–5]. Packet classification determines a set of rules that match each input packet using five header fields: a source prefix, a destination prefix, a source port number, a destination port number, and a protocol type. The highest priority rule among the matching rules becomes the best matching rule (BMR) for the packet. The Internet routers treat the packet as specified in the class of the BMR.

Using a ternary content addressable memory (TCAM) is the *de facto* standard in implementing the packet classification functionality [6]. However, the TCAM suffers from memory blowup when representing range fields in rules [6,7], and thus the number of range fields should be limited. Studies have been performed on efficient range representation for TCAM and a method to save TCAM area while providing the same performance [8–10]. However, the power dissipation problem of the TCAM severely limits its use in practice. Studies to replace TCAMs with ordinary memories using algorithmic approaches have been widely performed [11–38].

Various space decomposition algorithms have been studied such as the area-based quad-trie (AQT) [11], hierarchical intelligent cutting (HiCuts) [12], multi-dimensional cutting (HyperCuts) [13], range splitting (HyperSplit) [14], discrete bit selection (DBS), [15] effective cutting (EffiCuts) [16], ruleset splitting depending on characteristics (smartSplit) [17], and boundary cutting (BC) [18]. Each of these algorithms uses the geometric representation of rules and attempts to provide an effective search method for the geometrically represented space. They recursively decompose the space and determine the rules covering an input packet, which is represented as a point in the search space. Hence, the packet classification problem can be thought of as a point location problem in a geometrically represented search space.

Among the space decomposition algorithms, while the area-based quad-trie (AQT) provides a simple and intuitive search method using a 2-D trie structure, it has a number of issues. For two prefix fields used in constructing the AQT, the prefix information is only utilized up to the length of a shorter-length prefix. In other words, the prefix information in the longer-length prefix longer than the shorter-length prefix is not utilized in constructing the 2-D trie. As a result, each search path has a multiple number

* Corresponding author.
E-mail address: hlim@ewha.ac.kr (H. Lim).

http://dx.doi.org/10.1016/j.comcom.2017.02.005
0140-3664/© 2017 Elsevier B.V. All rights reserved.

Table 1
Example set of rules.

Rule no	Source prefix	Destination prefix	Source port	Destination port	Protocol type
R_0	010*	011*	0, 65535	1704, 1704	6
R_1	01100*	0110*	161, 161	1711, 1711	6
R_2	0110*	1001*	1024, 1024	1521, 1521	6
R_3	1010*	1101*	119, 119	1717, 1717	6
R_4	1*	10*	53,53	2110, 2110	6
R_5	00*	0*	1024, 1024	1717, 1717	6
R_6	*	110*	80, 80	1221, 1221	6
R_7	000*	*	0, 65535	0, 65535	6
R_8	001*	00*	0, 65535	0, 65535	*
R_9	00*	111*	0, 65535	0, 65535	*

of rule nodes and each input packet must be compared with an excessive number of rules.

The contribution of this paper is as follows. We propose the application of leaf-pushing [39] to the AQT in order to better map the geometrical search space into a 2-D trie. In our proposed leaf-pushing AQT, the number of rule nodes encountered in each search path is limited as a single leaf node and the number of rules compared with each input packet is limited to the number of rules stored in the leaf node. We also discuss an interesting characteristic of the leaf-pushing AQT, and describe an efficient method of utilizing the characteristic using a Bloom filter and a hash table. A refined structure of the leaf-pushing AQT which provides the controllability in classification speed is also proposed.

The remainder of the paper is organized as follows. In Section 2, related works are described such as the AQT and other space decomposition algorithms, the leaf-pushing, and the Bloom filter. Section 3 describes the proposed leaf-pushing AQT. Section 4 describes an implementation technique of our proposed leaf-pushing AQT. Section 5 shows the performance comparison of the proposed algorithm with the AQT. Section 6 presents a refined structure proposed for the performance comparison with other space decomposition algorithms and the comparison results. A brief conclusion is given in Section 7.

2. Related works

2.1. Area-based quad-trie (AQT)

Each input packet is represented as a point in the geometrically represented rule space. Space decomposition algorithms provide a search method using a trie or a tree structure to find the rules covering the point. In this paper, the term *trie* is differentiated from the term *tree* as follows. A trie is also an ordered tree-based data structure, but the trie has a unique property. In a trie structure, all the descendants of a node have a common prefix of the string associated with that node. For example, all the descendants of a node associated with string 0 are associated with a string starting with 0.

An area-based quad-trie (AQT) considers a 2-D search space composed of a source prefix in the *x*-axis and a destination prefix in the *y*-axis. Each rule is represented as a rectangular area specified by the source prefix and the destination prefix of the rule. Table 1 shows an example set of rules to describe an AQT. The wildcard represented by * means that the following bits can be either 0 or 1.

Fig. 1 shows the decomposed space specified by two fields of each rule, *F*1 and *F*2, which represent the source prefix and the destination prefix, respectively. The interval covered by a prefix in an axis is inversely related to the length of the prefix; the shorter prefix generates the larger interval. For example, the wildcard is a length 0 prefix and it covers the entire interval in an axis.

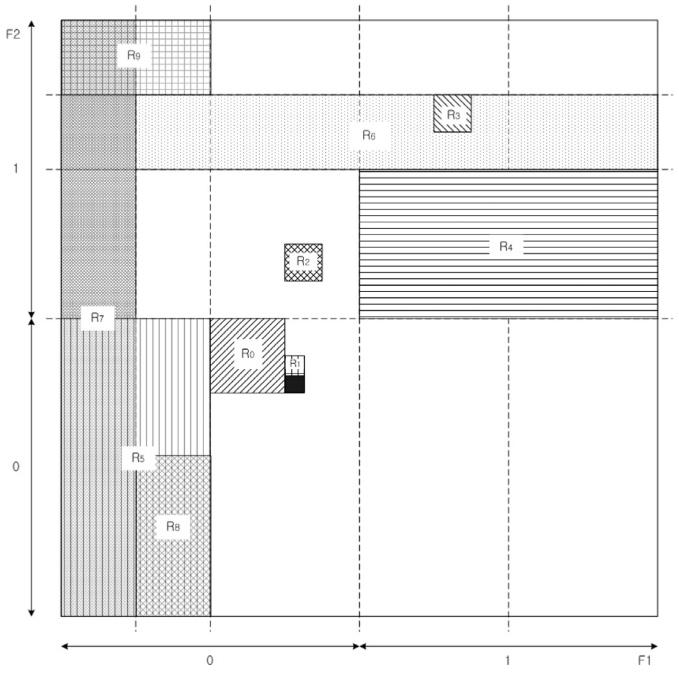

Fig. 1. Rules in decomposed space of AQT.

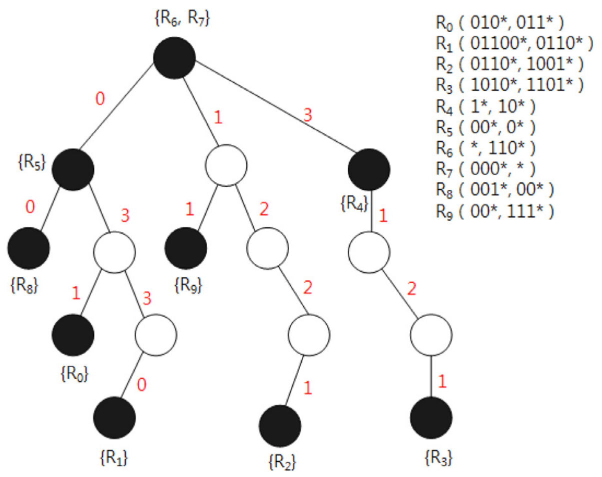

Fig. 2. Area-based quad-trie (AQT).

The AQT uses a recursive decomposition method combined with a trie structure. The AQT is a 2-D version for the packet classification of a binary trie for an IP address lookup [11]. Fig. 2 shows the AQT constructed to provide a search method for the decomposed space of Fig. 1. The search space shown in Fig. 1 is recursively decomposed by 4 equal-sized square planes (planes 0, 1, 3, and 2 by naming clockwise from the bottom-left plane). The entire search space is mapped to the root node, and the four square planes that divide the entire search space correspond to the 4 children of the root node, and so on. Among the rules included in a square plane, if any side of the rectangular area covered by a rule crosses the entire interval of the plane, the rule is included in the crossing filter set (CFS) of the plane.

The construction procedure of the AQT shown in Fig. 2 can be described using codewords that are generated by combining each bit of the source prefix and the destination prefix of the rules. Rules included in the CFS of a plane have the same codeword, which is generated based on the shorter length of the two prefixes in each rule. They are stored in the corresponding node, where the