Hierarchical Binary Search Tree for Packet Classification

Hyesook Lim, *Member, IEEE*, Ha Neul Chu, *Student Member, IEEE*, and Changhoon Yim, *Member, IEEE*

***Abstract*— In order to provide value-added services such as policy-based routing and the quality of services in next generation network, the Internet routers need to classify packets into flows for different treatments. Since packet classification should be performed in wire-speed for every packet incoming in several hundred gigabits per second, it becomes a major challenge in the Internet routers. In this letter, we propose a new packet classification scheme based on hierarchical binary search tree. The proposed scheme hierarchically connects binary search trees without empty internal nodes, and hence the proposed architecture significantly improves the search performance as well as greatly reduces the memory requirement compared with trie-based schemes.**

***Index Terms*— Packet classification, hierarchical trie, binary search tree, hierarchical binary search tree.**

I. Introduction

PACKET classification is indispensable for the Internet routers to provide the quality of services. Packet classification is to classify incoming packets into flows based on pre-defined rules so that routers can treat each packet according to the service defined in the class that the input packet belongs to [1]. Classes are defined by rules composed of multiple header fields, mainly destination prefix, source prefix, destination port number, source port number, and protocol type. Each field requires different matching operations such as the exact matching for the protocol type, prefix matching for prefixes, and range matching for port numbers. The difficulty in the packet classification is that it not only involves complicated matching operations but also has to identify the highest priority rule among all matching rules. Moreover, the packet classification should be performed in real-time for every packet incoming in several million packets per second, and hence the search speed is the major concern of the packet classification. This letter proposes a new packet classification scheme providing high throughput and low memory requirement.

Linear search is the simplest solution for packet classification and it consumes the smallest amount of memory. However, for a large rule set, the search performance of the linear search does not satisfy the real-time requirement. There are various approaches of packet classification, and they can be categorized into three groups. The first group is based on independent 1-dimensional searches. Bit-vector algorithm and cross-product algorithm are included in this group [2], [3]. The second group is based on the heuristic characteristics of classifiers, and the tuple-space search, hierarchical intelligent cutting, and hyper-cutting algorithms are included in this group [4]. The third group is based on trie for fields represented by prefixes, and the hierarchical trie, set-pruning trie, and grid-of-trie algorithms are belonged to this group [5]. Area-based quad-tree (AQT) [6] and priority-based quad-tree [7] algorithms are the 2-dimensional variation of this group. Some of the trie-based algorithms follow the hierarchical approach of the packet classification. The hierarchical approach is known to be slow because of back-tracking problem. However, we claim that, if there is not many prefix nesting relationship so that the back-tracking is not frequently occurred, the hierarchical approach achieves high speed search performance since search space is significantly reduced every time a field search is completed. The hierarchical approach of the packet classification is to recursively perform search in each field. Using the source prefix field, all matching candidate rules are identified, and for those candidates, rules are further filtered using the destination prefix field, and so on. The issue is what kind of data structures should be used for each field.

Hierarchical-trie (H-trie) [1], [4] scheme uses binary tries for the fields represented by prefixes for packet classification. Search firstly follows the source prefix trie, and the destination prefix trie is examined only when the source prefix is matched. Since packet classification is to search for the matched rule with the highest priority, even though a match is found, search has to go back to the source prefix trie and repeat until the entire source trie is examined. Set-pruning trie removes the back-tracking by copying all ancestor rules into leaves [4], and grid-of-trie removes the back-tracking by pre-computing switch pointers in each node [5]. Table I shows an example rule set which is generated using the class-bench databases [8].

TABLE I
AN EXAMPLE RULE SET

Rule No.	Src prefix	Dst prefix	Src port (start,end)	Dst port (start, end)	Protocol
0	1110*	*	53, 53	443, 443	17
1	111*	101*	53, 53	25, 25	6
2	*	10*	53, 53	25, 25	17
3	101*	11*	67, 67	5632, 5632	6
4	01*	0011*	1024, 65535	1024, 65535	6
5	101*	11*	53, 53	25, 25	4
6	11*	0100*	0, 65535	5632, 5632	6
7	101*	1011*	0, 65535	5632, 5632	6
8	*	10*	53, 53	25, 25	6
9	11*	11*	0, 15576	2783, 2783	4
10	010*	00*	53, 53	443,443	17
11	11*	00*	53, 53	25, 25	6
12	101*	0*	0, 65535	5632, 5632	6
13	01*	1*	53, 53	443, 443	17
14	01*	0*	0, 65535	5632, 5632	6

Manuscript received March 14, 2007. The associate editor coordinating the review of this letter and approving it for publication was Prof. Iakovos Venieris. This research was supported by the Ministry of Information and Communications under a HNRC-ITRC support program supervised by IITA.

H. Lim and H. N. Chu are with the Information Electronics Engineering Department, Ewha W. University, Seoul, Korea (e-mail: hlim@ewha.ac.kr).

C. Yim is with the Department of Internet and Multimedia Engineering, Konkuk University, Seoul, Korea.

Digital Object Identifier 10.1109/LCOMM.2007.070389.

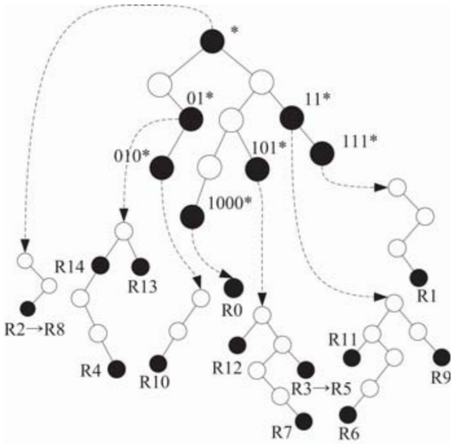

Fig. 1. Hierarchical trie (H-trie) for the rule set in Table I.

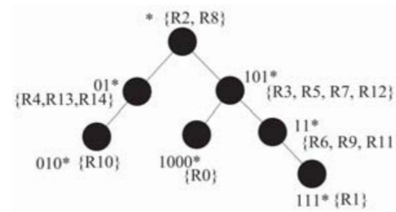

Fig. 2. Binary search tree using source prefixes for the rule set in Table I.

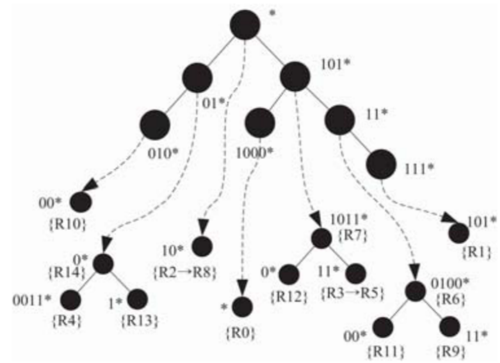

Fig. 3. Proposed hierarchical binary search tree for the rule set in Table I.

It is assumed that a smaller rule number has a higher priority. Fig. 1 shows the H-trie built using the example rule set. White nodes represent empty nodes and black nodes represent rule nodes. Dotted edges represent pointers to the next trie. In H-trie, because of empty internal nodes in both tries, the memory requirement is increased and the search performance is degraded. Especially, since the second trie is very sparse, a lot of empty internal nodes are generated.

In this letter, we propose a new packet classification scheme based on hierarchical binary search tree. By hierarchically connecting binary search trees which do not involve any internal nodes, the proposed scheme provides the improved search performance and the reduced memory requirement.

II. Proposed Algorithm

A. Binary Search Tree (BST)

Binary search has been popularly used for exact match search. For the longest prefix match in IP address lookup, Yazdani *et al.* proposed a binary search tree by defining the comparison of two prefix values with different lengths [9]. For two prefixes of different lengths, shorter length prefix is compared with the equal length sub-string of the longer length prefix. The prefix with bigger (smaller) numerical value is defined bigger (smaller). If they are the same, then the next bit of the longer length prefix is considered. If the bit is 1, then the longer length prefix is bigger, and otherwise, the shorter length prefix is bigger. By using this definition and by locating ancestor prefixes in a higher level than descendant prefixes, a binary search tree for prefixes with various lengths is constructed. In Fig. 2, we have shown an example binary search tree for unique source prefixes in Table I. Compared with the binary trie, the constructed binary search tree does not have any internal nodes.

B. Hierarchical Binary Search Tree (HBST)

We propose to use the binary search tree in the hierarchical approach of packet classification. We firstly build a binary search tree of source prefixes for a rule set, and then rules with the same source prefix would be mapped into a node of the tree. For those rules, we secondly build a binary search tree of destination prefixes and the tree is hierarchically connected with the node of the first tree. The binary search tree is much more effective than the binary trie if the number of prefixes is small and the level of prefix nesting is not many. The number of rules in packet classification is much less than the number of prefixes in IP routing table. The prefix nesting in packet classification is limited by a small number as will be shown in the next section. Therefore, using the binary search tree in the hierarchical approach is a lot more efficient than using the binary trie. We have shown the proposed hierarchical binary search tree (HBST) in Fig. 3. Compared with the H-trie, the proposed HBST does not have any empty internal nodes as shown. Especially, the destination trees are very small, and hence search can be finished with very high-speed.

Table Build The proposed HBST is implemented using two tables, a tree table and a rule table. The tree table has the information about the first binary search tree which is the source prefix, two pointers for children, and a rule table pointer. The rule table is constructed based on the second tree, and it has the rule number, the destination prefix, the remaining rule fields, two pointers for children, and a linked-list pointer. The linked-list pointer is for the rules having the same source and destination prefix pair, and they are ordered by priority. The number of tree table entries is the same as the number of unique source prefixes and the number of rule table entries is the same as rule numbers. The packet classification tables built using the proposed scheme are shown in Table II and Table III.

Table Search The source IP address of a given packet is compared with the source prefix of the root node. If the input is smaller, search follows the left pointer, and otherwise, search follows the right pointer. If the input matches, the search follows the rule table pointer in order to find out whether the input is matched for all the remaining fields. When a matched rule is encountered, the record of the current best match is kept. The linked-list pointer is only followed when

TABLE II

PROPOSED TREE TABLE FOR THE RULE SET IN TABLE I

Prefix	Length	Left	Right	RulePtr
*	0	1	2	2
01*	2	3	-	14
101*	3	4	5	7
010*	3	-	-	10
1000*	4	-	-	0
11*	2	-	6	6
111*	3	-	-	1

TABLE III

PROPOSED RULE TABLE FOR THE RULE SET IN TABLE I

Rule	Prefix	Length	SrcPort	DstPort	Protocol	Left	Right	Linked
0	*	0	53, 53	443, 443	17	-	-	-
1	101*	3	53, 53	25, 25	6	-	-	-
2	10*	2	53, 53	25, 25	17	-	-	8
3	11*	2	67, 67	5632, 5632	6	-	-	5
4	0011*	4	1024, 65535	1024, 65535	6	-	-	-
5	11*	2	53, 53	25, 25	4	-	-	-
6	0100*	4	0, 65535	5632, 5632	6	11	9	-
7	1011*	4	0, 65535	5632, 5632	6	12	3	-
8	10*	2	53, 53	25, 25	6	-	-	-
9	11*	2	0, 15576	2783, 2783	4	-	-	-
10	00*	2	53, 53	443, 443	17	-	-	-
11	00*	2	53, 53	25, 25	6	-	-	-
12	0*	1	0, 65535	5632, 5632	6	-	-	-
13	1*	1	53, 53	443, 443	17	-	-	-
14	0*	1	0, 65535	5632, 5632	6	4	13	-

the input is not matched with the current entry since rules are linked in the order of priority. If the current entry has valid child pointers, search follows a child pointer. If there are no more pointers to proceed, search goes back to the tree table. As already mentioned, the back-tracking is an intrinsic problem of the hierarchical approach, and hence the rule table is visited whenever a match occurs in the tree table. However, the number of visiting is limited by the number of source prefix nesting. As will be shown in the simulation result in the next section, the maximum number of prefix nesting is 3 or 4 in packet classification table, and hence the rule table is visited 3 or 4 times at maximum.

III. PERFORMANCE EVALUATION

Simulations have been performed using rule sets provided by class-bench [8]. Three different types of rule sets, access control list (ACL), firewall (FW), and IP chain (IPC), are generated, and the sizes of rule sets are between 50 and 5000 rules. In Table IV, we have shown performance comparison between the proposed HBST and the H-trie in terms of the average number of memory accesses (T_{avg}), the maximum number of memory accesses (T_{max}), and the required memory sizes in kbyte (M) according to the number of rules (N). The number of prefix nesting is also shown. As shown in the average and the maximum number of memory accesses, the proposed scheme shows 3 to 8 times faster search speed than the H-trie. In the required memory size, the proposed scheme requires 2 to 9 times less memory than the H-trie.

Table V shows the performance comparison with other schemes for rule sets with 5000 rules. The memory requirement of the proposed scheme is close to the linear search which is the minimum bound. In the search speed evaluated by the maximum and the average number of memory accesses, the proposed scheme is the best. H-trie shows the next best performance for IPC type and FW type. While the performance of other schemes heavily depends on the characteristics of classifier types, the performance of the proposed HBST scheme is very consistent. The simulation results also show that the proposed scheme provides very good scalability.

TABLE IV

PERFORMANCE COMPARISON FOR ACL, IPC, AND FW TYPE RULE SETS

	N	Prefix	T_{avg}		T_{max}		M (kB)	
		Nesting	H-trie	HBST	H-trie	HBST	H-trie	HBST
ACL50	49	4	83.59	10.99	108	18	11.33	1.34
ACL100	100	4	87.13	12.75	132	20	18.95	2.53
ACL500	450	4	91.94	20.64	154	55	40.94	9.46
ACL1k	958	4	77.17	17.46	124	41	82.91	20.20
ACL5k	4660	4	84.01	21.16	177	69	410.5	101.9
IPC50	50	4	63.31	10.12	103	13	13.83	1.46
IPC100	100	4	66.63	10.90	104	19	24.53	2.83
IPC500	497	4	72.49	17.44	129	30	92.33	12.63
IPC1k	988	4	71.91	21.11	128	29	121.57	22.28
IPC5k	4468	4	85.64	26.61	192	52	224.70	92.88
FW50	50	4	53.75	10.39	139	19	5.02	1.12
FW100	98	4	50.57	15.58	134	27	9.93	2.34
FW500	425	4	83.43	34.20	201	75	23.37	9.37
FW1k	871	3	52.14	19.80	117	35	39.44	19.26
FW5k	4351	3	69.20	36.75	162	72	119.10	89.88

TABLE V

PERFORMANCE COMPARISON FOR 5K RULE SETS

		Linear Search	H-Trie [4]	BV [2]	AQT [6]	PQT [7]	HBST
ACL5k	T_{max}	4660	177	76	94	113	69
	T_{avg}	2399.0	84.01	64.10	50.11	59.61	21.16
	M	86.47	410.48	2793.2	200.22	149.09	101.94
IPC5k	T_{max}	4468	192	230	415	295	52
	T_{avg}	1957.2	85.64	151.86	344.79	202..07	26.61
	M (kB)	82.90	224.70	2351.5	234.26	142.94	92.88
FW5k	T_{max}	4351	162	1044	1193	999	72
	T_{avg}	2292.3	69.20	738.75	660.12	571.06	36.75
	M (kB)	80.73	119.10	2394.7	479.77	138.98	89.88

REFERENCES

[1] H. J. Chao, "Next generation routers," *Proc. of the IEEE*, vol. 90, no. 9, pp. 1518-1558, Sept. 2002.

[2] T. V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," *ACM SIGCOMM*, pp. 203-214, Oct. 1998.

[3] F. Baboescu and G. Varghese, "Scalable packet classification," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 2-14, Feb. 2005.

[4] P. Gupta and N. Mckeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp.24-32, Mar./Apr. 2001.

[5] F. Baboescu, S. Singh, and G. Varghese, "Packet classification for core routers: is there an alternative to CAMs?," in *Proc. IEEE INFOCOM*, vol. 1, pp. 53-63, 2003.

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

[7] H. Lim, M. Kang, and C. Yim, "Two-dimensional packet classification algorithm using a quad-tree," *Computer Commun.*, vol. 30, no. 6, pp.1396-1405, Mar. 2007.

[8] D. E. Taylor and J. S. Turner, "ClassBench: a packet classification benchmark," in *Proc. IEEE INFOCOM*, pp.2068-2079, Mar. 2005.

[9] N. Yazdani and P. S. Min, "Fast and scalable schemes for the IP address lookup problem," in *Proc. IEEE HPSR2000*, pp. 83-92.