High-Speed Packet Classification Using Binary Search on Length

Mun, Ju Hyoung
Samsung Electronics
juhyoung.mun@samsung.com

Introduction

- Packet Classification is one of the major challenges for next generation routers
  - Involves complicated multi-dimensional search
  - Should be performed in wire-speed for every incoming packet

- We proposed a new packet classification algorithm which applies binary search on length to the area-based quad-trie.

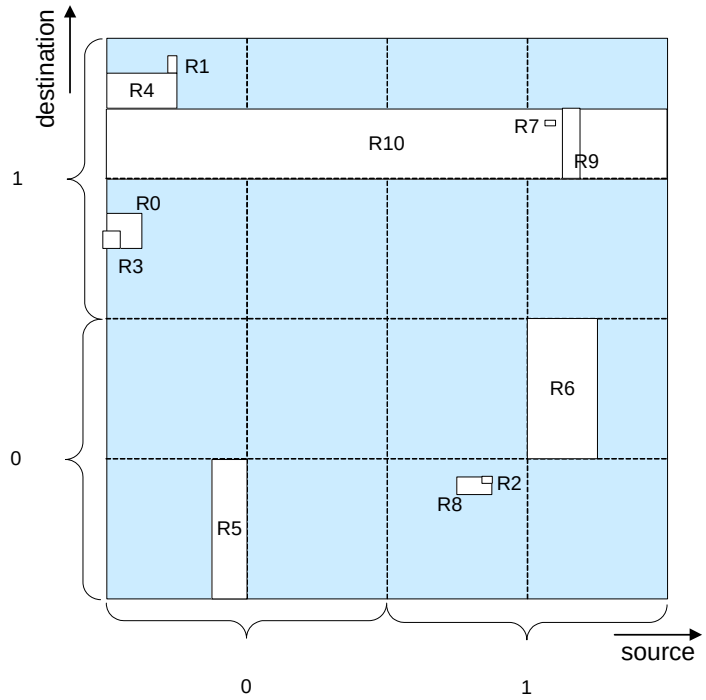
Two-dimensional Binary Trie

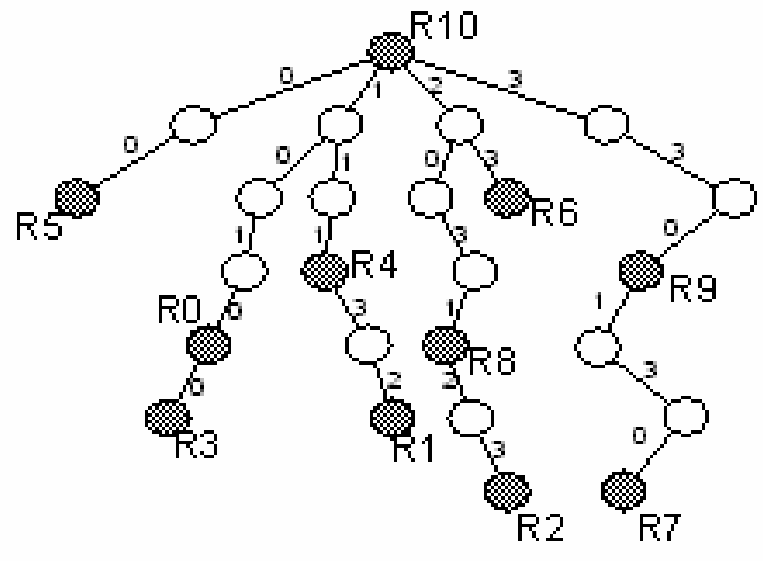
- Area-based Quad-Trie (AQT)
  - the most well known 2-dimensional search trie
  - Rules are represented by rectangles in the source and destination prefix plane
  - 2-Dimensional area of source and destination prefix is partitioned recursively
  - Each partitioned area is mapped into a node of a Quad-trie
- Crossing Filter Set
  - If any dimension of a rule completely crosses the area, it is defined as a crossing filter.

Area-Based Quad-Trie (Buddhikot, PHSN1999)

	src prefix	dst prefix	Src Port	Dst Port	Prtl
R0	0000*	1010*	53, 53	443, 443	17
R1	000111*	11110*	53, 53	25, 25	6
R2	101011*	001101*	53, 53	25, 25	17
R3	00000*	10100*	67, 67	5632	6
R4	000*	1110*	1024	1024	6
R5	0011*	00*	53, 53	25, 25	4
R6	110*	01*	0, 65535	5632	6
R7	110010*	1101100*	0, 65535	5632	6
R8	1010*	00110*	53, 53	25, 25	6
R9	11010*	110*	0, 15576	2783	4
R10	*	110*	*	*	*

Area-Based Quad-Trie (Buddhikot, PHSN1999)





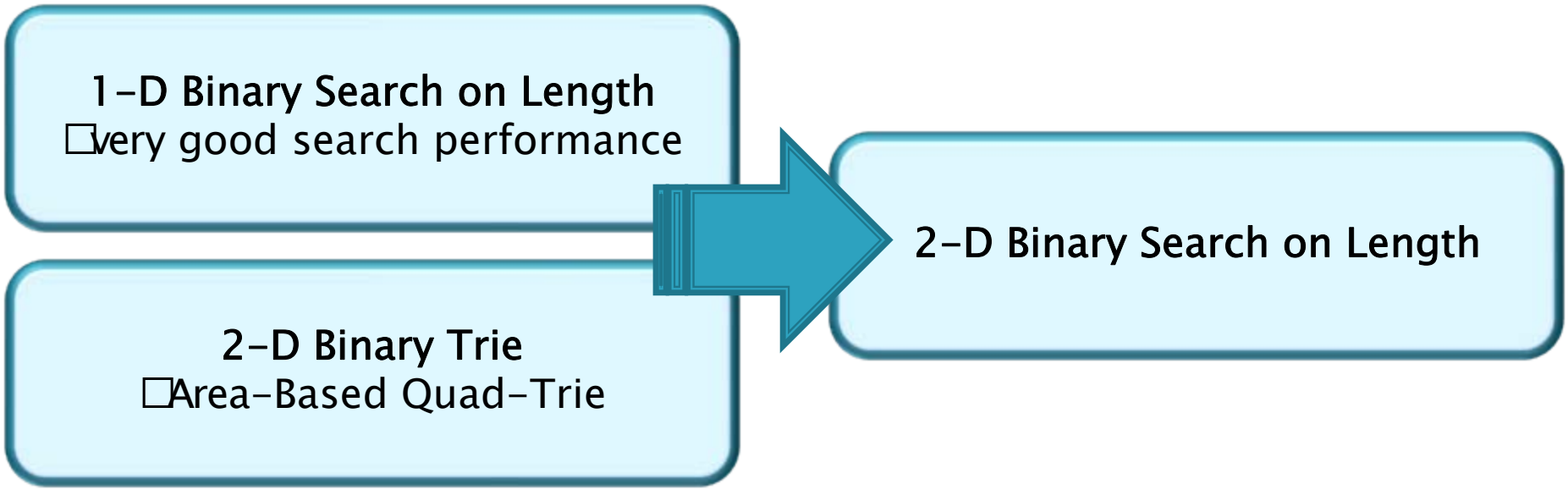
- AQT performs the linear search on prefix length, and so it does not provide good search performance.

Binary Search on Length (Waldvogel, Sigcomm1997)

- Binary search for length
- hash in accessing the actual prefix value of that length
- Excellent search performance : $O(\log_2 W)$
  - W : maximum prefix length

- Pre-computation is required
  - Markers are required in empty internal nodes to indicate the existence of prefixes in longer lengths
  - Best Matching Prefix (BMP) is noted with markers to avoid the back-tracking in case that markers mislead the search

Proposed Work

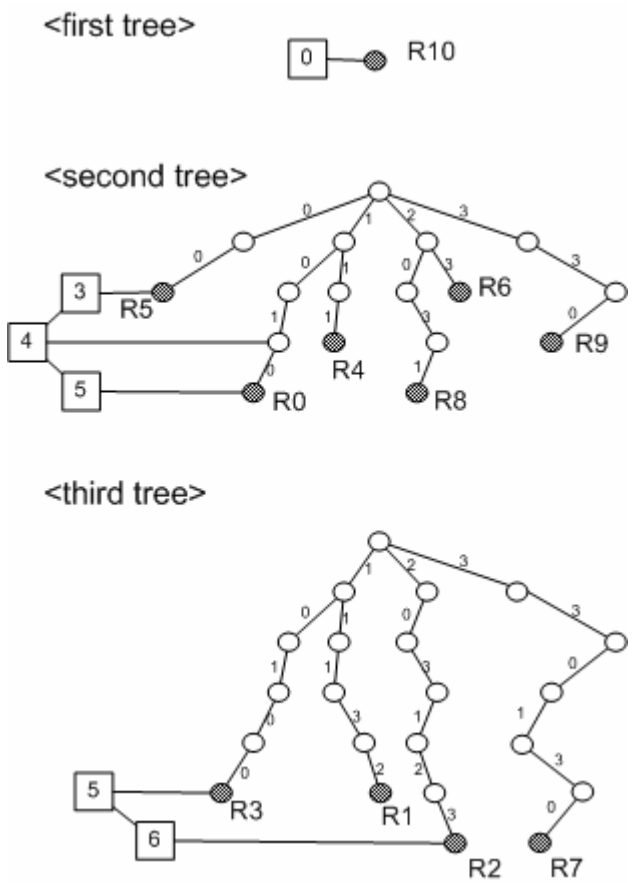


Proposed Work

- Unlike IP address lookup, the Packet Classification looks for the highest priority rule among all matching rules
  - Pre-computed markers and their best matching rules are not suitable for finding all matching rules
- Why markers and their BMPs are required?
  - Because of nesting relationship of prefixes
  - If there is no nesting relationship, the pre-computation is not necessary
- By removing the nesting relationship, we can remove the pre-computation

Proposed Work

- AQT trie is decomposed into multiple tries depending on the relative level
- So that rules located in each decomposed trie never has nesting relation.
- The number of tries would be equal to the maximum number of rule nesting
  - But there are not many levels of rule nesting in classifiers



Building the Data Structure

- The data structure is composed of two tables
  - A quad-trie table and a rule table

- To build the quad-trie tables

```
Build ( filter list )
do {
   root = Build_AQT_trie( filter list );
} while( filter != NULL )
n = 1;
do {
   Leaves = delete_nth_level_enclosures_from_AQT_Trie( root );

   delete_unnecessary_internal_nodes( root );
   nth_table = Build_Hashtable( Leaves );
   n++;
} while( root != NULL )
```

Building the Data Structure

- The rule table is generated as follows
  - The number of entries of the rule table is the same as the number of rules.
  - Each entry of the rule table has the rest of header fields
  - If there is more than one rule mapped into a node, rules should be connected by a linked-list in the order of their priorities

Searching

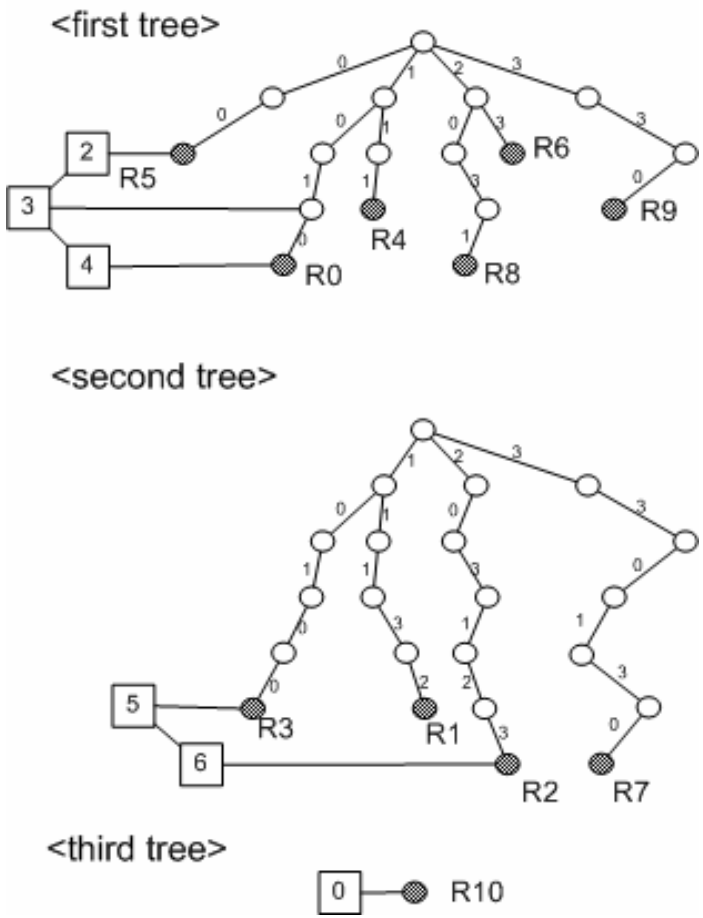
```
Search (input)
BMF = *; MF = *;
codeword = concatenation( src_address(input), dst_address(input) )
n = 1;  //start at 1st hash table
ptr = index( nth _table);
do {
    CFS = BSL( ptr, codeword );
    // search crossing filter set
    MF = Search_ruletable( CFS, input );
    if ( MF is higher than BMF )
        BMF = MF;
    n++;
    ptr = index( nth _table);
} while (ptr ! = NULL)
return BMF;
```

Optimization Techniques

- If rule priorities are considered, the performance improvement is expected.

- Optimization Technique 1 (Search Lower Priority Rules Later)
  - From the analysis of rule distribution, we found out that rules with a wild-card in any of prefix fields have low priorities in general
  - The first optimization technique is to search the trie composed of wild-card rules at the latest

Optimization Technique 1

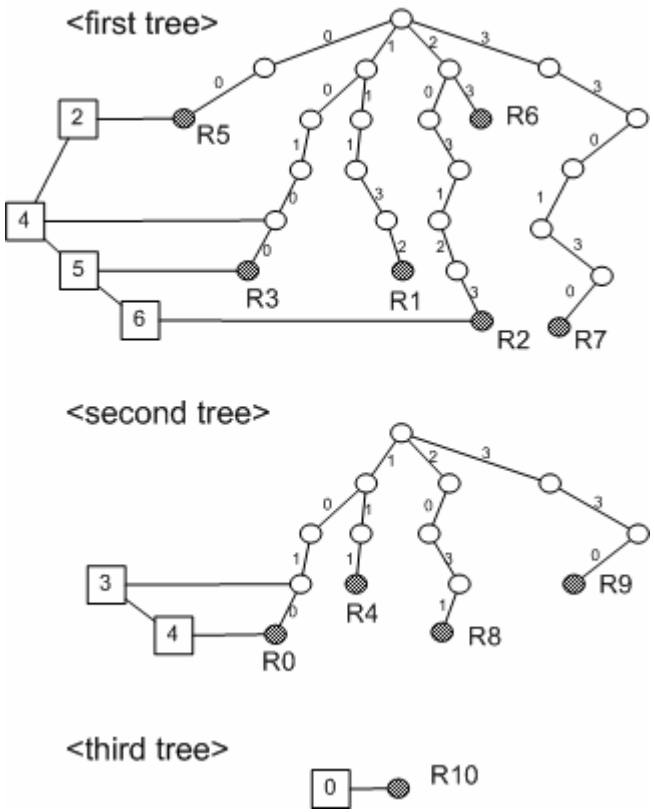


<Optimization Technique 1>

Optimization Techniques

- Optimization Technique 2 (Search Higher Priority Rules Earlier)
  - From the further analysis, we found out that there is some correlation between the length of prefixes and the priority of rules
  - Rules with longer prefixes tend to have higher priorities
  - Therefore, if we search rules in lower levels first, the better search performance is expected

Optimization Techniques 2



<Optimization Technique 2>

Simulation Results

- Performance evaluation result of proposed algorithm (ClassBench, Infocom2005)

	N	Nt	Twst	Tavg	Mtrie (Kbyte)	Mrule (Kbyte)	M/rule (byte)
ACL1k	958	5	47	21.3	23	20.2	45
ACL5k	4659	6	78	34.7	84.3	97.9	39
FW1k	870	3	293	192.7	3.5	18.3	25
FW5k	4343	3	1002	560.7	4.2	91.4	22
IPC1k	988	5	76	63.2	32.3	20.7	54
IPC5k	4467	5	263	182.3	25.5	93.8	27

Simulation Results

- Performance evaluation result of proposed optimization technique 1

	N	Nt	Twst	Tavg	Mtrie (Kbyte)	Mrule (Kbyte)	M/rule (byte)
ACL1k	958	5	43	17.34	23	20.2	45
ACL5k	4659	6	59	20.10	84.3	97.9	39
FW1k	870	3	286	115.95	3.5	18.3	25
FW5k	4343	3	971	392.54	4.2	91.4	22
IPC1k	988	5	71	33.38	32.3	20.7	54
IPC5k	4467	5	233	65.18	25.5	93.8	27

Simulation Results

- Performance evaluation result of proposed optimization technique 2

	N	Nt	Twst	Tavg	Mtrie (Kbyte)	Mrule (Kbyte)	M/rule (byte)
ACL1k	958	5	21	12.54	23.2	20.2	45
ACL5k	4659	6	39	17.91	93.8	97.9	41
FW1k	870	3	378	19.31	3.8	18.3	25
FW5k	4343	3	804	67.02	4.5	91.4	22
IPC1k	988	5	69	21.89	33.9	20.7	55
IPC5k	4467	5	234	32.30	24.5	93.8	27

Simulation Results

- Performance comparison result for ACL types

	ACL1k			ACL5k		
	Tavg	Twst	M (Kbyte)	Tavg	Twst	M (Kbyte)
H-trie [1]	77.2	124	82.9	84.0	177	401.5
HiCuts [4]	57.1	309	178.4	93.4	443	956.3
AQT [8]	38.6	64	56.4	50.1	94	200.2
PQT[9]	35.6	75	29.9	59.6	113	145.6
BV [12]	66.0	68	153.3	64.1	76	2793
Proposed	21.3	47	43.2	34.7	78	182.2
Prop opt 1	17.3	43	43.2	20.1	59	182.2
Prop opt2	12.5	21	43.4	17.9	39	191.7

Simulation Results

- Performance comparison result for FW types

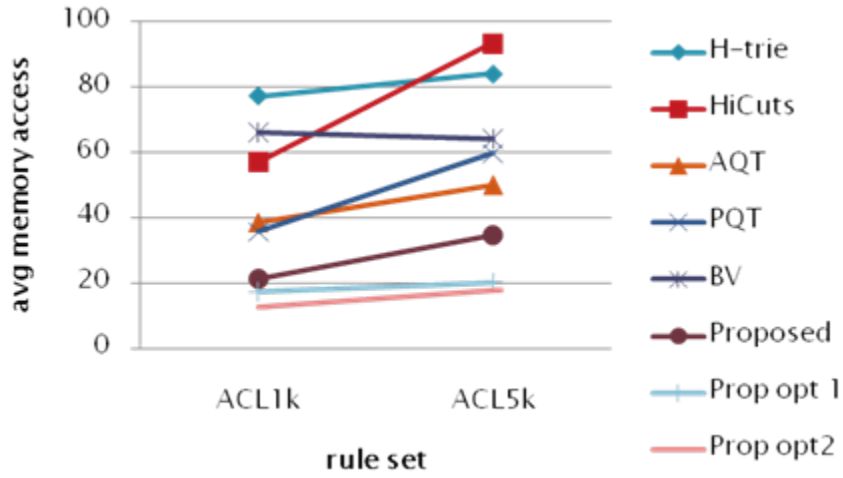
	FW1k			FW5k		
	Tavg	Twst	M (Kbyte)	Tavg	Twst	M (Kbyte)
H-trie [1]	52.1	117	39.4	69.2	162	119.1
HiCuts [4]	68.0	342	1373	411.6	1691	3704
AQT [8]	369.3	444	35.2	660.5	1193	479.8
PQT[9]	197.9	293	27.2	571.1	999	136.0
BV [12]	196.6	318	111.9	738.8	1044	2340
Proposed	192.7	293	21.8	560.7	1002	95.6
Prop opt 1	115.9	286	21.8	392.5	971	95.6
Prop opt2	19.3	378	22.1	67.0	804	95.9

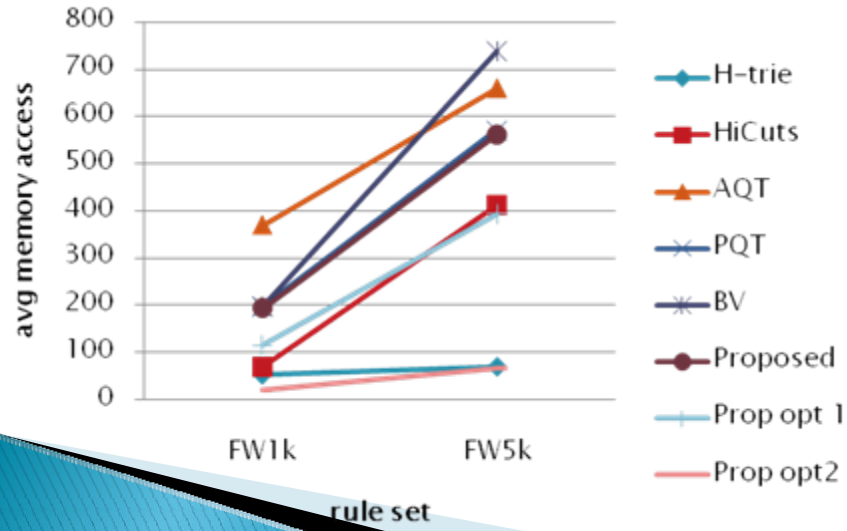
Simulation Results

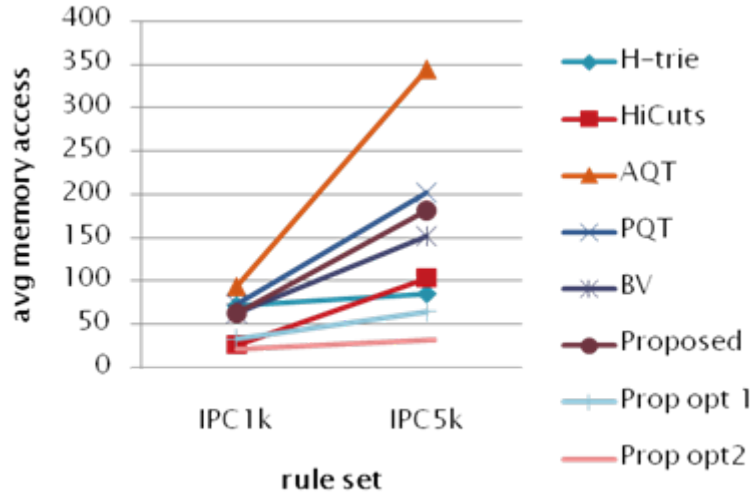
- Performance comparison result for IPC types

	IPC1k			IPC5k		
	Tavg	Twst	M (Kbyte)	Tavg	Twst	M (Kbyte)
H-trie [1]	71.9	128	121.6	85.6	192	224.7
HiCuts [4]	27.1	147	656.6	103.9	505	3060
AQT [8]	94.5	119	71.2	344.8	415	234.3
PQT[9]	73.6	106	30.9	202.1	295	139.9
BV [12]	63.6	80	154.3	151.9	230	2351
Proposed	63.2	76	53.0	182.3	263	119.3
Prop opt 1	33.4	71	53.0	65.2	233	119.3
Prop opt2	21.9	69	54.6	32.3	234	118.3

Simulation Results







Conclusion

- We proposed an efficient packet classification algorithm which shows excellent performance in the search speed and the required memory size.
- We have explored a method to apply the binary search on length for packet classification.

- The average number of memory accesses for the proposed algorithm is about 17 for a 5000 rule classifier of ACL type which is much better than related works.