International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 11 160 – 166

Real Time Packet Classification and Analysis based on Bloom Filter for Longest Prefix Matching

Ms. Namita N. Kothari
ME Information Technology2nd,
Amrutvahini College of Engineering,
Sangamner, India
namitakothari8@gmail.com

Mr. R. S. Bhosale
Associate professor, Department of IT
Amrutvahini College of Engineering,
Sangamner, India
bhos_raj@rediffmail.com

Abstract— Packet classification is an enabling function in network and security systems; hence, hardware-based solutions, such as TCAM (Ternary Content Addressable Memory), have been extensively adopted for high-performance systems. With the expeditious improvement of hardware architectures and burgeoning popularity of multi-core multi-threaded processors, decision-tree based packet classification algorithms such as HiCuts and HyperCuts are grabbing considerable attention, outstanding to their flexibility in satisfying miscellaneous industrial requirements for network and security systems. For high classification speed, these algorithms internally use decision trees, whose size increases exponentially with the ruleset size; consequently, they cannot be used with a large rulesets. However, these decision tree algorithms involve complicated heuristics for concluding the number of cuts and fields. Moreover, fixed interval-based cutting not depicting the actual space that each rule covers is defeasible and terminates in a huge storage requirement. We propose a new packet classification that simultaneously supports high scalability and fast classification performance by using Bloom Filter. Bloom uses hash table as a data structure which is an efficient data structure for membership queries to avoid lookup in some subsets which contain no matching rules and to sustain high throughput by using Longest Prefix Matching (LPM) algorithm. Hash table data structure which improves the performance by providing better boundaries on the hash collisions and memory accesses per search. The proposed classification algorithm also shows good scalability, high classification speed, irrespective of the number of rules. Performance analysis results show that the proposed algorithm enables network and security systems to support heavy traffic in the most effective manner.

Keywords- *packet classification; decision tree algorithms; bloom filter; hashing; Longest Prefix Matching*

*****

I. INTRODUCTION

Packet classification is an enabling function in network and security systems that enable routers to support access control, virtual private networks, quality of service differentiation and other value added services [1]. As shown in fig. 1, a rule consists of a set of fields, in which the most common fields are IP source prefix, IP destination prefix, source port number, destination port number, and protocol type in the packet header. The following points are derived from the study:

1) The bits in the source/destinations IP addresses in the rule-set are distributed between bits 0-4 of the first octet and bits 16-32 of the third and forth octets.

2) Specific source port numbers are identified more than specific destination port numbers in the rule-set databases.

3) Source and destination port extend in the rule-set databases are mostly of larger in size.

4) The rules with just a single destination port is more than their counterpart source-ports in the rule-set databases [16].

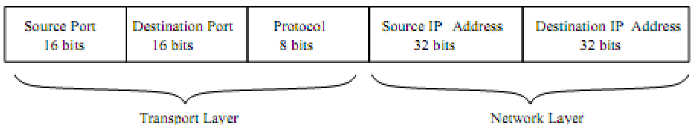


Fig. 1. Fields that are used in packet classification

As shown in fig. 2, packet classification is accomplished using a packet classifier, which is also known as a policy database, flow classifier, or simply a classifier. A policy classifier is a collection of rules or policies. Each rule determines a class that a packet may belong to based on some principle on fields of the packet header, and relates with each class an identifier. This identifier uniquely specifies the action associated with the rule that matches the packet header [2].

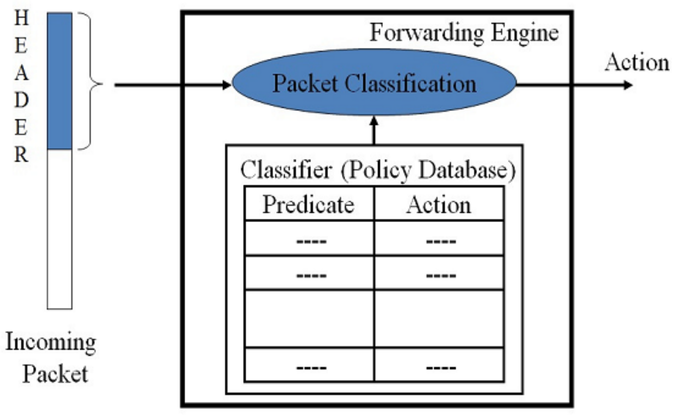


Fig. 2. Packet Classification

Many algorithms and architectures have been introduced over the years in an effort to determine an effective packet classification solution. Hardware-based solutions such as application-specific integrated circuits (ASICs) with off-chip TCAM (Ternary Content Addressable Memory) have been extensively adopted since they support wire-speed classification performance but they have a difficulty to satisfy various industrial requirements [3]. However, the high power consumption and the cost of TCAM made to seek some other algorithmic solutions. Therefore, we require a new solution to acquire high classification performance and high flexibility simultaneously. To overcome this problem, we need to develop a packet classification algorithm that performs fast classification on large ruleset size.

The algorithm should also support table sizes and high-speed table updates. Generally, packet classification algorithms use complex and large internal tables to magnify classification performance, and the size of the tables grows exponentially with the size of the rulesets [4], [5].When a table is created for a ruleset with tens of thousands of rules, the size of the classification tables is making it ineffective for most network and security platforms. One of the best ways is to break the entire ruleset into small rulesets by dividing and apply a packet classification algorithm on each subset [6]. There is no packet classification algorithm which supports a large ruleset as well as fast classification. To solve this problem, we propose a new packet classification system. The features of proposed system are summarized below.

- It holds constant high performance of packet classification disregarding of ruleset size.
- It supports larger rulesets that is almost infeasible for existing fast packet classification algorithms.
- It exterminates the inter-partition search overhead, which is a critical weakness of partitioning-based algorithms such as HiCuts and HyperCuts.
- It adapts a new classification technique that reduces redundant rules and supports fast classification using Bloom Filter data structure and LPM algorithm.

The proposed algorithm includes a new approach in holding large rulesets while maintaining packet classification performance by combining partition search tables and packet classification tables [3], [7]. Generally, the Bloom filter is used to skip lookup in some subsets which include no matching rules and to make a possibility to maintain high throughput by using Longest Prefix Matching (LPM) and hash tables [8]. However, Bloom filters provide most efficient solution for packet classification and filters large amount of packets in required time without any packet drop or missing with required optimal memory space.

II. RELATED WORK

Packet classification is a vast body of literature review. It should cover the features like, support general rules which includes prefixes, range, exact values, wildcards, better data structures to rule bases, multiple matches and preprocessing [9]. Packet classification algorithms are classified according to their implementation or characteristic types. We divide algorithms into non-partitioning and partitioning types according to the accepted partitioning techniques.

Fig. 3 shows the performance comparison of non-partitioning and partitioning algorithm [10]. Partitioning-based algorithms simultaneously fulfill two requirements:

Moderate table size and fast packet classification. Therefore partitioning-based algorithms are little slow in the maximum packet classification speed they can achieve. Whereas, the cross-producting algorithm achieve fast performance but requires a very memory requirement and very long table building time.

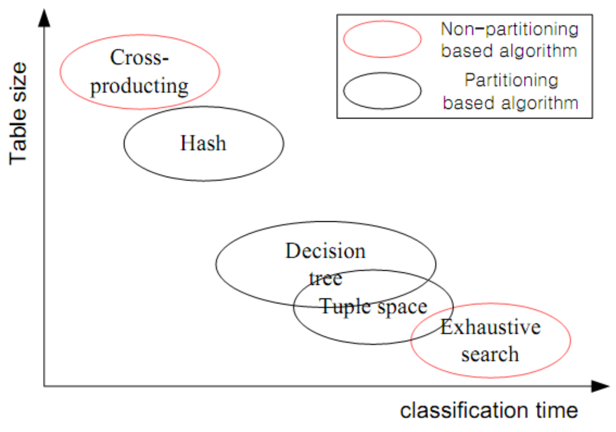


Fig. 3. Performance comparison of packet classification algorithms.

Some algorithms belonging to each category are as follows:

A. Non-partitioning-based techniques

This technique finds matching rules for the given keys by searching the entire ruleset. To acquire a high classification speed, it uses very large and complex tables; hence, it droops from large table size as the size of the ruleset increases. Hence, it is infeasible for large rulesets. Exhaustive search [11], cross-producting [5] and caching-based algorithms [12] are well-known algorithms fall into this category.

1) Exhaustive search:

A widely known exhaustive search algorithm is the linear search algorithm similar to TCAM approach, which linearly searches all the rules that are organized in the decreasing order of priority. Hence, it shows low classification speed, i.e., *O(N)*, where N is the total number of rules; whereas, the memory requirement and rule update speed are *O(N)* and *O(1)*, which are moderate compared to other algorithms. Due to these characteristics, this algorithm is adopted for small rulesets. For large rulesets, it should be used together with partitioning-based algorithms, such as hashing-based algorithms or, decision tree for intra-partition search [6].

2) *Cross-producting:*

Cross-producting algorithm finds a matching rule by blending all search results field by field using pre-built tables [4]. In short, it gives the high performance in packet classification, at the cost of large tables. For example, RFC (Recursive Flow Classification), one of the widely known cross-producting algorithms, exhibits fixed classification speed disregarding the ruleset size [5].However, it needs large table size and a long building time. The table size increases exponentially with the size of ruleset; therefore, this algorithm cannot be suitable for large rulesets. Moreover, it is not possible to support partial table update; thus, it has a long table update time. To overcome these shortcomings, tables can be built with partitioning approaches such as FRFC (Fast table building for Recursive Flow Classification) generates tables in a partitioning manner but searches in a non-partitioning manner [13]. It separates the ruleset into small partitions and builds RFC tables for every partition and these tables are then combined. As a result, FRFC increases the table building speed while sustaining high classification speed. For each partition, the time for table building decreases continuously, and the total time also decreases consequently than RFC. But FRFC is not a perfect solution because it cannot ameliorate the low scalability because of large table size.

3) *Caching-based:*

Caching-based algorithms extract pre-searched results related with keys into the cache [12]. While searching with a key, the result is rapidly retrieved via an exact matching based on a hash function. This approach is simple and productive only when the identical key is repeatedly used for searches. However, the locality of keys declines extensively in large-scale networking applications. The cache hit ratio is small, the cache update overhead is large, and the cache size increases quickly. Therefore the overall packet classification performance is declined drastically. The caching-based approach is feasible for small networks only.

B. *Partitioning-based techniques:*

Partitioning-based techniques perform packet classification effectively by decreasing the search space for the given keys by partitioning ruleset. It is not so easy to develop an optimal partitioning algorithm. Hence most of existing partitioning algorithms are based on heuristic approaches, so they cannot promise an optimal result. Most widely used partitioning-based algorithms include decision-tree, tuple-space and hash-based algorithms [14], [6].

1) *Decision tree based:*

Decision-tree-based packet classification algorithms such as HiCuts and HyperCuts show search performance by exploiting the geometrical representation of rules and searching for a geometric subspace to which input packet belongs. Decision trees decrease the size of ruleset to be searched by using tree-based data structures. A large ruleset is divided into multiple sub-rulesets. For intra-partition search, each uses a linear search algorithm. Decision-tree algorithms have various types such as basic radix trees, multi-field search trees, hierarchical trees and modified trees having smaller table size [1]. Most algorithms show adequate performance with regard to the classification speed and table size. Particularly, HyperCuts gives high search speed but the speed declines as the size of ruleset increases. Moreover, the table size gets bigger exponentially, thus, it is impossible to support large rulesets.

2) *Tuple-space based:*

Tuple-space-based algorithms like Conflict-Free Rectangle search, divide rulesets according to tuples, which consist of bit indices for different fields of a rule [14]. The algorithm is then finds an equivalent tuples for the given keys, to find a matching rule. Previous researches have shown that a tuple space is smaller than size of a ruleset. Therefore tuples searching is way faster than searching the ruleset. Each rule associated with a tuple has the identical bitmask length for any tuple’s field; therefore, adopting a hash algorithm achieves faster classification for a ruleset in the tuple. Though this algorithm has the low memory usage but needs a high preprocessing and classification time which could vary based on the nature of the rule set.

3) *Hash-based algorithm:*

Several memory accesses are needed to find a partition including the matching rule using tuple-space-based or decision-tree algorithms; therefore, these algorithms contribute limited support for quick inter-partition search. This problem is cleared by a hash-based algorithm, which creates a hash key from all or some selected keys for the similar fields. A partition is searched with one or two memory accesses [6]. Although this algorithm almost eliminates the inter-partition search overhead, the total numbers of partitions are therefore increased, generates a large table size. Thus hash-based algorithms are preferable when high packet classification performance must be achieved disregarding the memory size.

Shortcomings of algorithms come mostly from performance evaluation of algorithms and it is based on the assumptions and features where these algorithms are concentrating certain classifier and perform effectively only on this classifier. Additionally, algorithms require big memory access resulting in low speed processing. Thus, these algorithms not expected to work effectively in the case of increased requirement for next generation routers.

III. EXISTING SYSTEM

An incoming packet associates to a certain flow when all the packet fields are in the range of rule flow. In other words, each rule has F components and the *ith* component of rule R, referred to as R[i], which is a regular expression of the packet header on the *ith* field. That means, a packet (P) matches a particular rule (R) if, and only if, P is in the range of R[i] for every *ith* field of the header [2]. Packet classification by Decision tree algorithm is nothing but construction of decision tree where the leaves of the tree have rules or subset of rules. Decision tree algorithm can provide great speed search performance, if internal nodes are stored in an off-chip memory. Decision tree based algorithms such as HiCuts, HyperCuts and EffiCuts gives the highest priority match. HiCuts and HyperCuts algorithms select the field and total number of cuts on a locally optimized decision, which consists of the memory requirement and search speed. In HiCuts, each rule defines a d-dimensional rectangle in space, where d=number of fields in the rule. It recursively cuts the space into subspace with fewer overlapped rule. To find a match for the incoming packets, a linear search is performed using rules. It uses two parameters that is threshold *(binth)* and space factor *(spfac)*. HiCuts algorithm considers only one field at a time while selecting the dimensions of the cuts. While HyperCuts considers multiple fields at a time by decreasing the depth of decision tree and divide it into multiple fields. EffiCuts was used to eliminate overlap among all the rules. The researchers separated all the rules. In this algorithm, they defined rules subset to be separable if all the rules in each dimension. For each subset a separate tree is developed where the rules are separated without incurring replication. In Boundary Cutting based packet classification algorithm finds out the space that each rule covers and performs the cutting according to the rule boundary. Thus, the cutting in this algorithm is deterministic and does not involve the complicated heuristics, and it is more effective in offering efficient memory requirement [1].

IV. PROPOSED SYSTEM

Bloom filter is a space-efficient probabilistic data structure that concisely supports set membership queries. A Bloom filter is an array of m bits for representing a set S = $\{x_1, x_2, \ldots, x_n\}$ of n elements. Initially all the bits in the filter are set to zero and use k hash functions, $h_i(x)$, $1 \le i \le k$ to map items $x \in S$ to random numbers uniform ranging 1, . . .*m*. The hash functions are assumed to be uniform. An element $x \in S$ is inserted into the filter by placing the bits $h_i(x)$ to one for $1 \le i \le k$. Whereas, *y* is assumed a member of set S if the bits $h_i(y)$ are set, and assured not to be a member if any bit $h_i(y)$ is not set [2]. Algorithm 1 gives the pseudo code for the insertion operation. Algorithm 2 presents the pseudo code for the membership test of a given element *x* in the filter.

```
Data: $x$ is the object key to insert into the Bloom filter.
Function: insert($x$)
for $j$ : 1 . . . $k$ do
  /* Loop all hash functions k */
  $i \leftarrow h_j(x)$;
  if $B_i$ == 0 then
    /* Bloom filter had zero bit at position i */
    $B_i \leftarrow 1$;
  end
end
```

Algorithm 1: Pseudo code for Bloom filter insertion

```
Data: $x$ is the object key for which membership is tested.
Function: is member($x$) returns true or false to the membership test
$m \leftarrow 1$;
$j \leftarrow 1$;
while $m == 1$ and $j \le k$ do
  $i \leftarrow h_j(x)$;
  if $B_i == 0$ then
    $m \leftarrow 0$;
  end
  $j \leftarrow j + 1$;
end
return $m$;
```

Algorithm 2: Pseudo code for Bloom member test

Hash tables are widely used in many packet processing applications such as per-flow state management, packet classification, IP route lookup and network monitoring. Bloom filter uses Hash table as a data structure. Basically, many packet classification algorithms initially perform a lookup on a single header field and avail the results to avoid the search to a smaller subset of packet classifiers [5]. Since a lookup on the each and every fields can also be performed using the hash table algorithm improving the hash table performance and benefits packet classification algorithms as well.

A software based LPM algorithm used for IP lookup. The algorithm improves the performance of a regular hash table using Bloom filters. Fig. 4 illustrates this design for high-speed prefix matching [10]. The process of packet classification is divided into some basic steps. The first step is the Longest Prefix Match (LPM) operation. Then by using perfect hash function mapping, the LPM results to the rule number in order to perform fast searching. If the packet does not match any rule, the hash function will map the packet to some rule number. Since such invalid mapping can occur, it is necessary to include the further steps in which the packet is examined against the resulting rule. Hence, the complete Rule Table has to be stored in the last step.

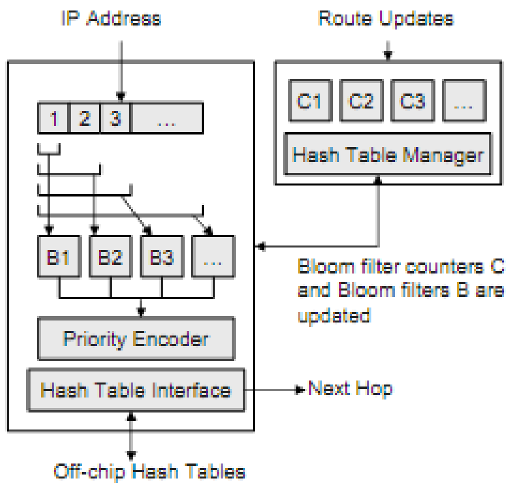


Fig 4. Longest Prefix Matching using Bloom filters

Let $P.f_i$ denote the value of field i in packet P. The packet classification process can be outlined in the following pseudo-code.

```
***ClassifyPacket(P)***
1. for each field $i$
2. $v_i \leftarrow LPM(P.f_i)$
3. $\{match, \{Id\}\} \leftarrow HashLookup((v_1, . . . , v_k))$
```

Algorithm 3: Pseudo code for classification

As the algorithm depicts, we first execute LPM on each field value. Then we search the key constructed by all the longest matching prefixes in the hash table. The result of this lookup indicates if the rule matched or not and also outputs a set of matching rule IDs relating with a matching rule [15].

V. SYSTEM ARCHITECTURE

Proposed system uses Longest Prefix Matching to avoid redundancy and optimize the tree. Hashing is used for routing and searching next hop using key and value. Bloom data structure suppresses large memory for sustaining high throughput. System architecture mainly contains 5 stages:

- Packet Capturing
- Apply LPM and Hashing
- Build data structure(Bloom filter)
- Perform packet classification
- Packet analysis and comparison using Classbench rulesets.

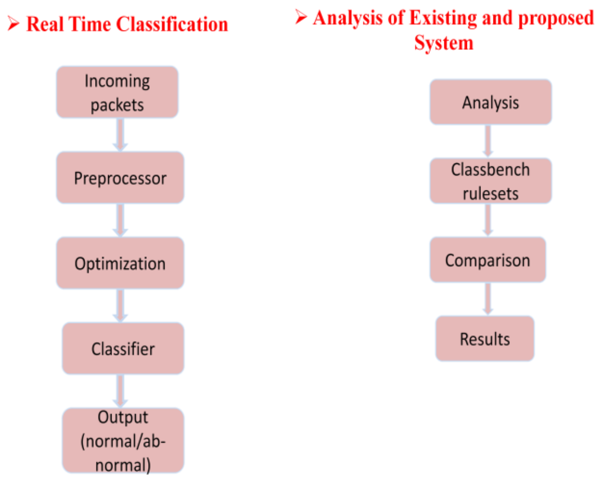


Fig. 5. Architecture of proposed system

VI. PERFORMANCE ANALYSIS AND RESULT

A. Rulesets:

Analysis using C# has been extensively performed for rule sets created by Classbench. Three different types of rule sets—Access Control List (ACL), Firewall (FW), and Internet Protocol Chain (IPC)—are formed with sizes of approximately 1000, 5000, 10,000 and 100,000 rules each. Rule sets are named using the set type followed by the size such as with ACL1K, that means an ACL type set contain about 1000 rules. The ruleset databases range in size from 68 to 4557 entries and make use of one of the following formats [16]:

1) Access Control List (ACL) – A standard format for VPN, security and NAT rule-sets for firewalls and routers (enterprise, edge, and backbone).

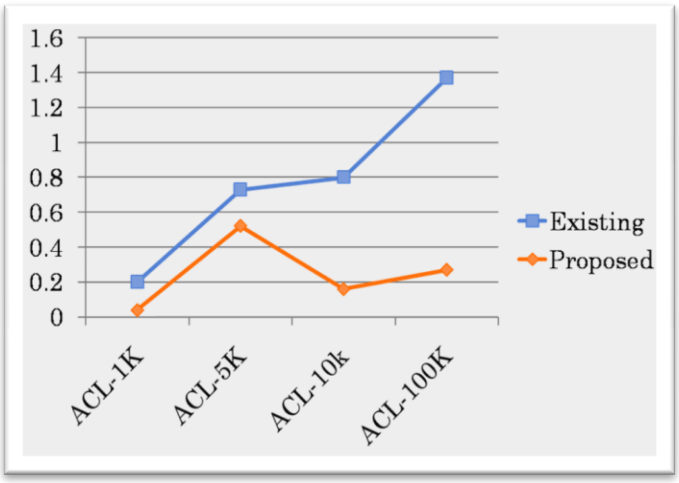
2) Firewall (FW) – A proprietary format for specifying security rulesets for firewalls.

3) IP Chain (IPC) – A decision tree format for VPN, security and NAT rulesets for software based systems.

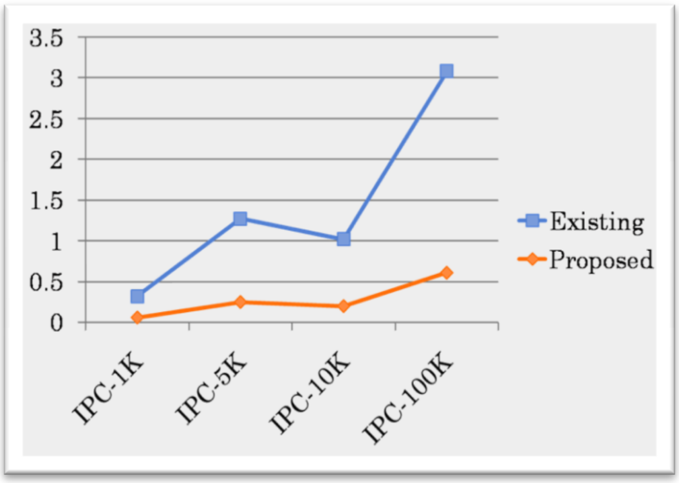
B. Memory Requirement:

The following graphs show the comparison between the existing system and proposed system in terms of memory requirement in Kb (kilobyte) per rule. The memory amount for decision tree algorithms that is boundary cutting (existing) depends on rule number and type. While, the memory amount for proposed system using bloom filter and LPM depends on hash key and value. The following graphs show that, the total memory amount (bytes per rule) required for storing rules in proposed system is less than that of existing system.

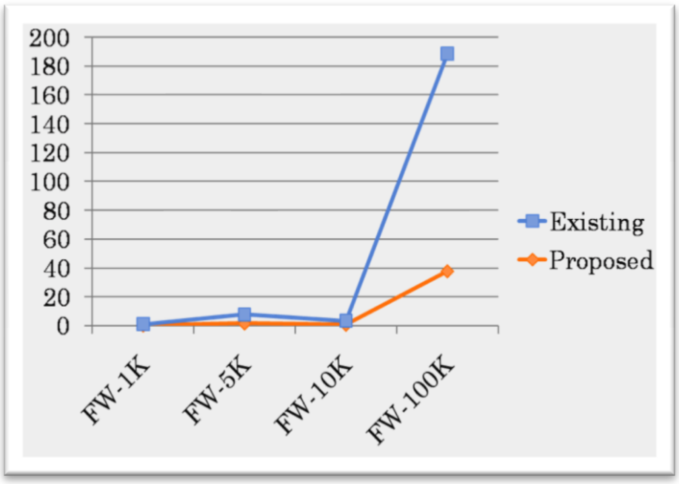
- *For ACL ruleset:*



- *For IPC ruleset:*



- *For FW ruleset:*



Conclusion

This proposed system is undertaken to design and evaluate real time packet classification in network applications to provide less memory requirement and quality of services. To construct such classification, it uses LPM and Bloom filter. Throughout the extensive analysis using Classbench databases performed between previous decision-tree algorithms that uses boundary cutting algorithm and bloom filter. Memory requirement for the proposed system is less than that of existing system. Proposed algorithm enables both the highest priority match and the multimatch packet classification.
Limitation of this system is that if the size of database is large then it might affect the performance in real-time packet classification.

References

[1] H. Lim, N. Lee, G. Jin, J. Lee, Y. Choi, and C. Yim, “Boundary Cutting for Packet Classification,” vol. 22, no. 2, pp. 443-456, April 2014)

[2] N. Kothari and S. E. Pawar, “Packet Classification based on Boundary Cutting analysis by using Bloom Filters,” ISSN: 2321-8169, Volume 3, Issue 7, July 2015.

[3] Wooguil Pak and Young-June Choi, “High Performance and High Scalable Packet Classification Algorithm for Network Security Systems,” IEEE Transactions on Dependable and Secure Computing, 2015.

[4] D.E. Taylor and J.S. Turner, ”Scalable Packet Classification using Distributed Crossproducting of Field Labels,” Proc. IEEE Conf. Computer and Communications Societies (INFOCOM ’05), pp 269-280, Mar. 2005, doi:10.1109/INFCOM.2005.1497898.

[5] P. Gupta and N. McKeown, ”Packet classification on Multiple Fields,” Proc. ACM Conf. Applications, technologies, architectures, and protocols for computer communication (SIGCOMM’99), pp. 147-160, Aug. 1999, doi:10.1145/316194.316217.

[6] L. Choi, H. Kim, S. Kim, and M.H. Kim, ”Scalable Packet Classification Through Rulebase Partitioning Using the Maximum Entropy Hashing,” IEEE/ACM Trans. Networking, vol 17, iss.6, Dec. 2009, doi:10.1109/TNET.2009.2018618.

[7] S. Dharmapurikar, H. Song, J. Turner, J. Lockwood, “Fast packet classification using Bloom filters,” in: Proc. of ANCS, 2006, pp. 61–70.

[8] A.G. Alagu Priya and H. Lim, “Hierarchical packet classification using a Bloom filter and rule-priority tries,” Comput. Commun., vol. 33, no. 10, pp. 1215–1226, Jun. 2010.

[9] H. A. J. Sistani, S. P. Amin, and H. Acharya, “Packet classification algorithm based on geometric tree by using Recursive Dimensional Cutting (DimCut),” vol. 2, no. 8, pp. 31-39, August2013.

[10] D.E. Taylor, ”Survey and taxonomy of packet classification techniques,” ACM Computing Surveys, vol. 37, iss. 3, pp. 238-275, Sep. 2005, doi:10.1145/1108956.1108958.

[11] S. Sahni, K.S. Kim, and H. Lu, ”Data structures for one dimensional packet classification using most specific-rule matching,” Proc. Parallel Architectures, Algorithms and Networks (I-SPAN ’02), pp. 1-12, May 2002, doi:10.1109/ISPAN.2002.1004254.

[12] J. Xu, M. Singhal, and J. Degroat, ”A novel cache architecture to support layer-four packet classification at memory access speeds,” Proc. IEEE Conf. Computer and Communications Societies (INFOCOM ’00), pp. 1445-1454, Mar. 2000, doi:10.1109/INFCOM.2000.832542.

[13] W. Pak and S. Bahk, ”FRFC: Fast Table Building Algorithm for Recursive Flow Classification,” IEEE Communications Letters, vol. 14, pp. 1182-1184, Dec. 2010, doi:10.1109/LCOMM.2010.100810.100572.

[14] V. Srinivasan, S. Suri, and G. Varghese, ”Packet classification using tuple space search,” Proc. ACM Conf. Applications, technologies, architectures, and protocols for computer communication (SIGCOMM ’99), pp. 135-146, Aug. 1999, doi:10.1145/316188.316216.

[15] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, “Longest prefix matching using Bloom filters,” in SIGCOMM ’03: ACM, 2003, pp. 201–212.

[16] Mahmood Ahmadi, S. Arash Ostadzadeh, and Stephan Wong, “An Analysis of Rule-set Databases in Packet classification ,” Computer Engineering Laboratory Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology.