Improving Packet Classification by Projected Tuple Pruning

Hung-Yi Chang
leorean@isu.edu.tw
Dept. of Info. Management
I-Shou University
Kaohsiung, Taiwan 840, ROC

Pi-Chung Wang, Chun-Liang Lee, Chia-Tai Chan
{abu, chlilee, ctchan}@cht.com.tw
Telecommunication Laboratories
Chunghwa Telecom Co., Ltd.
Taipei, Taiwan 106, ROC

Abstract

In next generation networks (NGNs), packet classification is important in fulfilling the requirements of multimedia services. Using pre-defined filters, the incoming packets can be categorized that determines to which forwarding class a packet belongs. Packet classification is essentially a problem of multidimensional range matching. The tuple space search is a well-known solution based on multiple hash accesses for various filter length combinations. The tuple-based algorithm, a tuple pruning search, performs well in the practical environment; however, the worst-case speed is not guaranteed. Another scheme, a rectangle search, could provide good time complexity, but it suffers from the memory-explosion problem.

In this work, we explore the relative property of filters and reorganize the filters through **filter projection**. The proposed scheme could improve the lookup speed of tuple pruning without any extra storage. We evaluate the proposed scheme with both real world and synthetic filter databases. The experimental results show that the proposed scheme increases the lookup speed by a factor of two.

Keywords: Internet, High-Speed Network, IP Address Lookup, Packet Classification.

1 Introduction

Packet classification is the process of identifying packets based on specific rules. It has been extensively employed in the Internet for secure filtering and service differentiation to reflect policies of network operations and resource allocation. The performance of packet classification is important in the deployment of differentiated services. The popularity of multimedia services would burden the filter database to a great extent. Also, packet classification with a potentially large number of filters is difficult and exhibits poor worst-case performance [4].

The classifier and the filter of packet classification must be defined before the formal description of the packet classification problem. A classifier includes a set of filters to divide an incoming packet stream into multiple classes. A filter $F = (f[1], f[2], \ldots, f[k])$ is called k-dimensional if it has k fields, of which each $f[i]$ is a variable length prefix bit string, a range or an explicit value of a packet header. A filter can be any combination of fields; the most common fields are the IP source address (SA, 32 bits), the destination address (DA, 32 bits), the protocol type (8 bits), port numbers (16 bits) of source/destination applications and protocol flags in the packet header. A packet P is said to match a particular filter F if for all i, the i_{th} field of the header satisfies $f[i]$. Each filter has an associated action. The least-cost matched filter will act to process the arriving packets, such that the packet classification problem is a least-cost problem.

Hashing is an extensively applied method for performing fast lookup. Many hash-based schemes have been presented to solve

the IP routing lookup and packet classification problem [1, 2, 10]. The tuple space search is a well-known two-dimensional (*source address prefix, destination address prefix*) solution which is based on multiple hash accesses for various filter-length combinations [6,10]. The tuple pruning search is the typical tuple-based algorithm and shows the lower bound $O(W^2)$ of the lookup speed, where W is the length of the IP address. However, the performance is not acceptable. Another algorithm, rectangle search, is proposed and shows the lower bound $O(2W-1)$ of the lookup speed, where W is the length of the IP address. The rectangle search is highly scalable with respect to the number of filters. However, it suffers from memory-explosion. For example, through experiments, the required entries are determined to be as large as twelve-fold filters

In this work, we explore the relative property of prefixes and reorganize the tuples through **filter projection**. Although two one-dimensional lookups are required, the overall performance is improved significantly in terms of speed and storage. We evaluate the proposed scheme with both real world and synthetic filter databases. The experimental results show that the proposed scheme increases the lookup speed by a factor of two without any extra storage.

The rest of the paper is organized as follows. Section 2 introduces previous works. Sections 3 describes tuple-based algorithms and filter projection. Section 4 presents the experimental setup and results. Finally, Section 5 concludes our work.

2 Previous Works

Since the past few years, researchers have been more interested in solving the packet classification problem and have proposed several algorithms for solving this issue. Maintaining the packet forwarding speed at a decent level is the major concern in the related studies. Gupta and Mckeown [6] presented the existing packet classification algorithms in detail. These algorithms could be mainly categorized into two classes: software-based and hardware-based. The following briefly describes the important properties of these algorithms.

Ternary Content Addressable Memory (TCAM) technology has advanced significantly in packet classification. However, the largest TCAMs available today can only support up to 16,000 rules. Algorithms that use conventional static random access memory (SRAM) can outperform TCAMs when large databases have to be supported. Research in scalable approaches is still required.

While the aforementioned scheme focuses on hardware approach. Several software-based schemes have been proposed [4, 5, 7, 10, 11]. The Crossproducting takes $O(N^k)$ memory [11]. The *Recursive Flow Classification* algorithm presents good lookup performance results and a good hardware implementation with moderate memory requirements for real life databases. However, it also takes $O(N^k)$ memory in the worst case, and does not allow incremental updates. In [4], the proposed algorithm decomposes the search space intelligently to find the best matching filter. The Hierarchical Intelligent Cuttings algorithm was proposed to perform packet classification at high speeds with affordable memory utilization [5]. Based on the studying of the filter databases, a scaled model is presented in [7]. The *Tuple space search* scheme partitions filters into distinct field length combinations, called tuple and searches through each tuple by hashing [10]. The disadvantages of tuple space search are non-deterministic search time and explosive storage, which are tackled in our study.

3 Improve Tuple Pruning Search by Filter Projection

The solution of tuple space is motivated by the observation that, although filter databases include several different prefixes or ranges, the distinct prefix lengths tend to be few [2]. The tuple space idea gener-

alizes the foregoing approach [2] to multi-dimensional filters [10]. A tuple is a set of filters with specific prefix lengths, and the resulting set of tuples is called a **"tuple space"**. Since each tuple has a specific bit-length for each field, these bit-lengths can be concatenated to create a hash key, which can be used in performing the tuple lookup. The matched filter can be found by probing each tuple, and tracking the least-cost filter. For example, the two-dimensional filters $F = (10*, 110*)$ and $G = (11*, 001*)$ both belong to the tuple $T_{2,3}$ in the second row and third column in the tuple space. When searching for $T_{2,3}$, a hash key is constructed by concatenating two bits of the source field with three bits of the destination field. Even a linear search of the tuple space demonstrates a considerable improvement over a linear search of the filters since the number of tuples is typically much smaller than the number of filters.

The simplest algorithm, tuple pruning search, performs lookups on individual fields to eliminate tuples that cannot match the query [10]. For each dimension, the referred information is collected in the pruning table. Next, the lookup procedure starts by searching the pruning tables. Then the set of referred tuples for each prefix are recorded and the tuples corresponding to the intersection will be probed. Since no extra entry is required besides the pruning table, it features low update cost. We use an example to explain the lookup procedure. The two filters F_1 (10*,110*) and F_2 (1010*,110010*) are located in $T_{2,3}$ and $T_{4,6}$, respectively. For the incoming packets with addresses (101000,110010), the matched prefixes of source address include 10* and 1010* which are referred by filters located in $T_{2,3}$ and $T_{4,6}$, respectively. For the destination address, the matched prefixes are 110* and 110010* which are also referred by filters located in $T_{2,3}$ and $T_{4,6}$. Hence the intersected tuples $T_{2,3}$ and $T_{4,6}$ would be probed, as shown in Figure 1. Srinivasan *et. al.,* claimed that the intersected filters are very rare in the industrial firewall database; therefore, it might perform well in the practical environment [10]. However, the worst case performance $O(W^2)$ remains the same as in the linear search.

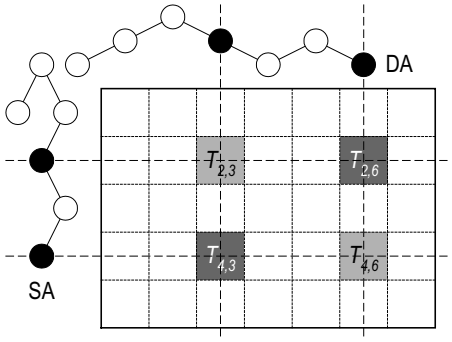


Figure 1. The Tuple Pruning Search.

The rectangle search, a tuple-based algorithm, was proposed to improve the worst-case performance of the tuple lookup [10]. The lower bound has been demonstrated to be $O(2W-1)$ given a $O(W \times W)$ rectangular tuple space, where W is the number of distinct prefix lengths. The primary aim is to eliminate a set of tuples during each probing. Tuples above T are eliminated if the probe of tuple T returns "**Match**". Otherwise, tuples to the right of tuple T are discarded. Markers and a pre-computation mechanism are required to reach this goal. Assuming that the number of filters is N, a rectangle search requires NW memory space.

To improve the tuple pruning search, we could adopt the idea based on the pre-computation concept. In the rectangle search, the pre-computation is only performed in one direction (horizontal or vertical). The pre-computation avoids the searching complexity for the less specific filter under certain conditions. For example, if the filters in $T_{2,3}$ and $T_{2,6}$ ($F_{2,3}$ and $F_{2,6}$ hereafter) are matched simultaneously, $F_{2,3}$ must be the prefix of $F_{2,6}$. Hence the hypercube defined by $F_{2,6}$ is contained by the hypercube of $F_{2,3}$. Intuitively, the action carried by $F_{2,6}$ could be one with least cost in $F_{2,3}$ and $F_{2,6}$ by pre-computation, and the

probe for $T_{2,3}$ could be eliminated. This operation could also be performed in the horizontal direction. For example, the action of $F_{2,3}$ could be duplicate to $F_{4,3}$ in $T_{4,3}$. The pre-computation is named as "**filter projection**".

In addition, the concise description of the filter projection is addressed. The projection for the horizontal direction (first dimension) is considered first. The filters are sorted according to second field's lengths in ascending order. The second dimension is used to sort in the first step. For the filters with identical length, the length of the first dimension is adopted. Also, the cost of each filter F_i is compared with the filters which has identical length in the second dimension. If the compared filter F_j is a prefix of F_i, the action of F_j is thus copied to F_i. The algorithm is shown as follows.

Horizontal Projection Algorithm
The filters are sorted based on the length of the second dimension (high priority) and first dimension (low priority) in ascending order.
For each filter $F_i(f_i[1], f_i[2])$ **BEGIN**
    Search the filter F_j which is the prefix of F_i in the first dimension and has identical second-dimension length.
    Compare the cost.
    If the cost of F_j is lower,
        the action of F_j is copied to F_i.
END

This algorithm could also be performed for vertical direction (second dimension) or both directions. In the next section, the performance for different combinations are illustrated.

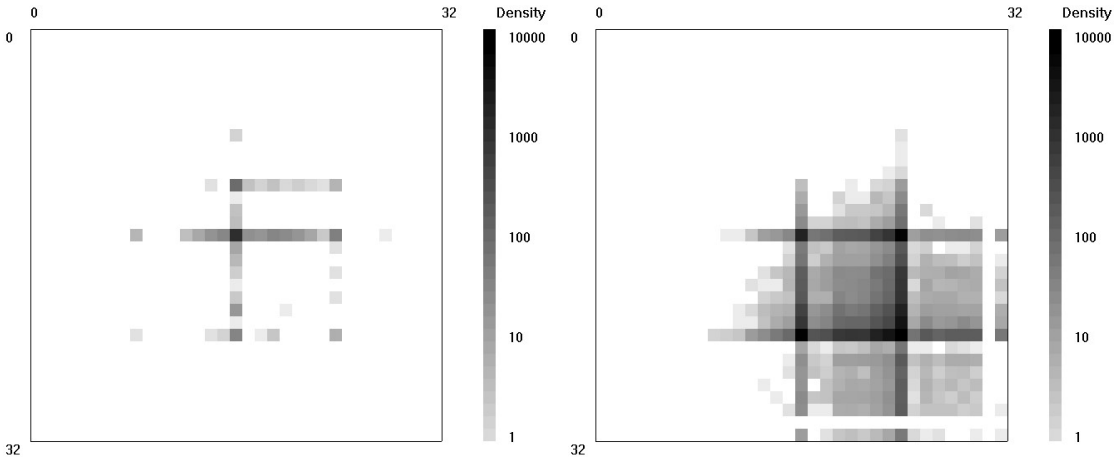
Search: The classification procedure consists of two one-dimensional table lookup and multiple tuple probes. Firstly, the BMP lookups are performed in the one-dimensional lookup tables for both dimensions. The lookup results are the sets of tuples where the prefixes of IP addresses exist. Accordingly, the sets are intersected to derive the resulted set of tuples. For the tuples in the intersected set, only the tuples at different rows and columns are probed since the actions of the filters at the same row and column are merged in the pre-computation. Hence the search cost is reduced to $O(2W-1)$ which is identical to rectangle search.

4 Performance Evaluation

This section demonstrates that the proposed algorithm increases the lookup speed through the experimental results. Ten real world filter databases are used to evaluate the performance, and the number of filters varies from 331 to 3,028. Since the industrial filter databases are relatively small, including those considered in [10] and [4], synthetic filter databases are adopted to illustrate the scalability of the proposed scheme. The routing table includes 102,309 prefixes, downloaded from the NLANR [3] as a basis for synthesizing filters. Sampling the routing prefixes randomly produces 12 ⟨source prefix, destination prefix⟩ filter databases. The minimal database contains 1K filters, and the sizes of the databases increase to 5k, 10k, 20k, 30K, and so on, and up to 100K filters. Their length distributions are shown in Fig. 2.

The speed performance is measured with five combinations: no projection, projection for column, projection for column and row, projection for row and projection for row and column. The experimental results based on the real-world filter databases are presented in Table 1. In most cases, the probed tuples are reduced due to filter projection. While the projection is performed in mere columns or rows, the tuple probes are less than three. If both the horizontal and vertical projections are used, the lookup could be completed within two tuple probes. In some cases, the sequence of performing horizontal and vertical projection could affect the performance slightly.

We present the detailed results for synthetic classifiers in table 1. The performance of the tuple pruning is relatively sta-



(a) Real World Classifier. (3,028 Entries) (b) Synthetic Classifier. (100K Entries)

Figure 2. Filter Length Distribution.

Table 1. Comparing Speed. (Real-world Databases)

Filter Databases	Tuple	Original	Col.	Col.+Row	Row	Row+Col.
331	34	3	2	2	2	2
548	58	4	3	2	3	2
651	59	4	3	2	3	2
722	65	2	2	2	2	2
986	63	4	3	2	3	2
1,130	27	2	2	1	2	1
1,142	68	2	2	2	2	2
2,278	93	3	2	2	3	2
2,300	91	3	3	2	2	2
3,028	189	2	2	1	2	1

ble as compared with the number of occupied tuple. The filter projection could further reduce at most half tuple probes. While the projection is performed in both directions, the performance is improved at least twice. Also, the sequence of performing horizontal and vertical projection does not affect the performance at all.

5 Conclusion

Packet classification is a highly effective primitive for associating a policy-defined context with each incoming packet, so as to permit packet handling using DiffServ actions to activate multimedia services. However, implementing at high-speeds with a large number of rules is difficult. This study investigates the properties of the filters and proposes the "filter projection" for improving lookup performance of the tuple pruning. Filter projection could eliminate unnecessary tuple probes by pre-computation. The experimental results obtained for real-world and synthetic databases showed that the lookup speed was increased by a factor of two. The trace-driven evaluation of the proposed scheme is left to be addressed for future research.

References

[1] Daxiao Yu, Brandon C. Smith, and Belle Wei. Forwarding Engine For Fast Routing Lookups and Updates. In *IEEE Globecom*, pages 1556–1564, November 1999.

Table 2. Comparing Speed. (Synthetic Databases)

Filter Databases	Tuple	Original	Col.	Col.+Row	Row	Row+Col.
1K	120	5	3	3	3	3
5K	120	5	3	2	3	2
10K	276	6	4	3	4	3
20K	308	8	4	3	4	3
30K	331	7	4	3	4	3
40K	340	7	4	3	4	3
50K	359	8	5	3	4	3
60K	363	8	5	3	5	3
70K	365	8	5	3	5	3
80K	369	8	5	3	5	3
90K	373	8	5	3	5	3
100K	382	7	5	3	5	3

[2] M. Waldvogel, G. Varghese, J. Turner and B. Plattner. Scalable High Speed IP Routing Lookups. In *ACM SIGCOMM*, pages 25–36, September 1997.

[3] NLANR Project. National Laboratory for Applied Network Research. See *http* : *//www.nlanr.net*.

[4] Pankaj Gupta and Nick McKeown. Packet Classification on Multiple Fields. In *ACM SIGCOMM*, pages 147–160, September 1999.

[5] Pankaj Gupta and Nick McKeown. Packet Classification using Hierarchical Intelligent Cuttings. In *Hot Interconnects VII*, August 1999.

[6] Pankaj Gupta and Nick McKeown. Algorithms For Packet Classification. *IEEE Network Magazine*, 15(2):24–32, 2001.

[7] Thomas Woo. A Modular Approach to Packet Classification: Algorithms and Results. In *IEEE INFOCOM*, pages 1213–1222, March 2000.

[8] T.V. Lakshman and D. Stidialis. High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching. In *ACM SIGCOMM*, pages 203–214, September 1998.

[9] V. Srinivasan and G. Varghese. Fast IP lookups using controlled prefix expansion. *ACM Trans. On Computers*, 17:1–40, Febuary 1999.

[10] V. Srinivasan, G. Varghese and S. Suri. Packet Classification using Tuple Space Search. In *ACM SIGCOMM*, pages 135–146, September 1999.

[11] V. Srinivasan, G. Varghese, S. Suri and M. Waldvogel. Fast Scalable Level Four Switching. In *ACM SIGCOMM*, pages 191–202, September 1998.