Transactions on Computer Science and Technology
December 2015, Volume 4, Issue 4, PP.59-65

Scalable Multi-Field Packet Classification Using Fixed-Length Compression Bit Vector

Jiaqi Gao [1†], Xiaohe Hu [1, 2], Jun Li [2, 3]

1. Department of Automation, Tsinghua University, Beijing 100084 China

2. Research Institute of Information Technology (RIIT), Tsinghua University, Beijing 100084, China

3. Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing 100084, China

[†]Email: gaojq12@mails.tsinghua.edu.cn

Abstract

Packet classification is one of the core techniques in network devices such as firewall, IDS and IPS. Tens of thousands of rules pose great pressure on classification algorithm in terms of memory requirements and throughput. In this paper, a new packet classification algorithm, named Fixed-Length Compression Bit Vector (FLCBV), is proposed based on the compression scheme FLC. Compared with traditional algorithms, FLC is more adaptive to network environment, especially in large scale ruleset and multi-field packet header. Experimental results demonstrate that FLCBV takes only 65% of memory compared with traditional compression algorithms without losing much throughput.

Keywords: Packet Classification; Bit Vector; Compression

1 INTRODUCTION

Packet classification is one of the most critical techniques for many network applications, such as access control, QoS, and traffic monitoring. It remains a challenging problem for future network elements, such as open flow switches and virtual network functions. The requirements on latency, throughput, and storage become stricter with the increasing complexity of rulesets. Among those requirements, storage is always the most crucial one.

To the best of our knowledge, few algorithms work under 100MB memory in case of large scale rulesets, i.e. rule number of 10K and larger, without additional compression or grouping. But most of the conventional compression algorithms are designed for data storage, performing well for mass volume of data. As for network applications, those algorithms cannot guarantee optimal compression ratio for rule storage. Also, they are not suitable for frequently access and intensive bitwise operations.

In this paper, a novel classification algorithm called Fixed-Length Compression Bit Vector (FLCBV) is proposed. It is based on a purposely designed bit vector compression algorithm FLC and a high throughput, high memory requirement packet classification algorithm Bit Vector (BV) [2]. Experiment shows that FLCBV, compared with other compressed BV algorithms, significantly reduces memory occupation without much performance degradation.

The rest of the paper is organized as the following: Section 2 provides a brief introduction of related work; Section 3 introduces the compression algorithm; Section 4 shows how FLCBV works, and Section 5 illustrates the experiment results of FLCBV comparing with other algorithms. Conclusion and future work are discussed in Section 6.

2 RELATED WORK

Currently, there are two major types of packet classification approaches. Algorithms such as HiCuts [4], HyperCuts [5], HyperSplit [6] directly operate on the entire rule space, and partition the k-dimensional space into smaller subspaces by hyperplanes that are parallel with the corresponding dimensions. Although these dimension-dependent algorithms perform well with small scale rulesets, when the ruleset size is over 10K or the number of dimensions increases to more than 5, their pre-processing time and memory requirement increase dramatically, along with

serious throughput drop. The drawbacks block the implementation of these algorithms in backbone network, which updates frequently and requires high performance.

The other type of approaches processes each dimension independently and then cross-product the partial results. Lakshman and Stiliadis [2] proposed a scheme of bit vector, which projects every rule onto each dimension to build k one-dimension tries. In each trie, each node stands for a prefix, which can be calculated from the path to the root. The node is associated with a bit vector, where each rule is represented with a bit in the vector. Each bit position maps to a corresponding policy, 1-bit is set when the policy hits the node's prefix. When a new packet arrives, each dimension of the header is searched through its corresponding trie and returns the bit vector associated to the output node. Since the bits in each vector are sorted by descendent order of priority, an AND of the k bit vectors will reveal the matching rules, and the first non-zero bit indicates the best matching rule. An example is illustrated in FIG. 1.

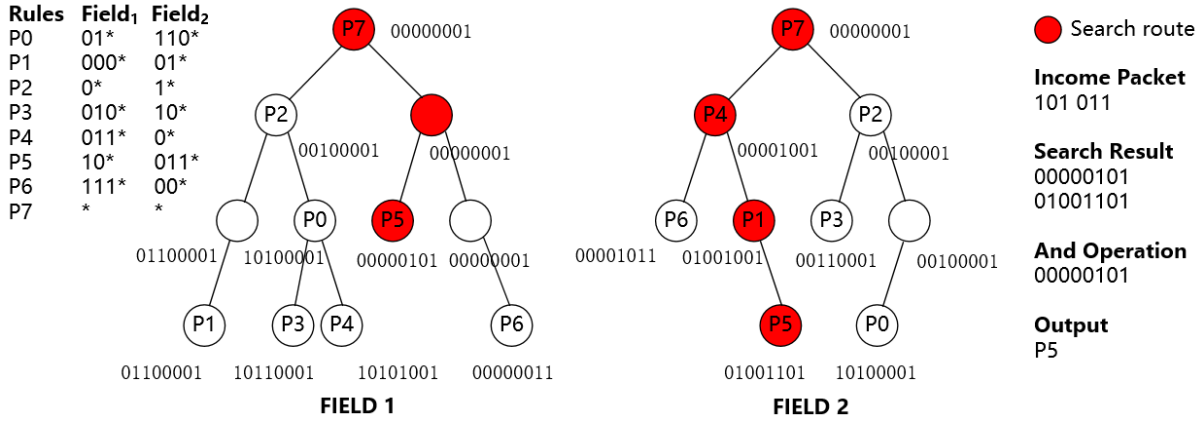


FIG. 1 EXAMPLE BUILD AND SEARCH PROCEDURE IN BV

As the length of the bit vectors is the number of the rules, the memory requirement of the algorithm is determined by node number, rule number, and dimension number. Surely this algorithm works out well with middle scale rule size, but when it comes to the backbone network, tens of thousands of rules will easily consume gigabytes of memory.

The memory issue seems to be easily solved by applying a high compression ratio algorithm to the bit vectors. But the result turns out not practical. The reason is going to be discussed it in Section 3.

3 COMPRESSION ALGORITHM ANALYSIS

In BV, the bit vector of a node carries the information of position, while in fact each packet will only hit a few rules. The distribution of 1-bit (bit value set to 1) in bit vector is sparse. However, some dimensions of ruleset, like those corresponding to Port or Protocol packet header fields, often block or permit the same kinds of packets, such as Port 22 or UDP protocol, which will introduce consecutive 1-bit bits in some dimensions of the bit vectors.

As mentioned in Section 2, the BV algorithm executes AND operation extensively, since the length of the bit vector is equal to the number of rules, say 10K, while the depth of the trie is determined by the bit length of each packet header field. The compression algorithm has to be optimized for bit operation, especially AND operation.

Considering the above characteristics of the bit vectors and the BV algorithm, the requirements of the compression algorithm is listed below.

- Consecutive series of 0-bit or 1-bit can be compressed effectively.
- The order of priority in the compressed vector can be preserved.
- Compressed vector cross-production can be carried out with AND operations directly, without decompressing the whole vectors.

3.1 Enhanced Word-Aligned Hybrid (EWAH)

EWAH is a bit vector compression algorithm that supports direct Boolean operation. Its compression is based on 32

bits (word). Firstly, it classifies words into two categories, *clean word*, in which all of the 32 bits are of the same value, 0 or 1, *dirty word*, which indicates other circumstances, i.e. the words not filled with bits of the same value. Then EWAH compresses those words into two types of words. The first type is *verbatim (or dirty) word*. The second type of words is called *marker word*: its first bit indicates which clean word (0 or 1) will follow, the subsequent 16 bits are used to store the number of clean words, and the rest 15 bits are used to store the number of dirty words following clean words. An example is illustrated in FIG. 2.

One common characteristic of the Variable-Length Compression (VLC) algorithms, including EWAH, is that they compress variable length bit sequences, which is usually an integer multiples of fixed length bit sequences. This kind of variable-length algorithms maximize the compression ratio, however, they often perform badly at AND operation, especially multiple AND operations of bit vectors at the same time, because the original length of each section is arbitrary, and we have to AND them one by one. Therefore, VLC algorithm is not suitable for BV.

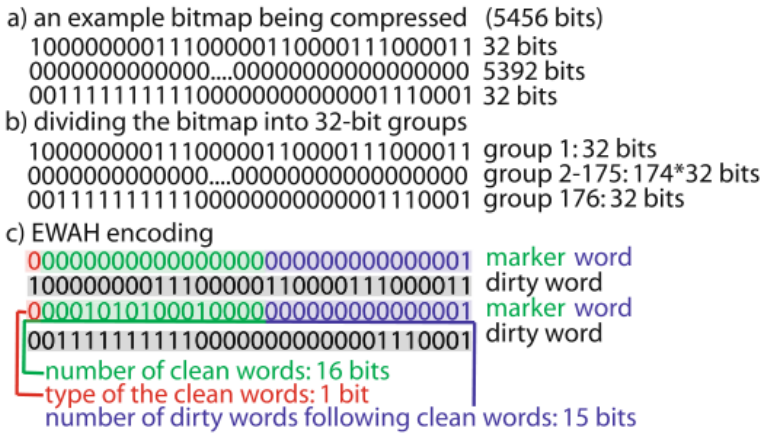


FIG. 2 EWAH COMPRESSION EXAMPLE

3.2 Fixed-Length Compression

Fixed-Length Compression is proposed to meet the requirement of BV for AND operation. As its name suggests, FLC is based on fixed length short sequences. The main steps are listed below.

Firstly, the bit vector is divided into 32 bits long bit sequences, called *subsequences*.

Secondly, the subsequences are compressed separately. In order to maximize the compression ratio, the bit vectors in several rulesets with 10K rules are studied statistically, and the following types of subsequences have the highest probability.

- **0-Fill**: All of the 32 bits are 0.
- **1-Fill**: All of the 32 bits are 1.
- **0-Dirty**: Divide the 32 bit subsequence into four 8 bit vectors, if only one of those vectors is not all 0, we call that vector a *dirty vector*, and the subsequence is 0-Dirty.
- **1-Dirty**: Similar to 0-Dirty, 1-Dirty means only one of the 8 bit vectors is not all 1.

TABLE 1 TYPE OF SUBSEQUENCE AND THEIR COMPRESS RESULTS

Type	Result
0-Fil	0
1-Fill	10
0-Dirty	110 + the position of dirty vector (2 bits) + content of dirty vector
1-Dirty	1110 + the position of dirty vector (2 bits) + content of dirty vector
Else	1111 + content of the subsequence

Tags are used to distinguish different types of subsequences. Leveraging the idea of Huffman Coding, compression results are listed in TABLE 1 and FIG. 3.

a) an example bit vector being compressed (hex)
000000000 FFFFFFFF 00080000 FFFF2FFF 148EA3F2
b) FLC encoding (bin)
0 10 1100100001000 1110100010111
1111000101000 10001110 10100011 11110010

FIG. 3 FLC COMPRESSION EXAMPLE

4 FIXED-LENGTH COMPRESSION BIT VECTOR

In order to reduce the memory usage of BV algorithm in case of large scale rulesets, FLC and BV are integrated to a new scheme called FLCBV.

FLBVC is designed with two phases, namely, building phase and searching phase.

4.1 Building Phase

In building phase, *k* one-dimension tries are built when there are *k* fields in the packet header. To build each trie, we need to traverse each rule. The whole iteration takes the following steps:

Create the node

When iterating, each dimension of the rule is searched from the root of corresponding trie. The nodes are built dynamically when searching. Search length is determined by the prefix. When complete, the corresponding bit in the node's bit vector is set.

Update the trie

With the prefix structure of the trie, each node will contain the classification information of its parent node, while step 1) will miss some nodes. As a result, the trie needs to be updated. The trie is traversed and the information is passed down by OR operation. These two steps are the same as traditional BV algorithm, which means after this two step, the result is the same as traditional BV algorithm.

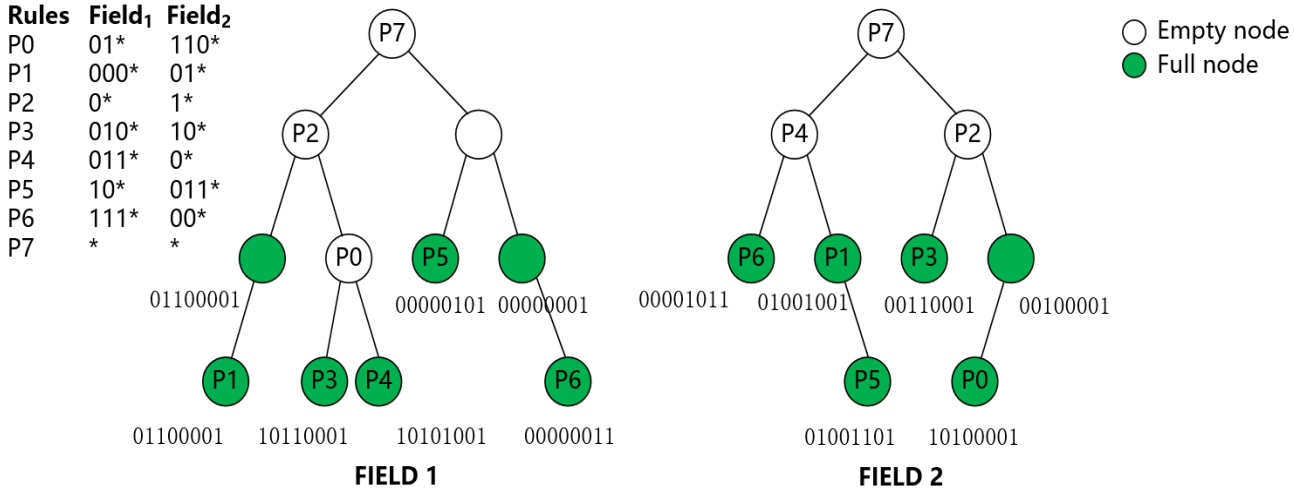


FIG. 4 EXAMPLE RULES AND BUILDING RESULT

Delete the unnecessary node

The primary difference between packet and rule is packet doesn't contain mask. That means during the classification, nodes with both children existing are out of the search range. Then the bit vectors of unnecessary nodes can be deleted, which are called Empty Nodes. The nodes with bit vector are called Full Nodes. FIG. 4 illustrates an example of FLCBV trie building.

4.2 Searching Phase

In the searching phase, when a packet arrives, each field of its header traverses the corresponding trie, then *k* bit vectors are obtained. In the original BV algorithm, AND operation will be applied on these vectors and to determining the highest priority matching rule. But this is infeasible for compressed bit vectors, and the decompression will take too much time on the other hand.

When analysing the details of AND operation on bit vectors, we found that it begins from the highest priority position, and ends when meeting the first 1-bit. This means that most of the outputs of the AND operations are 0-bits. A new decompression scheme named Partial Decompression is proposed accordingly. In this scheme, the bit vectors are not decompressed completely, as we only need to know the type of the decompressed bit, or where the 0-bits are in the decompressed bits. Leveraging this scheme, only the first several bits are consumed without processing the whole compressed vector. The partial decompression result is an 8 bit long feature code, which indicates the position of 0-bits. The feature code is listed in TABLE 2.

TABLE 2 TYPE OF SUBSEQUENCE AND THEIR FEATURE CODE

Type	**Feature code**			
0-Fil	00000000			
1-Fill	11111111			
0-Dirty	11000000	00110000	00001100	00000011
1-Dirty	11111111			
Else	11111111			

Then we can AND those feature code, if the result is 0, then we do not need to decompress those subsequence, because the result must be 0. If the result is not 0, then we will have to perform decompression but only once to find the position of the first 1-bit. A typical example is shown in FIG. 5.

a) Two compressed bit vector(bin)
0 10 1100100100000 1101100100101 1110101101010
10 0 1101001001000 1101111000010 1110000000001
b) Feature code (bin)
00000000 11111111 00110000 00000011 11111111
11111111 00000000 00001100 00000011 11111111
c) AND result (bin)
00000000 00000000 00000000 00000011 11111111
d) Uncompress the fourth subsequence and check the result
00000000 00000000 00000000 00100101
00000000 00000000 00000000 11000010
00000000 00000000 00000000 00000000
e) Uncompress the fifth subsequence and check the result
11111111 11111111 11010010 11111111
00000001 11111111 11111111 11111111
00000001 11111111 11111111 11111111
f) Output the result
32 × 4 + 7 = 135

FIG. 5 EXAMPLE OF PARTIAL UNCOMPRESS, **AND** OPERATION

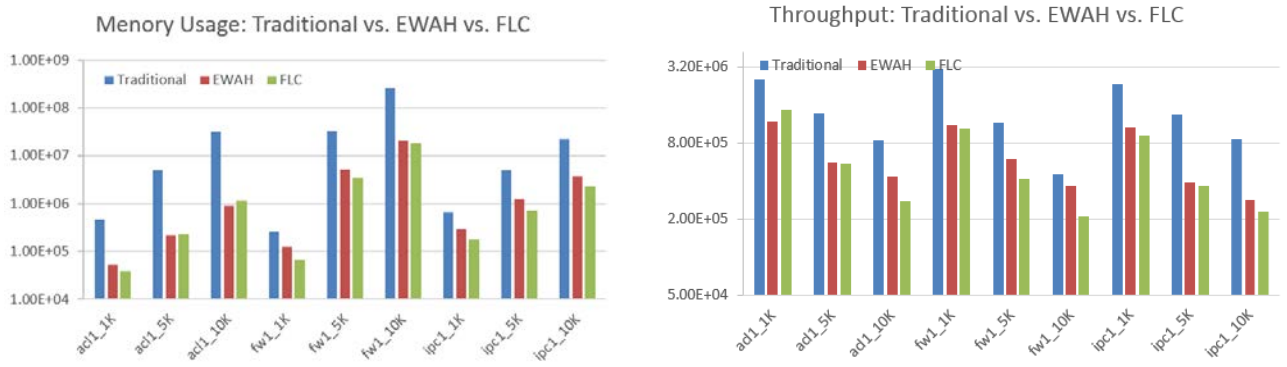
5 PERFORMANCE EVALUATION

In this section, the performance of the proposed FLCBV algorithm, as well as the original BV and BV with EWAH, is evaluated and compared using different types of rulesets.

The rulesets used in our test are publicly available [7], containing three types of rules: Access Control List (ACL), Firewall rules (FW) and IP Chans (IPC). After each type name of rules is the rule size, e.g. acl1_1K refers to the ACL rule containing about 1K rules. All the rules are 5 dimensional, 32 bit source/destination IP addresses with

prefixes, 16 bit source/destination port numbers, and 8 bit transport layer protocol.

Three primary measures of the performance, including memory usage, throughput, and pre-processing time, are obtained on a HP Z228 workstation with Intel i7-4790 CPU and 16GB memory. All the code is written in C++ and compiled with –O3 parameter in Ubuntu 14.04. Main performances are shown in FIG. 6.



(a) (b)

FIG. 6 MEMORY AND THROUGHPUT RESULT: TRAD. VS. EWAH VS. FLC

5.1 Memory Usage

The memory usage depends on the ruleset, even rulesets with the same size may result in various memory consumption. Different types of rulesets will generate different sizes of tries, no matter in the depth or the width. Theoretically, the memory usage depends on the number of nodes and the size of the bit vector. The type of rulesets usually determines the number of nodes, such as ACL focuses on some certain ranges while FW usually covers all of the IP ranges; on the other hand, the size of ruleset determines the size of the bit vector.

When a compression algorithm is involved in, both the type and the size of rulesets will affect the size of the compressed bit vectors, as different types of rulesets will affect the distribution of 0-bits and 1-bits, so does the size of rulesets.

The result is illustrated in FIG. 6 (a). It can be seen that both EWAH and FLC achieve very good compression ratio, and FLC outperforms EWAH significantly in FW and IPC rulesets. Statistically, FLC takes about 65% memory of EWAH.

5.2 Throughput

It has been observed from FIG. 6 (b) that FLC is a little bit slower than EWAH. This is because EWAH can compress a long sequence of 0-bit to a 32 bit word, while FLC can only compress 32 0-bit to one bit. In each AND operation cycle the step length is different, which is 32 bits for FLC, several 32 bits for EWAH. The approach to improve throughput is discussed in the future work.

5.3 Pre-processing Time

Pre-processing speed is an important measure for the packet classification algorithms. For BV, the building phase only takes several milliseconds, which means that update of rulesets can be conducted very efficiently by rebuilding the trie. This is very convenient comparing to other packet classification algorithms like HyperSplit, which will take seconds or even minutes to compile a ruleset of size 1K or larger.

6 CONCLUSION AND FUTURE WORK

In this paper, we design a new compression scheme called Fixed-Length Compression, and combine it with BV, propose a novel packet classification algorithm, FLCBV. Unlike other existing algorithms, FLC employs fixed bit length to achieve better flexibility and higher compression ratio. Experiments show that FLCBV significantly outperforms prior compressed BV algorithms in memory usage without losing much throughput.

In the future, we will deploy FLCBV on hardware platforms, such as GPU and FPGA, to fully leverage on the strength of FLC, and FLCBV is expected to breakthrough its performance bottleneck with the help of parallel computing and hardware acceleration.

REFERENCES

[1] Hsieh C L, Weng N. Scalable Many-Field Packet Classification using Multidimensional-Cutting Via Selective Bit-Concatenation [C]//Proceedings of the 11th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). 2015: 187-188

[2] Lakshman T, Stiliadis D, High-speed policy-based packet forwarding using efficient multi-dimensional range matching[C]//ACM SIGCOMM 1998. 1998: 203-214

[3] Lemire D, Kaser O, Aouiche K. Sorting improves word-aligned bitmap indexes[J]//Data & Knowledge Engineering, 2010, 69(1): 3-28

[4] Xu B, Jiang D, Li J. HSM: A fast packet classification algorithm[C]//IEEE 19th International Conference on Advanced Information Networking and Applications (AINA). 2005: 987-992

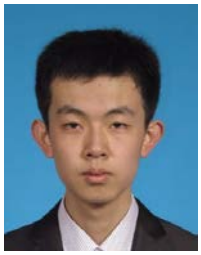
[5] Gupta P, McKeown N. Packet classification using hierarchical intelligent cuttings[C]//Hot Interconnects VII. 1999: 34-41

[6] Qi Y, Xu L, Yang B, Xue Y, Li J. Packet classification algorithms: From theory to practice[C]//IEEE INFOCOM 2009. 2009: 648-656

[7] http://www.arl.wustl.edu/~hs1/PClassEval.html

AUTHORS

[1]Jiaqi Gao, born in 1993, is an undergraduate student in Department of Automation, Tsinghua University. His research interests include packet classification and SDN.



[2]Xiaohe Hu, born in 1992, received the B.S. degree from the Department of Automation, Tsinghua University in 2014. He is now a PhD candidate in Department of Automation, Tsinghua University. His research interests include SDN and cloud datacentre networks.

[3]Jun Li, born in 1962, Ph. D., professor. His research interests include network security and architecture.