Abbasi et al. EURASIP Journal on Wireless Communications and Networking (2019) 2019:287
https://doi.org/10.1186/s13638-019-1617-8

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Ingredients to enhance the performance of two-stage TCAM-based packet classifiers in internet of things: greedy layering, bit auctioning and range encoding

Mahdi Abbasi[1*], Shakoor Vakilian[2], Ali Fanian[2] and Mohammad R. Khosravi[3,4]

Abstract

Using packet classification algorithms in network equipment increases packet processing speed in Internet of Things (IoT). In the hardware implementation of these algorithms, ternary content-addressable memories (TCAMs) are often preferred to other implementations. As a common approach, TCAMs are used for the parallel search to match packet header information with the rules of the classifier. In two-stage architectures of hardware-based packet classifiers, first the decision tree is created, and then the rules are distributed among its leaves. In the second step, depending on the corresponding leaves, the second part of the rules, which includes the range of source and destination ports is stored in different blocks of TCAM. Due to inappropriate storage of port range fields, the existing architectures face the problem of wasting memory and growing power consumption. This paper proposes an efficient algorithm to encode the port range. This algorithm consists of three general steps including layering, bit allocation, and encoding. A greedy algorithm in the first step places the ranges with higher weights in higher layers. Next, an auction-based algorithm allocates several bits to each layer depending on the number of the ranges in that layer. Finally, in each layer, depending on the weight order of the ranges, the bits are given values for the intended range. The evaluation results show that unlike previous methods of storing range fields, the proposed method not only increases the speed of the classification but also uses the capacity of TCAM in the second stage more efficiently.

Keywords: Packet classification, Ternary content-addressable memory (TCAM), Range field, Greedy layering, Bit auctioning, Layer encoding, Efficiency

1 Introduction

A wide range of packet processing devices in IoT including routers [1], anomaly detection systems [2], and monitoring systems [3], treat analysis systems [4], characterizing traffics in smart cities [5], and quality of service provisioning in software-defined networking [5, 6] are considerably accelerated by packet classification paradigm.

The novel IoT hardware systems necessarily depend on an underlying processing engines that can process a large volume of data in an efficient, effective, and flexible way at network speed. Such a system should, in particular, be able to adapt to changes in the environment and keep a reasonable quality of service when; for example, the traffic for a specific network rout increases intensely due to a substantial request from corresponding users [5, 7].

Recently, with the advent of software-defined networks (SDN) such effective network systems has become vital [8]. The idea behind SDN is to transfer the control of networks from localized fixed-behavior controllers distributed over a set of switches to a centralized and programmable controller that can respond to any change in network flows in a timely manner. The key process behind such an accelerated controls of SDN is the packet classification which classifies the flows of internet packets into a set of predefined flows by the network speed [9].

* Correspondence: Abbasi@basu.ac.ir
[1]Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan, Iran
Full list of author information is available at the end of the article

In packet classification, the type of each packet is analyzed by the implemented classification algorithm through software-based or hardware-based methods depending on the information contained in the various fields which were added to the packet header in the source machine. In this analysis, the information in the specified fields of the headers is matched with rules with different priorities, and ultimately, the best matching rule is selected. Finally, a type or flow label corresponding to the selected rule is assigned to the packet.

Among the packet classification algorithms, tree algorithms have received considerable attention in the field of network processors due to their high power and flexibility for implementing prioritized rules. These algorithms can be implemented using both hardware and software. Hardware implementations are widely referred to as TCAM-based methods, and software implementations are called SRAM-based methods [10]. Other hardware-based schemes are intended for implementing on the various platforms, such as application specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) [11–13]. The superiority of TCAM-based methods lies in their high search speed, and their disadvantage is their inefficient memory storage as well as low flexibility in applying the rules to the packets.

In TCAM-based methods, performing a parallel search on all the inputs requires high power consumption [14–27]. Therefore, several methods have been proposed based on the categorization of the TCAM inputs to reduce power consumption [10, 14–16, 18, 20–23, 27, 28]. Through categorization, we can activate or deactivate certain inputs during the search and improve the efficiency of TCAM [19, 21, 27, 29, 30]. To reduce power consumption, manufacturers of TCAMs have proposed a technique for partial activation of TCAM and searching the activated regions. In this process, the internal structure of TCAM chip is divided into fixed blocks; therefore, a variable number of TCAM blocks can be activated at any time. A review of the literature shows that the potential of the idea of reducing power consumption by means of categorization has not been fully utilized. One of the problems of these methods is the storage of the fields of the rules region. For example, in the rules of a classifier, the source, and destination ports are expressed as a range of numbers. Ranges cannot be stored directly in TCAM; instead, rows in TCAM should be determined corresponding to the expressed range, and the other fields should be copied after storing each number from the acceptable port range into the corresponding field of that TCAM [17, 31]. This inefficient method of storing range fields will increase memory consumption and, consequently, increase power consumption during the search process. To solve this problem, range encoding schemes are used to encode each range and map it onto a set of TCAM inputs. This paper proposes an algorithm for storing range fields in TCAM with the aim of reducing the wasting of memory as well as power consumption in the architecture of hardware-based packet classifiers which make use of tree algorithms [18, 32–34]. This innovation will result in optimal use of TCAM memory. Since other tree methods consume a large amount of memory for storing range fields in TCAM [34], the proposed method makes possible more efficient memory usage and power consumption by reducing the number of rows required for the storage of ports range.

The structure of the paper is organized as follows. In what follows, we shall have a look at previous works in the field of range encoding. In the third section, the proposed method for range encoding is described in detail through some examples. Section 4 addresses the implementation and evaluation of the proposed method, as well as the analysis of the results. Section 5 contains some concluding remarks and suggests possible directions for future research.

2 Related work

As noted in the introduction, a major challenge in using TCAM is power consumption. Therefore, much research has been conducted on reducing power consumption in TCAMs.

In [36], Francis et al. suggested a special method for building a routing table based on TCAM. They proposed two designs for low-power TCAM-based forwarding engines, i.e., bit selection architecture and the Trie-based architecture. Underlying both architectures is the division of the routing table into smaller parts so that every search would cover only one part of the table. Kai et al. [30] used this idea to control power consumption.

Srinivasan et al. [37] used an old technique to solve the problem of range encoding. In this technique, the range fields are defined by a set of prefixes, each of which can be stored into a row of TCAM. In general, for w bits, there will be at least (2w-2) prefix modes. Here is a simple proof. Consider the range [1, 2w-2]. The smallest set to cover this range is [4 ..., 0w-11, 10*, 110*, ..., 1w-10]. Taylor showed in [10, 34] that a rule consisting of two groups of 16-bit range fields could have at least $(2 \times 16\text{-}2)^2$ inputs. In practice, however, an examination of the actual databases described by Taylor in [10] indicates that some rules require TCAM inputs with factors higher than seven. The previous works related to the improvement of range field storage can be divided into two main groups: (1) database-independent range encoding algorithm and (2) database-dependent range encoding algorithm. A database-dependent range encoding algorithm is dependent upon how the ranges are distributed in their database and, therefore, depends on the type of the database related to the range field.

In contrast, the encoding of specific ranges using a database-independent algorithm does not depend on and vary with, the type of the database. Using the technique of fence encoding, Lakshminarayanan et al. [37] developed an algorithm called database-independent range pre-encoding (DIRPE). In this algorithm, ranges are expressed as sets of ternary values in the form of 1XX1X0 instead of being transformed into prefix sets. Also, unused bits in TCAM can be used to encode ternary strings. By unused bits, we mean the bits that remain unused when storing the rule fields due to the fixed number of bits in each row of TCAM. The performance of DIRPE is a function of the number of unused bits available to TCAM. A decrease in the number of available unused bits will reduce the efficiency of DIRPE encoding.

In comparison, the SRGE algorithm (short range gray encoding) which was proposed in [20] is an efficient, database-independent encoding scheme. This scheme does not require extra bits. Bremler-Barr and Hendler [20, 21] proposed SRGE as an independent range encoding algorithm based on their observations. They found that encoding the ranges that are available in queues in real databases would be much more efficient when using the Gray code. This design is the first algorithm that has a general effect on narrowing the ranges and, more particularly, works well in smaller ranges without the use of extra bits.

The first algorithm for the dependent encoding of ranges was proposed by Liu [35]. The basic idea of this method is to use extra bits for bit-mapping. In this work, one bit is specified for every selected range *r*. Additionally, there are values in the range fields which find exact matches. If the number of these values is taken as m, then the number of required extra bits is $\log_2^{m+1}$. Figure 1 illustrates the details of this technique by displaying how this algorithm works on six range fields from the range fields of six sample rules. Given this, if the range *r* receives i bits, the ith extra bits for all TCAM inputs that contain the range *i* will take a value of 1; the rest of the extra bits in this input will take null value. In searching a key, if it is within the range *r*, then the ith bit will be set to 1; otherwise, it will be set to 0. This basic scheme eliminates extra inputs of the ranges. However, this solution cannot be very useful because the redundancy of the range is eliminated by the number of extra bits. Lakshminarayanan discovered that the number of ranges in today's classifications of databases is approximately 300 [37]. The number will predictably increase in the future. The growth rate of the number of ranges is increasing due to the increased use of TCAM in intrusion detection systems and in proportion to the addition of new range fields such as packet length.

To solve the scalability problem, a region partitioning algorithm was suggested in [35]. An example of how this algorithm functions in five different regions from a single field of five rules is illustrated in Fig. 2. In the partitioning of the regions, the ranges are divided into several sub-ranges. To determine the sub-ranges, a layering algorithm must be used. The example given in Fig. 2 consists of three sub-ranges. Each sub-range is encoded with several bits. The number of bits needed to encode each sub-range depends on the number of ranges in that sub-range and increases with an increase in the number of sub-ranges. In practice, however, the outcome of this algorithm is relatively complicated, and every sub-range of a range requires a separate TCAM input. In [38], a set of encoding schemes based on layering is proposed,

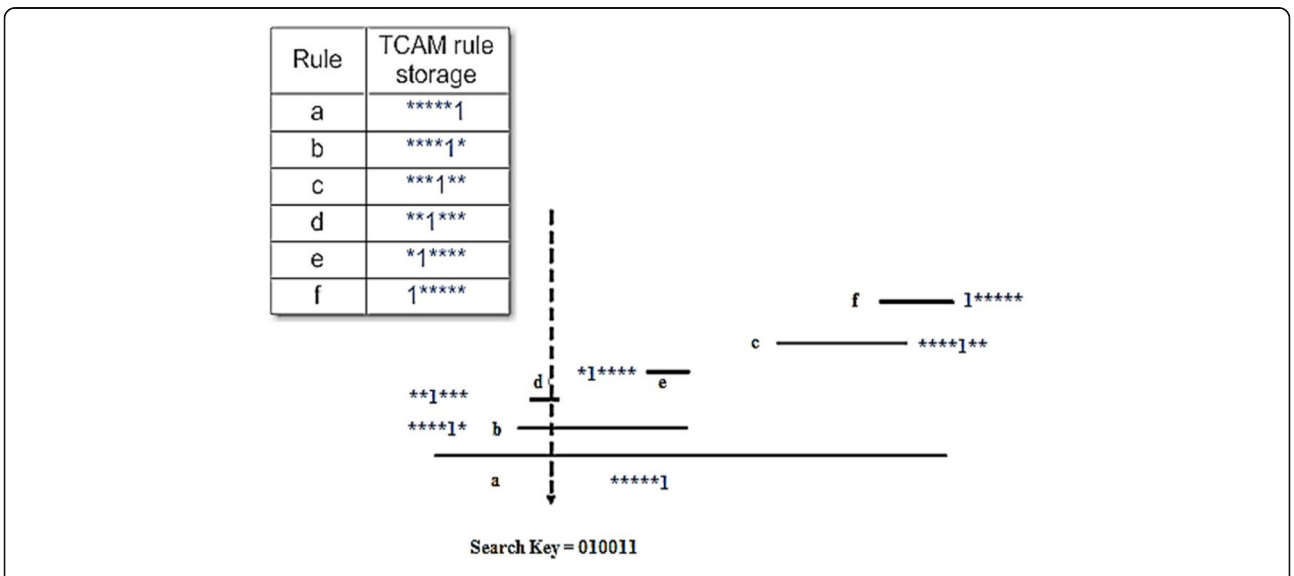

Fig. 1 An example of encoding using the basic scheme of dependent encoding proposed by Liu [35]

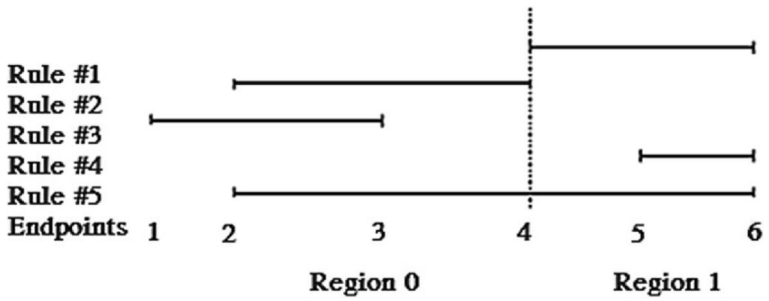

Fig. 2 An example of encoding using the region division encoding algorithm [20, 21]

which is referred to as encoding of the parallel packet classification. Of course, the algorithm does not explicitly address layering. The purpose of this method is parallel search of the fields. P2C is an improvement of the method presented in [39] which describes the relationships among the ranges and reduces the need for memory consumption. In the same vein, Chang and Su [24] developed range algorithms based on the Gray code. Their project improves the P2C algorithm by using binary reflected Gray codes. Their algorithm considers the initial ranges obtained by the classifier. They indeed focused on minimizing extra bits without considering the number of available bits in each TCAM input. Che et al. [40] developed a scheme called DRES based on the notion of bit-mapping. This scheme is a dynamic range selection algorithm for regions to which extra bits are assigned. The scheme is based on two simple but effective ideas. First, the ranges are expressed as a set of ternary values in the form of 1XX1X0 instead of a set of prefixes. Second, unused extra bits in TCAM can be used to encode ternary strings. This algorithm is a greedy algorithm and allocates extra bits to ranges with the highest prefix width.

Rottenstreich and Keslassy [42] proposed a database-dependent encoding scheme. In this method, each range of W bits can be encoded with W inputs of TCAM. The abovementioned encoding algorithm is based on the fact that any action that is applied to a packet by a rule is dependent on a default action. This action is predetermined and usually applies to packets that do not match any rule. Similar studies were conducted by Cohen and Raz [22]. In their algorithm, it is not necessary to partition the ranges into distinct sub-ranges, but the ranges can be partitioned into overlapping sub-ranges. The algorithm could gain an extension coefficient of 2 W-4 at the worst case for every range of W bits.

An algorithm for independent encoding of ranges was proposed in [21]. This algorithm consists of three sections. In the first section, layering of the ranges is performed. Layering can be done in four ways. The first method is to calculate the maximum size of independent sets by a greedy algorithm using a graph. In the second method, layering is performed by calculating the maximum size of colorable sets using a graph and the chromatic number. The third method performs layering via calculating the maximum weight of independent sets by assigning weights to the ranges. Finally, the fourth method is based on the calculation of the maximum colors of colorable sets. As stated in [21], the results of the four methods are identical, suggesting that the methods can be interchangeably used for independent encoding of ranges. In the next section of the algorithm, bits are assigned to each layer for encoding. The number of these bits is determined by the number of ranges in each layer. Finally, the encoding operation is done. What distinguishes this scheme from DRES is that DRES allocates one bit to each range while the scheme in [40] may allocate a varying number of bits depending on the number of layers and the number of ranges in each layer. Obviously, in DRES algorithm, the number of allocated bits is directly related to the number of independent ranges whereas the number of allocated bits in [21] for N ranges is approximately $\log_2 N$.

In [19], Yadi et al. introduced a plan called SmartPC to reduce power consumption. In this design, selected TCAM blocks are activated through pre-classifying instructions to reduce power consumption. The limitation of this method is to group the rules into separate blocks. Some solutions were proposed in [17, 41] to overcome this limitation. In these solutions, decision tree is used to classify the rules, and TCAM blocks are mapped onto the leaves of the decision tree. Both solutions are a combination of software-based classification (using decision tree) and hardware-based classification (using TCAM). Analytical results in [17, 41] suggest that these solutions work well to reduce the number of blocks required by TCAM, as well as, power consumption. Figure 3 shows the two-stage architecture of a packet classifier as proposed in [17, 41]. The difference between these two schemes is in the tree algorithm used to distribute the rules in the second stage of classification. On receiving an input packet, the classifier uses part of the fields of the header to search Index TCAM. In this figure, source IP address is used along with destination IP address (i.e., 64 bits) to search Index TCAM. The reason for using

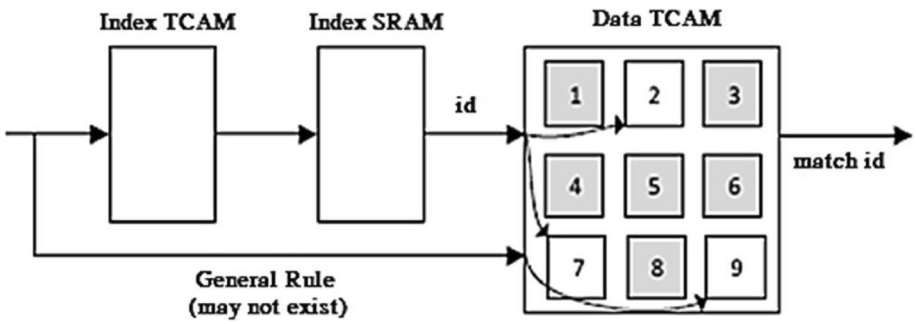

Fig. 3 The architecture of TCAM-based hardware classifier [41]

these two fields in the initial search is that they provide better distinction among the input packets. Next, the identifier obtained by the process of matching in Index TCAM is used as an input to activate TCAM blocks. It should be noted that, in Fig. 3, each TCAM block in the second stage of classification corresponds to a leaf in the decision tree.

In both schemes, some TCAM blocks are allocated to packets that are not classified in the first stage. These blocks are activated simultaneously with specific blocks that correspond to the decision tree leaves. Finally, from among the matching rules from several activated blocks, the best matching rule is specified using an encoder and transmitted to the output of the classifier. One of the basic problems in this architecture is the complexity of the memory needed to store range fields during the second stage of classification. In the architecture of network processors, the high storage capacity of TCAM (in comparison with SRAM) is a valuable resource. In the proposed method, the goal is the optimal use of TCAM memory. As an implicit result, the proposed approach reduces power consumption in the TCAM chip by reducing the number of inputs required by TCAM to store range fields in the second stage of classification.

3 The proposed method

In this paper, we present a solution for encoding ranges in a two-stage classifier of Internet packets in the framework of a TCAM-based architecture, which is discussed in [17, 41]. The proposed approach provides a more convenient use of memory in the second stage of classification by encoding the range fields. In this section, the suggested architecture is introduced as an encoding algorithm. In the architecture of a TCAM-based classifier which uses a decision tree to map the rules onto memory blocks, storage of range fields is a great challenge that has not been sufficiently addressed in the literature. By using an encoding algorithm to store range fields, the proposed solution reduces the number of TCAM inputs in the second stage of packet classification and decreases power consumption.

3.1 The architecture of the proposed classifier based on the encoding algorithm

The architecture of the classifier in Fig. 3 has been modified using the encoding algorithm and described in Fig. 4 as our proposed architecture. Characteristic of the proposed architecture is that TCAM blocks in the second stage can accommodate more rules, making it possible to use the blocks more efficiently. In the diagram shown in Fig. 4, the source and destination IP addresses must first be stored as element indexes in TCAM. When a packet enters the classifier, the source and destination IP address fields of the packet header are compared with the index of the elements of TCAM. During the first stage of the search, the source and destination port fields are encoded to be used for searching in the second stage. Matching at the first stage is based on the priority of the rules. Based on the best match found, an address is obtained to access one unit from the SRAM. According to the classification performed in the preprocessing stage, a numerical value has been stored in this unit by using the tree algorithm. This value shows the number of the TCAM block in which the encoded source, and destination port numbers of the rules are stored.

In the second stage of the search, the encoded fields of the source and destination ports of the packet header are compared with the valid elements of the selected TCAM block. If matches are found, the best matching rule is selected as the output. It should be noted that in this architecture while examining the selected block of TCAM, another block is also examined. This block is called the public block. In the public block, the information of the source and destination port of those rules is written, which have been repeated in the sub-regions created by cutting the two-dimensional geometric space corresponding to the source and destination IP addresses. The most important reason for dealing with such rules during the cutting process is that a bit may be selected for cutting which is of no value (or X) in the mentioned rule. It is obvious that, in the subsets created by cutting, this rule will fall into two subsets [43]. As a result, when storing rules in TCAM blocks, these rules

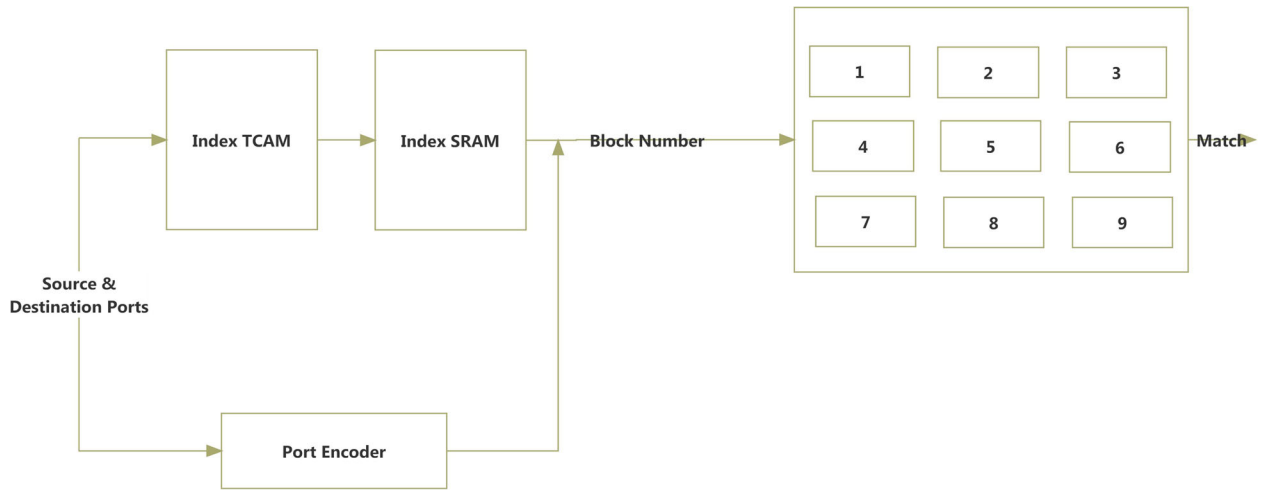

Fig. 4 Architecture of the proposed classifier

should be written into two blocks of TCAM. Therefore, these rules are written into the public block to avoid this problem and use memory optimally. Now, we are familiar with the architecture of the hardware packet classifier, we shall explain the proposed algorithm for encoding range fields.

3.2 The proposed encoding algorithm for storing range fields in TCAM

In general, rules that have a range field cannot be represented by a single TCAM input. It is therefore necessary to have a scheme for encoding the range fields so that they could be stored in TCAM. This encoding scheme maps the fields of any range r onto a set of TCAM inputs, resulting in what is called the coverage r. The encoding algorithm is shown in the flowchart in Fig. 5.

This algorithm consists of three general steps: (1) layering; (2) bit allocation; and (3) encoding. In the first step, which is marked in blue in the Fig. 5, range fields are divided by their weight. The weight of a field is equal to the difference between the beginning and end points of the field. This algorithm is a greedy algorithm in which, for layering, the value of the intended field is checked in all rules. In this algorithm, ranges with higher weights should be placed in higher layers. For this purpose, fields are arranged in descending order. After placing the range with the highest weight in the first layer, the next range must be layered. For this purpose, the second range is compared with the range that was located in the first layer. If there is an intersection between the ranges of the two fields, the new range will be placed in the second layer; otherwise, it will be put in the first layer. This procedure is performed for the values of the range fields of all the rules of the classifier.

In the second step, marked in yellow in Fig. 5, several bits are allocated to each layer depending on the number of the ranges in that layer. Bit allocation operation in this algorithm is like an auction. For this purpose, Eq. (1) is used:

$$\mathrm{Auction}[\mathrm{i}] = \sum_{j=2^{\mathrm{assigned}[i]}}^{2^{\mathrm{assigned}[i]+1}-1} W(L[i][j]) \tag{1}$$

where $W(L\ [i]\ [j])$ is the weight of the jth range from the ith row which is arranged in terms of range weights. Also, assigned $[i]$ specifies the number of bits allocated to each row. In the second stage, the auction starts. Using this equation for all rows, the value of auction is calculated, and one bit is allocated to the row with the highest auction value. Next, this relationship is reused and recalculated for all rows, and the next bit is allocated to the line that obtains the highest weight. This procedure continues until there is no other bit for allocation. No bit is allocated to ranges with a weight of zero.

In the third step, which is marked in red in Fig. 5, the encoding operation begins. In each layer, depending on the weight order of the ranges, the bits are given values for the intended range. Afterward, the ranges with a weight of zero should be encoded. For this purpose, all the bits allocated to a memory row are divided into two groups. The first group consists of the bits that specify the range fields, and the second group is used to encode bits with a weight of zero. The values of the bits in the first group are determined by performing a search in the layers and encoding operation. The bits in the second group are encoded with regard to the place of the ranges in each layer, beginning from number 1.

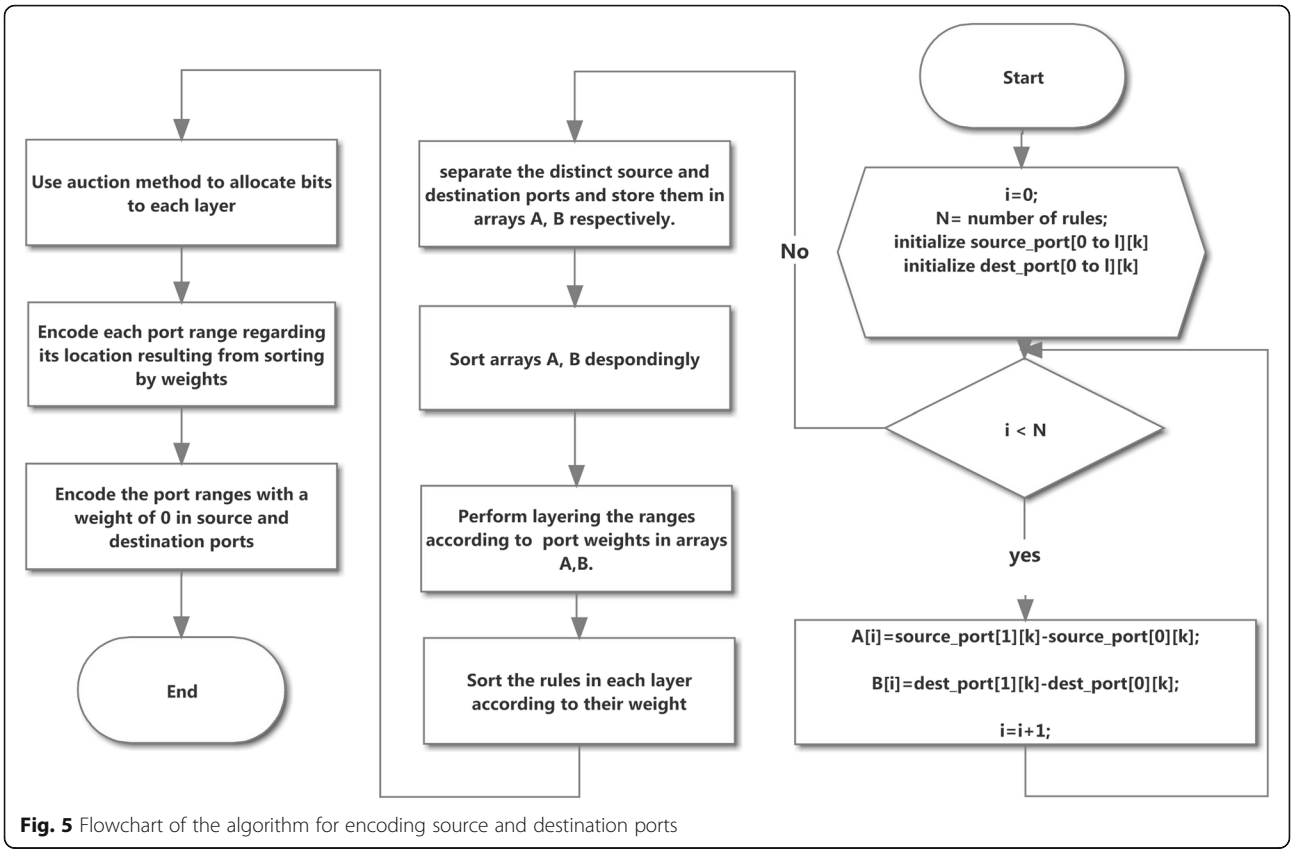

Fig. 5 Flowchart of the algorithm for encoding source and destination ports

3.3 An example of the proposed encoding algorithm for storing range fields

Table 1 lists a set of binary rules, including the two fields of source and destination port ranges. In the present work, 16 bits are used to encode the range of source ports and 16 bits to encode the range of destination ports. For encoding, the layering of ranges begins first. In this phase, the ranges of rules in the source and destination port fields are examined individually. For the sake of simplicity, the decimal values of the ranges of source and destination ports along with their weight are shown in Table 2. To this end, after separating the unique ranges of source and destination, the rules should be sorted by weight in descending order. As can be seen in the table of rule sets, the unique ranges of the source port field of the rules include [0, 53], [1024, 65535], [23939, 24032], [0, 65535], [52, 53], and [48, 53] and the unique ranges of the destination port field of the rules include [20, 21], [32, 37], [0, 65535], [0, 22] , [80, 88], [23936, 24032], [52, 53], and [48, 52]. To encode the above ranges, all these modes are first sorted by their weight. Next, the layering operation begins. Figure 6 a

Table 1 A set of binary rules including the two fields of source and destination port ranges

Rule number	Beginning of the source port	End of the source port	Beginning of the destination port	End of the destination port
1	0000000000000000	0000000000110101	0000000000010100	0000000000010101
2	0000010000000000	1111111111111111	0000000000100000	0000000000100101
3	0101110110000000	0101110111100000	0000000000000000	1111111111111111
4	0000000000000000	1111111111111111	0000000000000000	0000000000010110
5	0000000000110100	0000000000110101	0000000000000000	1111111111111111
6	0000000000110000	0000000000110101	0000000000000000	1111111111111111
7	0000000000000000	1111111111111111	0000000001010000	0000000001011000
8	0000000000000000	1111111111111111	0101110110000000	0101110111100000
9	0000000000000000	1111111111111111	0000000000110100	0000000000110101
10	0000000000000000	1111111111111111	0000000000110000	0000000000110101

Table 2 The decimal values of the range of source and destination ports along with their weights

Rule number	Source port range	Source weight	Destination port range	Destination weight
1	[0, 53]	53	[20, 21]	1
2	[1024, 65535]	64511	[32, 37]	5
3	[23936, 24032]	96	[0, 65535]	65535
4	[0, 65535]	65535	[0, 22]	22
5	[52, 53]	1	[0, 65535]	65535
6	[48, 53]	5	[0, 65535]	65535
7	[0, 65535]	65535	[80, 88]	8
8	[0, 65535]	65535	[23936, 24032]	96
9	[0, 65535]	65535	[52, 53]	1
10	[0, 65535]	65535	[48, 53]	5

and b illustrate the layering of the ranges of the source port field and destination port field of the rules of the classifier, respectively. After layering, the ranges in each layer are sorted again by weight. In the second step, bit allocation begins. To do this, the auction value is calculated for all layers according to Eq. (1). The first bit in the range of source and destination ports is given to the first layer. Finally, the number of bits allocated to the source port ranges is 6 in a way that one bit is allocated to layer 1, two bits to layer 2, two bits to layer 3, and one bit to layer 4. The number of bits allocated to the destination port ranges is also 6 bits: one bit allocated to layer 1, three bits to layer 2, and two bits to layer 3.

Encoding begins in the third step. For this purpose, each range is encoded according to its place in descending weight order, and the bit which is allocated to the layer in which it is placed will receive a value according to its order. Other bits allocated to other layers will receive a null value. Table 3 shows the unique range of source and destination ports as well as their encoding.

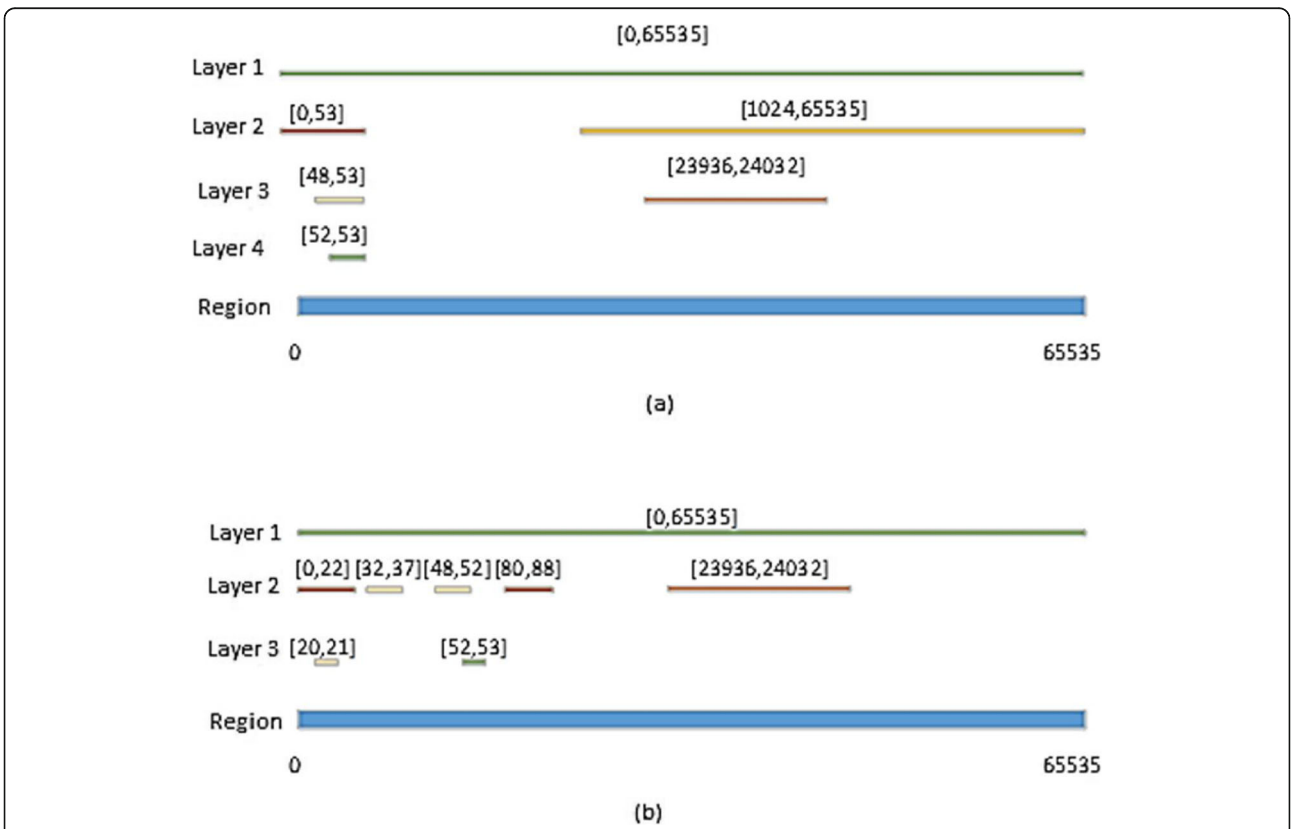

Fig. 6 Layering the range of the ports according to the rules in Table 1 **a** source port ranges; **b** destination port ranges

Table 3 Encoding the range of unique source and destination ports of the rule set of Table 1

Range of the unique source ports	Encoding of the source ports range	Range of the unique destination ports	Encoding of the destination ports range
[0, 65535]	1XXXXXXXXXXXXXXX	[0, 65535]	1XXXXXXXXXXXXXXX
[1024, 65535]	X01XXXXXXXXXXXX	[23936, 24032]	X001XXXXXXXXXXX
[23936, 24032]	XXX01XXXXXXXXXX	[0, 22]	X010XXXXXXXXXXX
[0, 53]	X10XXXXXXXXXXXX	[80, 88]	X011XXXXXXXXXXX
[48, 53]	XXX10XXXXXXXXXX	[32, 37]	X100XXXXXXXXXXX
[52, 53]	XXXXX1XXXXXXXXX	[48, 52]	X101XXXXXXXXXXX
-	-	[20, 21]	XXXX01XXXXXXXXX
-	-	[52, 53]	XXXX10XXXXXXXXX

3.4 Encoding the input packet

The information extracted from the header fields of the input packet should be encoded in order to search in TCAM blocks. For this purpose, the stored values are given to the encoder module as the source and destination ports of the input packet. In this module, the source and destination ports of the input packets are encoded. These ports are encoded independently from each other. In doing so, the source port and the destination port of the input packet are separately compared with the range of the unique ports of the rules. If the packet is within the rule range of a layer, the code of the corresponding rule will be copied into the intended location to be encoded. If the packet does not match, and cannot be covered by any of the rules of a layer, the bits associated with that layer will be set to 0 in the packet code.

Suppose a packet with the source port 123 and the destination port 22 enters the encoder module according to the rules in the Table 1. Given that 123 lies within the range of the first layer [0, 65535], the first bit of the source port code of this packet becomes 1. Since this packet does not match any of the ranges in the other layers of source port ranges, the remaining bits of the source port code of the packet are set to 0. Thus, the encoded code of the source port range is 1000000000000000. The destination port of this packet is also compared with the destination port ranges in the layers created in Fig. 6. Given that 22 is within the range [0, 65535], the value of the bit associated with the first layer in this packet becomes 1. Also, in the second layer, the packet matches the range [0, 22]. Therefore, the value of the bits associated with the second layer in this range is 010, which is stored in the location of the bit codes associated with the second layer. Because the destination port number does not overlap with the ranges in the next layers, the rest of the bits are set to 0. Therefore, the encoded code for the destination port of the packet will be 1010000000000000. Now, the encoded value of the source and destination ports can be used to find matches in that block of TCAM memory whose number was obtained from SARM in the first stage of classification by using source and destination port addresses.

4 Implementation and evaluation

In this section, first the implementation of the proposed algorithm is explained. For this purpose, after introducing the tools used to generate the packets, and the rule sets for the experimental classifier, the programming language, and code structure are introduced. Finally, the evaluation parameters are introduced, and the results of the implementation of the proposed algorithm are analyzed.

4.1 Rule set generation tools and evaluation parameters

The experimental rule sets and packets required for the evaluation of the proposed algorithm and architecture were generated using ClassBench suite, which was developed in [43]. ClassBench tool is a simulator for producing synthetic filters with desired distributions in the geometric space of filters. This tool produces dummy packets corresponding to the produced filters. Indeed, it creates filters with distributive parameters that are given to it as input. The presence of this simulator satisfies the need for real and heterogeneous filters of Firewalls, IP-Chains, and Access Control Lists. In the majority of the studies [44–48], the ClassBench tool has been used for producing the required data structure due to a need for filters and packets that are close to reality in terms of structural characteristics and statistical distribution.

Three general types of rule set can be generated by ClassBench, including Access Control List (ACL), Firewall (FW), and IP chain (IPC). Each rule set is named according to the type and size of the generated set. For example, ACL1K refers to the set of access control list rules that contain about 1000 rules. It should be noted that in the evaluations using this tool, sets of 100, 500, 1000, and 2000 rules from all the three abovementioned

types were created. For each set, the number of generated packets was ten times greater than the number of generated rules. Also, to use the rules generated by ClassBench using C ++ codes, the numbers corresponding to the four fields of source IP address, destination IP address, source port, and destination port were converted to binary numbers. Finally, the binary file created was used for simulation. The proposed algorithm was implemented using

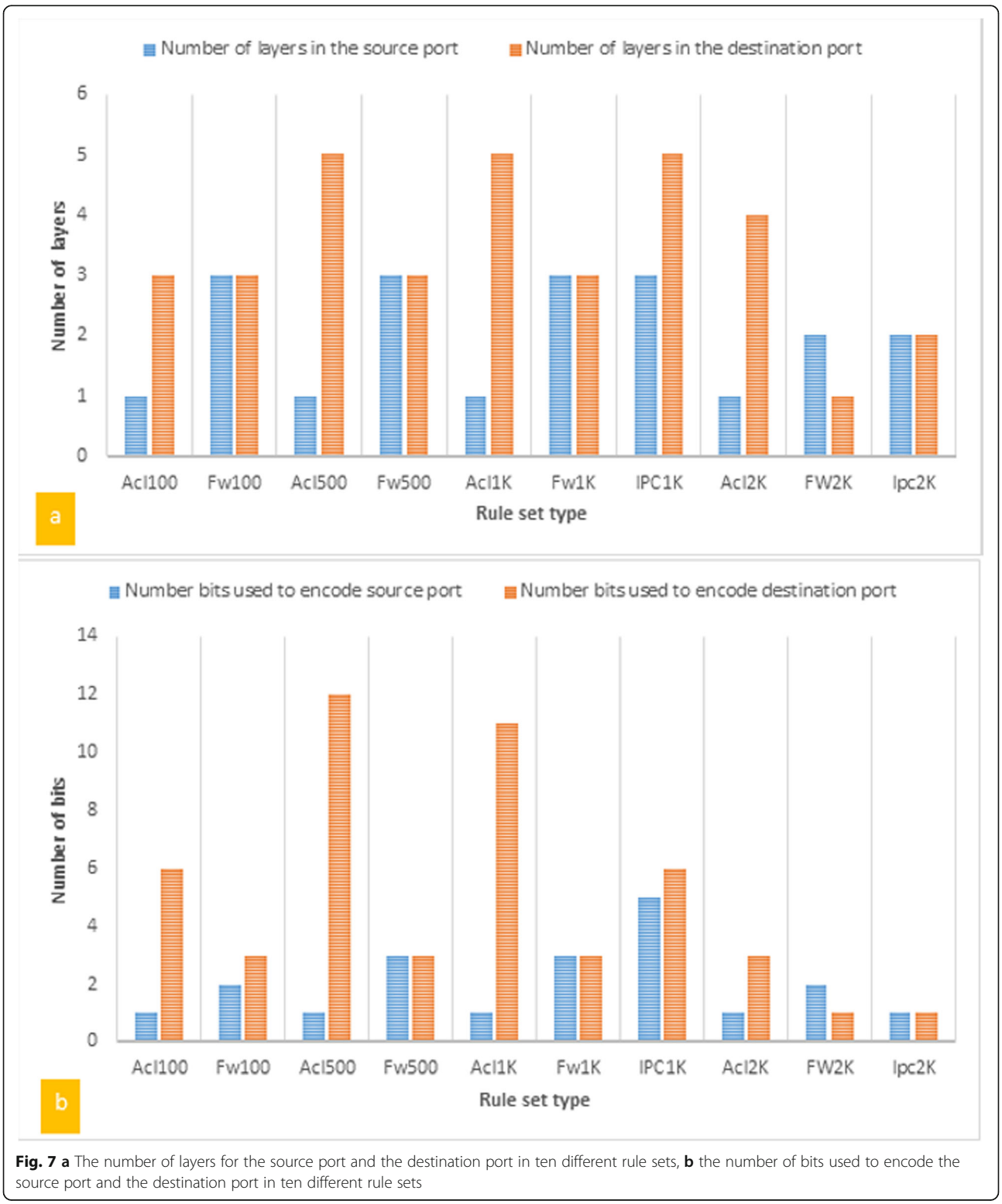

Fig. 7 **a** The number of layers for the source port and the destination port in ten different rule sets, **b** the number of bits used to encode the source port and the destination port in ten different rule sets

Table 4 Extension ratio in TCAM blocks in the second stage of classification

Rule set	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7	Block 8	Public block
ACL100	1.181818	1.3	1.181818	1.4444444	1	1.1818182	1.3	1.625	1.529412
FW100	1.625	1.083	1	2.16	1.083	1.083	1.444444	13	1.130435
ACL500	1.625	1	1.065574	1	1.625	1.3	1.25	2.241379	1.566265
FW500	1.27451	1.3	1.25	1.3	1.2264151	1.3	1.3829787	1.444444	1.940299
ACL1K	1.226415	1.2745098	1.444444	1.1711712	1.0743802	1.1818182	1.1504425	1	130
FW1K	1.25	1.2380952	1.238095	1.25	1.2380952	1.2380952	1.25	1.25	1.940299
IPC1K	1	1.2380952	1.056911	1.3541667	1.0655738	1.1403509	1.2149533	1.313131	3.023256
ACL2K	1.209302	1.319797	1.25	1.1981567	1.0833333	1.1872146	1.1453744	1.287129	8.965517
FW2K	1.244019	1.2440191	1.244019	1.25	1.2380952	1.2380952	1.25	1.244019	1.89781
IPC2K	1.420765	1.4364641	1.368421	1.4364641	1.3978495	1.3541667	1.4285714	1.444444	2.20339

VHDL language based on the IEEE 1994 standard and in ISIM development environment v13.1.

In this work, the allowed number of bits for encoding was 16 bits for both the range of source ports and the range of destination ports. To encode and save range fields and encode the incoming packets, an encoder module was used. This module initially performs the necessary processes on the range of source and destination fields. These processes include the encoding steps. Next, to encode each range field before being written into the block specified in the first stage of classification, the field is sent to the encoder module, and the output of the module is written in a specified TCAM block. Furthermore, this module is activated in RAM while the source and destination IP addresses are being searched and encodes the source and destination ports of incoming packets by means of the algorithm described in the previous section. It should be noted that, in the second stage of classification, the encoded values of the source and destination ports that were produced by the encoder module are used to search the selected block specified by the output index of the first stage (as well as the public block of rules) and to find the rule that best matches the source and destination ports of the input packet.

The parameters used to assess the encoder module include the number of the layers of source and destination ports, the number of bits used for encoding source and destination ports, the number of unique ranges with a weight of zero in the source and destination, and the number of unique ranges with a weight of non-zero in the source and destination.

4.2 Implementation of the two-stage architecture and the encoding algorithm

Figure 7a shows the number of layers for source and destination ports of ten rule sets of varying sizes generated by ClassBench. In this figure, the minimum number of layers for the source port is one, and the maximum number of layers is three. As shown in Fig. 7b, the minimum number of bits needed to encode source ports is one, and the maximum number is five. In Fig. 7a, the minimum and the maximum number of the layers of the source port is one and five, respectively. Also, the minimum and the maximum number of bits needed to encode the destination port, which is shown in Fig. 7b, is one, and 12, respectively. In these results, the maximum number of bits used for encoding belongs to the ACL500

Table 5 The minimum and maximum values of the extension ratio and the memory required for encoding

Rule set	Minimum value of the extension ratio	Maximum value of the extension ratio	Minimum amount of SRAM memory required for encoding (KB)	Maximum amount of SRAM memory required for encoding (KB)
ACL100	1	1.625	43.45211	43.49935
FW100	1	13	61.32269	61.89263
ACL500	1	2.241379	51.1507	51.69123
FW500	1.226415	1.940299	75.74132	77.20763
ACL1K	1	130	52.95357	54.47285
FW1K	1.238095	1.940299	79.6737	81.67321
IPC1K	1	3.023256	98.49408	106.0673
ACL2K	1.083333	8.965517	38.23026	38.83012
FW2K	1.238095	1.89781	25.58792	25.66043
IPC2K	1.354167	2.20339	25.55715	25.61868

rule, which is 12 bits for the destination port and one bit for the source port. Extra bits are also used to encode zero-weight fields.

Analysis of the obtained results would suggest that, in this method, a range field can be stored using a single row of TCAM while previous methods [17, 41] had problems with this task and had to allocate a separate row to each range field according to the difference between its beginning and end points. This issue is shown in the Table 4.

This table shows the extension ratio of the encoding of different blocks by the proposed method. The extension ratio is equal to the maximum capacity of a block divided by the number of rules of that block. In Table 5, the minimum extension ratio is shown for each rule set between its blocks. The minimum extension ratio in Table 5 is 1, meaning that a maximum of one row from TCAM blocks is allocated to each rule. A larger ratio means that the block is used

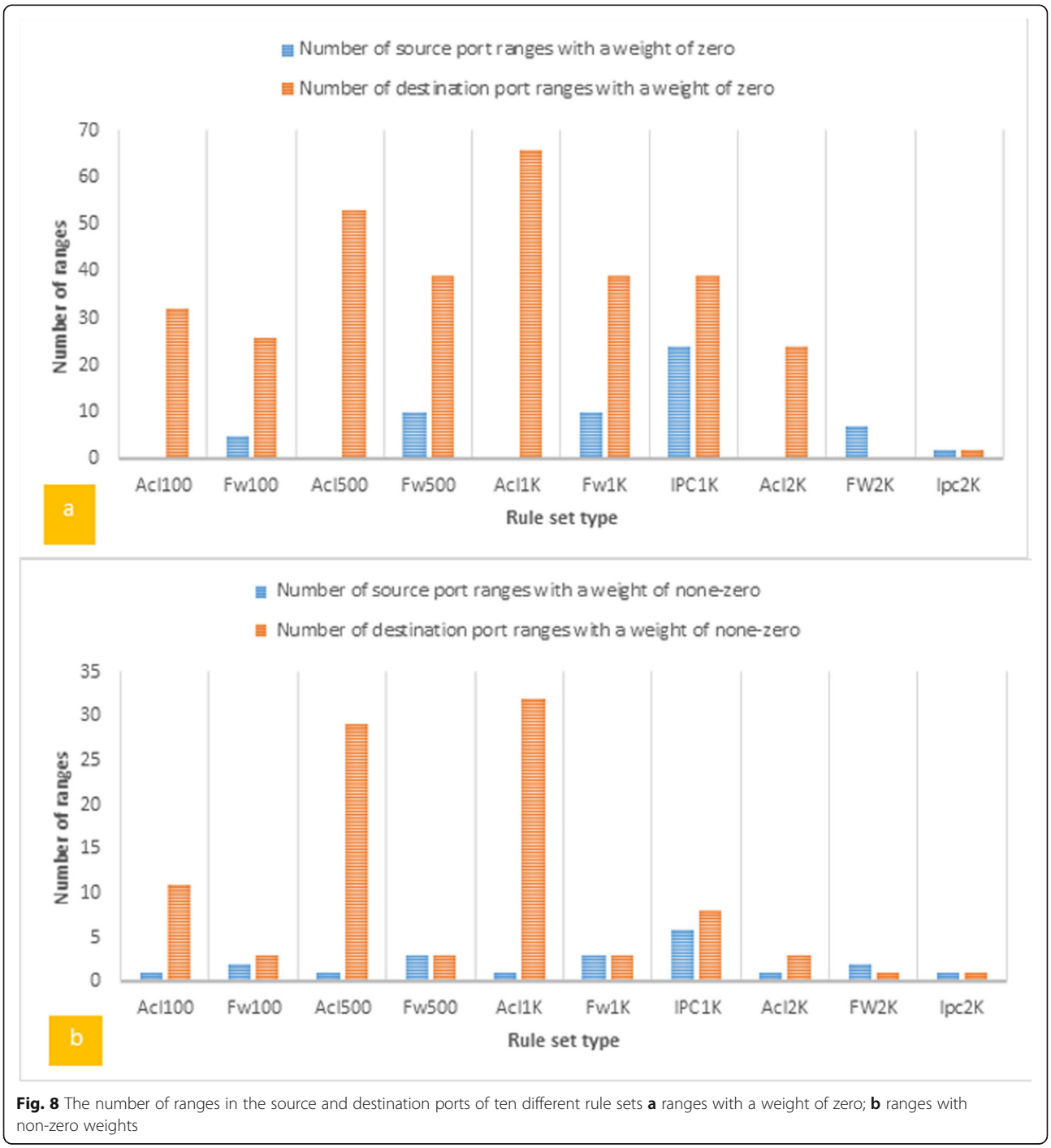

Fig. 8 The number of ranges in the source and destination ports of ten different rule sets **a** ranges with a weight of zero; **b** ranges with non-zero weights

less frequently. Table 5 also shows the minimum and maximum SRAM memory required for encoding. This value is calculated from Eq. (2):

$$\text{SRAM Size} = 65536*([\ \log_2{}^{s}] + [\ \log_2{}^{s}]) + s*d*\text{TCAM\_input\_size}\ )/(8*1024) \quad (2)$$

In this equation, s is the number of unique rules for the source port range, and d is the number of unique rules for the destination port range.

Figure 8a shows the number of ranges with a weight of zero for the source and destination ports separately. The maximum number of these ranges in the source port belongs to IPC1k, and the minimum number, which is 0, belongs to ACL100, ACL500, ACL1k, and ACL2k. Also, Fig. 8a shows the number of ranges with a weight of zero for the destination port. The maximum number of these ranges in the destination port belongs to ACL1k, and the minimum number, which is 0, belongs to FW2k rule set. In addition, according to Fig. 8b, the maximum number of unique ranges with a weight of non-zero in the source port belongs to IPC1k, and the minimum number, which is 1, belongs to ACL100, ALC500, ACL1k, and ACL2k. Also, as shown in Fig. 8b, the maximum number of unique ranges with a weight of non-zero in the destination port belongs to ACL1k, and the minimum number belongs to FW2k, and IPC2k.

Due to existence of the maximum source port ranges in IPC 1 k, it is expected that the maximum number of unique source ranges would be created for this rule set, respectively. Similarly, due to existence of the maximum destination port ranges in ACL 1 k, it is expected that the maximum number of unique destination ranges would be created for this rule set, respectively.

Figure 9 indicates that the maximum number of unique source ranges belongs to IPC 1 k and the minimum number belongs to ACL100, ACL500, ACL1k, and ACL2k. Also, Fig. 9 shows that the maximum number of unique source ranges belongs to ACL1k and the minimum number belongs to FW2k.

Simulation results show that encoding can significantly reduce power consumption due to the range fields. In previous methods such as [37, 41], no solution was proposed for optimal use of memory in the second stage of classification. The advantage of this method is that it does not reduce the speed of the classification while making optimal use of TCAM capacity in the second stage. This optimal use will also reduce power consumption. The previous methods of storing range fields had serious problems, and sometimes limited block capacity could prevent many rules from being written.

5 Conclusion

In this paper, a new port-encoding method is proposed that can be exploited in any two-stage TCAM-based packet classifier. The proposed encoding method can be used for optimizing memory usage in the second stage of any two-stage TCAM-based packet classifier, which maps the rule sets to decision trees. The proposed method includes three steps: the greedy layering of ports, allocating bits for distinct layers according to a novel auctioning process, and range encoding. The main benefit if the proposed method is the optimal use of the capacity of TCAM blocks as well as the reduction of their power consumption. In this work, VHDL was used for simulation purposes. The rules and packets needed to evaluate the algorithm were generated by ClassBench suite. The key criteria for evaluating the proposed architecture were the number of layers of the source port and destination ports and the number of bits used to encode

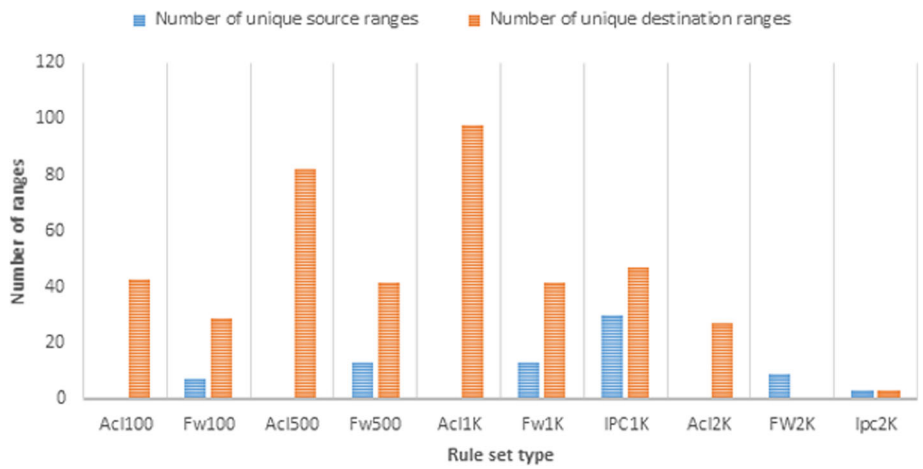

Fig. 9 The number of unique ranges for source and destination ports of ten different rule sets

these layers. The simulation results show that, by encoding the range of source and destination ports, they can be stored in a single TCAM row. It can be inferred from the results that the proposed method proved to be more efficient than previous methods in using memory in the second stage of classification. This superiority corresponds to lower power consumption in TCAM blocks which hold the range value of classifier rules.

Abbreviations

ACL: Access control list; ASIC: Application specific integrated circuit; DRAM: Dynamic random access memory; FPGA: Filed programmable gate array; FW: Firewall; IoT: Internet of Things; IP: Internet protocol; IPC: IP chain; SDN: Software defined network; SRAM: Static random access memory; TCAM: Ternary content addressable memory

Acknowledgements

The authors would like to thank the editor and reviewers for their time and consideration.

Authors' contributions

MA and AF have participated in design of the proposed method and practical implementation. SV has coded the method. MA and MK have completed the first draft of this paper. All authors have read and approved the manuscript.

Authors' information

Not applicable.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan, Iran. [2]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran. [3]Department of Electrical and Electronic Engineering, Shiraz University of Technology, Shiraz, Iran. [4]Computer Engineering Department, Persian Gulf University, Bushehr, Iran.

Received: 3 October 2019 Accepted: 5 December 2019
Published online: 30 December 2019

References

1. B. Indira, K. Valarmathi, D. Devaraj, An approach to enhance packet classification performance of software-defined network using deep learning. Soft Computing **23**(18), 8609–8619 (2019)
2. J. Wee, J.-G. Choi, W. Pak, Wildcard fields-based partitioning for fast and scalable packet classification in vehicle-to-everything. Sensors **19**(11), 2563 (2019)
3. S.R. Hussain, S. Nirjon, E. Bertino, in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Securing the insecure link of Internet-of-Things using next-generation smart gateways (IEEE, 2019)
4. E. Hodo et al., in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. Threat analysis of IoT networks using artificial neural network intrusion detection system (IEEE, 2016)
5. A. Sivanathan et al., Characterizing and classifying IoT traffic in smart cities and campuses. in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2017. IEEE.
6. D.M.F. Mattos, P.B. Velloso, O.C.M.B. Duarte, An agile and effective network function virtualization infrastructure for the Internet of Things. Journal of Internet Services and Applications **10**(1), 6 (2019)
7. S.Y. Shin et al., Dynamic adaptive network configuration for IoT systems: a search-based approach. arXiv preprint arXiv **1905**, 12763 (2019)
8. Das, R.K., A.K. Maji, and G. Saha, Prospect of improving internet of things by incorporating software-defined network, in advances in communication, devices and networking. 2019, Springer. p. 537-544.
9. R. Bilal, B.M. Khan, *Software-defined networks (SDN): a survey, in handbook of research on cloud computing and big data applications in IoT* (IGI Global, 2019), pp. 516–536
10. E. Spitznagel, D. Taylor, J. Turner, Packet classification using extended TCAMs. in Proceedings of the 11th IEEE International Conference on Network Protocols 2003. IEEE Computer Society.
11. C. Li et al., Memory optimization for bit-vector-based packet classification on FPGA. Electronics **8**(10), 1159 (2019)
12. J. Li et al., Exploiting packet-level parallelism of packet parsing for FPGA-based switches. IEICE Transactions on Communications, 2018EBP3333 (2019)
13. H. Alimohammadi, M. Ahmadi, Clustering-based many-field packet classification in software-defined networking. Journal of Network and Computer Applications **147**, 102428 (2019)
14. W. Yu, S. Sivakumar, D. Pao, *Pseudo-TCAM: SRAM-based architecture for packet classification in one memory access* (IEEE Networking Letters, 2019), pp. 1–1
15. R. Wei, Y. Xu, H.J. Chao, Finding nonequivalent classifiers in Boolean space to reduce TCAM usage. IEEE/ACM Transactions on Networking **24**(2), 968–981 (2016)
16. Y.C. Cheng, P.C. Wang, Scalable multi-match packet classification using TCAM and SRAM. IEEE Transactions on Computers **65**(7), 2257–2269 (2016)
17. S. Vakilian, M. Abbasi, A. Fanian, Increasing the efficiency of TCAM-based packet classifiers using dynamic cut technique in geometric space. Journal of Advanced Defence Science and Technology **6**(1), 65–71 (2015)
18. Z. Ruan, X. Li, W. Li, *An energy-efficient TCAM-based packet classification with decision-tree mapping. in TENCON 2013 - 2013 IEEE Region 10 Conference (31194)* (2013)
19. Y. Ma, S. Banerjee, A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification. SIGCOMM Comput. Commun. Rev. **42**(4), 335–346 (2012)
20. A. Bremler-Barr, D. Hendler, Space-efficient TCAM-based classification using gray coding. Computers, IEEE Transactions on **61**(1), 18–30 (2012)
21. A. Bremler-Barr, D. Hay, D. Hendler, Layered interval codes for TCAM-based classification. Comput. Netw. **56**(13), 3023–3039 (2012)
22. R. Cohen, D. Raz, *Simple efficient TCAM based range classification. in 2010 Proceedings IEEE INFOCOM* (2010)
23. C. Hao et al., DRES: Dynamic range encoding scheme for TCAM coprocessors. Computers, IEEE Transactions on **57**(7), 902–915 (2008)
24. Y. Chang, C. Su, in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*. Efficient TCAM encoding schemes for packet classification using Gray Code (2007)
25. A. Bremler-Barr, D. Hendler, in *Proceedings of the IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. Space-efficient TCAM-Based classification using Gray Coding (IEEE Computer Society, 2007), pp. 1388–1396
26. C.-T. Chan et al., in *information networking*, ed. by H.-K. Kahng. Scalable packet classification for IPv6 by using limited TCAMs (Springer Berlin Heidelberg, 2003), pp. 76–85
27. R. Panigrahy, S. Sharma, in *High Performance Interconnects*. Reducing TCAM power consumption and increasing throughput (Proceedings. 10th Symposium on, 2002, 2002)
28. Z. Kai et al., DPPC-RE: TCAM-based distributed parallel packet classification with range encoding. Computers, IEEE Transactions on **55**(8), 947–961 (2006)
29. Kesselman, A., et al. Space and speed tradeoffs in TCAM hierarchical packet classification. in Sarnoff Symposium, 2008 IEEE. 2008.
30. Kai, Z., et al. An ultra high throughput and power efficient TCAM-based IP lookup engine. in INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. 2004.
31. F. Zane, G. Narlikar, A. Basu, Coolcams: power-efficient TCAMs for forwarding engines. **1**, 42–52 (2003)
32. Vamanan, B. and T.N. Vijaykumar, TreeCAM: decoupling updates and lookups in packet classification, in Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. 2011, ACM: Tokyo, Japan. p. 1-12.
33. O.J. Murphy, R.L. McCraw, Designing storage efficient decision trees. IEEE Trans. Comput. **40**(3), 315–320 (1991)

34. D.E. Taylor, Survey and taxonomy of packet classification techniques. ACM Comput. Surv. **37**(3), 238–275 (2005)
35. H. Liu, *Efficient mapping of range classifier into ternary-CAM* (2002), pp. 95–100
36. A.J. McAuley, P. Francis, *Fast routing table lookup using CAMs. in INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies* (Networking: Foundation for the Future, IEEE., 1993)
37. K. Lakshminarayanan, A. Rangarajan, S. Venkatachary, Algorithms for advanced packet classification with ternary CAMs. SIGCOMM Comput. Commun. Rev. **35**(4), 193–204 (2005)
38. Lunteren, J.v. and T. Engbersen, Fast and scalable packet classification. IEEE Journal on Selected Areas in Communications, 2003. **21**(4): p. 560-571.
39. T.V. Lakshman, D. Stiliadis, High-speed policy-based packet forwarding using efficient multi-dimensional range matching. SIGCOMM Comput. Commun. Rev. **28**(4), 203–214 (1998)
40. H. Che et al., DRES: Dynamic range encoding scheme for TCAM coprocessors. IEEE Transactions on Computers **57**(7), 902–915 (2008)
41. Ruan, Z., X. Li, and W. Li. An energy-efficient TCAM-based packet classification with decision-tree mapping. in 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013). 2013. IEEE.
42. O. Rottenstreich, I. Keslassy, in *2010 Proceedings IEEE INFOCOM*. Worst-case TCAM rule expansion (2010)
43. D.E. Taylor, J.S. Turner, ClassBench: a packet classification benchmark. IEEE/ACM Transactions on Networking **15**(3), 499–511 (2007)
44. M. Varvello et al., Multilayer packet classification with graphics processing units. IEEE/ACM Transactions on Networking **24**(5), 2728–2741 (2016)
45. Y. Deng et al., in *Theoretical and Mathematical Foundations of Computer Science*. NPGPU: Network processing on graphics processing units (Springer, 2011), pp. 313–321
46. K. Kang, Y.S. Deng, in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Scalable packet classification via GPU metaprogramming (IEEE, 2011)
47. Zhou, S., S.G. Singapura, and V.K. Prasanna. High-performance packet classification on GPU. in High Performance Extreme Computing Conference (HPEC) 2014. IEEE
48. J. Zheng et al., *Accelerate Packet Classification Using GPU: A Case Study on HiCuts, in Computer Science and its Applications* (Springer, 2015), pp. 231–238

Publisher's Note