

RESEARCH

Open Access



Layered methods for updating AloT-compatible TCAMS in B5G-enabled WSNs

Mahdi Abbasi^{1*} , Shobeir Vakilian², Shakoor Vakilian³, Mohammad R. Khosravi⁴ and Hatam Abdoli¹

*Correspondence:
abbasi@basu.ac.ir

¹ Department of Computer
Engineering, Faculty
of Engineering, Bu-Ali Sina
University, Hamedan, Iran

² Department of Electrical
and Computer Engineering,
Isfahan University of Technology,
Isfahan 84156-83111, Iran

³ Department of Electrical
and Computer Engineering,
Islamic Azad University Science
and Research Branch, Tehran, Iran

⁴ Department of Computer
Engineering, Persian Gulf
University, Bushehr, Iran

Abstract

Classification is a fundamental processing task in advanced network systems. This technique is exploited in 5G/6G wireless sensors networks where flow-based processing of the internet packets is highly demanded by intelligent applications that analyze big volumes of data in a limited time. In this process, the input packets are classified into specific streams by matching to a set of filters. The ternary content-addressable memory (TCAM) is used in hardware implementation of internet packets. However, due to the parallel search capabilities, this memory leads to an increase in the speed and drop of hardware bundles compared to other types of software bundles, but with the increase in the number of rules stored in its layers, the power required for searching, inserting and eliminating increases. Various architectures have been proposed to solve this problem, but none of them has proposed a plan to reduce power consumption while updating the rules in the TCAM memory. In this paper, two algorithms are presented for reducing power consumption during TCAM memory upgrades. The key idea in the proposed algorithms is the reduction in the search range as well as the number of displacements while inserting and deleting rules in TCAM. Implementation and evaluation of proposed methods represent a reduction of more than 50% of the number of visits to TCAM in both proposed algorithms, as well as reducing the update time in the second proposed algorithm compared to the first proposed algorithm which confirms the efficiency of both methods.

Keywords: Packet classification, Layering, Ternary content-addressable memory, Number of searches, Update duration

1 Introduction

The future 5G/6G technologies present comprehensive heterogeneous network communications, with no restrictions. They provide high data transmission rates, accompanied by high consistency, improved scalability [1] and power efficiency [2–4]. The evidence behind these emerging technologies is to scale up the Internet of Things (IoT), where billions of mobile devices can process real-time sensory data using machine learning or artificial intelligence algorithms [5], and then communicate the condensed information over network [6, 7]. In such a complicated ecosystem, most of the nodes are linked together by using wireless sensor networks.

The Content-Addressable Memories (CAMs) are used in the hardware implementation of varied protocols used in Wireless Sensor Networks (WSN) [8], pattern

analytics[9], Deep Neural Networks (DNNs) [10] and Artificial Intelligence (AI) applications that need ultra-high speed and considerably low energy consumption, especially in edge devices [8, 11, 12].

Packet classification is the most basic processing in modern systems in Internet of Thing (IoT) networks [13, 14]. These networks include terrestrial WSNs [15], Unmanned Aerial Vehicle (UAV) networks [16], cellular networks [17], and especially green networks [18]. This basic processing is the most important factor in accelerating stream-based network services such as quality of service, traffic monitoring, and virtual private networks (VPNs) [19]. Due to the importance and challenges of the cluster, a lot of research has been done in the hardware and software tools [20–22]. Although the implemented software solutions due to cost and flexibility are desirable, they are not responsive to high speed in modern networks. The hard drive solutions, despite costing, were presented to increase the efficiency and speed in packet classification. In this regard, a team of researchers [23–25] has implemented the rule categorization to reduce power consumption using the decision tree. In these architectures, a two-stage structure is used for categorization. Figure 1 shows an outline of a two-step classifier architecture. Firstly, using a three-location addressable memory based on content, the IP addresses of the source and destination packets are matched to the prefix conditions of the source and destination URLs in the category rules. The output of this step is the SRAM home address, which contains the number of one of several TCAM blocks for matching the packet port in the second stage of the classification. In this step, the ports of origin and destination are examined. Due to the reduction in the number of bits sought in this method, power consumption is greatly reduced when searching. However, providing an efficient algorithm for upgrading with low power and high speed in these types of memory is one of the most important challenges to date.

Ternary content-addressable memories (TCAM) are widely used as a search engine for packet classification in the network. These types of memory store all the rules as an array of triplet bit strings. The contents of all TCAM addresses are searched in parallel with the input packet, and its output is the highest address of the matches occurred in the TCAM. The procedures used on TCAM are generally used to prevent their high levels of power consumption. The cost of making each bit of these memory is another factor that could be mentioned. Each cell of these memory needs 16 transistors while SRAMs require six transistors to be built [26]. Simultaneous enabling of all cells for searching on TCAM inputs will increase power consumption.

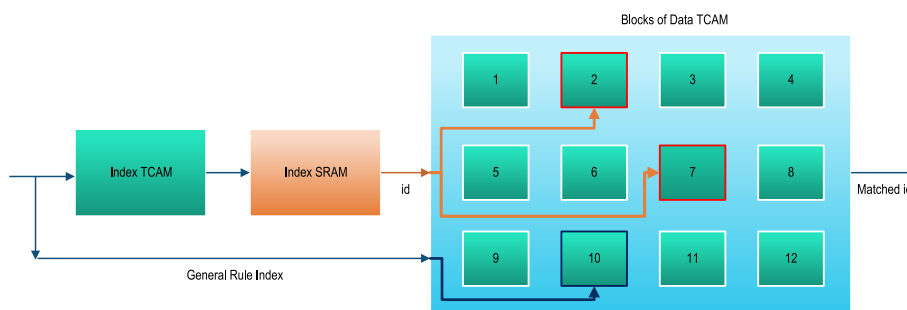


Fig. 1 The outline of a two-step classifier architecture

In this paper, two algorithms are proposed for reducing the updating time and enhancing the TCAM's efficiency in the first phase of the two-stage categorization, including origin and destination IP. In this task, the layering scheme has been used [27, 28]. In the rules layering scheme, there are no relationships between the two rules located on the same layer. In the presented algorithms, the scope of the enquiry has been considerably reduced by each search. Doing so, there are fewer searches to find the right place for rule location. Also, by decreasing the length of the chain of searched layers, the maximum need for the displacement of the rules decreases. This will lead to the reduction in the number of writings in the TCAM.

The paper structure is presented as follows. The second part presents the works done for updating. The third section addresses the details of the proposed algorithms. Part four describes the implementation and evaluation of proposed algorithms. Finally, Sect. 5 explains the general conclusion of the proposed solutions.

2 Review of the literature

In 2001, Shah and Gupta [27] presented two algorithms for updating TCAM with a field. The key idea behind the proposed algorithms was TCAM layering and improved use of its space. The purpose of these two algorithms is to increase the performance in packet routing and to reduce the update timing in its worst circumstance. OPT- PLO and CAO_OPT are the two algorithms presented in this study. The results of these two algorithms show that the CAO_OPT performance is better than PLO_OPT in packet classification.

Wong et al. [29] introduced an updating plan in 2004. In this design, an algorithm was used to update a set of rules in TCAM. The key idea behind this plan is to maintain the coherence of the PF (Policy Filtering) table during the upgrade process, rather than reducing the number of rule replacements during the upgrade time. This plan was presented to solve the problem of updating the PF table, despite a major challenge in the effective use of TCAM. The results in this design show that each rule requires, maximum, one second for updating. This time, of course, is achieved provided that two percent of the PF table will be empty.

Song and Turner [30] presented a plan for updating high-speed filters in the category of packet classification in 2006. In this scheme, introduction of new rules in TCAM's preferred data is done without examining the priority in rules with intrusion. Each rule has given a priority (block number). The highest priority in the rules occurs through several searches in TCAM. The authors of the proposed scheme believe that the highest priority in the rules matching is determined by a number of searches in TCAM. Priority in rule matching can be determined by the number of $\text{Log}(N)$ searches, where N is the number of the distinct priorities of rules, to determine the filter with the highest priority. In this scheme, the speed of access to TCAM will be reduced as a result of a numerous searches.

Mishra et al. [31] presented an algorithm for distributing rules in two TCAMs (ITCAM, LTCAM) in 2011 to store filters in the course of TCAM's gradual updating process. The results of the evaluation of this project show that the supposed design has improved by more than 48% at the speed of access to TCAM compared with the plans presented for updating by a TCAM.

Kuo et al. [28] provided an efficient TCAM collaborative processor memory for updating the routing table. The purpose of this project is to reduce the number of transmissions and to make optimal use of secondary memory. In this method, the average number of displacements per update is equal to the CAO_OPT algorithm [27]. The time to compute the displacement of the prefixes in the TCAM in [28] is less than [27]. In this method, the number of searches and the number of displacements is from $O(\text{Log}L)$, where L defines the last filled layer.

One of the major drawbacks in the reviewed plans is focusing on a single field while in architectures [32, 33] at least two prefix fields have been used. The work conducted in [32] aims to encode the range fields in packet classification, and consequently, the authors have not evaluated the performance of their architecture against the metrics that are used in our study including the number of writes and lookups for adding/deleting a rule from a TCAM. Hence, no meaningful comparison is possible. In this paper, in two presented algorithms, the speed of upgrading time has been improved according to the number of referrals based on two suggested fields.

3 Proposed method

Both proposed update algorithms are divided into three major parts. The first section is finding the right place to enter the rule. The speed of finding the packet location is one of the most important issues in updating the problem in the first part. In fact, the main goal in this section is to reduce the number of clocks needed for each search. The proposed updating plans aimed at reducing the number of searches. By each searching, the limit of the rule entered is reduced. In fact, the speed of finding the right place for locating the rule is increased. In the second part, after finding the appropriate location for the packet, the location of the spots is determined. The importance of the number of displacements is in the number of writings. Three-time clocks are required for each write at TCAM [28]. There are two goals in upgrading. The first aim is the transposition of rules related to the entered rule, and the second goal is the transposition of unconnected rules with the entered rule. In this paper, by prioritizing the second goal, we try to eliminate the transposition of inadequate rules, and the number of displacements associated with the rule entered is decreased. The third section of the presented update algorithm is related to the examining of free space in TCAM.

3.1 First proposed algorithm

Updating the lookup table in the routers or firewalls includes two sections of the adding and the removal of the rule. In the first proposed algorithm, there are 65 bits corresponding to each rule in two dimensions. The structure of these 65 bits is as follows: 32 bits for the first field, 32 bits for the second field and one bit for showing the validity of the TCAM line. The first field is for the source IP address and the second field for the destination IP address. IP addresses are considered as prefixes. By entering the rule in order to be added to the first proposed algorithm, a general search is performed in TCAM. The obtained output has two states. The first state is when the match has not occurred and the second state is when the matching occurs. In the absence of matching, there is no need to continue the search the rule. If the match occurs, there are three situations, depending on the state of the two fields of rule. In the first position, the entered

rule is covered by the rule. In the second situation, the entered rule will cover the found rule. In the third position, the match is exact and there is a rule in the router. By search and precise matching occurrence, the removal operation is performed and some replacements occur in the router. The number of searches and the number of writings in this algorithm is of the order of $O(R)$, where R represents the number of TCAM. Following the examination, the two sections of adding and removal are performed in the first proposed algorithm.

3.1.1 Adding the rule

Adding a rule with two prefix fields in the first algorithm starts with a general search in the TCAM, and in the case of no occurrence of the rule match in the first address of the free space, the adding action is performed. Figure 2 illustrates this function. In the case of a match occurrence, the occurred matching, the type of the occurred match is examined and the appropriate reactions for locating the rule in TCAM. Some states occur in the case of match occurring. These states as illustrated in the flowchart of Fig. 3 are as follows:

First State: The length of the first and second prefix of the rule found equates the length of the first and second prefix of the entered rule. Therefore, there is no need to add the rule entered and the exact match has occurred.

Second State: The first and second prefix of the found rule covers the corresponding first and second prefixes of the entered rule, or the first or second prefix of the found rule is equal to the corresponding first or second prefix of the entered rule and covers another prefix. So, the entered rule locates in the address of the found rule. The found address, along with the rules established after the address of the found rule, transform one address size to the lower address.

Third state: The first and second prefix of the entered rule covers the first and second prefix of the found rule, or the first or second prefix of the entered rule equals with the corresponding first or second found prefix. It covers the next prefix, or one of the prefixes is covered and the other may not be covered. Therefore, from the address found to the last address, searching is done in order to investigate the occurrence of one of the three following conditions. The first condition is the exact match. The second condition occurs if the entered rule is covered by the found rule. Therefore, the entry rule is located at the found address, and the rest of the rules one address size are moved to the

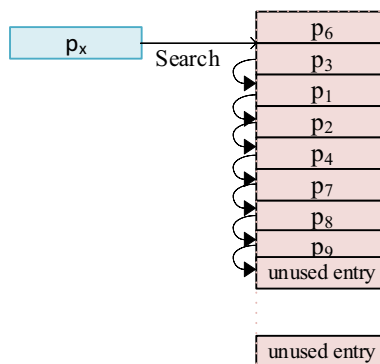


Fig. 2 Adding rule in the first algorithm

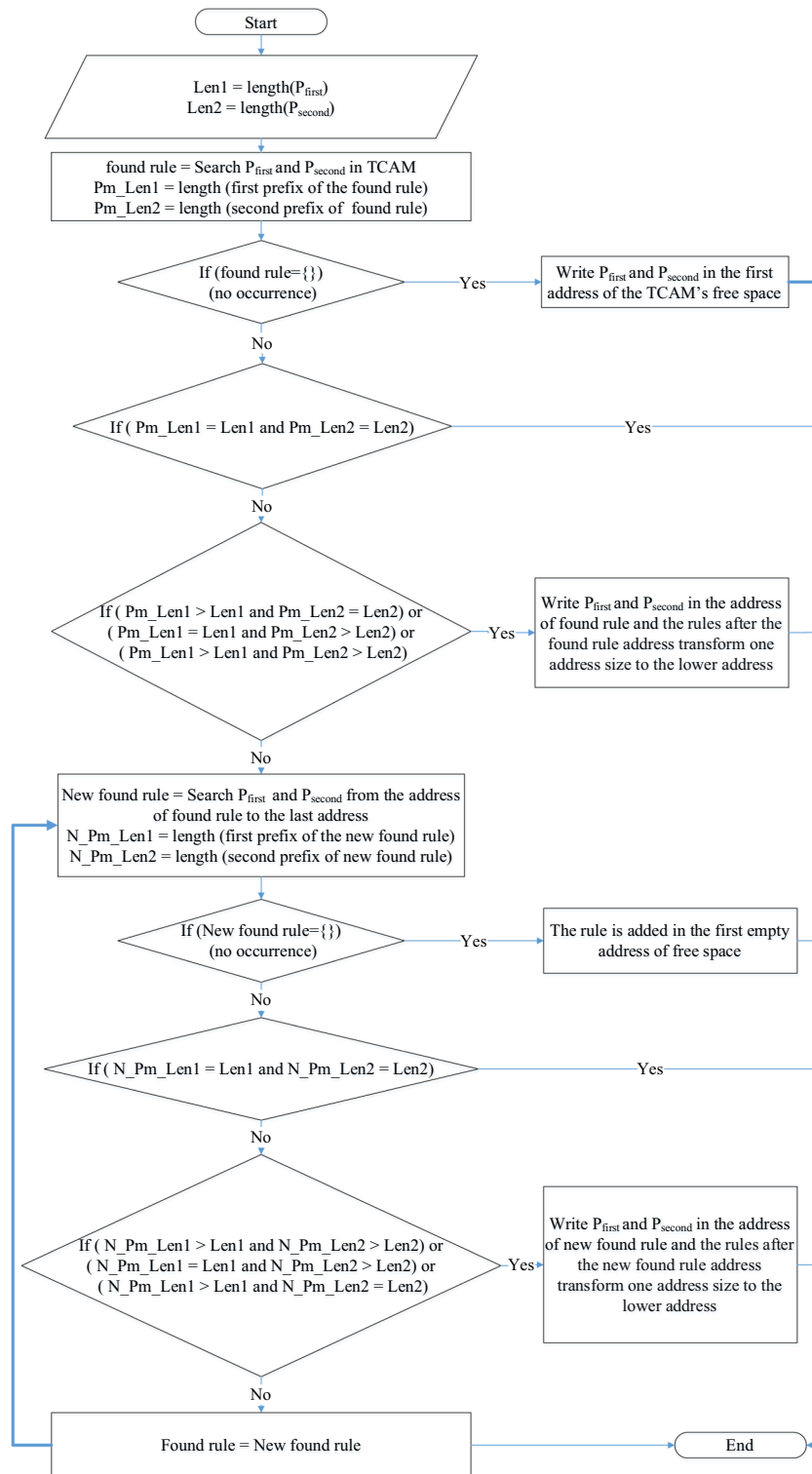


Fig. 3 The flowchart of adding rule in the first algorithm

lower. The third condition come out in the case of a non-compliance until achieving the last considered address for inserting the rules, i.e., no matching is found. Hence, the rule is added in the first empty address of free space.

Figure 2 shows an example of adding an input rule into the TCAM memory. The Px rule is inserted for adding to TCAM. A search is done in TCAM. Match has taken place. The Px rule is surrounded by P3; therefore, the p3 rule, along with the rules in the lower TCAM addresses, is passed to a lower address by a single unit.

3.1.2 Deleting the rule

The search for finding the right place for the entered rule for deletion is the same as adding the rule in the first algorithm. Deletion starts with a search in TCAM. Obviously, the absence of compliance and the lack of compliance with the rule that entered into it means that there is no rule in the TCAM. If the rule to be removed in the TCAM is found, all the rules found in the address after the found rule are converted to the same rule by one single address size.

Figure 4 shows an example of the exact match occurrence during deletion. Also, this algorithm is illustrated by the flowchart of Fig. 5 All rules in the TCAM move as one address size as to the address where the match occurred, and the last rule is removed from the TCAM.

3.2 The second proposed algorithm

In order to update TCAM by the second proposed algorithm, in addition to the 64 bits mentioned, 11 bits are needed to resolve 1089 layers, and a bit is also used to determine the TCAM line validity. Therefore, each TCAM line is composed of 76 bits. There are no relationships between both of the two rules in one layer. If a rule is located in the higher layer, for example, the second layer, it means that there is a rule in layer one and the rule in layer 2 is covered by the rule specified in the number one. In the case of a search in TCAM, if there is a relationship between the two prefixes of the entered rule with both prefixes from found rules, the matching is performed. The number of layers filled during the addition of the rule in TCAM is called the length of the chain of rule. The circle surrounding the rule is a chain in which the rules are linked together throughout a chain. The number of searches to find the appropriate layer in the TCAM is of the grade of $O(\log 2L)$, and the number of displacements between the layers is from $O(L)$.

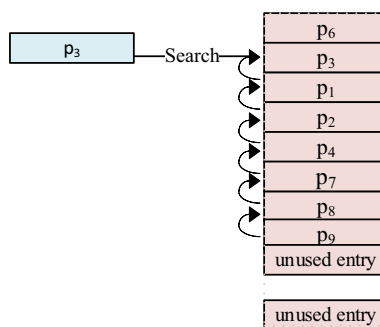


Fig. 4 Deleting the rule in the first algorithm

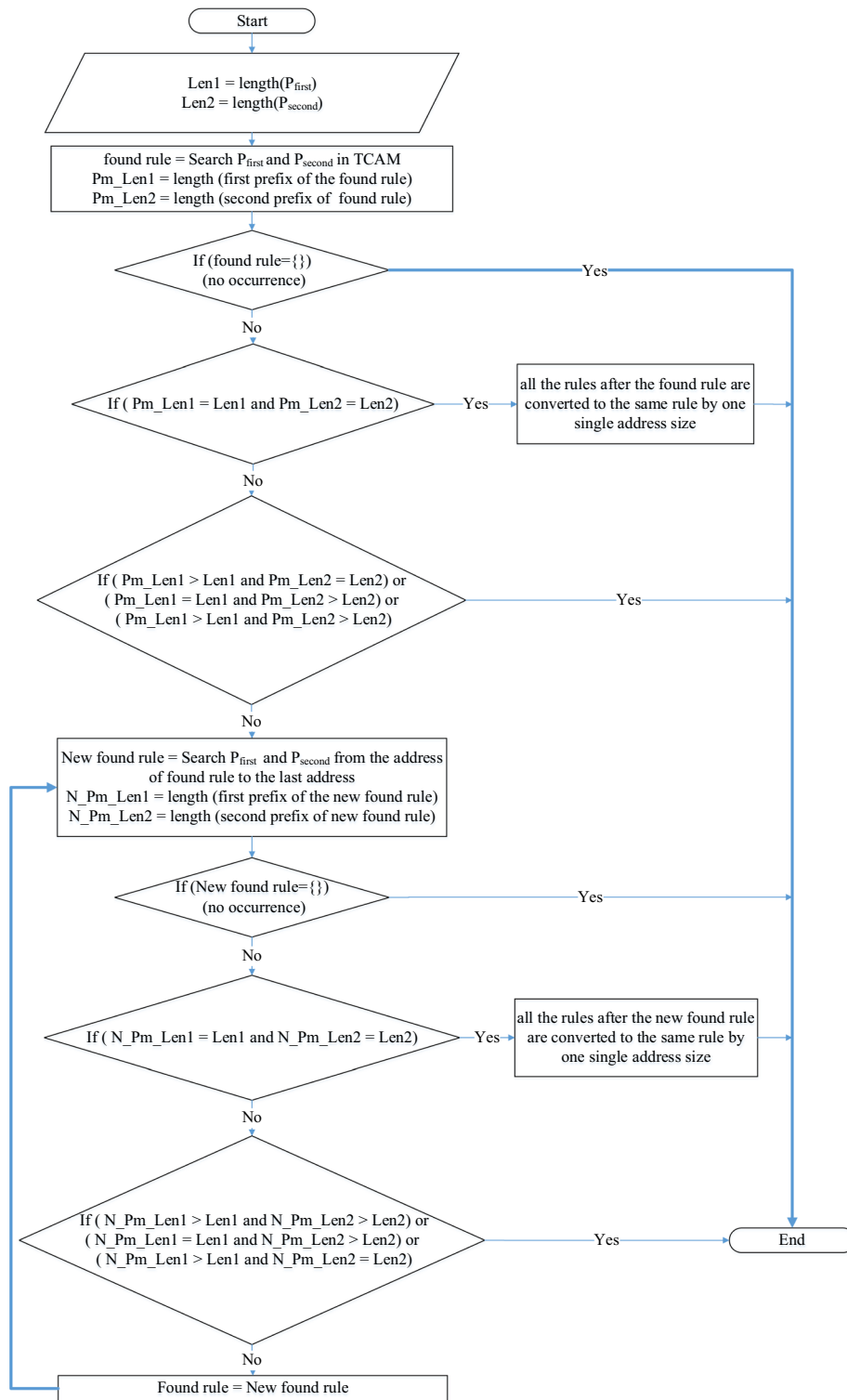


Fig. 5 The Flowchart of deleting the rule in the first algorithm

Here, L represents the number of TCAM layers. In the following, two major sections of the TCAM update, including adding and deleting the rule in the second proposed algorithm, are discussed.

3.2.1 Adding a new rule

Adding a rule with two prefix fields in the second algorithm starts with a total TCAM search, and if the match does not occur, the rule in the first address of the free space is added to layer one. The flowchart of Fig. 6 shows this algorithm. If the match occurs, the matching states are considered as follows:

- First State: If there is an exact match, there is no need to add the rule; the first and second found prefix of the rule is equal to the first and second prefix of the entered rule, respectively. And it is implied that there is a rule in TCAM.
- Second state: In the case of surrounding matching occurrence, the entered rule is set in the address of the found rule and the found rule is transferred to a higher layer. Surrounding matching occurrence is a matching in which both the first and the second prefix of the found rule surround both the first and second prefixes of the entered rule, or one of the first or second prefixes of the found rule equal with the first and second prefix of the entered rule, respectively, and another prefix of the found rule contain the corresponding prefix of the entered rule.
- Third State: If the surrounded matching occurred, it means that the proper layer for the rule layer settlement is placed in the layers after the found rule layer. Therefore, the middle layer is chosen among the last layer, the rule layer and the layer after the found rule layer and the search is performed in the selected layer. Surrounding matching is an adaptation in which both the first and the second prefix of the entered rule encompass both the first and the second prefix of the found rule, respectively, or one of the first or second prefixes of the entered rule equals with the first or second prefix, respectively, and another prefix of the entered rule surrounds the corresponding prefix in the found rule. As overall search is done, given that matching occurrence happen in which one of the two prefixes of the found rule is covered by the corresponding prefix of the entered rule and the other prefix, the middle layer is selected and the search is done in the middle layer. According to the search of the middle layer in the third state, the following states can occur:
 - Accurate matching occurs. So, there is no need to add the rule and the rule exists in TCAM.
 - In the absence of matching or surrounded matching, the desired layer can be located in the upper half of the TCAM. So, the middle layer is selected between the layers before the searched layer and the lower layer and the search is performed in the selected layer.
 - In the case of a match occurrence in which one of the two prefixes of the found rule is covered by the equivalent prefix from the entered rule and the other layer covers the other prefix from the lower layer on, then due to the impossibility of predicting the location of the entered rule for examining the surrounding chain, the rule is being searched.

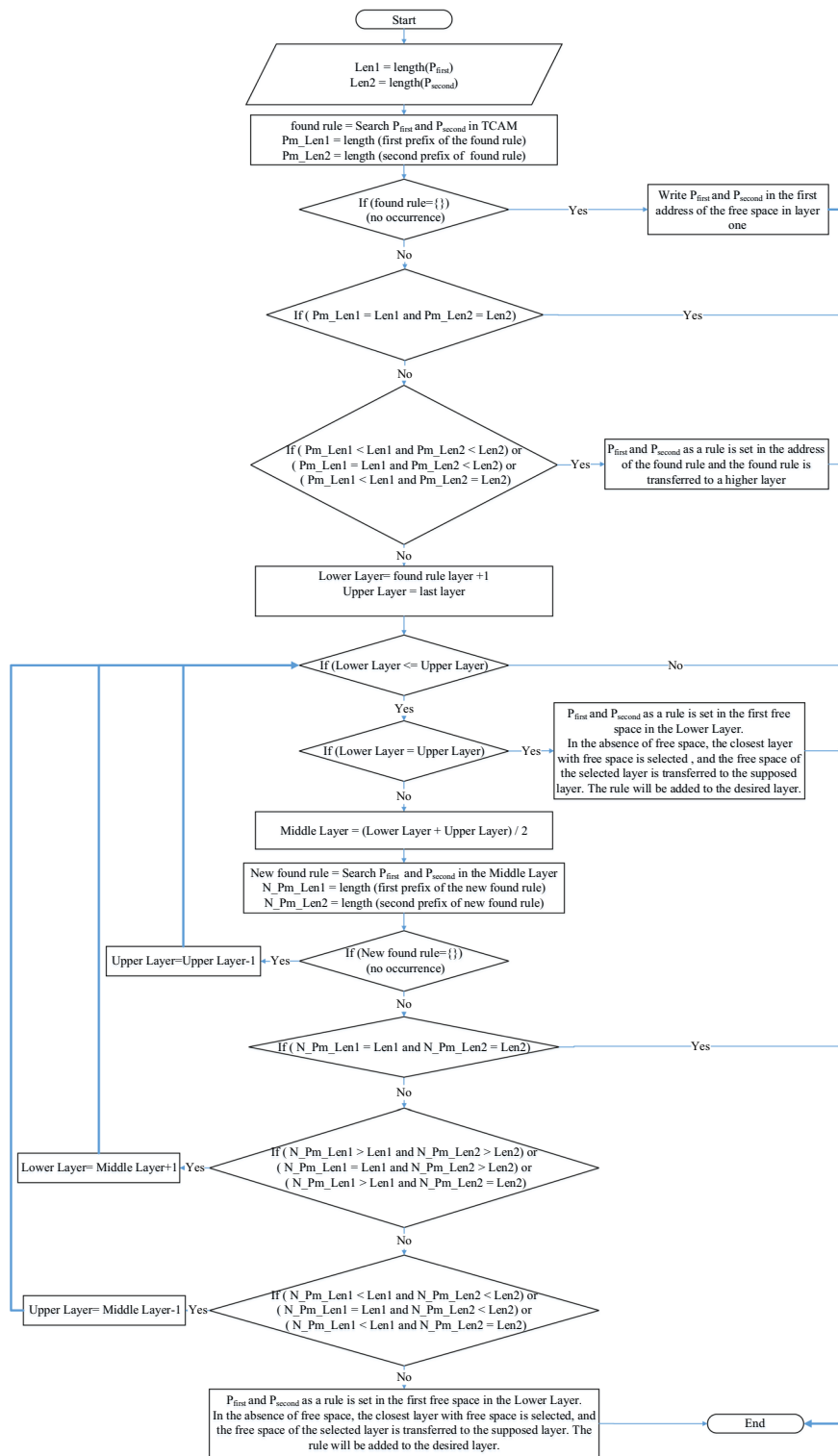


Fig. 6 The Flowchart of adding a rule in the second algorithm

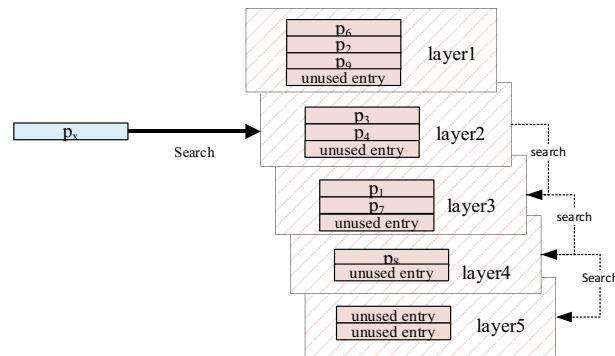


Fig. 7 An example of adding the rule to the layered TCAM

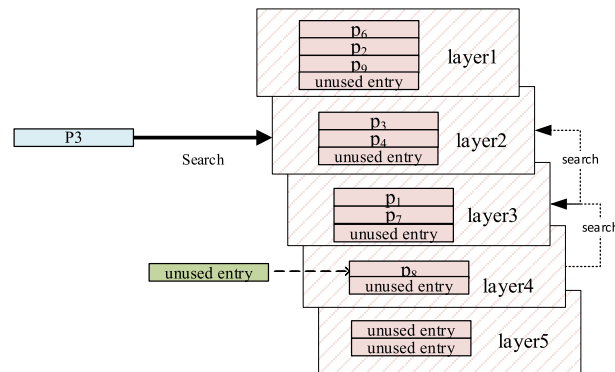


Fig. 8 An example of deleting a rule

By selecting a suitable layer, the free space is being examined. In the absence of free space, the closest layer with free space is selected based on computations that takes the lowest number of displacements into account, and the free space of the selected layer is transferred to the supposed layer. The rule will be added to the desired layer later on.

Figure 7 shows an example of adding the rule to the TCAM. Rules P_8, P_1, P_3, P_6 , make up a chain of surrounding rule. The P_x rule will be entered for adding to the TCAM. Matching occurs in layer one. Due to the fact that the rule P_x covers the rule P_6 and the fact that the last filled layer is Layer 4, the search takes place in the middle layer, Layer 2, of TCAM. Matching has been occurred. The P_x rule is surrounded by P_3 . So the P_3 rule is moved to the third layer and the P_1 rule that covers the P_3 rule is transferred to the fourth layer. Also, during the rule chain, the P_8 rule is moved to the lower layer.

3.2.2 Deleting the rule

In the second algorithm, the search for finding the right place for the entered rule for deletion is the same as adding the rule. If the rule is found, the found rule will be deleted from the TCAM. Figure 5 shows the removal of the P_3 rule from the TCAM. P_1 and P_8 rules which are located in the lower layer and in the same chain of rule are transferred to the upper layer (Fig. 8).

As shown in the flowchart of Fig. 9, firstly, a search is performed with the content of the deleted rule in the higher layer. Considering the performed search, the following states can occur:

State 1: Failure in matching occurrence means that there is no rule in the higher level to cover the deleted rule. Accordingly, there is no need for movement.

State 2: If matching with the content of found rule occurs, a search is done in the deleted rule layer. If no matching occurs, the found rule in the upper layer will be passed

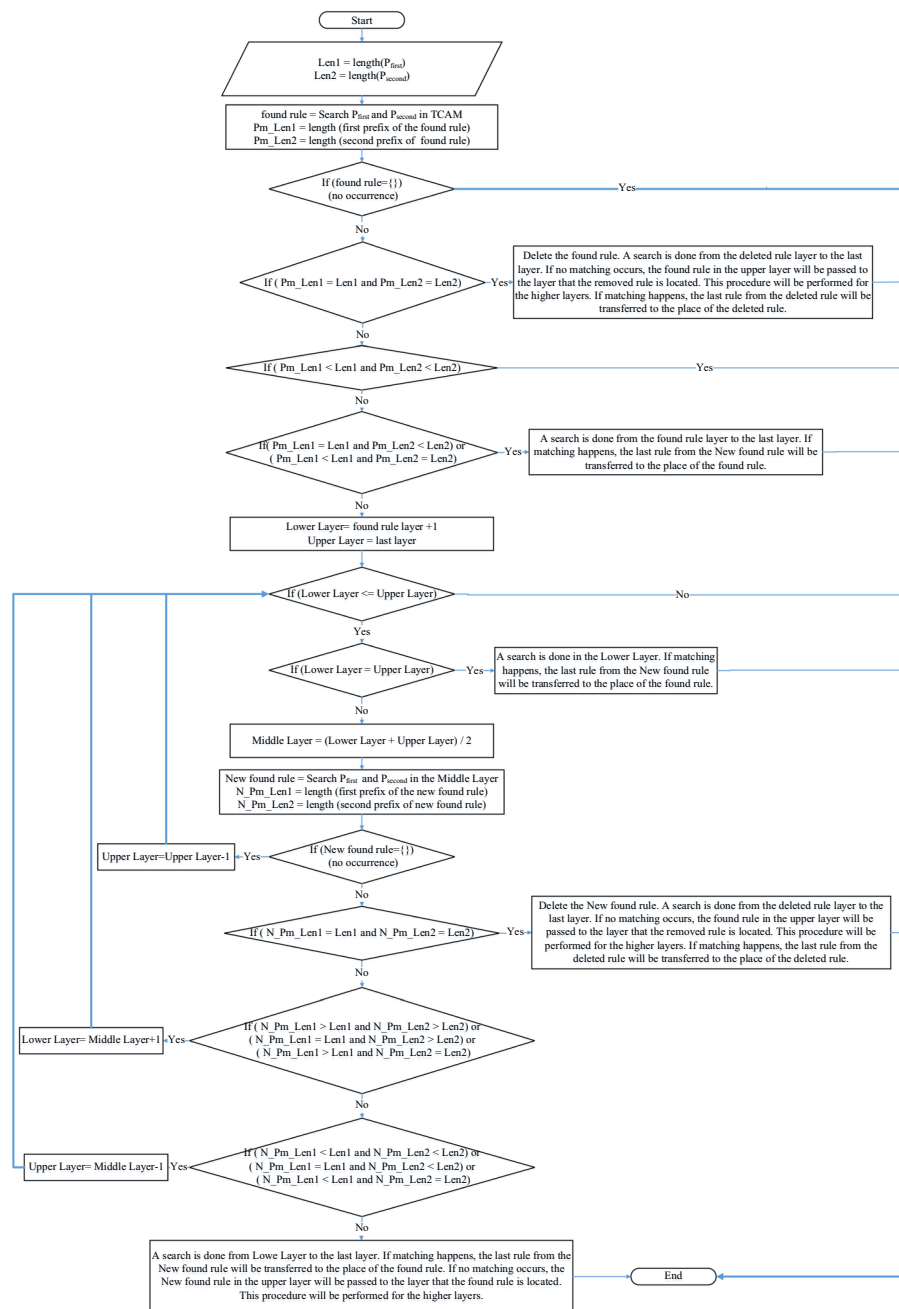


Fig. 9 The flowchart of deleting a rule in the second algorithm

to the layer that the removed rule is located. This procedure will be performed for the higher layers. If matching happens, there is no need for displacement, and only the last rule from the deleted rule will be transferred to the place of the deleted rule. This is the rule, for freeing up the space.

4 Implementation and evaluation

The implementation of proposed algorithms using the VHDL language, based on the IEEE 1994 standard, has been conducted in the ISE development environment, version 13.1.

The set of rules and test packs required for evaluating and testing the proposed architecture is created using the ClassBench tool in [34]. Using this tool, you can create a set of rules that have the same properties as the actual rule set in the classification of IP packets.

Three general types of rule sets are produced by the ClassBench tool. These three types are as follows: Access Control List (ACL), Firewall (FW), and IP (IPC) Chain, which are described based on the number of used rules. In the following, the implementation process of the two algorithms is described and finally, after introducing the evaluation parameters, the results of the implementation of the two proposed methods are compared.

The simulation results are shown in Figs. 10 and 11. By increasing the number of rules, the number of searches and the number of writings in the first algorithm increases linearly. Increasing the number of rules in the first proposed algorithm increases the probability of occurrence of states with a high number of writings in TCAM. An important point in the plots of Figs. 10 and 11 is that the number of lookups on ACL1k is consider remarkably lower as compared to other two datasets including FW1k and IPC1k. The distribution of IP addresses in ACL1K is the main reason that makes the number of lookups considerably lower as compared to each of the other same-size rulesets. Hence, the probability that the router will be locked, increases (i.e., during the update time it gets offline). Also, as shown in Fig. 12, the number of searches in the second algorithm has not remarkably changed by an increase in the number of rules. Therefore, with increasing number of rules, the number of searches does not increase linearly. With the

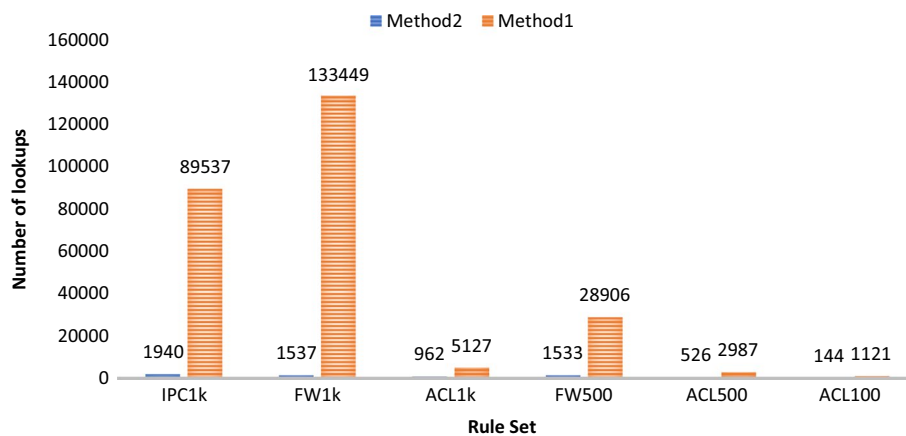


Fig. 10 Number of lookups for adding a rule

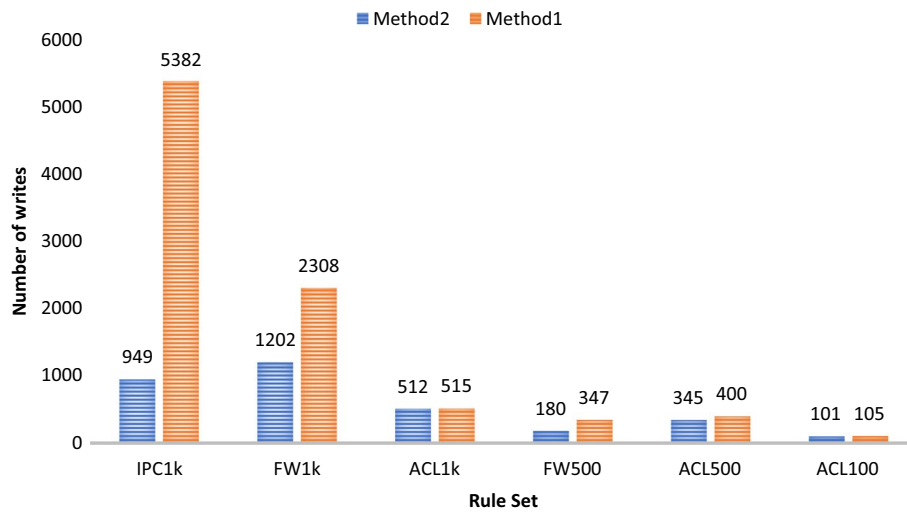


Fig. 11 Number of writes required for adding a rule

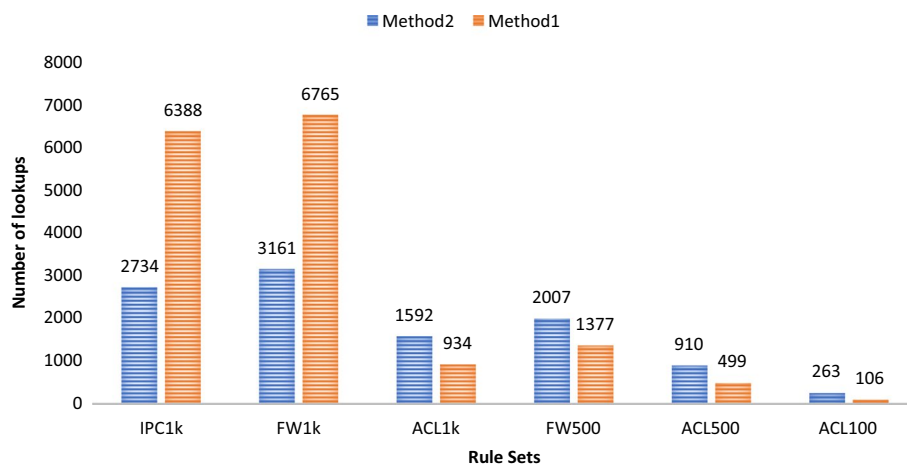


Fig. 12 Number of lookups for removing a rule

entrance of each rule, due to the low number of searches, the router’s lock time will be reduced.

In Fig. 12, the number of lookups in the first algorithm, compared to the other algorithm, is increased by increasing the number of rules. In other words, lower number of displacements is done in the second algorithm compared to the first algorithm. Therefore, the second algorithm shows better efficiency than the first algorithm. Increasing the number of searches in the first method due to the occurrence of the surrounded matching at the beginning of the TCAM; then the entered rule can be found between the found matching and the last URL containing the content in the TCAM. Therefore, from the address after the found rule, the search is done to find the right place of the entered rule. If the match occurs at the end of the TCAM, fewer searches are made. In the second method, due to the use of layering scheme, the number of required searches reduces. The number of writings depends on the occurrence of the matching in the

TCAM. In the second method, the number of displacements depends on the length of the prefix surrounding chain, but in the first method, in the case of the occurrence of the surrounding matching, all the rules after the matched rule are also displaced, and this case is displayed in diagrams of Figs. 11 and 13. If the matching occurs in the first algorithm at the last TCAM addresses, since there is no need to select another layer as the second algorithm, the number of searches in the first algorithm decreases compared to the second algorithm.

Figures 12 and 13 show the number of searches and writings while deleting from TCAM. The results in deleting rules are approximately like the results of adding rules to TCAM. However, the number of searches in the first method can be less than the number of searches in the first algorithm in the case of matching in the end of TCAM and in the case of no need to examine the higher rows of TCAM. Note that the number of searches in the second algorithm includes the number of searches that are required after removal for displacement. In the first algorithm, after deleting the rule, there is no need to search. However, there is an unpredictable state in the first algorithm. In this case, the number of searches in the first algorithm is considerably increased compared to the second algorithm.

Figures 14 and 15 show the number of references to TCAM and secondary memory in both algorithms. According to Fig. 14, the number of referrals as a result of writing and searching in memory by increasing the rule in the first algorithm compared to the second algorithm has increased in adding the rule. Therefore, it is predicted that by increasing the number of rules, due to the use of layering plan, the second algorithm will have better results in the number of searches and displacements, and therefore the number of referrals in adding the rule, compared to the first algorithm.

Complexity of adding/deleting a rule in the first algorithm is $O(R)$, where R denotes the number of TCAM entries, while the complexity of the same operations by the second algorithm is $O(L_1 \times L_2)$, where L_1, L_2 , respectively, denote the number of inner and outer layers in TCAM.

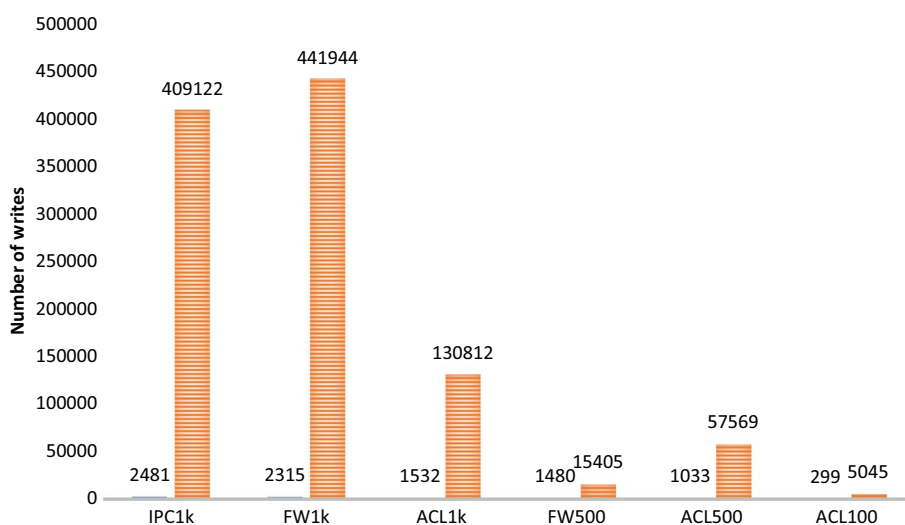


Fig. 13 Number of writes required for removing a rule

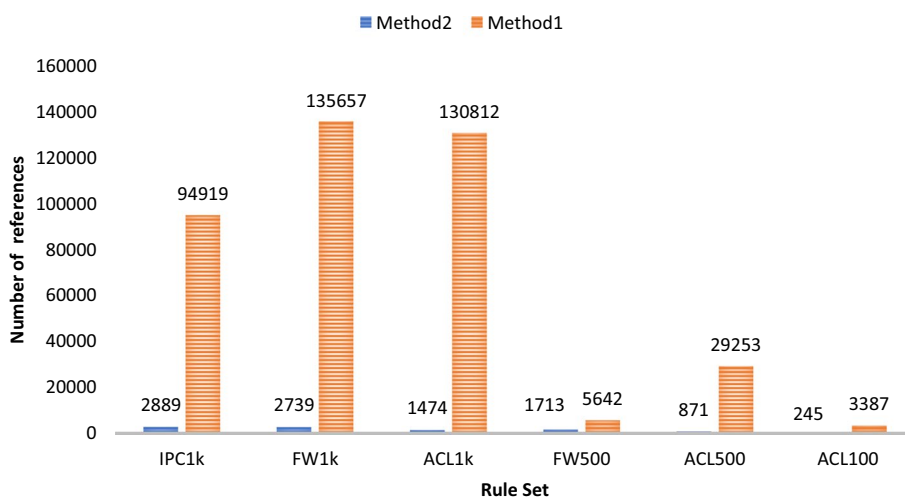


Fig. 14 Number of references for adding a new rule

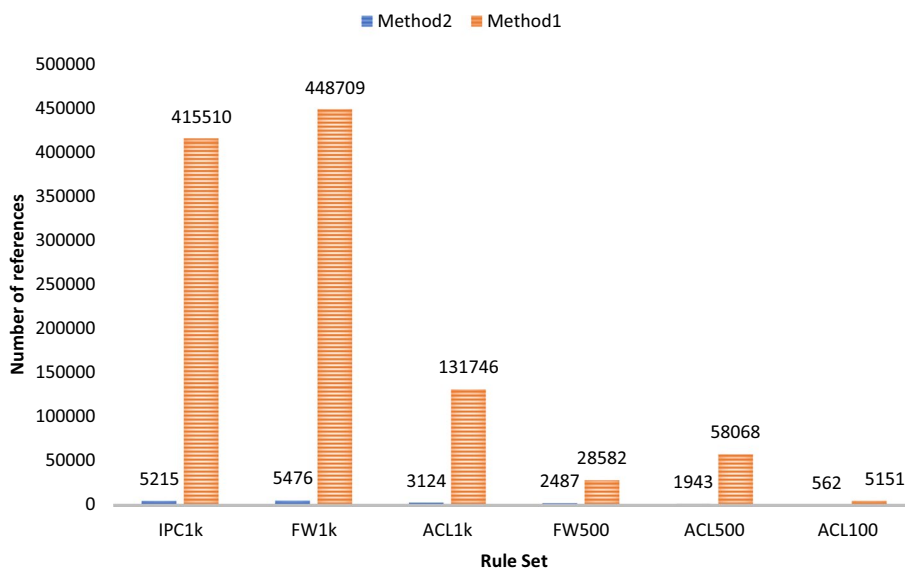


Fig. 15 Number of references for removing a new rule

In fact, in the first algorithm compared to the second algorithm, the unpredictable state can significantly reduce the number of entered rules in the specified time for adding or removing TCAM. The number of displacements in the first algorithm has no relationship with the substitution of the rules by the entered rule, but in the second algorithm, the cause of the displacement is due to the lack of free space in the selected layer for adding the rule and the relationship of the found rule to the entered rule. According to the diagram in Fig. 15, the number of references as a result of writing and searching in the memory by increasing the rule in the first algorithm has increased compared to the second algorithm in eliminating the rule. Therefore, it is observed that by increasing the number of rules, due to the use of layering scheme, the second algorithm compared to the first algorithm displays better results in the

number of searches and the number of displacements, and therefore the number of referrals in the elimination of the rule. Therefore, the efficiency of the first algorithm decreases compared to the second algorithm.

5 Conclusion

In this paper, two algorithms were presented in order to update TCAM IP packet class rules. The main problem with the existing methods in updating clustering, based on TCAM, is the absence of an algorithm for updating TCAM with two fields. Also, the number of visits to TCAM was one of the most important criteria for improving classifying updating based on TCAM in this article. The number of searches in TCAM during updating is one of the factors for reducing TCAM referrals and thus reducing the time of updating. In the searching operation, all TCAM rows are searched in parallel, which reduces the searching time.

The second algorithm is based on the layering scheme, but the first algorithm is based on simple and raw method. Based on the obtained results of the implementation of the algorithms, due to the decline in the number of referrals, the update time in the second algorithm was lower than the first algorithm. The main reason was the act of limiting and regularizing the search field in TCAM using layering scheme. All the searches in the second algorithm are performed in TCAM, while in the first algorithm, only the first search in the TCAM is performed. Therefore, taking into account that all bits are activated at the time of the search, the power consumption of the second algorithm increases compared to the first algorithm.

Another important factor in updating time is the number of writings and displacements in TCAM. In the first algorithm, the displacement factor, in addition to the rules which are matched to the entered rule, the displaced rules are in line with creating the space. These rules are not related and adapted with the entered rule. In the second algorithm, the layer displacement was reduced as much as possible by creating layers. In the created layers, no two rules are matched. Therefore, the number of transmissions and writings decreased in TCAM, and as a result of decreasing in the number of transmissions, the number of writings also decreased. By reducing the number of referrals to the TCAM, due to the decrease in the number of searches and the number of writings in the TCAM, and therefore reducing the update time, the efficiency of the offline router over updating time is increased in the second algorithm compared to the first algorithm and it gets online in less time.

Abbreviations

DRAM	Dynamic random access memory
IoT	Internet of Things
CAM	Content-addressable memory
TCAM	Ternary content-addressable memory
IP	Internet protocol
ACL	Access control list
FW	Firewall
IPC	IP chain
SRAM	Static random access memory
ASIC	Application specific integrated circuit
FPGA	Filed programmable gate array
WSN	Wireless sensor network
DNN	Deep neural network
AI	Artificial intelligence

Acknowledgements

Authors thank editor and reviewers for their time and consideration.

Authors' information

Not applicable.

Author contributions

MA and SV 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.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Declarations**Competing interests**

The authors declare that they have no competing interests.

Received: 28 March 2022 Accepted: 1 June 2022

Published online: 13 June 2022

References

- H.H. Attar et al., Bit and packet error rate evaluations for half-cycle stage cooperation on 6G wireless networks. *Phys. Commun.* **44**, 101249 (2021)
- T.S. Kumar, S.L. Tripathi, Comprehensive analysis of 7T SRAM cell architectures with 18 nm FinFET for low power biomedical applications. *Silicon* (2021).
- A. Gangadhar, K. Babulu, Design of low-power and high-speed CNTFET-based TCAM cell for future generation networks. *J. Supercomput.* **77**(9), 10012–10022 (2021)
- A. Agha, H. Attar, A.K. Luhach, Optimized economic loading of distribution transformers using minimum energy loss computing. *Math. Probl. Eng.* **2021**, 1–9 (2021)
- C. Feng et al., Blockchain-empowered decentralized horizontal federated learning for 5G-enabled UAVs. *IEEE Trans. Ind. Inf.* **18**, 3582–3592 (2021)
- Attar, H., et al. Network coding hard and soft decision behavior over the physical layer using PUMTC. In: *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)* (2018).
- X. Li, Z. Zhou. An efficient privacy-preserving classification method with condensed information. In: *International Conference on Image and Graphics* (Springer, 2017).
- M. Arulvani, M.M. Ismail, Low power FinFET content addressable memory design for 5G communication networks. *Comput. Electr. Eng.* **72**, 606–613 (2018)
- R. Yang et al., Ternary content-addressable memory with MoS₂ transistors for massively parallel data search. *Nat. Electron.* **2**(3), 108–114 (2019)
- Choi, W., et al. Content addressable memory based binarized neural network accelerator using time-domain signal processing. In: *Proceedings of the 55th Annual Design Automation Conference* (2018)
- C. Chen et al., Distributed computation offloading method based on deep reinforcement learning in ICV. *Appl. Soft Comput.* **103**, 107108 (2021)
- H.E. Yantir, A.M. Eltawil, K.N. Salama, A hardware/software co-design methodology for in-memory processors. *J. Parallel Distrib. Comput.* **161**, 63–71 (2022)
- W. Wei, et al., Multi-objective optimization for resource allocation in vehicular cloud computing networks. *IEEE Trans. Intell. Transp. Syst.* (2021).
- M. Abbasi et al., Efficient flow processing in 5G-envisioned SDN-based Internet of Vehicles using GPUs. *IEEE Trans. Intell. Transp. Syst.* **22**(8), 5283–5292 (2020)
- H. Attar et al., Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **16**(6), 1550147720913233 (2020)
- C. Chen et al., Data dissemination for industry 4.0 applications in internet of vehicles based on short-term traffic prediction. *ACM Trans Internet Technol (TOIT)* **22**(1), 1–18 (2021)
- E. Qafzezi, et al. A survey on advances in vehicular networks: problems and challenges of architectures, radio technologies, use cases, data dissemination and security. In: *International Conference on Advanced Information Networking and Applications* (Springer, 2022)
- A. Malik, R. Kushwah, A survey on next generation IoT networks from green IoT perspective. *Int J Wireless Inf Netw* **29**, 36–57 (2022)
- M. Abbasi, S. Vesaghati-Fazel, M. Rafiee, MBitCuts: optimal bit-level cutting in geometric space packet classification. *J. Supercomput.* **76**(4), 3105–3128 (2020)
- D.E. Taylor, Survey and taxonomy of packet classification techniques. *ACM Comput. Surv. (CSUR)* **37**(3), 238–275 (2005)
- M. Abbasi, M. Rafiee, A calibrated asymptotic framework for analyzing packet classification algorithms on GPUs. *J. Supercomput.* **75**(10), 6574–6611 (2019)

22. M. Abbasi, R. Tahouri, M. Rafiee, Enhancing the performance of the aggregated bit vector algorithm in network packet classification using GPU. *PeerJ Comput. Sci.* **5**, e185 (2019)
23. S. Vakilian, M. Abbasi, A. Fanian, Increasing the efficiency of TCAM-based packet classifiers using dynamic cut technique in geometric space. *J. Adv. Def. Sci. Technol.* **6**(1), 65–71 (2015)
24. 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)
25. S. Vakilian, M. Abbasi, and A. Fanian. Increasing the efficiency of TCAM-based packet classifiers using intelligent cut technique in geometric space. In *2015 23rd Iranian Conference on Electrical Engineering* (2015)
26. J.V. Lunteren, T. Engbersen, Fast and scalable packet classification. *IEEE J. Sel. Areas Commun.* **21**(4), 560–571 (2003)
27. D. Shah, P. Gupta, Fast updating algorithms for TCAM. *IEEE Micro* **21**(1), 36–47 (2001)
28. F. Kuo, Y. Chang, C. Su, A memory-efficient TCAM coprocessor for IPv4/IPv6 routing table update. *IEEE Trans. Comput.* **63**(9), 2110–2121 (2014)
29. Z. Wang et al., CoPTUA: consistent policy table update algorithm for TCAM without locking. *IEEE Trans. Comput.* **53**(12), 1602–1614 (2004)
30. H. Song and J. Turner. NXG05-2: Fast Filter Updates for Packet Classification using TCAM. In *IEEE Globecom 2006* (2006)
31. T. Mishra, S. Sahni, and G. Seetharaman. PC-DUOS: Fast TCAM lookup and update for packet classifiers. In: *2011 IEEE Symposium on Computers and Communications (ISCC)* (2011)
32. Z. Ruan, X. Li, and W. Li. An energy-efficient TCAM-based packet classification with decision-tree mapping. In *TENCON 2013–2013 IEEE Region 10 Conference (31194)* (2013).
33. M. Abbasi et al., Ingredients to enhance the performance of two-stage TCAM-based packet classifiers in internet of things: greedy layering, bit auctioning and range encoding. *EURASIP J. Wirel. Commun. Netw.* **2019**(1), 287 (2019)
34. D.E. Taylor, J.S. Turner, ClassBench: a packet classification benchmark. *IEEE/ACM Trans. Netw.* **15**(3), 499–511 (2007)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
