AIR: An AI-based TCAM Entry Replacement Scheme for Routers

1st Yuchao Zhang
Beijing University of Posts and Telecommunications
Beijing, China
yczhang@bupt.edu.cn

2rd Peizhuang Cong
Beijing University of Posts and Telecommunications
Beijing, China
congpeizhuang@bupt.edu.cn

3th Wendong Wang
Beijing University of Posts and Telecommunications
Beijing, China
wdwang@bupt.edu.cn

4th Gong Zhang
Huawei Theory Research Lab
Hongkong, China
nicholas.zhang@huawei.com

5th Li Chen
Huawei Theory Research Lab
Hongkong, China
chen.li7@huawei.com

6th Yanwei Xu
Huawei Theory Research Lab
Hongkong, China
xuyanwei1@huawei.com

***Abstract*—With the development of 5G and IoT (Internet of Things), more and more devices are connected to the Internet, raising requirements on network routers. To satisfy the requirements on speed and efficiency, commercial routers have to keep expanding TCAM (Ternary Content Addressable Memory) size.**

In this paper, we analyzed real traces of a backbone network router located in New York, and disclose the characteristics of traffic. Based on this, we propose a very lightweight AI-based solution to replace the traditional TCAM updating scheme, named AIR. In the experiments with 20,000 flow entries, the router with AIR with 2,000 TCAM entries performed similarly to a traditional router with 16,000 TCAM entries. Further, the improvements are more significant in larger scale scenarios.

***Index Terms*—TCAM, router, AI, prediction**

I. Introduction

With the development of 5G and IoT (Internet of Things), more and more terminals are connecting to the Internet, the number of entries in a real core router is close to 700,000 [1]. The explosive growth in the number of entries raises higher requirements for routers both on scale and efficiency. To guarantee the query performance under such explosive growth of traffic, the current TCAM-based commercial routers have to increase their TCAM size to keep pace of the growth of query number. The TCAM size of common routers is as large as 512,000[1]. This makes routers not only heavy but also superfluous, especially for large-scale core routers.

How to use a small TCAM size router to achieve the query efficiency of traditional routers? First of all, TCAM's cost, circuit logic complexity and power consumption are positively correlated with its size [2]. Second, as the routing table entries are stored in TCAM in logical order, a large number of entries will be moved to satisfy the order requirements when updating/adding entries, which directly results in high updating overhead. What's worse, limited to the circuit logic of TCAM, the query process can not be executed in parallel during refreshing, and has to be paused, so the line-speed forwarding process will also be affected. [3]

[1]The Cisco Catalyst 6500 series, such as WS-SUP720-3BXL, VS-S720-10G-3CXL and RSP720-3CXL-GEthe, the default IPv4 TCAM size is 512,000 and the maximum value is 1,000,000.

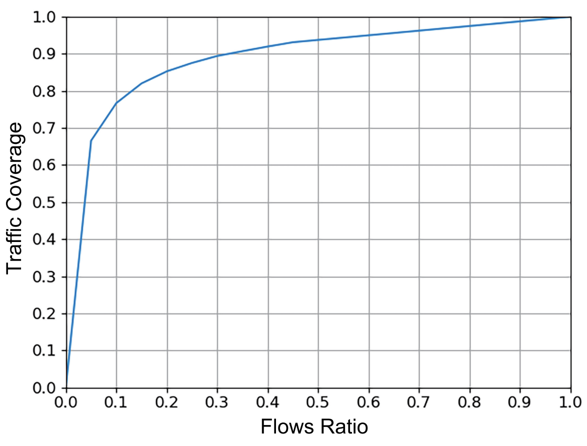


Figure 1. Traffic feature

Although there are lots work to improve router performance, such as increasing the capacity of TCAM to improve the hit ratio [4] or accelerating refresh operation to reduce the overall query delay [5], [6], these solutions based on sufficient TCAM size. While in this paper, we aim at designing an extremely lightweight router.

We analyzed real trace of a backbone network router located in New York City [7], and found potential opportunities to compress TCAM size. We plot the proportion of flows that go through this router in Figure 1, which shows that a small number of flows are contributing to the majority of traffic, in other words, only a small number of entries are active within a certain time period. Inspired by the success of AI-based solutions on caching [8] and predicting [9], we propose an AI-based solution to reduce the refreshing delay, and combine two LSTM to accelerate algorithm convergence. We conduct series of experiments, and the results show that, when the scale of the entry is 20,000, the AIR with 2,000 TCAM entries can achieve the hit ratio of querying of a traditional router with 16,000 TCAM entries, and reduce time of replacement from more than 10,000 to less than 1,000. Those improvements becomes more prominent due to the real number of entries in real routers.

The remainder of this paper is organized as follows. We reviews related work and motivation in Section II. In Section III, we describe the overall structure of AIR. Then in Section IV, we detailed introduce the specific prediction model (an AI algorithm and the optimization mechanism). We then conduct extensive evaluations and show the results in Section V. Finally, we conclude the paper and give the future work in Section VI.

II. Motivation and Related Work

A. Related Work

1) Routing Table Lookup: Traditional routing table lookup methods can be classified into two categories: software-based searching algorithms and hardware-base match-action mechanism.

Earlier, routers mainly used software-based routing algorithms that based on the tree structure. The characteristic of this type of method is that the prefix is represented by a binary Trie, which is a tree-based storage structure. The binary Tire tree indicates that each node can be composed of left and right child nodes at most, and the bit value of the prefix determines the structure of the tree. On this basis, some researchers have done related optimization work. For example, [10] introduces the level-compressed method, using a single node to replace all previous complete subtrees, thereby further reducing the forwarding table space. [11] proposes a hierarchical binary tree algorithm, which layers the tree in depth, and the layered Tires is expanded into a full binary tree. Another method is based on the hash table. [12] organizes the hash table according to the prefix length, and stores the routing prefixes in different linear hash tables with different lengths. A good performance hash function is difficult to find, and some prefixes need to be added when the forwarding table is updated. This requires reselecting the hash function to organize the hash table, which will reduce hash performance and increase update difficulty.

These mentioned above are software search algorithm based on SRAM whose common characteristic is the search speed is slow. Nowadays, the interface speed of the core router of backbone network has reached Gbps or even Tbps, which means the traditional software-based search algorithm has

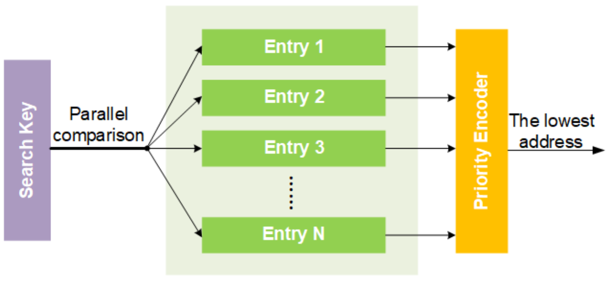


Figure 2. TCAM lookup structure

been unable to meet the search demand of high-speed communication system. In this context, hardware-base match-action mechanism was proposed. Given the GPU's excellent parallel capabilities, many studies have used GPUs to accelerate routing lookups [13]. For FPGA-based routing lookup algorithm needs to solve how to all routing table entry information to stored on the chip and how to construct pipeline stages. [14] proposed to only store a part of the data using hashing and [15] proposed to adjust the tire structure by rotating some branches to balance stage size. TCAM is a three-state content addressing memory that can complete the matching all routing table entries according to the entered index key in one clock cycle, then return the index's address in TCAM. When there are multiple matching entries in TCAM, it will return the entry with the lowest address. The structure of TCAM lookup as shown in Figure 2.

At present, a lot of work is devoted to reducing the average movement times when inserting rules through complex algorithms [5], [16], [17]. However, the above works all have the problem that the reduction times effect is not good enough or the complexity is too high. Moreover, all of them only support item-by-item calculation or insertion, which cannot cope with explosive updates. Another part of works [18] focused on improving the hit ratio of TCAM through complex algorithms, but these methods require group updates of entries, so there are unsatisfactory deficiencies in the update operation part.

2) AI-based Methods: In recent years, with the solution of computing power problems, artificial intelligence has developed rapidly again. More and more researchers focus on AI and have proposed and improved network models, which widely used in various fields, such as analyzing images, summarizing documents, speech recognition, etc. An effective network model has a powerful ability to solve many problems that cannot be solved by traditional methods. In many problem scenarios, if the premise that the next state can be predicted, then new ideas and methods can be provided to solve it. Artificial intelligence can handle complex problems well through deep neural networks which can obtain more hidden attributes or rules, so it is more accurate than traditional methods of prediction. Currently, there are many excellent AI-based predictions in various scenarios.

And some studies use AI to deal with caching strategy issues. Through the reinforcement learning model [19] or graph convolutional neural network model [20], combined with the temporal or spatial characteristics of the data, to predict the popularity of the relevant data. The DeepCache architecture proposed in [21] can accounts for predicted information of objects to make smart caching decisions.

These above related works have made progress in their respective scenarios, but they do not meet our current needs. In our application scenario, the activity level of each entry has the characteristic of the time series, which the LSTM model can well handle. And after extensive literature review, we choose the LSTM as our basic prediction model.

B. Challenges

- **TCAM size.** The theoretical value of the number of different flows passing through a switch is infinite, especially the switch of the core network. The destination IP address determines the forwarding port of the packet in the routing process. In the IPv4 and IPv6 co-existence network, the number of IP addresses up to $2^{32} + 2^{128}$ which is almost innumerable. Currently, all IPv4 addresses have been allocated, and the number of IPv6 addresses allocated is increasing rapidly. For all kinds of routers, the explosive routing scale becomes an enormous challenge.
- **Timeliness.** For the TCAM-based architecture, the huge size will increase the complexity of the replacement process which will pause the query operation. Then the query efficiency will become a performance bottleneck of routers.

III. AIR Overview

To solve the above challenges, here we give an overview of our solution and the details will be introduced in the next section.

AIR looks at the problems existing in traditional switches from a new perspective and aims at reducing TCAM capacity to reduce some problems brought by it. According to our statistics of a large number of real traffic data, among all this traffic flows through the switch in a period of time, a small number of flows can account for the majority of the traffic. Moreover, multiple flows can match the same entry, the number of accesses of different entries is quite different. It can be known that in the current storage policy of TCAM, most of the entries are an extremely low activeness or are not even accessed at all. At present, machine learning has achieved satisfactory results in various fields of prediction. By using machine learning methods to predict the activity of routing entries, and put the top K active entries of the next period into TCAM in advance. Finally, to minimize the capacity of TCAM.

The focus of AIR is not to improve the accuracy of the prediction model, nor to improve the performance of the model by modifying the model. Moreover, we set the activity of entries based on the frequency of match and the period of the statistic is relatively large. Therefore, the error between the predicted and the real results is acceptable. As long as the predicted results are within a certain range, the overall efficiency of the AIR system will not be affected.

The overall structure of AIR is shown in Figure 3. According to the original traffic data provided by the switch or router, the access frequency of all entries of the device can be obtained after preprocessing. The historical frequency sequence of each entry as the input of the prediction model. The activity of all entries in the next period can be obtained through the calculation of the prediction model. According to the threshold set by the actual situation of the network and the size of TCAM, all entries with high activity are placed into the TCAM. The above process is a working cycle of AIR that runs repeatedly. It should be noted that the number of active entries in the actual running process is small, and most of them are "dead". In other words, a large part of all entries that visited frequency is 0 do not require prediction, which reduces the calculation of AIR.

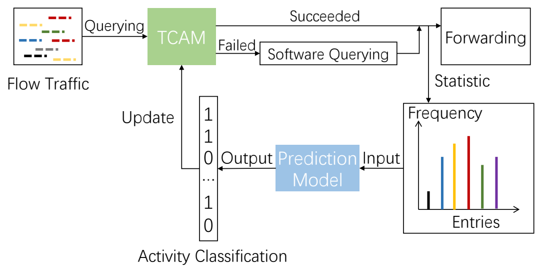


Figure 3. AIR architecture

IV. AI Router

A. Data Preprocessing

In a period which we call a statistical period, different flows may match the same one entry in the switch, so the frequency of all entries of the switch will be significantly different. That is, the entries can be divided into two types according to the activity periodically. At present, switches have reliable traffic statistics functions. According to data provided by the switches, the frequency sequence of several historical periods of each entry is used to predict the next period activity of this entry.

The traffic has self-similar characteristics, so the length of the statistical period is relatively flexible. For different flows, if the activity of the entry fluctuates greatly, the short period is appropriate; otherwise, a long period should be selected. Besides, the statistical period can be measured in terms of time or frequency. In the experiments of this article, the latter is selected based on the characteristics of the selected data set. The number of required historical periods is also relatively flexible, which determines the input layer of the prediction model. If the activity of entries is relatively stable, a small amount of historical data is sufficient, and vice versa. After original data collection and preprocessing, the prediction data of each entry is converted into a vector sized N*1, which is also the basic form of prediction model data set.

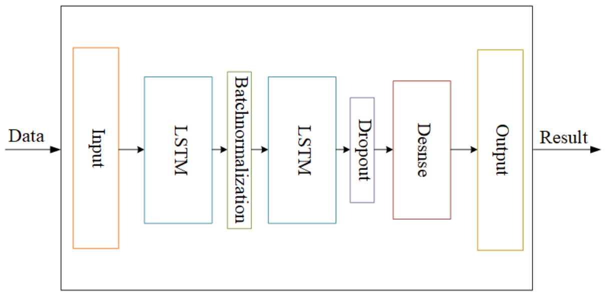


Figure 4. Prediction model structure

B. Algorithm Design

1) Input and output: In the prediction module, the data corresponding to the entry is a one-dimensional vector with time series properties, and each value is the number of times that the entry matched in a statistical period. The sequence is segmented by a sliding window with size N and step 1. The data in the window is the input data of the model, and the next value of the window is the output label corresponding to the input. The input is an N*1 vector and the output is an integer greater than or equal to 0. The data generated through this series of operations serves as the data set for the training model.

2) Model structure and parameter settings: In the prediction module of AIR, LSTM is used to process the prediction about time series characteristics. The overall model structure shown in 4 which consists of the following parts: two LSTM layers, with a Batchnormalization layer added in the middle to avoid the gradient disappearance problem and to speed up the model training, then a Dropout layer added to reduce the occurrence of overfitting, finally a value output through a fully connected layer whose activation function is the $relu$ function. The overall model structure is shown in the figure. Moreover, the $Adam$ optimizer and the mean square error loss function are used for the model.**Add hyperparameters illustration**

C. Optimization

As mentioned above, it is not necessary to predict all entries in a process cycle of AIR. To reduce the complexity and time of the prediction part, we propose a pre-filtering strategy for entries. Filter all entries of the switch before performing the prediction operation. When frequencies of the last few statistical periods of an entry all are 0, then the next period frequency of this entry is set to 0 without performing prediction operation. Add a special identifier to record whether the historical frequency is 0. Suppose that if 8 historical periods need to be observed, the identifier array is f [N], and the total number of N entries, then f [i] = 0x00 makes it clear that the entry need not be predicted.

The specific pseudo-codes are shown as Algorithm 1 and Algorithm 2.

Algorithm 1 Predict with filter

```
Input: identifier array $f$
Output: next frequency array $n\_f$

1: for $i$ in $N$ do
2:     if $f[i]$ == 0x00 then
3:         $n\_f[i] = 0$
4:     else
5:         $n\_f[i]$ = predict($data[i]$)
6:     end if
7: end for
8: return $n\_f$
```

Algorithm 2 Identifier update

```
Input: identifier array $f$ and current frequency array $c\_f$
Output: $f$
1: for $i$ in $N$ do
2:     if $c\_f[i] == 0$ then
3:         continue
4:     else
5:         $f[i] = (f[i] << 1) ||$ 0x01
6:     end if
7: end for
8: return $f$
```

D. Pipeline Prediction

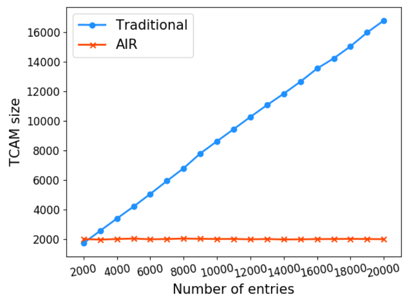
The prediction of each entry is independent, so the calculation can be designed to be pipelined to reduce the delay.***

V. Evaluation

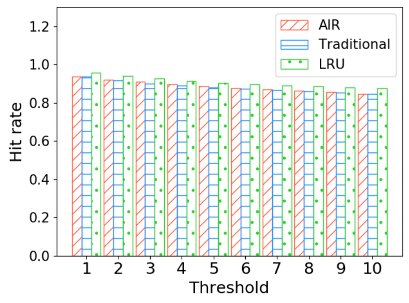
A. Experimental Setup

In the paper experiment, the original traffic data provided by CAIDA [7], which is collected from a core router on a backbone network in New York on January 17, 2019. Due to the related policies of the data provider, we only obtained the original quintuple of the data packets with time series, source IP address, destination IP address, source port, destination port and protocol, without entry data of the switch. To facilitate calculation, we remove the IPv6 address and set the mask length of all destination IP to 24, which means that the corresponding entry is its first 24 bits value. And it is assumed that the set of all aggregated destination IP addresses is the entries set of this switch. This setting to preprocess data of the experiment does not affect the overall execution process and practical feasibility, because the data provided by the switch in AIR is also the number of accesses of each entry.

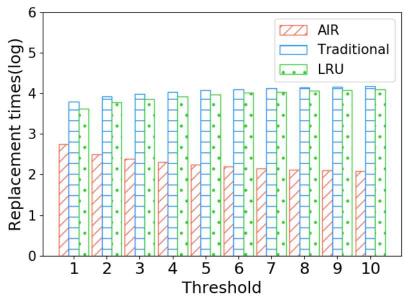
According to statistics, the total number of packets in the data set is 130 million, including 660,000 different destination IP addresses, and the number of entries after aggregation is 8,598. According to the characteristics of used data, the number of packets is defined as the statistical unit. We tried a variety of values and chose ~~100,000~~ as the statistical period after comprehensive consideration. The data set is preprocessed through a program to count the number of accesses of each



(a) AIR performance of TCAM size



(b) Hit rate



(c) Replacement times

Figure 5. AIR comparisons

entry in each statistical period. Finally, the first 80% is used as training data and the rest is used as test data.

The prediction model based on LSTM is implemented by Keras with a batch size of 32 and an epoch of 50. By comparing the performance of different values, we choose 100 as the number of required historical periods.

B. Period Setting

Set different periods and analyze the results.

C. AIR Performance

Different critical value of hot and non-hot entries will correspond to different TCAM size and hit rate. Under the condition that the hit rate is above 90% and the number of replacements in the entire traffic is equal, the comparison of required TCAM size of traditional and AIR in different entries as shown in Figure 5(a). Under the above conditions, the TCAM size of AIR is not affected by the number of entries which is maintained at about 2,000. The traditional will increase proportionally as the number of entries increases. When the number of rules reaches 20,000, the traditional TCAM size is about 17,000.

Due to the problem of data scale, we have only reached 20,000 entries. However, the entries scale is about 600,000 in real routers which size required will be enormous.

D. AIR Analysis

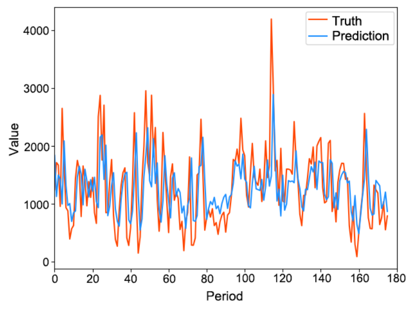
1) Prediction Performance: The prediction performance of the entry in the model meets the needs of AIR. We select the prediction results of three entries randomly, as shown in Figure 6(a) to 6(c). There is a large difference in the activity between the three selected entries. Each group of data has 176 statistic periods, moreover, the number of hit times of an entry in a period ranges from thousands to hundreds, even to dozens. The maximum fluctuation value of these three entries can reach more than 4000, also this value will be greater in other entries. It can be seen that the prediction model can predict entries with different activities and the prediction results have reached our expectations that the approximate trend of the activity of all entries can be accurately obtained. In theory, the higher the accuracy, the better, but we do not calculate the accuracy of the prediction results directly because the errors here do not necessarily affect the overall efficiency of the AIR system. The specific reasons have been explained above.

2) TCAM Hit Rate: According to different thresholds, we analyzed and compared the accuracy of AIR and traditional architecture under the same TCAM size. As the traditional TCAM replacement strategy is that if the TCAM query is missed, then the corresponding entry will be put in the TCAM. Because all netmask set to 24 bits, We do not consider the association between entries, so it can choose the entry to be replaced randomly or according to the LRU (Least Recently Used) rule. And also to ensure the validity of the data, the final results are an average of 176 periods.

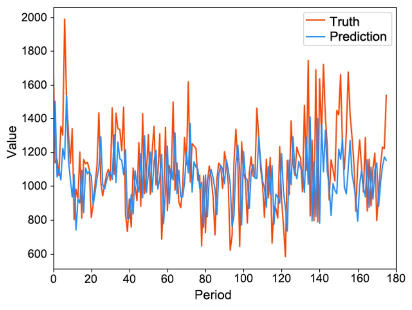
The specific results of setting the threshold from 1 to 10 are shown in the Figure 5(b), Whether the traditional architecture of LRU and random or the AIR, the hit rate results are similar under the same conditions. When the threshold is 1, the accuracy of AIR can reach 93.6%, and when the threshold is 10, the accuracy of AIR can still reach 84.8%. A higher threshold makes the accuracy continue to decline, and too low accuracy will affect the overall performance of the switch, so we do not do higher threshold analysis.

3) Times of Replacement: Each switch developer chooses different calculation strategies which means the delay generated by each insertion of new entry is different, so we choose to compare the times of replacement. Under the same hardware equipment and computing strategy, the replacement times is proportional to the delay time, so the compare result can reflect the performances of AIR and traditional architecture. The result is shown in Figure 5(c). Because the large difference in magnitude, the logarithm operation is carried out for final results for the effect of the legend. The traditional way of replacing is the 10,000 level, while AIR is the 100 level. When the threshold is set to 10, AIR's replacement times performance improves more than 100 times.

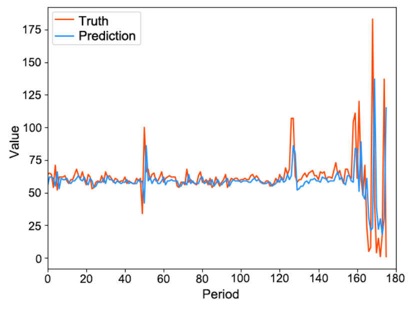
As mentioned earlier, the computational overhead of inserting entries and the movement overhead of related entries account for the main part of the system suspension of TCAM during run-time. AIR reduces the delay time while ensuring a certain accuracy and reducing the TCAM size, which greatly improves the overall efficiency.



(a)



(b)



(c)

Figure 6. Prediction result of entries

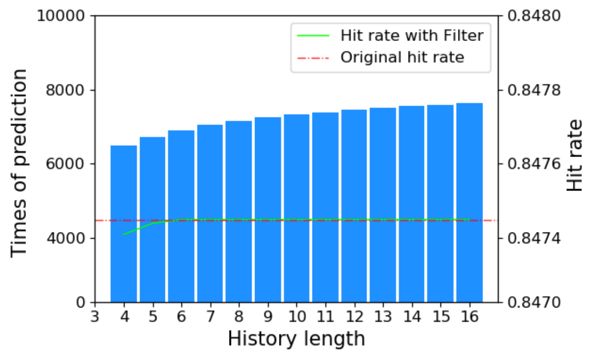


Figure 7. Performance of filter

E. Filter Optimization

Before the prediction, excluding "dead" entries from the prediction list can reduce the calculation of the prediction part in every execution cycle. The criteria for "dead" are different, that is, the length of the historical period is reviewed, which will affect the specific judgment. With the preprocess via filter, the number of the required prediction entries is shown in Figure 7.

Though the filter may mistakenly set 0 to some entries which will decreases hit rate, the longer the historical period of observation, the lower the probability of such errors. Under the condition of the threshold is 10, the influence of the filter on the hit rate is shown in Figure 7. According to the experimental results, the hit rate is consistent with that without the filter when the length of the historical period is longer than 6, whereas the calculation of prediction under this condition is less than 6907. If treat the access time that below threshold as 0, the optimization performance can be further improved.

VI. Conclusion

The explosive growth of routing entries has made route lookup a bottleneck for switches. TCAM's parallel lookup feature solves the problem of the speed of route lookup, but its shortcomings will become worse as the TCAM size increases. In this paper, we first analyze the existing network traffic characteristics and then design the AIR (AI-based Router) system which can make the required TCAM size not increase with the increase of entries, thereby greatly reducing the size of TCAM. We conduct a series of experiments based on real traffic and then confirm that the improved performance of AIR over traditional architecture. Moreover, we also evaluated the accuracy and replacement times of AIR. In the future, we will continue this project and do new work on predict models or centralized entries replacement strategy to further improve AIR performance.

References

[1] "Public route servers," http://www.routeservers.org/, 2019.

[2] "Cisco," https://www.cisco.com/, 2020.

[3] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary cams," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 193–204.

[4] B. Vamanan, G. Voskuilen, and T. Vijaykumar, "Efficuts: optimizing packet classification for memory and throughput," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 207–218, 2011.

[5] X. Wen, B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu, "Ruletris: Minimizing rule update latency for tcam-based sdn switches," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 179–188.

[6] P. He, W. Zhang, H. Guan, K. Salamatian, and G. Xie, "Partial order theory for fast tcam updates," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 217–230, 2017.

[7] "The caida ucsd anonymized internet traces," https://www.caida.org/data/passive/passive_dataset.xml, 2019.

[8] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Deepcache: A deep learning based framework for content caching," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*. ACM, 2018, pp. 48–53.

[9] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "Gstnet: Global spatial-temporal network for traffic flow prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 10–16.

[10] S. Nilsson and G. Karlsson, "Ip-address lookup using lc-tries," *IEEE Journal on selected Areas in Communications*, vol. 17, no. 6, pp. 1083–1092, 1999.

[11] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, *Small forwarding tables for fast routing lookups*. ACM, 1997, vol. 27, no. 4.

[12] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, *Scalable high speed IP routing lookups*. ACM, 1997, vol. 27, no. 4.

[13] Y. Go, M. A. Jamshed, Y. Moon, C. Hwang, and K. Park, "Apunet: Revitalizing {GPU} as packet processing accelerator," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 83–96.

[14] H. Fadishei, M. S. Zamani, and M. Sabaei, “A novel reconfigurable hardware architecture for ip address lookup,” in *2005 Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2005, pp. 81–90.

[15] D. Pao, Z. Lu, and Y. H. Poon, “Ip address lookup using bit-shuffled trie,” *Computer Communications*, vol. 47, pp. 51–64, 2014.

[16] P. He, W. Zhang, H. Guan, K. Salamatian, and G. Xie, “Partial order theory for fast tcam updates,” *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 1, pp. 217–230, 2018.

[17] K. Qiu, J. Yuan, J. Zhao, X. Wang, S. Secci, and X. Fu, “Fast lookup is not enough: Towards efficient and scalable flow entry updates for tcam-based openflow switches,” in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 918–928.

[18] J.-P. Sheu and Y.-C. Chuo, “Wildcard rules caching and cache replacement algorithms in software-defined networking,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 19–29, 2016.

[19] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, “Optimal and scalable caching for 5g using reinforcement learning of space-time popularities,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2017.

[20] Y. Zhang, P. Li, Z. Zhang, B. Bai, G. Zhang, W. Wang, B. Lian, and K. Xu, “Autosight: Distributed edge caching in short video network,” *IEEE Network*, 2020.

[21] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, “Making content caching policies ’smart’ using the deepcache framework,” *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 5, pp. 64–69, 2019.