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

Yuchao Zhang*, Peizhuang Cong*, Bin Liu‡, Wendong Wang*, Ke Xu‡†¶
*School of Computer Science (National Pilot Software Engineering School),
Beijing University of Posts and Telecommunications, Beijing, China
‡ Department of Computer Science and Technology, Tsinghua University, Beijing, China
† BNRist, Tsinghua University, Beijing, China
¶ Peng Cheng Laboratory, Shenzhen, China
{yczhang, congpeizhuang, wdwang}@bupt.edu.cn, {liub, xuke}@tsinghua.edu.cn

***Abstract*—Ternary Content Addressable Memory (TCAM) is an important hardware used to store route entries in routers, which is used to assist routers to make fast decision on forwarding packets. In order to cope with the explosion of route entries due to massive IP terminals brought by 5G and the Internet of Things (IoT), today's commercial TCAM has to keep the corresponding growth in capacity. But large TCAM capacity is causing many problems such as circuit design difficulties, production costs, and high energy consumption, so it is urgent to design a lightweight TCAM with small capacity while still maintains the original query performance.**

Designing such a TCAM faces two fundamental challenges. Firstly, it is essential to accurately predict the incoming flows in order to cache correct entries in limited TCAM capacity, but prediction on aggregated time-sequential data is challenging in the massive IoT scenarios. Secondly, the prediction algorithm needs to be real-time as the lookup process is in line-rate. In order to address the above two challenges, in this paper, we proposed a lightweight AI-based solution, called AIR, where we successfully decoupled the route entries and designed a parallel-LSTM prediction method. The experiment results under real backbone traffic showed that we successfully achieved comparable query performance by using just $1/8$ TCAM size.

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

I. Introduction

Ternary Content Addressable Memory (TCAM) in router is a hardware used to store routing entries in Access Control Lists (ACLs), which is essential for routers to make fast packet forwarding decisions according to the source and destination IPs. But the capacity of TCAM is now under great pressure, which comes from the massive physical terminals accessed to the Internet with the help of 5G and Internet of Things (IoT).

The explosive entry growth is raising high requirements for TCAM both on capacity and efficiency. Firstly, current TCAM-based commercial core routers have to expand TCAM's capacity to keep up with entry growth, while the expansion from 512,000 to 900,000 [1] [1] will naturally result in high production costs and electricity consumption [2]. Secondly, to ensure the line speed performance of packet forwarding, large capacity TCAM faces challenges in circuit design and query efficiency [3]. Therefore, simple expansion of TCAM capacity is not a sustainable solution to meet future network requirements [4], [5], and it is urgent to design a small-sized TCAM while keeping the original packet forwarding performance.

To design such a lightweight small-sized TCAM is not an easy thing which is faced with two fundamental challenges. (1) Accurate prediction on each flow in aggregated time-sequential flows. Making the TCAM with limited capacity can cover all the packet queries, the key is to make accurate prediction on future traffic and leverage cache and replace in time. But the traditional "replace is miss" strategy leads to frequent replacements in small-sized TCAM, which would seriously affect query efficiency and thus is impractical in core routers [6].

(2) Line-rate efficiency of the prediction algorithm. While the network link rate is growing rapidly from 1/10Gbps to 100Gbps [7], routers must be able to achieve the same line-speed in packet query and forwarding. While the existing prediction algorithms on each individual entry can not satisfy such timeliness efficiency [8], [9].

To address the above two challenges, in this paper, we present AIR, a lightweight AI-based TCAM entry replacement scheme, which turns flow prediction problem into ACL entry prediction and decouples the entry aggregation in the customized LSTM algorithm, thus improve the prediction accuracy. The key to make AIR practical is to make the AI-based prediction in line-rate, and AIR achieves this by parallelizing the proposed decoupled-LSTM. Such decoupling and parallelizing attains both accuracy and efficiency.

The work was supported in part by the National Key R&D Program of China under Grant 2019YFB1802603, the National Natural Science Foundation of China (NSFC) Youth Science Foundation under Grant 61802024, Beijing Natural Science Foundation under Grant M21030, the Fundamental Research Funds for the Central Universities under Grant 2020RC36. Peizhuang Cong's work was supported in part by BUPT Excellent Ph.D. Students Foundation under Grant CX2021232 . Bin Liu's work was supported in part by the National Science Foundation of China under Grant 61872213, 62032013 and 61432009.

978-0-7381-3207-5 / 21 /$31.00 © 2021 IEEE

[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.

We have implemented a prototype of AIR and evaluated it by playing back real traffic from a backbone network. The experiment results show that AIR achieves similar forwarding performance with only 1/8 capacity or even less. We also show that AIR can easily handle different WANs with various traffic characteristics.

Our contributions are summarized as follows:

- Characterizing backbone network's workload from the perspective of router traffic to motivate the need of a lightweight TCAM entry replacement scheme. (§III)
- Presenting AIR, an AI-based TCAM entry replacement scheme that achieves the identical forwarding performance by decoupling entry aggregations and parallel execution. (§IV)
- Demonstrating the practical benefits of AIR by a real-world backbone network playback. (§V)

The rest 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. In Section IV, we introduce the specific prediction model (an AI algorithm and its optimization mechanisms) in detail. We then conduct extensive evaluations and show the results in Section V. Finally, we conclude the paper in Section VI.

II. Related Work and Motivation

In this section, we review some related work of TCAM and the correlative AI-based improvement methods, and thus motivate our AIR designing.

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. S. Nilsson et al. introduced the level-compressed method, using a single node to replace all previous complete subtrees, thereby further reducing the forwarding table space [10], [11]. Another method is based on the hash table. M. Waldvogel et al. organized the hash table according to the prefix length and stored the routing prefixes in different linear hash tables with different lengths [12]. Time-varying forwarding table needs to frequently reselect hash functions, which will reduce hash performance and increase update difficulty.

As the link rate of backbone has reached Gbps or even Tbps, the traditional software-based search algorithms have been unable to meet the search demand of high-speed communication system. Given the GPU's excellent parallel capabilities, Y. Go et al. used GPUs to accelerate routing lookups [13]. For FPGA-based routing lookup algorithm needs to solve how to all routing table entry information to store on the chip and how to construct pipeline stages [14]. H. Fadishei et al. proposed to only store a part of the data using hashing and

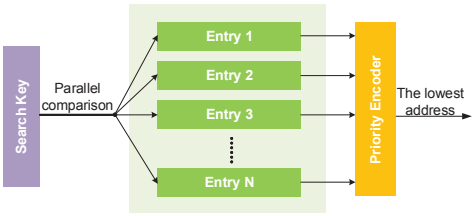


Fig. 1: TCAM lookup structure

D. Pao et al. proposed to adjust the trie structure by rotating some branches to balance stage size [15]. However, the GPU-based method only adds parallel computing based on software search algorithm without making essential improvements. And complex search algorithm will increase the logic complexity of FPGA, which will reduce its clock frequency.

2) ***Ternary Content Addressable Memory****:* TCAM is widely used for rule-based table lookup and packet classification [16], [17]. 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 1.

There is a principle of longest matching in TCAM, which means that if some entries have the same prefix, one entry should be stored at lower addresses than those with shorter netmask. As shown is Figure 2, $102.1.23.24/32$ is stored lower than $102.1.23.0/24$ to ensure the packet with destination IP address $102.1.23.2$ be forwarded to right port eth0, otherwise, the packet would be forwarded to the wrong port eth4. It should be noted that the address of the entry represents its priority. So when an entry is being added/deleted/updated, some entries must be moved to ensure the right priority. As an analysis, inserting a single entry for a 1K entry set requires a maximum of 466 entry-move [6], [18] designed a sophisticated algorithm to reduce to 10 entry-moves on average of per entry insertion. But the algorithm is time consuming which will cause unacceptable delay to compute a moving scheme. Moreover, a more complex problem can be observed. If only $102.1.23.0/24$ stored in TCAM according to the cache rules, then those packets which should matched to $102.1.23.24/32$ will also be forwarded incorrectly. Since the dependency of entries, if the entry e should be inserted to TCAM, some dependent entries of e in the routing table must be also loaded to TCAM simultaneously for the correctness of lookups, even if the caching strategy does not determine those dependent-entries. Especially, the move operations and query operations can only be performed serially, so updating the TCAM will increase the system's delay greatly.

3) ***Entry Replacement Schemes****:* TCAM capacity is hard to meet the needs of explosive growth of routing table entries. To tackle this issue, some researchers tried to compact the

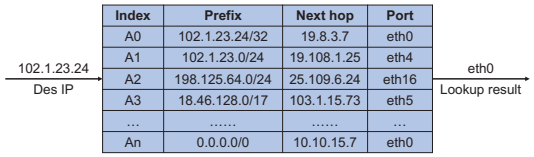


Fig. 2: Dependency of entries

rule tables which did not fundamentally eliminate the broadening gap. Introducing caching mechanism can alleviate this problem, then [19] therefore tried to merge some top flows into ternary rules to improve the TCAM utilization. But it has to traverse all entries in TCAM to decide whether a new entry should be inserted, which fails to tackle the numerous entries scene. After using the caching mechanism, there are considerations for hit rate and content update. [20] designed a complicated algorithm to select entries for better TCAM hit-rate which requires group updates of entries, so there are unsatisfactory deficiencies in the update operation part. Additionally, the dependence between entries makes the insertion and deletion of TCAM more complicated. [21] took the update cost of an entry into consideration when choosing cached entries. And [22] devoted to reducing the average movement times when inserting entries through complex algorithms. However, the above works all have the defect that the reduction time 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. The T-cache [23] crafted dependency-free rules in cache update, and used statistical-based strategy to select cached rules, while AI-based method in this paper will work smarter under burst traffic.

*4) **AI-based Methods**:* In recent years, more and more researchers focusing on AI have proposed good network models, which are widely used in various fields, such as analysing images, summarizing documents, speech recognition, etc. He et al. [24] proposed an meta-learning scheme which can adapt to unseen traffic without the need for retraining the individual predictors. And some studies use AI to deal with caching strategy issues. Zhang et al. [25] used the temporal and spatial characteristics of items to predict the popularity through a graph convolutional neural network model. The DeepCache architecture proposed in [8] can accounts for predicted information of objects to make smart caching decisions. The FreeCache proposed in [9] achieved good results by separating the rules and their rules. These above related works have made progress in their scenarios, but can not work in TCAM due to the flow aggregation of different IPs and the requirement of line-rate processing speed. But in prediction part, it obtains probabilities of indefinite quantity entries by the input entry sequence, which will increase model cells. And the coupled entries lead it cannot be pipelined to reduce the calculation delay. These above related works have made progress in their respective scenarios, but we can improve further with this paper.

B. Motivations

We characterize the real world workload of one core router in New York backbone network [26], and disclose the opportunities of designing a decoupled and paralleled AI-based solution.

- **Traffic skew distribution.** The traffic presents a Zipf-like skewed distribution, i.e, a small number of flows are contributing to the majority of traffic, even 5% flows can contribute more than 90% traffic [26], [27]. This natural distribution property provides an opportunity to use a small capacity TCAM while still maintains high query performance by predicting hot entries.
- **Stable and independent route entry.** Numerous flows and their dependence present difficulty to prediction algorithm. But the number of entries of a forwarding table is relatively stable, and more importantly, the access frequency of entry is independent of each other. It is artful to switch the flow prediction problem into the entry prediction problem, and decouple the relationship between entries, which provides feasibility for prediction in routing lookup scenarios.

These observations together motivate the need for a customized AI-based approach to realize a lightweight TCAM.

III. AIR Overview

Before presenting AIR in detail in Section IV, we first present the overview framework and the principles behind its design.

A. Overall Framework

The overall structure of AIR is shown in Figure 3, and it also outlines the TCAM's workflow.

1) **Entry Decoupling:** The original data a router receives is the aggregated packets from all different flows, so in this module marked with a purple dashed circle in Figure 3), we try to eliminate the dependence between entries by calculating the access frequency for the entry with the same first-24 bits. The historical frequency sequence of each entry, which retains time-dimension characteristics, will be used as the input of the subsequent modules.
2) **Parallel Prediction:** The future access frequency of each entry can be obtained through the calculation of the prediction model, which is the basis for judging whether the entry is active, i.e., whether it should be cached into the TCAM. With the decoupled data from the first phase, the prediction process can be executed in parallel to ensure timeliness of the whole algorithm. This phase is marked with an orange dashed circle in Figure 3.
3) **Entry Classification:** After predicting the future access frequency of entries, these entries can be classified into two categories by setting a threshold, i.e., the entry with an access frequency greater than the threshold is

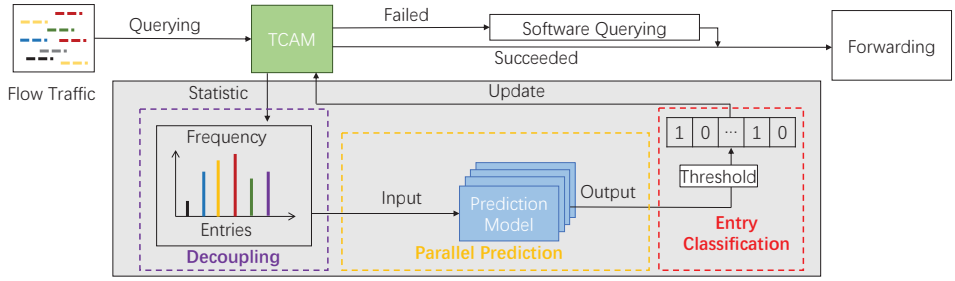


Fig. 3: AIR architecture

classified as hotspot entry, otherwise as non-hot one . Then, all hotspot entries would be updated into TCAM. This phase is marked with a red dashed circle in Figure 3.

The skewed distribution of traffic makes it valuable to optimize TCAM using prediction. AIR first converts the flow prediction problem into the entry prediction problem, which improves the accuracy of prediction. Due to the requirements of linear speed processing in network scenarios, AIR cleverly decouples entries, which not only parallelizes the prediction process so that the prediction results can meet the need of network scenarios but also makes prediction granularity and interval more flexible. The threshold can be changed with network status, which can better adapt to network dynamics.

B. Design Principles

Such design of AIR is driven by several principles:

- *Proper prediction granularity.* When the prediction is done at the flow level, such massive number of flows would lead to an extremely large computing space. From a new perspective, AIR focuses on corresponding entries after flow aggregation. The number of entries is relatively stable and much smaller than that of flows, significantly reducing the computing overhead. Besides, prediction on entries would also be more accurate.
- *Data decoupling.* The traffic dynamism makes it very difficult to select the model input. If the sequence is selected at a fixed frequency, the number of selected flows would be ever-changing, making the prediction algorithm invalid. If the sequence is selected at a fixed length, it has to make real-time adjustments according to traffic state to retain characteristics. Moreover, it needs tens of thousands or even longer sequences length to reflect data characteristics. The dynamic and ultra-long sequence data is quite challenging for prediction models. In AIR, we calculate access frequency of each entry independently, thus decouples the sequence dependence. Along this way, the subsequent prediction module becomes able to predict entry's future access based on this historical data.
- *Dynamic prediction interval.* When the network is relatively stable, some prediction calculations may not be necessary because hot entries are already cached in TCAM. However, when the network fluctuates, it is necessary to increase the prediction frequency to refresh TCAM timely. This problem is solved in AIR by using a dynamic prediction interval that can be set flexibly and adjusted according to traffic situations.

IV. AI Router

In this section, we introduce in detail the AI algorithm integrated in AIR.

A. Entry Decoupling

As described in above sections, we should make accurate prediction on each flow in the aggregated time-sequential traffic, which can be formulated as follows.

$$(P_{e_i,n}^{(S_{IP},D_{IP})}, P_{e_j,n+1}^{(S_{IP},D_{IP})}, P_{e_k,n+2}^{(S_{IP},D_{IP})}, ...) \tag{1}$$

where $P_{e_i,n}^{(S_{IP},D_{IP})}$ denotes that the coming flow at time t is packet e_i with source IP S_{IP} and destination IP D_{IP}. Such traffic is the aggregated sequential data from all the flows with different sources and destinations, making the prediction extremely difficult.

In AIR, we try to solve this problem from a totally different perspective by decoupling each entry from the aggregated traffic. We set a periodic interval Δ_t, and turn the origin flow sequence data into entry frequency, and for each entry e^i, the frequency prediction problem can be denoted as follows.

$$x_{t+(n+1)*\Delta_t}^{e_i} = f(x_t^{e_i}, x_{t+\Delta_t}^{e_i}, ..., x_{t+n*\Delta_t}^{e_i}) \tag{2}$$

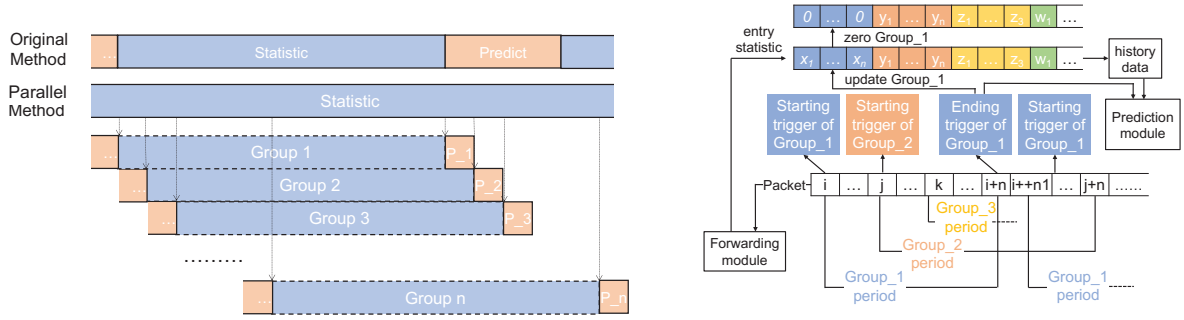
where $x_{t+\Delta_t}^{e_i}$ means that at time interval t to $t+\Delta_t$, entry e^i appears x times.

What should be noted is that the interval Δ_t is strategically set to be dynamically adjusted. It would be lower for active flows than non-active ones. Also, when predicting the frequency value in the next interval, the needed number of required historical intervals is also different, which would be less for stable flows than unstable ones.

B. Parallelization

Although the sequential data has been decoupled into frequency data, the large amount of particular entries also makes the prediction algorithm impractical in efficiency. Therefore we design a parallel prediction scheme.

AIR divides all entries of the router into disjoint groups DG and set different starting time for each group. When the



(a) Parallelization sketch

(b) AIR parallelization process structure

Fig. 4: AIR Parallelization

number of arrived packets from this group reaches a threshold TS, it will trigger the prediction module to predict the entry frequency in the next interval Δ_t, and then update the TCAM caching according to the prediction results. As DGs differ among each other, so the time when the arrived packets reach TS is also different, thus making parallel processing possible. The sketch map is shown in Figure 4(a).

The detail architecture of parallelization is shown in the Figure 4(b). The access frequency data of the entries obtained from the forwarding module is counted. Due to the independent of each entry, the counting stage does not need to consider the limitation of grouping and directly performs a '+1' operation on the value of the corresponding access frequency. A series of operations will be triggered at the end of each statistical period of each group. The latest statistical results of this group from the statistical array combined with corresponding historical access frequency data from historical data array will be input to the prediction module, which will process predict calculation according to this data. And the latest data will also be stored in the historical data array, then the corresponding value of this group in the statistical array will be cleared to start the next period statistics. And cycle back and forth.

C. Algorithm Design

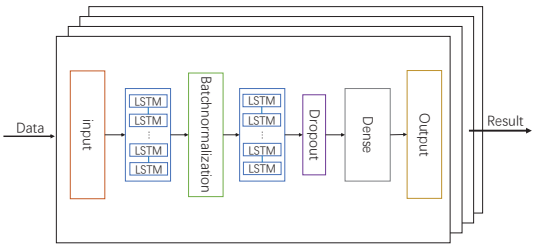


Fig. 5: Prediction model structure

In AIR, we design a customized LSTM-based model to process the prediction with time series characteristics. The overall structure is shown in Figure 5 which consists of the following parts: two $Layer_{LSTM(units=128)}$s, with a $Layer_{Batchnormalization}$ added in the middle to avoid the gradient disappearance problem and to speed up the model training, then a $Layer_{Dropout}$ is added to reduce the occurrence of overfitting, finally the predicted value is generated after a $Layer_{full\_connection}$ whose activation function is the $relu$ function. Moreover, the $Adam$ optimizer and the mean square error loss function are also used in AIR.

The pseudo code of AIR is shown in Algorithm 1, and the function is described in detail. The processes are as follows: 1) Flow statistics trigger predictions in groups. 2) Obtain decoupled entries data according to the flow data. 3) Predict entries' frequency based on historical data. 4) Obtain hotspot entries according to the predicted results and the threshold; 5) Update hotspot entries into TCAM.

*1) **Input and output**:* In the prediction module, the input data is a one-dimensional vector with a specific entry, a timestamp and the access times of this entry was matched in the last interval Δ_t, e.g., $\vec{N} = (n_1, n_2, ..., n_n)^T$. This input sequence is segmented by a sliding window with size n and step 1. So the final input $\vec{N}_{input}$ of the prediction module for each entry is the last N number of $\vec{N}$, and the output of the prediction module V_{output} is an integer indicating the predicted access frequency of this entry in the next interval. Such prediction results of all entry groups serve as the reference data for update decision making of TCAM.

*2) **Model structure and parameter settings**:* In the prediction module of AIR, LSTM-based model is used to process the prediction about time series characteristics. The overall model structure shown in Figure 5 which consists of the following parts: two $Layer_{LSTM(units=128)}$s, with a $Layer_{Batchnormalization}$ added in the middle to avoid the gradient disappearance problem and to speed up the model training, then a $Layer_{Dropout}$ added to reduce the occurrence of overfitting, finally a value output through a $Layer_{full\_connection}$ whose activation function is the $relu$ function. Moreover, the $Adam$ optimizer and the mean square error loss function are used for the model.

Algorithm 1 AIR algorithm

entry_statistic(): return of group i which should be predicted or $None$
decouple_entry(i): return decoupled entry data of group i
get_hisdata(e): get entry e historical data array
LSTM_predict($sequence\_data$)**:** predict frequency according to the parameter
entry_cla($threshold, group\_i\_data$)**:** return hotspot entries of group i based on threshold

while True **do**
  $group\_i$ = entry_statistic()
  if $group\_i$!= $None$ **then**
    New_Thread: Pre_Fuc($group\_i$)
  end if
end while

Pre_Fuc($group\_i$):
$entry\_data$ = decouple_entry($group\_i$)
for e in $group\_i$ **do**
  $\vec{N}$ = get_hisdata(e)
  V_{output} = LSTM_predict($\vec{N}$)
  fre_data.append(V_{output})
end for
$hot\_entries$ = entry_cla($threshold, fre\_data$)
update_TCAM($hot\_entries$)

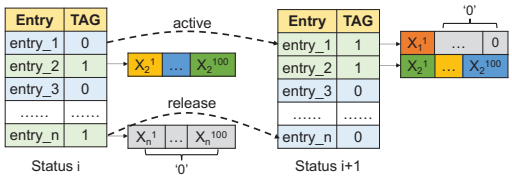


Fig. 6: Storage structure

D. Optimization

Current commercial switches or routers are equipped with statistics and analysis functions, which are all linear processes. However, a large number of predictions will consume computing resources and cause delays. An inefficient strawman method is to predict all entries of a group after each statistical period. To alleviate this problem, we optimize the prediction part from the spatial aspect on basis of parallelization. According to observations, it can be confirmed with a high probability that the activity of some entries will be very low and will not even be accessed in the future. The computing resources used in these entries are completely wasted. Therefore, we attach constraints to filter out entries that are not worthy of prediction to reduce the computational burden. Besides, the prediction part needs to save entries' historical access data, which will occupy a large amount of memory. We designed an efficient data storage structure and strategy to optimize AIR in terms of data storage. The specific design schemes will be presented in detail below.

*1) **Entry filter**:* As mentioned above, it is unnecessary to predict all entries of a DG during a processing cycle. To reduce the complexity and time of the prediction part, we propose a pre-filtering strategy for entries. Filter entries of the DG which need be predicted before performing the prediction operation. When frequencies of the last few statistical periods of an entry are all 0, then the next period frequency of this entry is set to 0 without performing prediction operation as equation (3).

$$x^{e_i}_{t+(n+1)*\Delta_t} = \begin{cases} f(x^{e_i}_t, x^{e_i}_{t+\Delta_t}, \ldots, x^{e_i}_{t+n*\Delta_t}), \sum_{j=0}^{K} x^{e_i}_{t+j*\Delta_t} \neq 0 \\ 0, \sum_{j=0}^{K} x^{e_i}_{t+j*\Delta_t} = 0 \end{cases} \tag{3}$$

K represents the number of historical periods that need to be observed. Add a special identifier to record whether the historical frequency is 0. Suppose $K = 8$, the identifier array of DG_i is $f[N_i]$, and N_i is the total number entries of DG_i, then $f[i] = 0x00$ makes it clear that the entry i need not be predicted. The specific pseudo-codes are shown as Algorithm 2 and Algorithm 3.

Algorithm 2 Predict with filter

Input: identifier array f
Output: next frequency array n_f

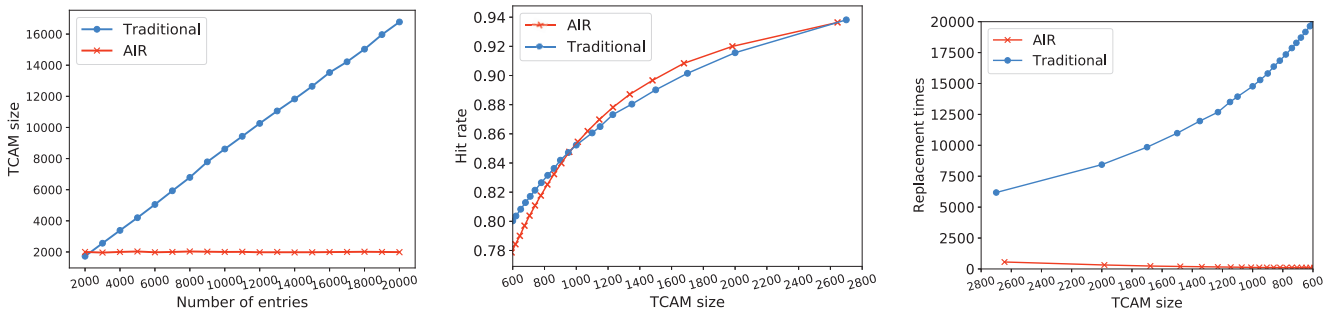
for i in N **do**
  if $f[i] ==$ 0x00 **then**
    $n\_f[i] = 0$
  else
    $n\_f[i] =$ predict($data[i]$)
  end if
end for
return n_f

Algorithm 3 Identifier update

Input: identifier array f and current frequency array c_f
Output: f

for i in N **do**
  if $c\_f[i] == 0$ **then**
    continue
  else
    $f[i] = (f[i] << 1) ||$ 0x01
  end if
end for
return f

*2) **Storage**:* Given that the number of entries in a real router far more than $600K$, if storing the access frequency data of several periods of each entry, assuming use the unsigned short int to record 100 periods data, it will take $120MB$ ($32bit * 600K * 100$) on-chip storage space which is inefficient and unacceptable.



(a) AIR performance of TCAM size (b) Hit rate (c) Replacements

Fig. 7: AIR comparisons

We observed the historical frequency of each entry and found that when the *dead* becomes *active* again, its historical access frequency is almost all 0. To reduce the storage space requirements of AIR, we optimized the storage rules by adding a 1-bit identifier for each entry to indicate its status(*dead* or *active*). For the *dead*s, there is no need to store a large amount of data and all historical access frequencies are 0 by default. The overall storage structure is shown in the Figure 6. Due to the filter, this part of the entries will not affect the prediction at all, and the remaining entries are stored according to the original strategy. When the statistical register data shows that an entry's status changes from *dead* to *active*, its identifier will be changed and its subsequent access frequency will also be stored. When the historical access frequencies of entry over periods are 0, then its identifier will be changed and its storage space will be freed. The huge number of *dead* entries makes this strategy save a significant amount of storage space.

V. Evaluation

In this section, we clarified the experimental setup and analyzed the performance of AIR with some existing strategies.

A. Experimental Setup

The data we used in the evaluation was collected from a core router on a backbone network in New York on January 17, 2019 [26], with each piece of data consists of a five tuple: ⟨ time series, source IP, destination IP, source port, destination port, protocol⟩. The total number of packets in the data set is $130,000,000$, including $660,000$ different destination IP addresses, and $8,598$ entries. For larger number of entries in real backbone network, AIR is still applicable.

To facilitate entry decoupling, we set the mask length of all destination IPs to 24, which means that the corresponding entry equals the first 24 bits. It is assumed that the set of all aggregated destination IP addresses is the whole entries set. Such processing ensures the overall execution and feasibility.

The prediction model based on LSTM is implemented by Keras with a batch size of 32 and an epoch of 50 is implemented on the Ubuntu 16.04-LTS operating system. In order to ensure the generality of the results, we also evaluated on other data sets.

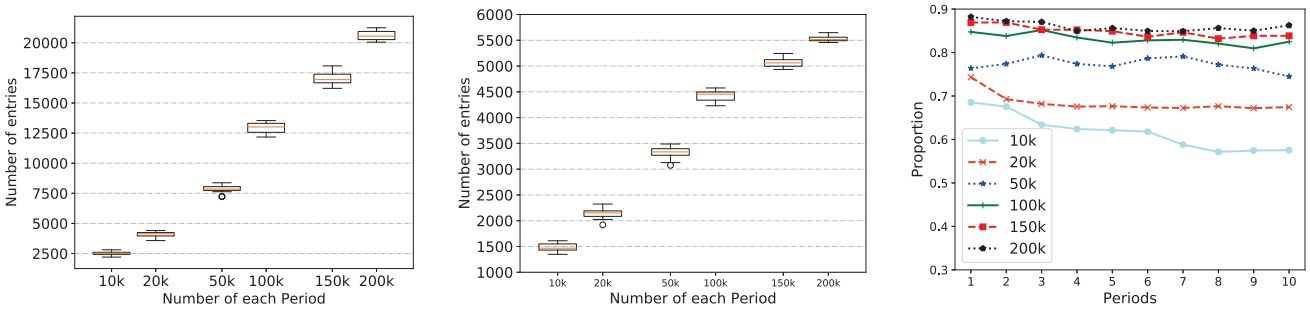
B. AIR Performance

We first evaluate the required TCAM size when there are different number of entries (from 2000 to 20000 entries) in the traffic. To make it fair, we conduct different groups of experiments when the hit rates of the two algorithms are the same. We then show the experiment results in Figure 7(a), from which we can see that under traditional "replace if miss" strategy, the required TCAM size increases linearly (from 2000 to over 16000) with entry number, while in AIR, the required TCAM size is always around 2000, saving more than **8 times** of TCAM capacity. In the real network where the number of entries is over 20000, AIR works even better.

From another perspective, different sizes of TCAM will affect the hit rate and the number of table replacements. The AIR and traditional schemes are simulated separately by setting different TCAM sizes. The statistical period is set to 100K, and AIR selects the entries equal to TCAM size from high to low according to the predicted frequency. Moreover, the final result of the two strategies is to average the results of 170 periods. With the increase of TCAM size, the accuracy of AIR and traditional strategies are similar and showing an upward trend. However, for any TCAM size, the entry replacement times of AIR has a great advantage over the traditional ones. The hit rate and replacement times of AIR and traditional schemes are shown in Figure 7(b) and Figure 7(c).

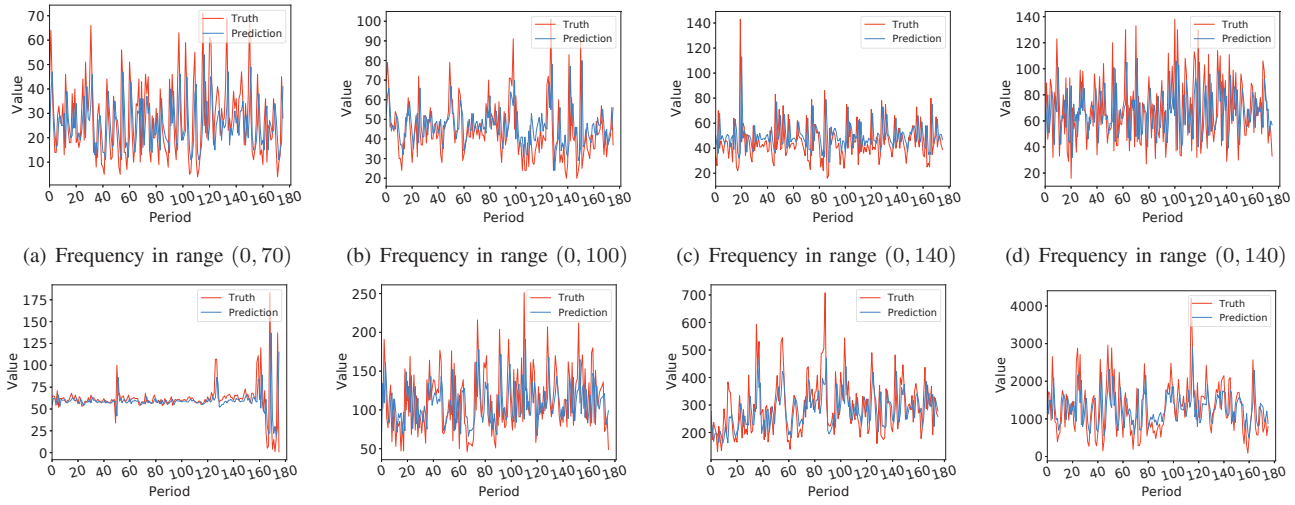
C. AIR Analysis

1) ***Period Analysis****:* We set different lengths statistic periods for the same traffic data set, $10K$, $20K$, $50K$, $100K$, $150K$, and $200K$. Obviously, for different length periods, the number of IP or corresponding entries appeared in each period is different. We counted 200 periods of different lengths respectively and the distribution of IP/entry numbers in each case is shown in the box Figure 8(a) and Figure 8(b). As the length grows, the number of IP/entry in each period increases significantly, which makes us have to face the problem of a large number of entries should be predicted in each prediction especially we combine the filter mentioned earlier. In order to control time and resource consumption of prediction process, the period should not be set too long. From another point



(a) Unique IP (b) Unique Entry (c) Decay of hot entries

Fig. 8: Analysis of period



(a) Frequency in range $(0, 70)$ (b) Frequency in range $(0, 100)$ (c) Frequency in range $(0, 140)$ (d) Frequency in range $(0, 140)$

(e) Frequency in range $(0, 175)$ (f) Frequency in range $(0, 250)$ (g) Frequency in range $(0, 700)$ (h) Frequency in range $(0, 4K)$

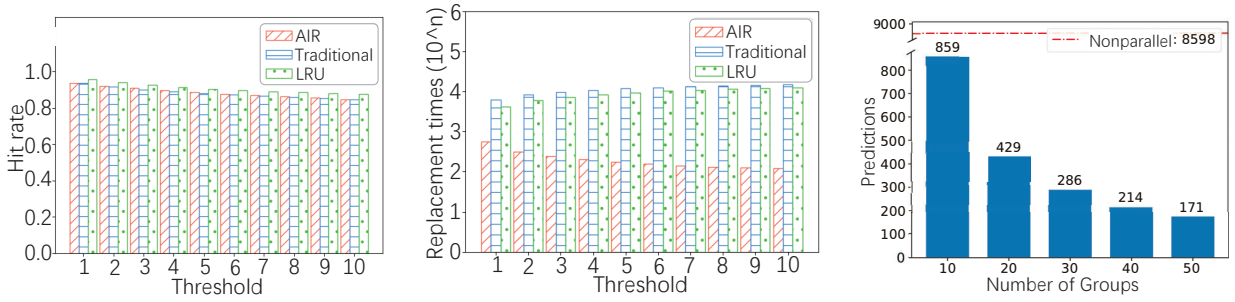
Fig. 9: Prediction result of entries

of view, we analysed the regularity of entries and traffic in the period and obtained the hot entries' traffic coverage in subsequent periods. For different period lengths, the traffic coverage of the top 20% hot entries in the next 10 periods is shown in Figure 8(c). The longer the period is, the more stable the hot entries. And vice versa, the shorter the period is, the more obvious the traffic coverage of the hot entries decreases with time. Therefore, setting a short period not only increases the frequency of prediction but also reduces the reliability of prediction based on the frequency of historical items. Therefore, setting a short period not only increases the frequency of prediction, but also reduces the reliability of historical frequency-based prediction. In combination with the above analysis, the appropriate period length should be determined according to traffic and equipment characteristics, and we choose $100K$ in the experiments of this paper.

*2) **Prediction Accuracy Benefited from Decoupling**:* As described in Section III, we solve the prediction challenge under aggregated flows circumstance by decoupling entries and making prediction on disjoint entry groups. So here we evaluate the prediction performance of AIR. We randomly select four groups with different access times during 176 periodic intervals. We show the prediction results together with the truth in Figure 9(a) - 9(h), which indicate that the prediction algorithm after data decoupling works well on entries with various frequencies.

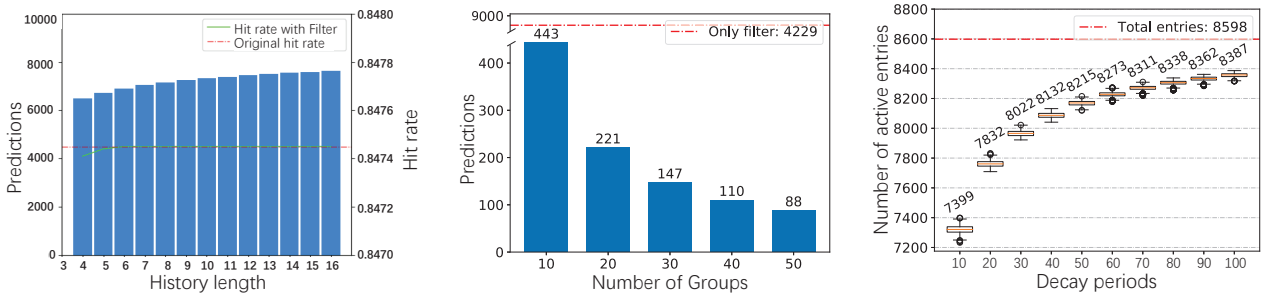
*3) **Threshold Analysis**:* As describe in Section IV, there is a threshold TS on the number of arrived packets during an interval, which would directly affect TCAM hit rate and the corresponding overhead. So here we evaluate AIR's performance under various thresholds and compare the results with another baseline solution – Least Recently Used (LRU), besides the traditional one.

- **Hit Rate:** We set the threshold TS to 1$\sim$10, and conduct a series of experiments to measure hit rate. In traditional method, when an entry query is missing in TCAM, it will be added to TCAM by randomly replacing another one. In LRU, the least recently used one would be ejected, while in AIR, we choose the one with least future frequency instead.
  To ensure the validity of prediction results, we take the average of 176 experiments as the final result and show



(a) Hit rate (b) Replacement times (c) Performance of parallelization

Fig. 10: AIR analysis



(a) Performance of filter (b) Performance of filter&parallelization (c) Analysis of 'active' entries

Fig. 11: Analysis of optimizations

the comparisons in Figure 10(a). From these results we can see that the average hit rate of the above three mechanisms are similar to each other, even with different thresholds. For example, when TS is set to 1, the hit rates are 93.6%, 93.5%, 94.0% for AIR, traditional method and LRU, respectively.

- **Overhead:** Although lower threshold can lead to higher hit rate, it also results in higher overhead on calculation and entry replacement latency. Using the same hardware of TCAM, algorithm's overhead is proportional to entry replacement times, so we count entry replacement times under different threshold (from 1 to 10, the same as in Figure 10(a)) and show the results in Figure 10(b), where the results are in logarithm operation. From these results, we can see that the replacement times of both traditional method and LRU are around 10^4 times, while that of AIR is around 10^2 times, which is **100 times less**.

*4) **Parallelization**:* As described in Section III, we parallelize the prediction process to further improve the prediction efficiency and ensure the timeliness of the results. So here we divide all entries into 10 to 50 disjoint groups DGs. In our experimental data set, the number of predictions required each time is shown in the Figure 10(c). As DGs increases, the number of prediction entries at a time decreases. When DGs is 50, only 171 entries need to be predicted, which is much smaller than the 8,598 in nonparallel mode. The optimal group configuration set based on traffic characteristic and computing power can make statistics and predictions completely parallel.

*5) **Optimizations**:*

- **The Filter:** 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 judgement. With the preprocess via filter, the number of the required prediction entries is shown in the histogram of Figure 11(a). Note that the value of the number of predictions without the filter does not present in the figure, because it depends on the number of entries of routing table that has far exceeded $600K$ in current commercial routers. 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 11(a). According to the experimental results, the hit rate is consistent with that without the filter when the length of historical period is longer than 6, whereas the calculation of prediction under this condition is less than 6,907. If treat the access time that below threshold as 0, the optimization performance can be further improved.
- **Filter&Parallelization:** The filter and parallelization optimize the prediction part from the perspectives of space and time, which can further reduce the number of predictions in each time. In this experiment, we set the filter's threshold to minimum 1 which means all entries with the

0 frequency in the last one period are regarded as $dead$ and combined with the parallelization scheme. In the case of different groupings, the number of predictions in each time is shown in Figure 11(b).

- **Storage:** Since the entries of our dataset are not the complete set of a router, we theoretically calculate the space that can be saved. The traffic lasts about 5 minutes for a total of 1300 periods ($100K$ each period), during which 8598 entries were accessed. The storage space can be released when the access frequencies of an entry in the last x periods are all 0, we call the x as the decay period. With different decay period configurations, the number of $active$ entries in 1300 periods is shown as the Figure 11(c). Even when the decay period is 100 which is already the maximum, the number of $active$ is less than the total entries'. It can then be concluded that there is an upper limit on the number of $active$ entries for this traffic, namely 8,387. In order to facilitate the calculation, we expanded the active number to $10K$, and still kept the total number of entries at 600K. The on-chip storage space required according to the above rule is $2.06MB$ ($600K * 1bit + 10K * 100 * 32bit$), which is much less than the original scheme's $120MB$.

In summary, AIR achieves the similar hit rate to the current commercial schemes, but with only $1/8$ TCAM capacity and less replacement times in two orders of magnitude.

VI. Conclusion

With the rapid development of access network, there are more and more terminals accessing to the Internet, so routers have to continue expanding TCAM's capacity to cope with the explosive growth in the number of entries. In order to solve the problems caused by large capacity TCAM, we designed an AI-based TCAM entry replacement scheme call AIR, where we decoupled the aggregated flows to address the prediction challenge. To make AIR practical, we then designed a parallel LSTM algorithm which largely reduces the calculation overhead. Through a series of experiments, AIR has been shown to achieve similar performance to existing strategies while using only $1/8$ TCAM capacity. The performance improvements will be more obvious in large-scale networks.

References

[1] "Bgp routing table analysis reports," https://bgp.potaroo.net/, 2020.

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

[3] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys (CSUR)*, vol. 37, no. 3, pp. 238–275, 2005.

[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] 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.

[6] 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.

[7] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang, "Aeolus: A building block for proactive transport in datacenters," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 422–434.

[8] 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.

[9] R. Li, B. Zhao, R. Chen, and J. Zhao, "Taming the wildcards: Towards dependency-free rule caching with freecache," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[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. Byun, Q. Li, and H. Lim, "Vectored-bloom filter implemented on fpga for ip address lookup," in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2019, pp. 1–4.

[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] C. Zhang, Y. Feng, H. Song, Y. Wan, W. Xu, Y. Wang, H. Dai, Y. Li, and B. Lin, "Obma: Minimizing bitmap data structure with fast and uninterrupted update processing," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–6.

[17] V. Demianiuk, S. Nikolenko, P. Chuprikov, and K. Kogan, "New alternatives to optimize policy classifiers," *IEEE/ACM Transactions on Networking*, 2020.

[18] 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.

[19] Q. Dong, S. Banerjee, J. Wang, and D. Agrawal, "Wire speed packet classification without tcams: a few more registers (and a bit of logic) are enough," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1, pp. 253–264, 2007.

[20] 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.

[21] Z. Ding, X. Fan, J. Yu, and J. Bi, "Update cost-aware cache replacement for wildcard rules in software-defined networking," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 457–00 463.

[22] 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.

[23] Y. Wan, H. Song, Y. Xu, Y. Wang, T. Pan, C. Zhang, and B. Liu, "T-cache: Dependency-free ternary rule cache for policy-based forwarding," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 536–545.

[24] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, and P. Tengkvist, "A meta-learning scheme for adaptive short-term network traffic prediction," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2020.

[25] 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.

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

[27] N. Sarrar, S. Uhlig, A. Feldmann, R. Sherwood, and X. Huang, "Leveraging zipf's law for traffic offloading," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 1, pp. 16–22, 2012.