A&B: AI and Block-Based TCAM Entries Replacement Scheme for Routers

Peizhuang Cong, Yuchao Zhang, *Member, IEEE*, Bin Liu, *Senior Member, IEEE*, Wendong Wang, *Member, IEEE*, Zehui Xiong, *Member, IEEE*, and Ke Xu, *Senior Member, IEEE*

***Abstract*— With the ever-increasing deployment of 5G and IoT, the number of end-hosts/terminals is increasing rapidly, so that routers have to cache more and more forwarding entries to guarantee communication reachability of these terminals, which makes Ternary Content Addressable Memory (TCAM)-based routers keep expanding resource requirements. However, the design and implementation of large-capacity TCAM-based routers are faced with such challenges: difficult circuit design, high production cost and energy consumption, thereby posing an urgent requirement on a lightweight TCAM that can still maintain those massive communication connections. In this paper, we aim to design a lightweight router with small storage requirement while still retaining the original communication connection performance, which is not straightforward due to the following two challenges: First, under the condition of massive sequential flow data, it's difficult to accurately and timely select the entries to cache for a small capacity TCAM. Second, given the strict prefix matching principle, how to efficiently insert the selected entries into TCAM is also challenging. To address these problems, we propose A&B: an AI-based Routing entry prediction strategy (AIR) and a Block-based entry Insertion Tactic (BIT). AIR can precisely select entries by conducting accurate entry predictions, which converts dynamic flow-based prediction into stable and parallelizable entry-based prediction by decoupling spatio-temporal characteristics. BIT optimizes entry insertion by isolating TCAM into several blocks, thus eliminating the time-consuming entry movements. The experiment results based on real backbone traffic show that our lightweight A&B achieves comparable performance compared to the traditional schemes by using only $1/8$ TCAM storage.**

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

I. Introduction

TERNARY Content Addressable Memory (TCAM) is an essential unit of routers for storing and querying routing entries, which assists in making fast forwarding decision based on IP address of packet header. However, the capacity of TCAM becomes one of the bottlenecks to accommodate the rapid growth in physical terminals that access to the Internet accelerated by 5G and the Internet of Things (IoT).

The growth of the number of entries is raising high requirements for TCAM both on capacity and efficiency [1]. The current TCAM-based commercial core routers have to scale TCAM capacity to keep up with the growth of entries, while the expansion from 512,000 to 900,000[1] will naturally result in high production costs and electricity consumption [2]. Moreover, the large-capacity TCAM also involves circuit design constraints [3]. Therefore, crudely expanding the capacity of TCAM in strawman way is not a sustainable solution to satisfy future network demands [4], [5], and it is urgent to design a kind of small-capacity TCAM-based routers while maintaining the original packet forwarding performance.

To design such an efficient lightweight small-capacity TCAM is not straightforward which is faces with two fundamental challenges: *Entry selection* and *Entry insertion*.

- *Entry selection*: In order to have a higher query hit rate, which entries should be stored in TCAM with limited capacity? The traditional *"insert if missed"* update strategy results in highly frequent replacement under small-capacity TCAM, which would severely decrease query efficiency and is therefore impractical in core routers [6]. Leveraging prediction to make replacement

Manuscript received 15 December 2021; revised 11 March 2022; accepted 23 April 2022. Date of publication 18 July 2022; date of current version 19 August 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1802603, in part by the National Natural Science Foundation of China (NSFC) under Grant 62172054 and Grant 62072047, and in part by the Key Project of Beijing Natural Science Foundation under Grant M21030. The work of Peizhuang Cong was supported in part by the Beijing University of Posts and Telecommunications (BUPT) Excellent Ph.D. Students Foundation under Grant CX2021232. The work of Bin Liu was supported in part by the National Science Foundation of China under Grant 62032013, Grant 61872213, and Grant 61432009. The work of Zehui Xiong was supported in part by Research Grant under Grant SUTD SRG-ISTD-2021-165, in part by the SUTD-ZJU IDEA under Grant SUTD-ZJU (VP) 202102, and in part by the SUTD-ZJU IDEA Seed under Grant SUTD-ZJU(SD) 202101. *(Corresponding authors: Yuchao Zhang; Wendong Wang.)*

Peizhuang Cong and Wendong Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing 100876, China, and also with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: congpeizhuang@bupt.edu.cn; wdwang@bupt.edu.cn).

Yuchao Zhang is with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: yczhang@bupt.edu.cn).

Bin Liu and Ke Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China (e-mail: liub@mail.tsinghua.edu.cn; xuke@tsinghua.edu.cn).

Zehui Xiong is with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372 (e-mail: zehui_xiong@sutd.edu.sg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3191351.

Digital Object Identifier 10.1109/JSAC.2022.3191351

[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.

decisions is a potential approach, which would face two challenges of accuracy and efficiency. (1) The prediction accuracy determines the coverage ability of packet queries of TCAM with limited capacity, and then how to accurately predict each future flow based on complicated and uncertain aggregated time-sequential flows data is a challenge. (2) As the network link rate grows rapidly from 100Mbps to 100Gbps [7], routers must be able to achieve the same line-speed in packet lookup and forwarding. While the existing prediction algorithms on individual entries cannot satisfy such timeliness requirement [8], [9]. Then, how to linearize the prediction algorithm is another challenge.

- *Entry insertion*: How to insert those selected entries into TCAM efficiently? The longest prefix matching principle generates dependencies between entries with the same prefix, which makes entries have to keep the relative pre-post relationship between the dependent entry set when stored in TCAM. Therefore, the update of TCAM is a complex process. When inserting one entry, its related dependent entries also need to be inserted jointly, which may require the entries stored in TCAM to be moved several times to free up a proper space for the corresponding entries [6], [10]. It is a challenge to optimize the process of entry insertion of TCAM and make it more efficient.

To address forementioned challenges, in this paper, we present an efficient lightweight TCAM-based router framework, A&B, which consists of two components: an Artificial Intelligence (AI)-based Routing entry prediction strategy (AIR) and a Block-based routing entry Insertion Tactic (BIT).

- AIR decouples the future flows prediction based on complicated and uncertain aggregated time-sequential flows data to a denumerable individual routing entry prediction issue, and on that basis, parallelizes the LSTM-based prediction calculation. Such decoupling and parallelizing enhance both accuracy and efficiency of the prediction model.
- BIT, based on the traffic skewed distribution and probability segmentation, extremely loosens the strict restrictions on the relative pre-post relationship of the dependent entry set stored in TCAM.

We have implemented a prototype of A&B and evaluated it using real traffic from a backbone network [11]. The experiment results show that A&B achieves similar forwarding performance with only $1/8$ capacity or even less. We also show that A&B can effectively handle different Wide Area Networks (WANs) with various traffic characteristics.

Our contributions are summarized as follows:

- Characterizing the backbone network's workload from the perspective of router traffic to motivate the requirement of a lightweight TCAM entry replacement scheme. (§III)
- Presenting AIR, an AI-based TCAM entry prediction scheme that achieves the identical forwarding performance by decoupling entry aggregations and parallel execution. (§IV)
- Proposing BIT, a block-based TCAM entry insertion tactic that successfully loosens the strict storage restrictions of dependent entries stored in TCAM, and thus enhances the update efficiency. (§V)
- Demonstrating the practical benefits of A&B by a real-world backbone network playback. (§VI and §VII)

This paper is organized as follows. We review related work and motivations in § II. In § III, we describe the overall structure of A&B. In § IV and § V, we detailedly introduce the two modules of A&B, AIR and BIT, respectively. We then conduct extensive evaluations and show the results in § VI and § VII. We conclude the paper in § VIII.

II. Related Work and Motivation

In this section, we review related works of routing entry lookup and TCAM, and present the motivation of our A&B design.

A. Related Work

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

The software-based entry lookup is a kind of classic approach, which mainly stores prefixes in trie structure in binary way [12]–[17]. Stefan *et al.* used a single node to replace all previous complete subtrees based on the level-compressed tried to further reducing the forwarding table space [16]. SAIL divided routing entries into three levels, level 16, 24 and 32, which can achieve a satisfactory lookup performance due to less level visiting requirement [13]. Another alternative way is based on the hash table [18]–[22]. 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 [18]. CoLT exhibited a great memory efficiency and can launch parallel lookup over tables during every lookup due to its hash tables permit multiple possible buckets to hold each prefix [21]. Additionally, some works used bloom filter to perform the entry lookup task [23]–[28]. However, time-varying forwarding table needs to frequently reselect hash functions, which will reduce hash performance and increase update difficulty.

As the link rate of the backbone increases, the traditional CPU-based software-based lookup algorithms have been unable to meet the lookup demands of high-speed communication systems. Then Graphics Processing Unit (GPU) is used in some works to assist in accelerating the entry lookup due to its excellent parallel capabilities [29]–[31]. Younghwan *et al.* employed integrated GPU in Accelerated Processing Unit (APU) platform to achieve multi-10 Gbps performance for many compute/memory-intensive algorithms [31]. The above software-based lookup algorithms are highly flexible, and the performance can be enhanced by the parallel processing capability of the GPU, however, they still suffer from the lookup speed limitation essentially due to the inherent characteristics. Usually, Field Programmable Gate Array (FPGA)-based entry lookup strategy needs to address two main issues: how to store all routing entries information on the chip and how to construct pipeline stages [32]–[34]. Some works proposed to

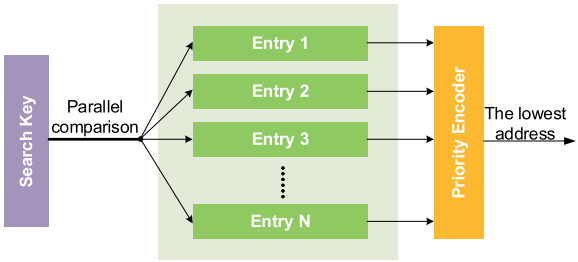
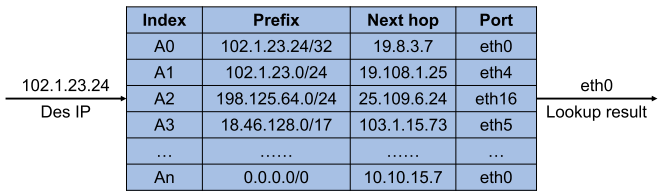


Fig. 1. TCAM lookup structure.



Index	Prefix	Next hop	Port
A0	102.1.23.24/32	19.8.3.7	eth0
A1	102.1.23.0/24	19.108.1.25	eth4
A2	198.125.64.0/24	25.109.6.24	eth16
A3	18.46.128.0/17	103.1.15.73	eth5
...	.....	.....	...
An	0.0.0.0/0	10.10.15.7	eth0

Fig. 2. Dependency of entries.

compact data structure and store a part of the data by using hashing [35]–[38], and some strategies proposed to adjust the trie structure by rotating some branches to balance stage size [39]–[41]. A complex search algorithm will increase the logic complexity of the FPGA, resulting in clock frequency trade-offs.

2) Ternary Content Addressable Memory: TCAM is a three-state content addressing memory that can provide 0, 1, and x (ignored) matches and can match of all routing entries according to the entered key value in one clock cycle, then it returns the address of the matched entry. When there are multiple matches caused by the x state, it will by default hit the entry with lowest stored address of all matched entries, i.e., the entry stored at lower addresses in TCAM has higher priority.

Given the three-state characteristic that is suitable for the longest prefix matching, TCAM has been widely used for rule-based entry lookup and packet classification [42]–[45], whose structure of lookup is shown in Figure 1.

Thus, the storage of entries in TCAM has to follow the principle of relative pre-post location relationship, which means that for entries with the same prefix, all entries must be stored in the lower address location than those entries with shorter masks. As shown in 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` to be forwarded to correct port `eth0`, otherwise, the packet would be forwarded to the wrong port `eth4`. It should be noted that the mask length represents entries' priority. Hence, some entries stored in TCAM may have to be moved to free up a suitable location for the newly inserted entry, even if there are some free spaces with high address. As an analysis, inserting a single entry for a $1K$ entry-set requires a maximum of 466 entry-moves [6]. To solve this problem, RuleTris reduced the average moves per entry insertion to about 10 times by the designed algorithm, however, it is time consuming and causes unacceptable latency in calculating the movement scheme [10]. In addition, a more complicated issue exists, e.g., if only `102.1.23.0/24` is stored in TCAM according to the cache rules, then those packets which should matched to like `102.1.23.1/32`, `102.1.23.16/28`, etc., will also be forwarded incorrectly. Due to the entry dependencies, when entry e should be inserted into TCAM, theoretically all dependent entries with mask are longer than e must also be inserted into TCAM jointly to guarantee the correctness of the matching result, even if the caching policy does not require these dependent entries. Especially, for TCAM, the move operations and query operations can only be performed serially, so updating the TCAM may significantly decrease the query performance.

3) TCAM Optimizations: To address the issue that TCAM's capacity can hardly meet the growth of routing entries, the series of Rasor-based works [46]–[48] are pursued from the perspective of compressing entry table, but minimizing the number of entries for a single switch is computationally difficult. With this, some studies investigated how to efficiently and accurately decompose a large table stored at network ingress into several smaller sub-tables and distribute them across network switches [49]. Using TCAM as a cache can also alleviate this problem from another perspective, which, however, introduces hit rate, content updates, and other challenges [50]. Sheu *et al.* leveraged a sophisticated algorithm according to temporal and spatial traffic localities to select entries for better TCAM hit-rate [51]. It requires multiple cumulative calculations from entry nodes to the root node in the entry dependency directed acyclic graph (DAG) to obtains the most appropriate cached entry-set in each update of TCAM. However, it is difficult to provide timely cached results for the lightweight TCAM scenario of this paper. T-cache crafted dependency-free rules in cache update and used statistical-based strategy to select cached rules [52]. CacheFlow divided all entries into several subsets according to dependencies to improve TCAM efficiency [53]. However, such proactive schemes are limited in the ability to generate entries dynamically based on the evolving network. Inspired by the advantages of prior studies, in this paper, we propose an AI-based entry selection strategy for TCAM caching that can still employ the existing schemes to alleviate the effect of entry dependencies from entry table perspective.

Several researches are conducted from the perspective of TCAM updates. Bohan *et al.* designed a TCAM update optimization framework that can guarantee consistent forwarding during the entire update process [54] and Ying *et al.* proposed a batch update algorithm which collectively evaluates and optimizes the TCAM placement for whole batches throughout [55]. Zixuan *et al.* proposed a strategy that comprehensively considers hit rate and update efficiency, and the algorithm can improve update efficiency by sacrificing hit rate [56]. Kun *et al.* devoted to avoiding unnecessary entry moves when inserting entries through complex algorithms which computes over the entire TCAM [57], but there is still room for achieving the ideal TCAM entry move reduction. Shah *et al.* designed a TCAM management scheme based on prefix-length of entries [58], but the skewed distribution characteristic of traffic is not considered. As such, we propose a block-based scheme for TCAM management, which can

TABLE I
PROPORTION OF EACH PREFIX-LENGTH ENTRIES

Proportion / Length / Value	8-15	16	17	18	19	20	21	22	23	24	25-32
Min	0.038	0.068	0.041	0.015	0.034	0.050	0.060	0.068	0.096	0.500	0.004
Average	0.040	0.070	0.043	0.016	0.037	0.051	0.062	0.070	0.098	0.507	0.006
Max	0.042	0.072	0.044	0.018	0.038	0.053	0.065	0.074	0.101	0.513	0.009

eliminate the effect caused by the dependency of entries when inserting new entries and avoid movements of the entries stored in TCAM, thereby eventually improving the update efficiency of TCAM.

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. Shunsuke *et al.* designed a forwarding information base (FIB) storage structure based on learned index, which is less than half of an existing trie-based FIB while it achieves the computation speed nearly equal to the trie-based FIB [59]. NeuroCuts [60] and NuevoMatch [61] proposed the AI-based method to construct a decision tree for fast packet classification. However, the essence of these studies is still to utilize the software-based tire structure to perform the lookup. He *et al.* proposed a meta-learning scheme to improve the accuracy by predicting different categories of traffic separately and then integrating results into a overall traffic prediction result [62]. It allows to train an individual predictor to adapt to a new category of traffic, but it is only applicable to traffic granularity prediction problems. Moreover, some studies applied AI to deal with caching strategy issues. Zhang *et al.* introduced an additional attribute, i.e., the spatial feature of short videos, to predict popularity through a graph convolutional neural network model [63]. DeepCache architecture proposed in [8] accounts for predicted information of objects to make smart caching decisions, which, however, is hard to cope with prediction tasks with large-scale objects like routing entries. The FreeCache proposed in [9] tackled the large-scale prediction objects problem by indexing and mapping, but it will introduce additional overhead in each prediction. In addition, the above scheme needs to maintain the temporal relationship between objects, which makes it difficult to parallelize the prediction process. Then, how to ensure the timeliness of prediction results becomes a challenge. Therefore, we proposed a parallelizable entry prediction algorithm with temporal and spatial characteristics decoupling.

B. Motivations

We characterize the workload of a core router located in New York backbone network [11], and disclose the opportunities of designing a decoupled and parallelized AI-based prediction solution and an effective entry insertion tactic.

- **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 [11], [64]. This natural distribution property provides an opportunity to leverage a small capacity TCAM while maintaining the high lookup performance by predicting hot entries.
- **Stable and independent routing entries.** Numerous flows and their dependence present the challenges to prediction algorithm. However, the number of entries of a RIB 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.
- **Isolation of the same prefix-length entries.** To satisfy entries' pre-post location constraint introduced by their dependencies, the update process of TCAM may involve complicated entry movements, which is computationally intensive and time-consuming. Then, trying to achieve lightweight TCAM through delicate replacement mechanism faces entry insertion challenge. Fortunately, entries with the same prefix length are isolated from each other, i.e., the above mentioned constraint is non-existent between them. Besides, the proportion of the number of each prefix-length accessed entries over a time interval is relatively stable,[2] as shown in Table I, which provides a new idea to accelerate the entry update of TCAM.

The aforementioned observations and analyses co-motivate the desirability of customizing the AI and block-based approach to realize a lightweight TCAM.

III. A&B SYSTEM OVERVIEW

In this section, we first describe the sketch of A&B briefly, and then present the framework and core-structure of two main modules, AIR and BIT, respectively.

A. Sketch of A&B

The overall architecture of A&B is shown in the Figure 3, which concentrates on the TCAM-based entry lookup part. It consists of two main core modules, AIR and BIT. AIR is mainly related to the prediction of TCAM storage elements by traffics, while BIT is concerned with updating the TCAM by the prediction results in an efficient way.

B. Framework of AIR

The overall structure of AIR is shown in the bottom rectangle of Figure 3, which can be outlined as three workflows.

[2]The number proportion of each different prefix-lengths accessed entries in per 10^6 queries, and the min, max and average values are calculated from more than 10 continuous periods.

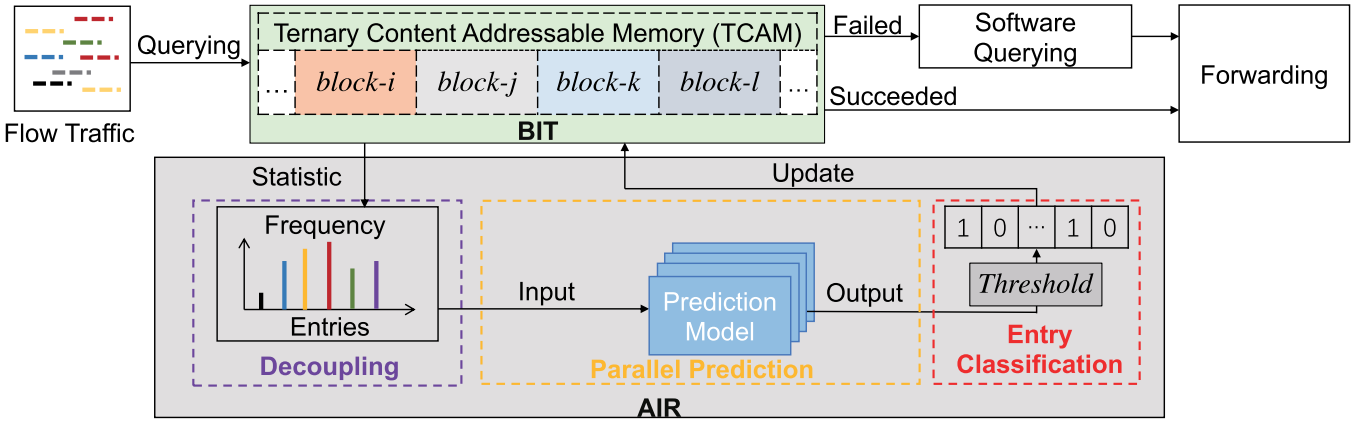


Fig. 3. A&B architecture.

1) **Entry Decoupling:** The original time-series data of flows in router is variable and extra-long, which is unpractical for prediction. Therefore, it is decoupled into entries access frequency data in the sub-module marked with a purple dashed circle in Figure 3. The consecutive historical access frequency data of the each entry still maintaining the time-dimension characteristics, which will be used as input to the subsequent module.
2) **Parallel Prediction:** The access frequency of each entry of the next period can be predicted by the AI model, which is the basis for determining whether the entry is hot or not, i.e., whether it should be inserted into the TCAM. The decoupling of first stage individualizes each entry computation and makes the parallelized prediction model practical, which guarantees the timeliness performance of the whole algorithm. This phase is marked with an orange dashed circle in Figure 3.
3) **Entry Classification:** On the basis of the predicted next-period access frequency, all entries can be divided into two categories depending on the set threshold TS, i.e., entries that are accessed less frequently than the threshold are classified as non-hot, otherwise, as hot, which will be inserted 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 utilizing prediction. Firstly, AIR converts a flow prediction issue into an entry prediction task, thus enhancing the accuracy of prediction. Secondly, due to the requirements of linear speed processing in network scenarios, AIR tactfully decouples entries, which not only parallelizes the prediction process so that the prediction results can fulfill the network scenarios, but also enables prediction granularity and interval to be more flexible. Moreover, the threshold TS can be tuned according to the network status, which can enables AIR to adapt network dynamic variations better. The details are given in § IV.

After our proposed entry decoupling, it is possible to use statistical methods to accomplish the functional requirements. However, statistical methods like moving average algorithms perform satisfactory on data with relatively stable trends. However, based on the statistics of real traffic, there are a large proportion of entries with relatively large fluctuations in hotness, in which case the learning model-based will perform better. Therefore, in order to accurately predict the hotness of all entries, we decided to use the LSTM model.

C. Core-Idea of BIT

The core idea of BIT is shown in Figure 3. There are dependent relationships between routing entries with the same prefix but different mask-lengths, which is the main issue that leading to the complicated content updates of TCAM. In a nutshell, BIT divides all routing entries into different groups based on the mask length and virtually isolates TCAM into different blocks. According to the mask length of the entry, the block stored is specified, where the longer the mask length is, the lower the address of the corresponding block in TCAM, which is determined by the query characteristic of TCAM.

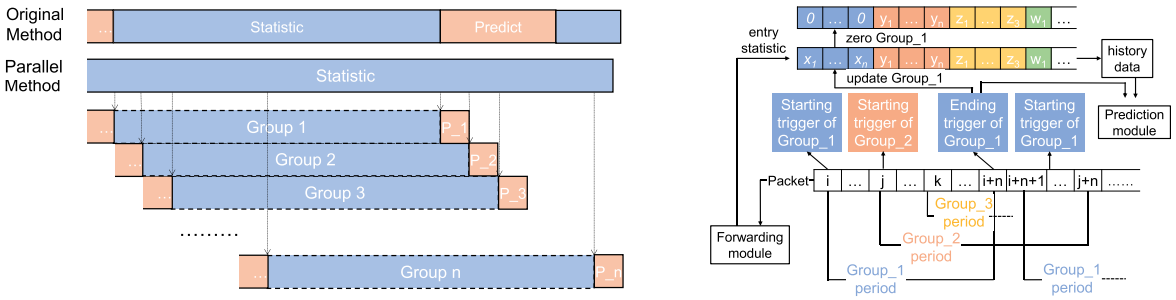
In this way, the complicated influence of dependent relationships on TCAM update can be eliminated, making the update process more efficient. The details are demonstrated in § V.

D. Explanation

A&B proposed in this paper mainly focuses on scaling down the required TCAM resources. The traditional TCAM update methods require highly frequent entry replacement to update such small-scale TCAM, which are accompanied by a significant amount of computation and execution of the corresponding dependent entries movements scheme, i.e., the traditional methods are hardly applied directly in the small-scale TCAM scenario. Given the skewed distribution of traffic, AIR selected high-frequency entries accurately for updating TCAM based on prediction results of the LSTM-based model. Hence, AIR minimizes the number of replacements of entries while ensuring the TCAM hit rate compared with the traditional method. In the complexity of calculation and execution of dependent entries movements scheme, BIT eliminates such operations that are required the traditional method.

IV. AI-Based Entry Prediction

In this section, we describe AIR in detail, including prediction algorithm and some optimization schemes. Table II



(a) Parallelization sketch

(b) AIR parallelization process structure

Fig. 4. AIR parallelization.

TABLE II
NOTATION DESCRIPTION OF AIR

Notation	Description
e^i	entry i
Δ_t	time interval in the unit of packets.
IL	the length of the statistics period in the unit of packets.
TS	classification threshold value for hot entries
DS	disjoint group of entries

summarizes the common notations and parameters in the description of AIR.

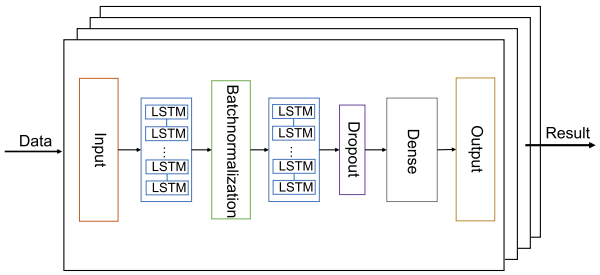


Fig. 5. Prediction model structure.

A. Design Principles

The design of AIR is driven by the following three principles:

- *Proper Prediction Granularity.*
  Prediction based on flow level is challenging, and such a massive number of flows would result in huge state space. AIR takes a new perspective and focuses on corresponding entries of aggregated flows. The number of entries is relatively stable, thus its state space is much smaller than that of flows, which significantly reduces the computing overhead. In addition, the results of prediction of entries will be more accurate.
- *Data Decoupling.*
  The dynamic of traffics makes the selection of model inputs very difficult. If the input sequence is selected at a fixed time interval, the number of selected flows would be ever-changing, which makes the prediction algorithm invalid. If the input sequence is selected in a fixed-length way, it is necessary to make real-time adjustments based on traffic status to retain characteristics. Additionally, it requires tens of thousands or even longer sequence lengths to capture the data characteristics. The dynamically variable and ultra-long traffic sequence data is quite challenging for prediction model. In AIR, we calculate each entry access frequency independently, thus decoupling the sequence dependencies. In this way, the subsequent prediction module can parallelly predict the next period access frequencies of the entries based on the historical data.
- *Dynamic Prediction Interval.*
  When the network traffic is relatively stable, frequent prediction operations are not necessary since the hot entries are already stored in the TCAM. When the network fluctuates, however, it is necessary to increase the prediction frequency to update the TCAM in time. In AIR, this matter is tackled by using dynamic prediction intervals, which can be set and adjusted flexibly according to the traffic status.

B. AIR Methodology

1) Entry Decoupling: As described in above sections, the issue of flow prediction based on the aggregated time-sequential traffic data can be formulated as follows.

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

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 address this problem from a different perspective by decoupling each flow from the aggregated traffic. We set a periodic interval Δ_t, and transform the origin flow sequence data into entry accessed frequency, and for each entry e_i, the frequency prediction problem can be denoted as follows:

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

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

It should be noted that the interval Δ_t is strategically set to be dynamically adjusted, which is lower for fluctuating traffic than for stable traffic. Moreover, the required number of period of historical data to predict next period frequency is different, which will be less for stable traffic than for fluctuating traffic.

In practice, the appropriate number of data history can be selected by refereeing the prediction accuracy in the offline training process of the model.

2) Parallelization: Although the flow sequential data has been decoupled into entry access frequency data, the centralized prediction of a large number of entries makes the timeliness of the prediction results not guaranteed. As shown in the *Original Method* of Fig. 4(a), assuming that the time to predict an entry is P_t, then the entry prediction results delay in the worst case is $N * P_t$, where N is the total number of entries to be predicted. The decoupled entries are independent from each other, then it is not necessary to put all the entry predictions in one time period. Therefore, we design a parallel prediction scheme.

AIR divides all entries of the RIB into disjoint groups (DG) and sets different packet count-based statistic start flags for each DG. To ensure consistency in the statistical period of each entry, each entry only belongs to a single DG. The best case is to averagely divide the entries of the same access frequency range into each DG based on the historical data, which can ensure that all DGs can avoid an extreme situation, such as the entries within a DG that all with high access frequency. One should note that the entries in the same DG have no dependencies except that they have the same start/end triggers of the statistics period, so the division of entries can also be adjusted flexibly according to the changes of network flows. As shown in *Parallel Method* in Fig. 4(a), the globally defined packet counter is maintained by the router. When each entry is accessed, its corresponding access frequency counter is added by 1. Since each DG has different start flag, assuming that DG_1 starts its statistics from the ith packet and DG_2 from the jth packet. When the $(i+n)th$ packet arrives at the router, where n is period-length IL, the next period frequency of entries of DG_1 will be predicted based on the access frequency at that interval (i.e., the latest statistics period) and historical data, and update the TCAM caching according to the prediction results. Then, a new statistics period will be started when the $(i+n+1)th$ packet arrives. Similarly, the entries of DG_2 are triggered to be predicted at the arrival of $(j+n)th$ packet, and its new statistics period will be started at the arrival of $(j+n+1)th$ packet. Lastly, the worst prediction result delay is $(N/d) * P_t$, where d is the number of DG.

The detail architecture of parallelization is shown in the Figure 4. The access frequency data of the entries obtained from the forwarding module. Since the independent of each entry, the counting phase does not need to consider the grouping restriction and directly performs a $+1$ operation on the counter of the corresponding access frequency. A series of operations will be triggered at the end of each statistical period of each DG. The latest statistical results of this DG 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 DG in the statistical array will be cleared to start the next period statistics.

Algorithm 1: AIR Algorithm

entry_statistic(): return DG_i that should be predicted;
decouple_entry(i): return decoupled entries of DG_i;
get_hisdata(e): get entry e historical data array;
LSTM_predict($data_e$)**:** predict e frequency;
entry_cla(TS, DG_i_data)**:** return hot entries of DG_i;
Input: trigger signal
Output: a new prediction thread
while *TRUE* **do**
  $group\_i$ = entry_statistic();
  if $group\_i$!= $None$ **then**
    New_Thread: `Pre_Fuc(`$group\_i$`)`;
Function `Pre_Fuc(`$group\_i$`)`**:**
  for e *in* $group\_i$ **do**
    $\vec{N}$ = get_hisdata(e);
    V_{output} = LSTM_predict($\vec{N}$);
    fre_data.append(V_{output});
  $hot\_entries$ = entry_cla(TS, fre_data);
  update_TCAM($hot\_entries$);

3) Algorithm Design: The pseudo code of AIR is shown in Algorithm 1, and the function is described in detail. The processes are as follows: (1) Triggering group prediction by traffic measuring. (2) Obtaining decoupled entry data based on traffic data. (3) Predicting entry frequency based on historical data. (4) Getting hot entries from prediction results and threshold values TS. (5) Updating the hot entries into TCAM.

a) Input and output: The historical access frequency data for each entry is a one-dimensional vector, e.g., $\vec{N} = (n_1, n_2, \ldots, n_i)^T$, where the value with the smallest index represents the access frequency of the latest Δ_t. The vector is segmented by a sliding window with size n and step 1 as the input $\vec{N}_{input}$ of the prediction model. n is the required number of historical periods for prediction. And the output V_{output} of the prediction module is an integer that indicates the predicted access frequency of the entry in the next period. The prediction results of all entries can be used as reference for TCAM update decisions.

b) 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 $LSTM_{(units=128)}$ 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 $Full\ Connection$ layer whose activation function is the $relu$ function. Moreover, the $Adam$ optimizer and the mean square error loss function are used for the model.

The prediction model will be used offline after it converges, which can still be trained periodically or when the TCAM hit rate decreases based on the latest entries access frequency data.

4) Optimization: Current commercial switches or routers are configured with measurement and analysis tools, which are linear processes. However, a great number of prediction calculations will consume computational resources and cause latency. An inefficient strawman way is to predict all entries of a DG after each statistical period. Then the prediction is optimized from the spatial aspect on the basis of parallelization to alleviate this issue. Based on observations, it can be confirmed with high probability that some entries will present very low activity and will not even be accessed in the future, then the computational resources occupied for these entries are actually wasted. Hence, we attach constraints to filter out the entries that are not worthy to predict in order to minimize the computational burden. In addition, the prediction process involves saving historical access data of all entries, which will occupy a large amount of storage memory. We designed an efficient data storage structure and strategy to optimize the data storage of AIR. The specific design schemes will be presented in detail below.

a) Entry filter: As described above, it is not necessary to predict all entries of a DG within a processing cycle. To reduce the complexity and time of prediction computations, we propose a pre-filtering strategy for entries. Before performing the prediction operation, a portion of the entries of DG which need to be predicted will be filtered. That is, when the last few statistical periods frequencies of an entry are all 0, then the next period frequency of this entry is set to 0 by default without performing prediction operation. The mathematical formulation of the entry filter is presented as the equation (3).

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

where K represents the number of historical periods that need to be observed.

It is possible to add a identifier to record whether the historical frequency is 0 or not. Assuming that $K = 8$, the array of identifiers for DG_i is $f[N_i]$, and N_i is the total number of entries for DG_i, then each identifier can be recorded by an unsigned char data. Such as, the identifier $f[i] = 0 \times 10(0001\,0000)$ indicates that the access frequency of the penultimate fifth period of entry e^i is not 0 and the rest is 0. And $f[i] = 0 \times 00$ indicates the e^i is $dead$. Identifiers will be updated when a new statistical period is completed. Taking the example of updating the identifier of e^i: First, moving f[i] one bit left, i.e., $f[i] \ll 1 = 0 \times 20\,(0010\,0000)$, the position corresponding to the original penultimate fifth period is shifted left to the penultimate sixth, and the position of the latest period will be set to 0 by default. If the access frequency of e^i in this new period is 0, then the update of this identifier is complete; otherwise, the identifier needs to be or-operated with $0 \times 01\,(0000\,0001)$, i.e., $f[i] \,||\, 0 \times 01 = 0 \times 21\,(0010\,0001)$, to revise the position corresponding to the latest period as non-zero. 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 $f[i] == 0$ **then**
    $n\_f[i] = 0$;
  else
    $n\_f[i] = \text{predict}(data[i])$;
return n_f;

Algorithm 3: Identifier Update

Input: identifier array f and current frequency array c_f
Output: f
for i *in* N **do**
  // $f[i] = f[i] \ll 1$;
  if $c\_f[i]! = 0$ **then**
    //revise the bit of identifier of latest period as 1
    $f[i] = f[i] \,||\, 0 \times 01$;
return f;

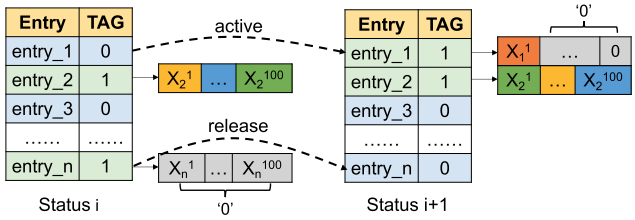


Fig. 6. Storage structure.

b) Storage: Given that the number of routing entries in a real router is almost close to 10^6, it would be inefficient and unacceptable to store a several periods of access frequency data for all entries. Supposing that 100 periods of data are recorded with unsigned short int, which will take $120MB$ $(32bit * 10^6 * 100)$ of on-chip storage space. From the observation of each entry historical frequency data, it is found that when $dead$ becomes $active$ again, its recorded 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, it is unnecessary to store a large amount of data, and all historical access frequencies are 0 by default. The overall storage architecture is depicted in the Figure 6. Due to the filter, the storage optimization of entries will not affect the prediction at all, because the $dead$s would have been filtered out already. When the statistical register data shows that the status of an entry changes from $dead$ to $active$, its identifier will be changed and its subsequent access frequency will also be recorded. When the last K periods historical access frequencies of an entry is 0, where K can be kept the same as in Equation (3), then its corresponding identifier will be modified and its storage space occupied by it will be free.

The massive number of *dead* entries makes this optimization strategy quite profitable.

V. Block-Based Entry Insertion Tactic

In this section, we describe the BIT optimization tactic for entry insertion of TCAM in detail present a typical case. Then we demonstrate the feasibility of BIT by analyzing real routing entries and flows traffic.

A. BIT Methodology

The skewed distribution of traffic allows us to leverage small capacity TCAM to handle massive entry lookups, which, however, requires timely entry updates as the cost. AIR can predict which entries will match maximum lookups percentage in the next statistical period, and these entries should be stored in TCAM. Then, how to insert selected entries into TCAM be the twin problem.

1) BIT Description: As explained in §II, the dependent relationship between entries makes it necessary to move some entries stored in TCAM when inserting new entries into TCAM to ensure that all entries in one dependent-set are stored in order of address from low to high according to the prefix length from long to short. All entries of a RIB form a directed acyclic graph based on the dependent relationship. And the equivalent prefix length entries are independent of each other, that is, there are no direct relationships between any two of them, which means that their relative position in TCAM does not affect the match results. The independent of all equivalent prefix length entries present an opportunity to optimize the entry insertion process during TCAM updates. Under this circumstance, we proposed a BIT strategy to optimize the entry insertion process of TCAM. To keep it consistent, we take routing entries corresponding to the IPv4 address as focus description.

Hence, according to the match regulation that the lowest address entry will be returned when there are multiple matched entries, we divided TCAM into 25 different capacity parts virtually from low address to high address by setting boundaries. BIT constrains all entries of different prefix-lengths can only be stored in the corresponding block when inserted into TCAM. Firstly, the order of divided blocks ensures that the newly inserted entry will not affect its dependent entries stored in other blocks. Then, since the entries in each block are all the same prefix-length, i.e., the entries stored in the same block are independent of each other. So, it is possible to directly insert the new entry into the free space or overwrite the removed entry according to the replacement algorithm.

The reason for not allocating TCAM space to *1-bit* to *7-bit* is that they are practically non-existent in almost all RIBs that we researched. Besides, it is necessary to reserve an additional one fixed space with the maximum address of TCAM to store `0.0.0.0/0` for default forwarding. The sketch of BIT TCAM is shown in Figure 7. In contrast to the classical literature and the state-of-the-art tactics [6], [55], [58], we take the influence of skewed traffic distribution on entry accesses into consideration and design the BIT, an entry access frequency-based TCAM storage constraint strategy.

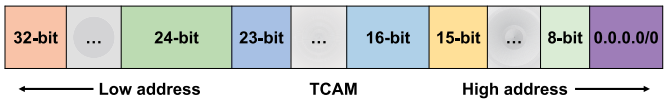


Fig. 7. Block sketch.

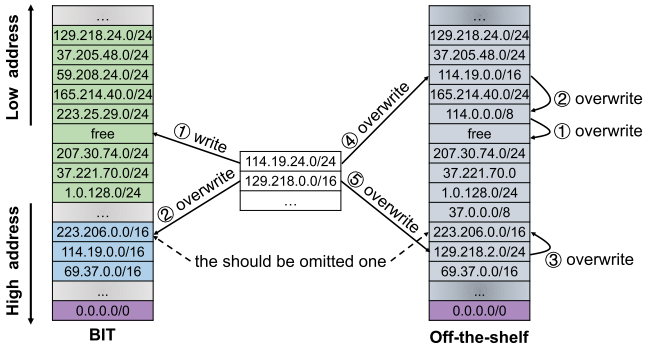


Fig. 8. Insertion comparison.

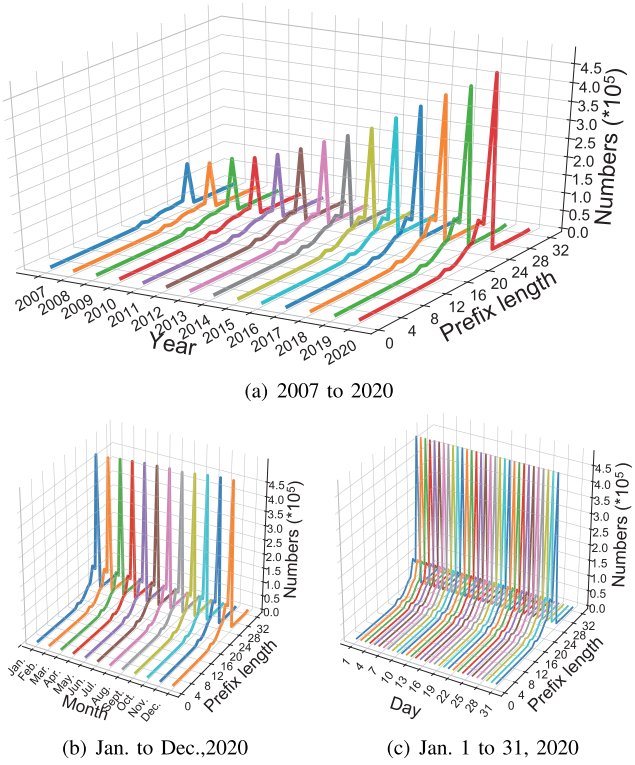
2) An Instance: In the BIT way, the constraint of storage range makes it possible to decouple dependent entries of different lengths from each other in TCAM, thus avoiding the movement of entries caused by finding the proper space during the insertion process. For an entry that needs to be inserted, it is simply to insert it into free space or rewrite the entry that should be omitted from TCAM in the corresponding length block. In the following, we will introduce the merit of BIT structure over the off-the-shelf strategy with a clear illustration.

As shown in Figure 8, suppose `114.19.24.0/24` and `129.218.0.0/16` should be inserted into TCAM. For the BIT scheme, `114.19.24.0/24` can be written into the free space of the *24-bit* range directly, and `129.218.0.0/16` can overwrite `223.206.0.0/16` that should be omitted according to replacement rules. For the off-the-shelf scheme, due to the dependent relationship, it is necessary to move `114.0.0.0/8` and `114.19.0.0/16` to high address to vacate a relative lowest address space to store `114.19.24.0/24`.

It should be noted that these movements are in this relatively simple case, however, in fact, the movements will be more complex in the circumstance of a large number of entries in TCAM. Then, the BIT scheme can greatly optimize the entry update process. The reason is that, in this example, the intentional omission of inserting (omitting) dependent entries with high (low) priority of the appointed one is because neither of these two ways affects the number of associate dependent entries.

B. Feasibility Analysis

Based on the real Routing Information Base (RIB) from CAIDA [11], we counted the numbers of entries corresponding to different prefix length from three timescales: from 2007 to 2020 in years, from January to December of 2020 in months, and from 1 to 31 of January of 2020 in days, respectively. The results are shown in Figure 9, where all the lines in Figure 9(a) show a similar trend and the lines in Figure 9(b)

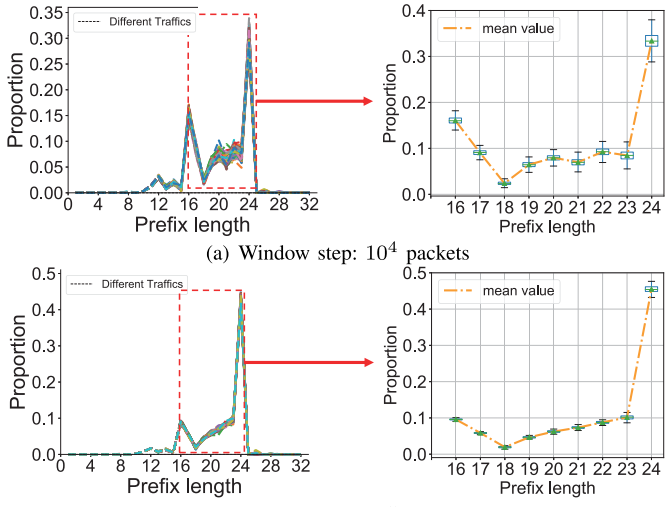


(a) 2007 to 2020

(b) Jan. to Dec.,2020

(c) Jan. 1 to 31, 2020

Fig. 9. Entry proportion.



(a) Window step: 10^4 packets

(b) Window step: 10^5 packets

Fig. 10. Accessed entries proportion.

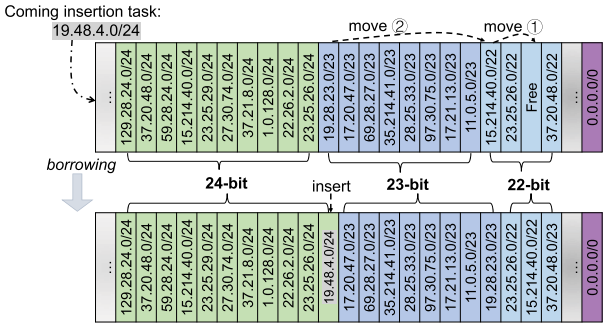


Fig. 11. Borrowing sketch.

and Figure 9(c) are almost completely overlapping. The context of the stable proportion of each prefix length entry in RIB drives us to speculate whether the entries stored in TCAM also shown a stable distribution based on the prefix length. Then, we respectively statistic dozens of periods of all matched routing entries based on a real traffic from CAIDA in different window steps, 10^5 and 10^4, the results of proportion of different prefix length in each period are shown in Figure 10. Entries with prefix length of *24-bit* and *16-bit* are the most numerous two categories, which account for nearly 50% of the total entries. The stable and skewed distribution of entries access frequencies enables BIT to be feasible.

C. Dynamic Range

According the observation of Figure 10, for a series continue flows, the ratio of different prefix length accessed entries fluctuates slightly between each window step. So, storing different prefix lengths entries into a fixed range may cause improper utilization of TCAM resources. Suppose there is no proper space to store when inserting an entry, including no free space and no entries that can be removed from TCAM, which indicates that the capacity of the corresponding block is insufficient. Based on the insufficient degree, it can dynamically tuning the store range of each prefix length block by *borrow* from neighbors. The reason that it can only borrow from direct neighbors is to ensure the continuity of all ranges. When neighbors are not available for borrowing, based on transferability, it is permitted to use the ranges between the two as transit to borrow from a non-directly connected block. The *borrow* only occurs when the distribution of the accessed entries changes and still follows the basic proportion constraints.

AIR provides entry access statistical information, which can be leveraged for the tuning process. The range pointers of blocks can be adjusted at a certain time interval or period, according to the access ratio of different length prefix entries. Each range can be updated as per Equation (4), where $R(i)$ indicates the latest ratio of accessed entries of prefix length i. That is, the range pointer of each block is updated in the order of prefix length from small to large based on the latest proportion.

$$Block_j = [(1 - \sum_{i=0}^{j} R(i)) * size(TCAM)] \tag{4}$$

The sketch of the *borrowing* process is shown in Figure 11. Supposing a new entry inserting of the 24-bit range triggers a borrowing operation. At this time, the neighbor of the 23-bit range is unavailable, but the 22-bit range has a free space. First, entry `15.214.40.0/22` should be moved to the free space to release the lowest address space of the 22-bit range. This internal movement is to guarantee the continuity of each range space during the borrowing process. Then, entry `19.28.23.0/23` will be moved to the newly released free space. After updating related range pointers, the new entry,

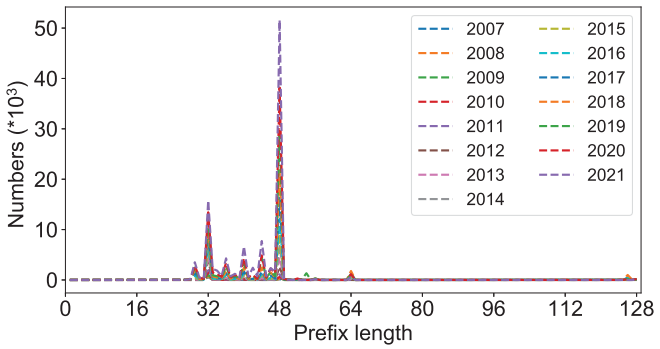


Fig. 12. Number of IPv6 entries from 2007 to 2021.

19.48.4.0/24, can be inserted into the highest address of the 24-bit range (the lowest address of the original 23-bit range).

D. Discussion

The BIT scheme is mainly proposed for IPv4 entries, so it may generate an intuition that it cannot be extended for IPv6 lookup or Access Control List (ACL) due to the longer prefix length or multiple match fields, which theoretically smashing block ranges and further increasing its quantity. However, in fact, there is still a turnaround in this intuition.

An IPv6 address is divided into two parts, the first 64 bits represent the network address and the rest 64 bits represent the host address. In fact, IPv6 entries present a better aggregation. The results of our statistics on real IPv6 RIBs[3] are shown in the Figure 12, which also confirms this conclusion. For multi-fields ACL lookup, the BIT is equivalent to dimensionality reduction of IP address dimension. Therefore, it is still possible to apply the existing algorithms in each block independently, but the complexity of the computing is greatly reduced.

VI. Evaluation for AIR

In order to present the experimental results in a clear and concise manner, we evaluate AIR and BIT independently in two sections, and then show the comprehensive performance of A&B. In this section, we clarify the experimental setup and analyzed the performance of AIR with some existing strategies.

A. Experimental Setup

The data that we use in our evaluation was collected on January 17, 2019 from a core router of a backbone network in New York [11], with each piece of data consists of a five tuple with timestamps: ⟨ time series, source IP, destination IP, source port, destination port, protocol⟩.

In this set of experiments, to shield the entries dependency feature, which be evaluated in § VII, we re-aggregate all appeared destination IP addresses in data-set to 24-bit mask length entries. We assume that the set of all aggregated entries is the entire entry-set, which ensures the overall implementation and feasibility. The total number of packets in the data set is 1.3×10^8, including 6.6×10^5 different destination IP addresses, and 8,598 re-aggregated entries. For larger number of entries in real backbone network, AIR is still applicable.

The LSTM-based prediction model is implemented by Keras on Ubuntu 16.04-LTS operating system, where the learning rate is set to 0.001, the batch size is 64, the epoch is 30, the initial weights and biases are generated randomly, and the training optimizer is *Adam*. We select 1,124 historical periods for each of the 8,598 entries and divide them into 1,024*8,598 data items by using a sliding window with size of 100 and step size of 1. The label of each data item is the corresponding access frequency of the next period. Among them, 60% as the training set, 20% as the validation set, and 20% as the test set. And the prediction model is used offline after it converges. The traditional scheme mentioned in experiments of AIR is based *"insert if missed"* replacement strategy, which selects the removed entries randomly.

B. AIR Performance

We first evaluate the required TCAM size when there are different number of entries (from 2,000 to 20,000 entries[4]) in the traffic. For the fairness, different groups experiments were conducted while keeping the hit rates of both algorithms consistent. The experiment results as presented in Figure 13(a), from which it can be seen that under the *traditional* scheme, the required TCAM size increases linearly (from 2,000 to over 16,000) with entries scale. For AIR, the required TCAM size is always around 2,000, which can save more than **8 times** of TCAM capacity. In the real network where the number of entries is extremely over 20,000, AIR works even better.

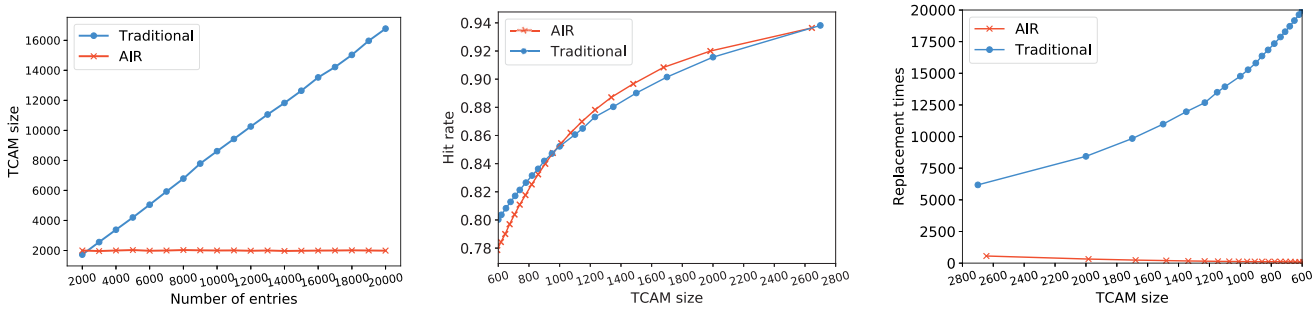
From another point of view, different TCAM capacity sizes will affect the hit rate and the number of entry replacements. AIR and *traditional* schemes are simulated separately by setting different TCAM sizes. The statistical period is set to 10^5, and AIR selects the entries equal to TCAM size from high to low according to the predicted frequency. In addition, the final results of the two strategies are averaged over 170 periods. With the increase of TCAM size, the accuracy of AIR and *traditional* strategies are similar and exhibit an upward trend. For any TCAM size, however, AIR has a significant advantage over the *traditional* in terms of input replacement time. The hit rate and replacement times of AIR and *traditional* schemes are shown in Figure 13(b) and Figure 13(c).

C. AIR Analysis

1) Period Analysis: We set different lengths statistic periods for the same traffic data set, 1×10^4, 2×10^4, 5×10^4, 1×10^5, 1.5×10^5, and 2×10^5. Apparently, for periods of different lengths, the number of IPs or corresponding entries appearing in each period is different. We count 200 periods of different lengths separately, and the distributions of the number of IPs/entries in each condition are shown in the

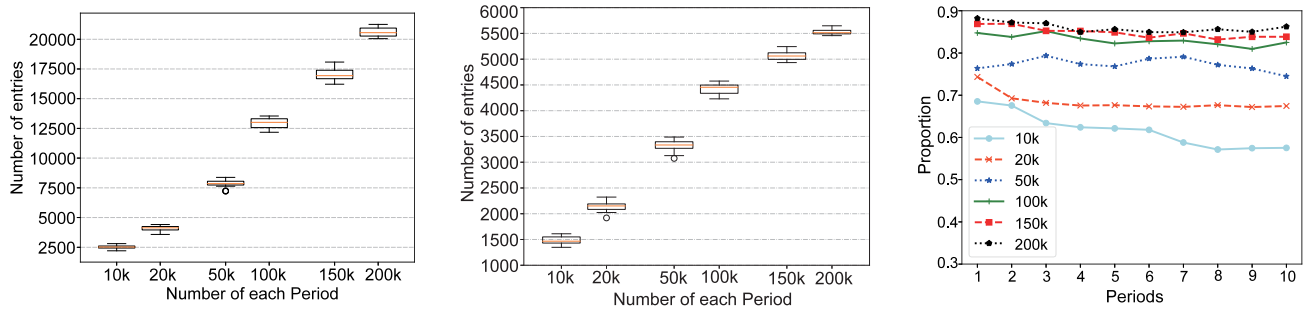
[3]This dataset created on a daily basis, starting from 2005-05-09 for IPv4 and 2007-01-01 for IPv6 [65], [66].

[4]The entries is expanded by extending the measurement span based on the original data set.



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

Fig. 13. AIR performance with the traditional strategy in TCAM size (a), hit rate (b) and replacements (c).



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

Fig. 14. Analysis of period.

box Figure 14(a) and Figure 14(b). As the length increases, the number of IP/entry per period also increases significantly, which makes us face the problem that a large number of entries should be predicted in each prediction process. And this situation also corresponds to the proposed filter mechanism. In order to manipulate the time and resource consumption of the prediction process, the period should not be set too long. From another perspective, we analysed the regularity of traffic and entries periodically 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 are shown in Figure 14. The longer the period, the more stable the hot entries are. And vice versa, the shorter the period, the more obvious the decrease in traffic coverage of hot entries over time. Hence, setting a short period not only increases the prediction frequency, 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 device characteristics, and we choose 10^5 in this experiments.

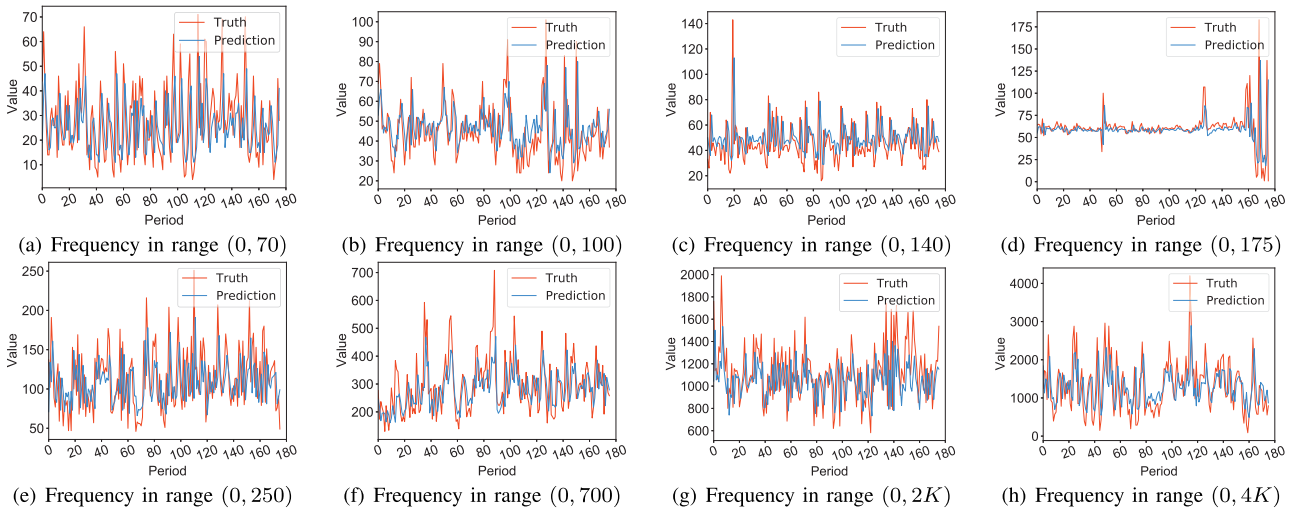
2) Model Prediction Accuracy: As explained in § III, we address the prediction challenge under aggregated flows circumstance by decoupling entries and making prediction on disjoint entry groups. Therefore, we evaluate the prediction performance of AIR here. In the process of model training, we set the different numbers of data history data with a granularity of 50, and prediction results show that there is only a slight improvement in accuracy after the history data exceeds 100 periods. Considering that a longer number of data history poses more challenges for both storage and computation, we set the history period length to 100 in the experiment.

We randomly chose several groups with different access time ranges during 176 periodic intervals. We chose the prediction results compared with the truth in Figure 15(a) to 15(h), which indicate that the decoupled prediction algorithm works well on various frequencies entries.

3) Threshold Analysis: An entry after being predicted will be evaluated whether it is a hot entry based on a threshold value TS, which will directly affect the TCAM hit rate and the corresponding overhead. In this experiment, we evaluate performances of AIR at various thresholds and compare the results with another baseline solution, *Least Recently Used (LRU)*, as well as the *traditional*. In practice, a too-large TS will make fewer entries cached and thus reduce the TCAM hit rate, while a smaller TS will select too many entries that exceed the TCAM space. Therefore, the TS value can be determined based on the TCAM resources of routers, i.e., the TS can be determined and adjusted by the range of access frequency of the corresponding entry when the TCAM is being filled.

- **Hit Rate:** We set the threshold TS to $1\sim10$ and conduct a series of experiments to measure hit rate. In *traditional*, when an entry that should be hit is not 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 prediction future frequency instead.

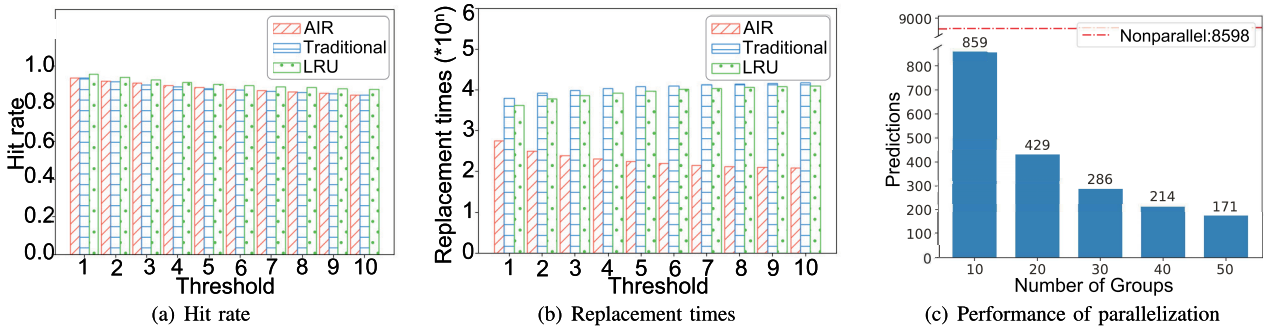
  To ensure the validity of prediction results, we take the average of 176 experiments as the final result. The comparison results are presented in Figure 16(a). From these results, it can bee observed that the average hit rate of the above three mechanisms are similar to each other,



(a) Frequency in range $(0, 70)$ (b) Frequency in range $(0, 100)$ (c) Frequency in range $(0, 140)$ (d) Frequency in range $(0, 175)$

(e) Frequency in range $(0, 250)$ (f) Frequency in range $(0, 700)$ (g) Frequency in range $(0, 2K)$ (h) Frequency in range $(0, 4K)$

Fig. 15. Prediction result of entries.



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

Fig. 16. Comparison of AIR, traditional strategy and LRU in hit rate (a) and replacement times (b), and group performance improvement (c).

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* and *LRU*, respectively.

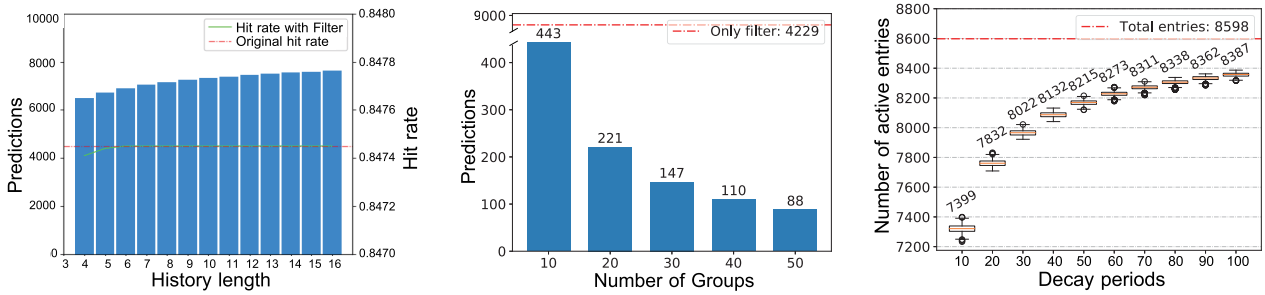
- **Replacement times:** Although a lower TS can improve the hit rate, it also results in higher overhead on calculation consumption and entry replacement latency. The operation of replacing an entry consists of two steps: the first is to select an entry that needs to be removed from the TCAM according to the replacement algorithm, and the second is to free a suitable space for the newly inserted entry by moving multiple entries stored in the TCAM. The overhead of these two steps is kept consistent in the experiments, so we count entry replacement times under different TS (from 1 to 10, the same as in Figure 16(a)) and show the results in Figure 16(b), where the results are in logarithm operation. From these results, it can be seen that the replacement times of both *traditional* and *LRU* are around 10^4 times, while AIR's is around 10^2 times, which is **100 times less**.

4) Parallelization: As described in § IV, we parallelize the prediction process to further improve the prediction efficiency and ensure the timeliness of the results. In this experimental, all entries are divided into 10 to 50 DGs, respectively. The number of predictions required each time is illustrated in the Figure 16(c). As DGs increases, the number of prediction entries per time decreases. When the number of DGs is 50, only 171 entries need to be predicted, which is much smaller than the 8,598 in nonparallel way. The optimal DG configuration which is based on traffic characteristic and computing power can enable statistics and predictions completely parallel.

5) Optimizations:

- **The Filter:** Eliminating $dead$ entries from the prediction list before execution can reduce the prediction computation in each operation cycle. The criteria for $dead$ is different, i.e., the length of the historical period should be reviewed, which will influence the specific judgement. After pre-processing by filters, the number of required prediction entries is shown in the histogram of Figure 17(a). Note that the number of predictions without the filter does not present in the figure, as it depends on the number of entries of the RIB which is close to 10^6 in a core router.

  The effect of the filter on the hit rate at a threshold value of 10 is shown in Figure 17(a). Due to the fluctuation of the traffic, the more successive periods an entry is accessed with a frequency of 0, i.e., the bigger the filter, the higher of accuracy that an entry is identified as $dead$. That is, the bigger the filter, the more computational resources are required to make unnecessary predictions for the $dead$ entry; and the smaller the filter, the higher



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

Fig. 17. Analysis of optimizations.

the risk of misclassification, which may affect the hit rate of TCAM. According to the experimental results, the hit rate maintains the consistency with no filter when the history period length is longer than 6, while the prediction calculations under this condition are less than 6,907. Optimization performance can be further improved if treating access frequency below the threshold as 0.

- **Filter&Parallelization:** The filter and parallelization optimize the prediction execution from the spatial and temporal point of view, which can further reduce the number of predictions per time. In this experiment, the threshold of filter is set to minimum value of 1, which means all entries with a frequency of 0 in the last one period are classified to $dead$, and combined with the parallelization strategy. In the case of different DG configurations, the number of predictions per time is shown in Figure 17(b).
- **Storage:** Although the entry set used for this experiment is not the complete set of a RIB, the storage space improvement can be theoretically calculated. The traffic lasts about 5 minutes for a total of 1,300 periods (10^5 per period), during which 8,598 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 1,300 periods is shown in the Figure 17(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 be concluded that the maximum of the $active$ entries for this traffic is 8,387. In order to facilitate the calculation, we expanded the active number to 10^4, and still kept the total number of entries at 10^6. The on-chip storage space required according to the above rule is $4.13MB$ ($10^6 \times 1bit + 10^4 \times 100 \times 32bit$), 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.

VII. Evaluation for BIT

In this section, we clarify the experimental description and analyze the performance of BIT with the off-the-shelf strategy, and then present the comprehensive performance of A&B.

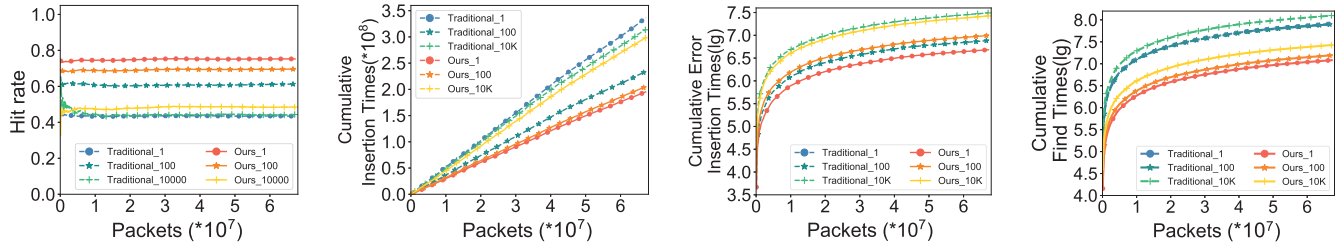
TABLE III
Notation Description of BIT

Notation	Description
PP	Protection Periods: the length of interval to protect newly inserted entry from being removed immediately.
IT	Insertion Times: the number of entry insertions, including missed and corresponding dependent entries.
EIT	Error Insertion Times: the number of entry insertion fails due to block space constraints.
FT	Find Times: the number of dependent entry (appropriate space) lookups when removing (inserting) an entry.
MT	Move Times: the number of TCAM entries movements to free a appropriate space for the newly inserted entry.

A. Experiment Description

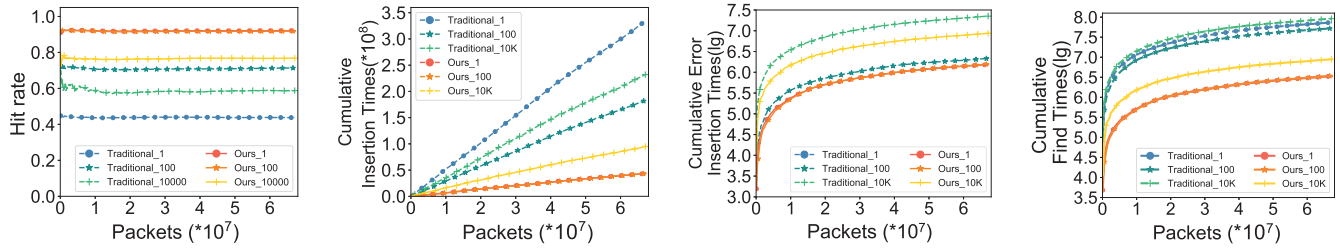
The RIB we used in the experiment is provided by CAIDA [66], and the traffic data is kept consistent with the described in § VI-A. To present the evaluation clearly, we will explain the parameters and measurements we set in the experiment of BIT. And Table III summarizes the common notations.

- `Protection Periods` (PP): When an entry was inserted into TCAM, to prevent it from being replaced out by subsequent inserted entries, we set a protection mechanism for the newly inserted entries and take the number of packets as the unit. For example, when the protection period is 100, a newly inserted entry will not be swapped out until after at least 100 TCAM packet lookups.
- `Error Insertion Times` (EIT): First, the `Insertion Times` (IT) is the number of operations to insert one entry, which is different from the number of misses, because each time when a new entry is inserted, its corresponding dependent entries need to be inserted together. Usually, the insertion times are greater than the miss times. Since the protection period mechanism, there may be insert failures due to the non-existence of entries that can be replaced out of TCAM, and the number of failures is the error insertion times.
- `Find Times` (FT): To find the appropriate space when inserting an entry or to find the dependent entries that need to be deleted together when deleting one entry, we uniformly categorize such tasks as find operations. The uniform expression in terms of times can mask



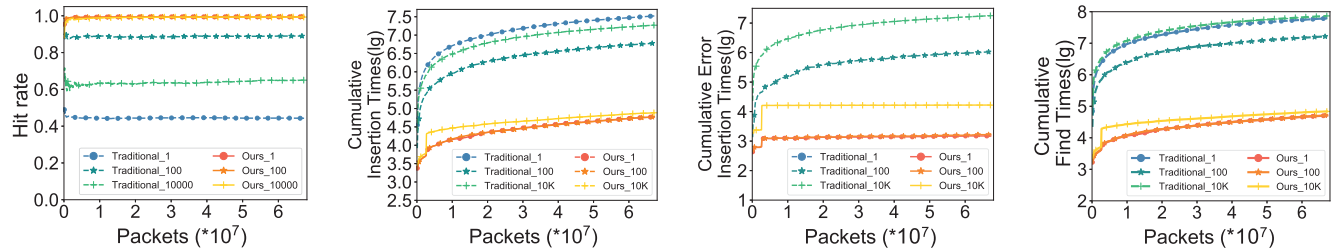
(a) Hit rate (b) Insertion times (c) Error insertion times (d) Find times

Fig. 18. The performance of BIT with TCAM size of 200 in hit rate (a), IT (b), EIT (c) and FT (d).



(a) Hit rate (b) Insertion times (c) Error insertion times (d) Find times

Fig. 19. The performance of BIT with TCAM size of 400 in hit rate (a), IT (b), EIT (c) and FT (d).



(a) Hit rate (b) Insertion times (c) Error insertion times (d) Find times

Fig. 20. The performance of BIT with TCAM size of 600 in hit rate (a), IT (b), EIT (c) and FT (d).

the specific lookup algorithm, e.g., hash or traversal. Moreover, for the traversal algorithm, BIT is traversed in each block instead of whole TCAM, and its complexity can be reduced at least to $\frac{\sum_{i=0}^{n} size(block_i)*times(block_i)}{size(TCAM)*times(all)}$ considering different find times and size of each block.

- `Move Times (`MT`)`: To vacate an appropriate space after *Find* for an inserted entry may trigger multiple overwrite operations. We refer to each overwrite as a move, so MT is the total number of moves to insert entries. Note that, as we explain in § V, MT does not exist in the fixed BIT, i.e., $MT_{BIT} \equiv 0$.

In this group of experiments, we adopt the *"insert if missed"* replacement strategy. The off-the-shelf strategy that we marked *traditional* in the evaluation, takes whole TCAM as integration, and BIT divides TCAM into several blocks. By setting different TCAM total capacity and protection period, we obtain the experimental results of the Traditional and BIT.

B. BIT Analysis

In this subsection, we will present the improvement of BIT from four performance indicators: hit rate, insertion times, error insertion times, and find times. We have plotted Figure 18, Figure 19, and Figure 20 with different TCAM capacities, 200, 400, and 600, respectively. The numbers in the legend of each figure represent protection period values. And the statistical period for each result is 10K packets. Both BIT and traditional methods are based on the *"insert if missed"* replacement strategy, which uses the LRU algorithm to select the removed entries. The original size of each block in BIT is set by the statistics of the entries accessed distribution.

1) Hit Rate: A larger capacity means that TCAM can store more entries, and correspondingly, the hit rate of lookup is higher for the same caching algorithm. Therefore, in this experiment, the hit rate shows an overall increase from the capacity of 200 to 600, as shown in Figure 18(a), Figure 18(a) and Figure 20(a).

For the same capacity and PP (two lines with the same maker in different colors in one figure), although both are the basis of the identical replacement strategy, the result shows that BIT performs better than the traditional scheme. And for the same capacity and different PPs of BIT (three solid lines in one figure), the smaller the PP is, the higher its corresponding hit rate is, which brings frequent replacement operations.

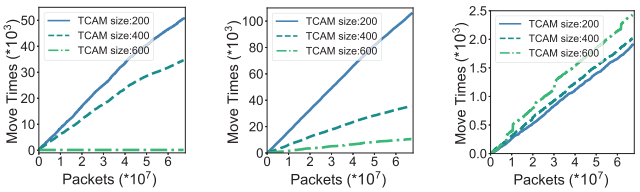
The reasons that BIT outperforms traditional ways on hit rate is elaborated as follows. In the traditional schemes, due to the entry dependency, when inserting an entry, all dependent entries should be inserted jointly, which, however, are rarely be accessed. Instead, BIT specifies the proportion of TCAM resources for each block according to the real entry access distribution. Due to the limitation of block size, these dependent entries are not allowed to occupy other blocks, which means that some of these dependent entries fail to be inserted, thus ensuring that TCAM resources are fully utilized. Finally, BIT improves the overall hit rate of TCAM.

2) Insertion Times: The results are presented in Figure 18(b), Figure 18(b), and Figure 20(b). It can be seen that the insertion times of BIT is much smaller than that of the traditional method with the same capacity and PP. Overall, this result is because BIT has a higher hit rate, then fewer insertion operations are triggered. Since the insertion operation corresponds to the write operation in the TCAM update process, fewer write operations mean less write time consumption and less computation consumption of entries relationships. In particular, at a capacity of 600 (the y-axis presents logarithm value of IT), BIT over performs the traditional more than 1,000 times. For BIT, the IT decreases across orders of magnitude as the capacity increases.

3) Error Insertion Times: The results of EIT for different TCAM capacities and PP are shown in Figure 18(c), Figure 18(c), and Figure 20(c). Although this metric does not directly reflect the performance, it is still necessary to analyze the causes and effects of error insertion, especially on the hit rate. First of all, the error insertion occurs because it is not possible to vacate a suitable space for the entry to be inserted even by moving under the mechanism of the protection period. So in the case of $PP = 1$ for traditional, no error insertion is generated because the number of dependent entries of one inserted entry does not exceed the total capacity of TCAM in the experiment. But for BIT, even if $PP = 1$, error insertion will occur when the number of dependent entries with the same prefix length of one inserted entry exceeds the corresponding block size. In a vertical comparison, taking PP as the only variable, as PP increases then a larger EIT is generated. If a relatively high hit rate is maintained, which corresponds to a more stable IT, it will, in turn, make the EIT stable at a relatively small value, as shown in Figure 20(c).

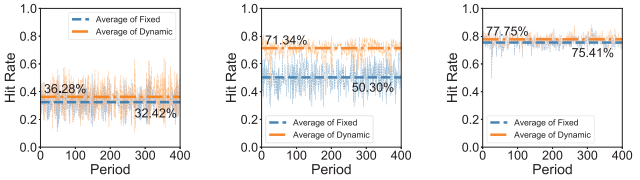
In addition, since BIT prevents dependent entries with low or even zero access frequency from being stored in TCAM more strictly than the traditional case, i.e., BIT reserves valuable resources for the entries with high access frequency. Therefore, under the same TCAM resources and replacement strategy, BIT has a higher hit rate than the traditional case.

4) Find Times: Both BIT and the traditional case are required to find if there are suitable spaces or dependent entries that need to be deleted jointly. Although they can be based on the same find algorithm, the complexity of traversal-based algorithms is usually related to the scale N. Therefore, we count and analyze the find operations triggered when updating TCAM. For all three TCAM sizes from small to large in the experiment, as described in Figure 18(d), Figure 18(d), and Figure 20(d), BIT over-performs than traditional on FT,



(a) $PP = 1$ (b) $PP = 100$ (c) $PP = 10^4$

Fig. 21. Move times.



(a) TCAM size = 200 (b) TCAM size = 400 (c) TCAM size = 600

Fig. 22. Comparison dynamic and fixed block.

which is attributed to the fact that BIT corresponds to less ITs, and, in turn, decreases the value of FT. Ultimately, the combination of multiple factors comprehensively improves the performance of BIT.

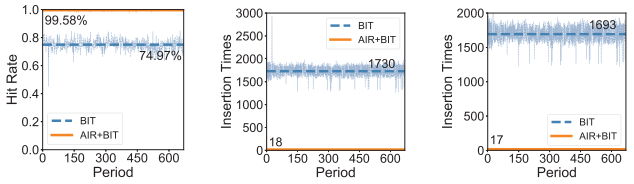
C. Move Times

The entries movement is an important impact factor of TCAM update performance in existing deployed algorithms, and theoretically, its impact is positively correlated with the entry insertion times scale. For BIT, when and only when borrowing process in dynamic mode will generate very few moves, and the MT is only related to the stable block scale, not to the increasing insertion times scale.

Since move times of the traditional are affected by PP and TCAM capacity, the MT results of the traditional update process are analyzed from these two aspects. In the $PP = 1$ case, when there is a relatively large TCAM capacity, there are enough entries that can be replaced when inserting a new entry, so few moves will be generated, as shown in Figure 21(a). The increment of PP restricts the replacement of the entries stored in TCAM, so when $PP = 100$, as shown in Figure 21(b), MT shows an increasing trend than $PP = 1$, especially when the TCAM size is relatively small. However, when $PP = 10^4$, MT shows a decrease instead, as shown in Figure 21(c). This is because the protection mechanism triggers a large number of insertion errors, which means lots of entry insertion tasks are suspended, i.e., no move operation is triggered.

D. Dynamic Block

To integrate AIR and BIT, a period-based replacement strategy is used in this set of experiments, i.e., counting information for a given length packets interval and then updating TCAM, which is consistent with AIR. Setting the statistic period window step to 10K packets, the hit rates are plotted for different TCAM sizes, 200 (Figure 22(a)), 400 (Figure 22(b)), and 600 (Figure 22(c)), respectively.



(a) Hit rate (b) Insertion times (c) Find times

Fig. 23. Performance of A&B.

The results reveal that the hit rate still follows the rule that it increases with the growing TCAM size. And no matter for which TCAM size, the dynamic scheme over-performs than the fixed. When $size = 200$, the overall hit rate is low due to the small size of the whole TCAM, which leads to a small gap between dynamic and fixed. Similarly, at $size = 600$, the TCAM size is relative large enough, the dynamic and the fixed keep a high hit rate, also, with a small gap. However, at $size = 400$, the in-between size has more maneuverability, which makes the performance improvement of dynamic more obvious.

E. AIR+BIT

A&B is integrated by updating the prediction result of AIR to TCAM on the basis of the constraint of BIT. The performance improvements of A&B in terms of hit rate, insertion times, and find times are depicted in the figure by comparing to the BIT-only policy, whose protection period is 10^4 that keeps consistent with the statistical period of A&B. The TCAM size of this experiment is 600. The hit rate can be further improved by the hot entry prediction of AIR, with an average value of 98.58% for more than 600 periods, versus 74.97% for BIT-only, as shown in Figure 23(a). Due to the threshold TS set in AIR, only hot entries were selected to insert into TCAM, combined with the stability of the traffic, only a very little amount of entries were inserted each period, as shown in Figure 23(b). The average IT per 10^4 packets is 18, far better than 1,730 of BIT-only. Moreover, A&B without error insertion times, i.e., $EIT \equiv 0$ due to the selection mechanism of AIR, which constraints the number of inserted entries does not exceed the size of the corresponding block. Due to the performance improvement brought by the insertion times, it also greatly reduces the number of times to find the suitable space during updating TCAM, as shown in Figure 23(c). The average FT of A&B is 17, while BIT-only reaches 1,693.

VIII. Discussion and Conclusion

With the rapid development of technologies such as 5G/6G and IoT, more and more terminals are accessing the Internet and generating enormous traffic, which imposes huge pressures on forwarding devices, so routers have to continuously expand the capacity of TCAM to cope with the explosive growth of the number of routing entries. In order to solve the problems caused by large capacity TCAM, we have designed an AI and block-based TCAM entry replacement scheme termed A&B. AIR decouples the aggregates flows to address the accuracy challenge of prediction, and on this basis, parallelizes LSTM algorithm, which can satisfy the efficiency demand for prediction results. BIT conducts a block-based TCAM routing entry insertion tactic based on prefix length, which can eliminate the entries moves issue when inserting new entries, and compress the scale of dependent entries according to probability and stable traffic skew distribution, which greatly improves the efficiency of TCAM updates. Moreover, given that AIR and BIT are not tightly coupled, they also can provide optimization for TCAM independently. The prediction mechanism of AIR is applicable in any capacity TCAM scenario to improve the hit rate. BIT provides a strategy for TCAM item insertion from a new perspective that can be combined with the already intensively studied table entry compression optimization algorithms. As the two steps of TCAM optimization, the combination of AIR and BIT, i.e., A&B, enhances the TCAM performance. Through a series of experiments, A&B has been shown to achieve similar performance to existing strategies while using only $1/8$ TCAM capacity and eliminate the complicated entries movements during the updating process of TCAM. The improvement in capacity will be more clear in large-scale networks.

Acknowledgment

The authors would like to thank Ian Lewis, University of Cambridge, for his valuable suggestions on expression and structure of this article. This work significantly extends [67] by adding block-based entry insertion tactic for optimizing updating process of TCAM.

References

[1] (2020). *BGP Routing Table Analysis Reports*. [Online]. Available: https://bgp.potaroo.net/
[2] (2020). *Cisco*. [Online]. Available: https://www.cisco.com/
[3] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, 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 Comput. Commun. Rev.*, vol. 41, no. 4, pp. 207–218, Oct. 2011.
[5] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 193–204, Oct. 2005.
[6] P. He, W. Zhang, H. Guan, K. Salamatian, and G. Xie, "Partial order theory for fast TCAM updates," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 217–230, Feb. 2018.
[7] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenters," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 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 Comput. Commun. Rev.*, vol. 48, no. 5, pp. 64–69, Jan. 2019.
[9] R. Li, B. Zhao, R. Chen, and J. Zhao, "Taming the wildcards: Towards dependency-free rule caching with FreeCache," in *Proc. IEEE/ACM 28th Int. Symp. Qual. Service (IWQoS)*, Jun. 2020, pp. 1–10.
[10] X. Wen *et al.*, "RuleTris: Minimizing rule update latency for TCAM-based SDN switches," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 179–188.
[11] (2019). *The Caida UCSD Anonymized Internet Traces*. [Online]. Available: https://www.caida.org/data/passive/passive_dataset.xml
[12] H. Asai and Y. Ohara, "Poptrie: A compressed trie with population count for fast and scalable software ip routing table lookup," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 57–70, Aug. 2015.
[13] T. Yang *et al.*, "Guarantee IP lookup performance with FIB explosion," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 39–50.

[14] A. Sivaraman *et al.*, "Packet transactions: High-level programming for line-rate switches," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 15–28.

[15] M. Zec, L. Rizzo, and M. Mikuc, "DXR: Towards a billion routing lookups per second in software," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 29–36, Sep. 2012.

[16] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, pp. 1083–1092, Jun. 1999.

[17] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, *Small Forwarding Tables for Fast Routing Lookups*, vol. 27, no. 4. New York, NY, USA: ACM, 1997.

[18] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable high speed IP routing lookups," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 1997, pp. 25–36.

[19] A. Broder and M. Mitzenmacher, "Using multiple hash functions to improve IP lookups," in *Proc. Conf. Comput. Commun., 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2001, pp. 1454–1463.

[20] F. Pong and N.-F. Tzeng, "Concise lookup tables for IPv4 and IPv6 longest prefix matching in scalable routers," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 729–741, Jun. 2011.

[21] T. Stimpfling, N. Bélanger, J. M. P. Langlois, and Y. Savaria, "SHIP: A scalable high-performance IPv6 lookup algorithm that exploits prefix characteristics," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1529–1542, Aug. 2019.

[22] L. Liu, J. Hu, Y. Yan, S. Gao, T. Yang, and X. Li, "Longest prefix matching with pruning," in *Proc. IEEE 20th Int. Conf. High Perform. Switching Routing (HPSR)*, May 2019, pp. 1–6.

[23] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using Bloom filters," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2003, pp. 201–212.

[24] J. Hasan, S. Cadambi, V. Jakkula, and S. Chakradhar, "Chisel: A storage-efficient, collision-free hash-based network processing architecture," in *Proc. 33rd Int. Symp. Comput. Archit. (ISCA)*, 2006, pp. 203–215.

[25] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using bloom filters," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 397–409, Apr. 2006.

[26] H. Yu, R. Mahapatra, and L. Bhuyan, "A hash-based scalable IP lookup using Bloom and fingerprint filters," in *Proc. 17th IEEE Int. Conf. Netw. Protocols*, Oct. 2009, pp. 264–273.

[27] H. Song, F. Hao, M. Kodialam, and T. V. Lakshman, "IPv6 lookups using distributed and load balanced Bloom filters for 100Gbps core router line cards," in *Proc. 28th Conf. Comput. Commun.*, Apr. 2009, pp. 2518–2526.

[28] H. Lim, K. Lim, N. Lee, and K.-H. Parl, "On adding bloom filters to longest prefix matching algorithms," *IEEE Trans. Comput.*, vol. 63, no. 2, pp. 411–423, Feb. 2012.

[29] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-accelerated software router," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 195–206, 2010.

[30] J. Zhao, X. Zhang, X. Wang, and X. Xue, "Achieving O(1) IP lookup on GPU-based software routers," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 429–430.

[31] Y. Go, M. A. Jamshed, Y. Moon, C. Hwang, and K. Park, "APUNet: Revitalizing GPU as packet processing accelerator," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 83–96.

[32] H. Byun, Q. Li, and H. Lim, "Vectored-Bloom filter implemented on FPGA for IP address lookup," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2019, pp. 1–4.

[33] K. Huang, G. Xie, Y. Li, and A. X. Liu, "Offset addressing approach to memory-efficient IP address lookup," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 306–310.

[34] Y.-H.-E. Yang, Y. Qu, S. Haria, and V. K. Prasanna, "Architecture and performance models for scalable IP lookup engines on FPGA," in *Proc. IEEE 14th Int. Conf. High Perform. Switching Routing (HPSR)*, Jul. 2013, pp. 156–163.

[35] M. Meribout and M. Motomura, "A new hardware algorithm for fast IP routing targeting programmable routers," in *Proc. Int. Conf. Netw. Control Eng. QoS, Secur. Mobility*. Cham, Switzerland: Springer, 2003, pp. 164–179.

[36] H. Fadishei, M. S. Zamani, and M. Sabaei, "A novel reconfigurable hardware architecture for IP address lookup," in *Proc. Symp. Archit. Netw. Commun. Syst.*, 2005, pp. 81–90.

[37] R. Sangireddy, N. Futamura, S. Aluru, and A. K. Somani, "Scalable, memory efficient, high-speed IP lookup algorithms," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 802–812, Aug. 2005.

[38] H. Song, M. Kodialam, F. Hao, and T. V. Lakshman, "Scalable IP lookups using shape graphs," in *Proc. 17th IEEE Int. Conf. Netw. Protocols*, Oct. 2009, pp. 73–82.

[39] F. Baboescu, D. M. Tullsen, G. Rosu, and S. Singh, "A tree based router search engine architecture with single port memories," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, 2005, pp. 123–133.

[40] H. Le, W. Jiang, and V. K. Prasanna, "A SRAM-based architecture for trie-based IP lookup using FPGA," in *Proc. 16th Int. Symp. Field-Programmable Custom Comput. Mach.*, Apr. 2008, pp. 33–42.

[41] D. Pao, Z. Lu, and Y. H. Poon, "IP address lookup using bit-shuffled trie," *Comput. Commun.*, vol. 47, pp. 51–64, Jul. 2014.

[42] V. C. Ravikumar, R. N. Mahapatra, and L. N. Bhuyan, "EaseCAM: An energy and storage efficient TCAM-based router architecture for IP lookup," *IEEE Trans. Comput.*, vol. 54, no. 5, pp. 521–533, May 2005.

[43] K. Kogan, S. I. Nikolenko, O. Rottenstreich, W. Culhane, and P. Eugster, "Exploiting order independence for scalable and expressive packet classification," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 1251–1264, Apr. 2015.

[44] C. Zhang *et al.*, "OBMA: Minimizing bitmap data structure with fast and uninterrupted update processing," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–6.

[45] V. Demianiuk, S. Nikolenko, P. Chuprikov, and K. Kogan, "New alternatives to optimize policy classifiers," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1088–1101, Jun. 2020.

[46] A. X. Liu, C. R. Meiners, and E. Torng, "TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 490–500, Apr. 2009.

[47] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to TCAM-based packet classification," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 237–250, Feb. 2010.

[48] C. R. Meiners, A. X. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, Apr. 2012.

[49] J.-P. Sheu, W.-T. Lin, and G.-Y. Chang, "Efficient TCAM rules distribution algorithms in software-defined networking," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 854–865, Jun. 2018.

[50] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Rule-caching algorithms for software-defined networks," Tech. Rep., 2014. Accessed: Aug. 1, 2021. [Online]. Available: http://www.cs.princeton.edu/~nkatta/papers/cacheflow-long14.pdf

[51] J. P. Sheu and Y. C. Chuo, "Wildcard rules caching and cache replacement algorithms in software-defined networking," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 19–29, Mar. 2016.

[52] Y. Wan *et al.*, "T-cache: Dependency-free ternary rule cache for policy-based forwarding," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 536–545.

[53] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-aware rule-caching for software-defined networks," in *Proc. Symp. SDN Res.*, Mar. 2016, pp. 1–12.

[54] B. Zhao, R. Li, J. Zhao, and T. Wolf, "Efficient and consistent TCAM updates," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1241–1250.

[55] Y. Wan, H. Song, Y. Xu, C. Zhang, Y. Wang, and B. Liu, "Adaptive batch update in TCAM: How collective optimization beats individual ones," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[56] Z. Ding, X. Fan, J. Yu, and J. Bi, "Update cost-aware cache replacement for wildcard rules in software-defined networking," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 457–463.

[57] 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 *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 918–928.

[58] D. Shah and P. Gupta, "Fast updating algorithms for TCAM," *IEEE Micro*, vol. 21, no. 1, pp. 36–47, Jan. 2001.

[59] S. Higuchi, J. Takemasa, Y. Koizumi, A. Tagami, and T. Hasegawa, "Feasibility of longest prefix matching using learned index structures," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 48, no. 4, pp. 45–48, May 2021.

[60] E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 256–269.

[61] A. Rashelbach, O. Rottenstreich, and M. Silberstein, "A computational approach to packet classification," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 542–556.

[62] Q. He, A. Moayyedi, G. Dan, G. P. Koudouridis, and P. Tengkvist, "A meta-learning scheme for adaptive short-term network traffic prediction," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2271–2283, Oct. 2020.
[63] Y. Zhang *et al.*, "AutoSight: Distributed edge caching in short video network," *IEEE Netw.*, vol. 34, no. 3, pp. 194–199, May 2020.
[64] N. Sarrar, S. Uhlig, A. Feldmann, R. Sherwood, and X. Huang, "Leveraging Zipf's law for traffic offloading," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 1, pp. 16–22, Jan. 2012.
[65] (2021). *University of Oregon Route Views Project*. [Online]. Available: http://www.routeviews.org/routeviews/
[66] (2021). *Routeviews Prefix to as Mappings Dataset for IPV4 and IPV6*. [Online]. Available: https://www.caida.org/catalog/datasets/routeviews-prefix2as/
[67] Y. Zhang, P. Cong, B. Liu, W. Wang, and K. Xu, "AIR: An AI-based TCAM entry replacement scheme for routers," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Service (IWQOS)*, Jun. 2021, pp. 1–10.

Peizhuang Cong is currently pursuing the Ph.D. degree with the State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include next generation network architecture, data-driven networks, and mobile internet.

Yuchao Zhang (Member, IEEE) received the B.S. degree in computer science and technology from Jilin University in 2012 and the Ph.D. degree from the Department of Computer Science, Tsinghua University, in 2017. She is currently an Associate Professor with the Beijing University of Posts and Telecommunications, Beijing, China, and a Visiting Scholar with the University of Cambridge, where she is also a Research Associate at the Wolfson College. Her research interests include large scale datacenter networks, federated learning, data-driven networks, and edge computing. She is a member of ACM.

Bin Liu (Senior Member, IEEE) received the M.E. and Ph.D. degrees in computer science and engineering from Northwestern Polytechnical University, Xi'an, China, in 1988 and 1993, respectively. He is currently a Full Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include high-performance switches/routers, network processors, high-speed security, and greening the internet. He has received numerous awards from China, including the Distinguished Young Scholar of China and won the inaugural Applied Network Research Prize sponsored by ISOC and IRTF in 2011.

Wendong Wang (Member, IEEE) received the B.E. and M.E. degrees from the Beijing University of Posts and Telecommunications, China, in 1985 and 1991, respectively. He is currently a Full Professor with the State Key Laboratory of Networking and Switching Technology. He has published over 200 papers in various journals and conference proceedings. His current research interests include next generation network architecture, network resources management and QoS, and mobile internet.

Zehui Xiong (Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore. He was a Visiting Scholar at Princeton University and the University of Waterloo. He is currently an Assistant Professor with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design. Prior to that, he was a Researcher with the Alibaba-NTU Joint Research Institute, Singapore. He has published more than 100 research papers in leading journals and flagship conferences and four of them are ESI Highly Cited Papers. His research interests include wireless communications, network games and economics, blockchain, and edge intelligence. He has won five Best Paper Awards in international conferences and technical committee. He was a recipient of the Chinese Government Award for Outstanding Students Abroad in 2019 and the NTU SCSE Best Ph.D. Thesis Runner-Up Award in 2020. He is serving as an editor or a guest editor for many leading journals, including IEEE TRANSACTIONS. He is the Founding Vice Chair of Special Interest Group on Wireless Blockchain Networks in IEEE Cognitive Networks Technical Committee.

Ke Xu (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University. He is currently a Full Professor at Tsinghua University. His research interests include next generation internet, P2P systems, the Internet of Things, network virtualization, and network economics. He is a member of ACM. He serves as an Associate Editor for IEEE INTERNET OF THINGS JOURNAL. He has guest edited several special issues in IEEE and Springer journals.