Adaptive Level-Aware Sketch for Efficient Traffic Measurement in Software Switches

Fuliang Li, *Member, IEEE*, Kejun Guo, Yuting Liu, Jiaxing Shen, *Member, IEEE*, Xingwei Wang, *Member, IEEE*, and Jiannong Cao, *Fellow, IEEE*

***Abstract*—Traffic measurement in software switches is critical for effective network management. Sketch has been proven to be a promising solution for traffic measurement in software switches. Considering the skewed distribution of network traffic, where low-frequency mouse flows dominate and high-frequency elephant flows are fewer, recent sketch-based solutions employ hierarchical designs to enhance memory efficiency and accuracy. However, these solutions inevitably introduce additional challenges, including increased memory access overhead, severe hash collisions between elephant and mouse flows, and limited adaptability to dynamic network environments. In this paper, we propose LA-Sketch, an adaptive level-aware data structure designed for efficient traffic measurement in software switches. First, LA-Sketch employs a level-aware classifier to intelligently map each flow to its corresponding level, thereby reducing memory access overhead caused by hierarchical designs and mitigating hash collisions between elephant and mouse flows. Second, we introduce an adaptive counter configuration method that dynamically adjusts the number of counters at each level according to diverse network traffic distributions, which theoretically minimizes overall hash collisions. Finally, to adapt to the continuously changing network traffic characteristics, we propose an adaptive online training method that enables LA-Sketch's classifier to maintain high performance using only sketch query values for training, avoiding the significant overhead of massive traffic data collection. Extensive evaluations on two real-world network traces across five measurement tasks demonstrate that LA-Sketch outperforms state-of-the-art hierarchical sketches.**

***Index Terms*—Sketch, hierarchical designs, network traffic measurement, software switches.**

Received 2 March 2026; revised 7 June 2026; accepted 30 June 2026. Date of publication 3 July 2026; date of current version 11 August 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62572105 and Grant U22B2005, and in part by LiaoNing Revitalization Talents Program under Grant XLYC2403086. An earlier version of this paper was presented in part at the 2025 IEEE/ACM International Symposium on Quality of Service (IWQoS'25) [DOI: 10.1109/IWQoS65803.2025.11143309]. Recommended for acceptance by Y. Hu. *(Corresponding author: Jiaxing Shen.)*

Fuliang Li, Kejun Guo, Yuting Liu, and Xingwei Wang are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: lifuliang@cse.neu.edu.cn; kejunguo@163.com; 2301921@stu.neu.edu.cn; wangxw@mail.neu.edu.cn).

Jiaxing Shen is with the Division of Artificial Intelligence, Lingnan University, Hong Kong (e-mail: jiaxingshen@LN.edu.hk).

Jiannong Cao is with the School of Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csjcao@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TC.2026.3709859

I. Introduction

A. Background and Motivation

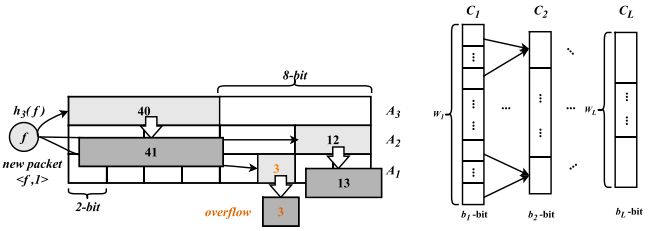
NETWORK traffic measurement is fundamental to various network management applications, including traffic billing, congestion control, anomaly detection, and so on [2], [3], [4], [5], [6], [7]. These applications depend on core measurement tasks such as flow size estimation, heavy-hitter detection, and entropy estimation. Since most data centers have not yet fully deployed programmable switches and such switches are constrained by limited resources, the majority of measurement tasks are still performed on dedicated servers or end hosts. Therefore, this paper focuses on software-based measurement solutions rather than switch-based approaches.

Sketch-based network traffic measurement solutions have been widely adopted due to their ability to achieve high accuracy with low memory overhead. Sketch is a probabilistic data structure that uses hash functions to map flows into buckets, which can be bits, counters, or key-value pairs. For instance, the widely used Count-Min (CM) Sketch [8] consists of k equal-length counter arrays, each associated with a hash function. When inserting a flow, CM Sketch maps it to k counters using k hash functions and increments each counter by one. When querying a flow, it checks the k mapped counters and reports the minimum value among them.

Although the CM Sketch achieves marginally higher memory efficiency, both its memory efficiency and accuracy remain constrained. To address this, recent solutions have considered the skewed characteristics of network traffic and adopted hierarchical designs to improve both memory efficiency and accuracy. However, these solutions introduce additional challenges. For example, Tower Sketch [9] and FCM Sketch [10] represent two state-of-the-art hierarchical sketches that exemplify these advancements and associated limitations.

Tower Sketch [9], as shown in Fig. 1(a), differs from the CM Sketch by using counter arrays with varying bit-widths. Lower-level arrays contain more counters with smaller sizes, while higher-level arrays have fewer counters with larger sizes. The insertion algorithm of Tower Sketch is consistent with that of CM/CU Sketch [8], [11]. While Tower Sketch improves memory efficiency, it suffers from three significant limitations:

- All flows are inserted into every level rather than being directed to their corresponding levels, leading to hash collisions between elephant and mouse flows.



(a) Tower Sketch. (b) FCM Sketch.

Fig. 1. Overview of Tower Sketch [9] and FCM Sketch [10].

- Due to the large counter values of elephant flows, their effective counts tend to appear only in the higher-level layers of the sketch. As a result, they suffer from higher estimation errors compared to the CM Sketch.
- How to configure the number of counters optimally remains an open question. Tower Sketch uses a fixed $2:1$ ratio between layers, which may not accommodate diverse network traffic distributions effectively.

FCM Sketch [10], as shown in Fig. 1(b), is similar to Tower Sketch in that lower-level arrays contain more counters with smaller sizes, while higher-level arrays have fewer counters with larger sizes. However, unlike Tower Sketch, the insertion algorithm of FCM Sketch starts with CM insertion at the lowest level and progressively moves to higher levels upon overflow. While FCM Sketch also achieves high memory efficiency, it suffers from three significant limitations:

- Since all flows are initially inserted at the lowest level and progressively moved to higher levels upon overflow, severe hash collisions occur between elephant and mouse flows.
- For elephant flows, each insertion requires sequential memory accesses from the lowest level to the corresponding higher-level array, resulting in excessive memory access overhead.
- How to configure the number of counters optimally remains an open question. FCM Sketch simply sets different proportions for each level and continuously tests to obtain the optimal configuration.

In summary, an ideal hierarchical sketch should achieve high memory efficiency while intelligently mapping flows directly to the corresponding level. This approach reduces hash collisions between elephant and mouse flows, as well as the overhead of sequential memory accesses. Additionally, it should automatically provide the optimal counter number configuration according to diverse network traffic distributions.

B. Proposed Solution and Contributions

In this paper, we propose the Level-Aware Sketch (LA-Sketch) to address the aforementioned challenges. LA-Sketch approximates the ideal hierarchical sketch through three key components: a level-aware classifier, an adaptive counter configuration method, and an adaptive online training method. Specifically, our contributions are as follows:

1) **LA-Sketch** (§III-B): We propose LA-Sketch, a novel level-aware hierarchical sketch designed for efficient traffic measurement in software switches. LA-Sketch employs a level-aware classifier to intelligently map each flow to its corresponding level and perform insertion, significantly reducing hash collisions between elephant and mouse flows as well as the number of sequential memory accesses.
2) **Adaptive counter configuration and online training methods** (§III-C and §III-D): We introduce an adaptive counter number configuration method, which is theoretically proven to minimize the overall hash collisions in LA-Sketch. Additionally, to adapt to the continuously changing network traffic characteristics, we propose an adaptive online training method that continuously improves the LA-Sketch's classifier using only the query values, without the need to collect massive network traffic data.
3) **Generalize to other sketches** (§V): The three key components of LA-Sketch can also be applied to existing hierarchical sketches. By incorporating the level-aware classifier, the adaptive counter configuration method, and the adaptive online training method into two state-of-the-art hierarchical sketches (Tower Sketch [9] and FCM Sketch [10]), we achieve further performance improvements.
4) **Extensive experimental verification** (§VI): We conduct extensive experiments on two real-world network traces across five measurement tasks to evaluate LA-Sketch. Experimental results demonstrate that LA-Sketch outperforms the state-of-the-art hierarchical sketches. The adaptive counter configuration method enhances LA-Sketch's performance, while the adaptive online training method achieves results comparable to traditional online training method.

II. Related Work

A. Sketch

Sketch-based solutions can be classified into two categories: simple sketches and complex sketches. Simple sketches typically consist of multiple arrays of counters, each associated with a hash function. These sketches assign uniform bit widths to all counter arrays, leading to low memory efficiency. Moreover, as they equally treat elephant and mouse flows, the accuracy of these sketches is poor due to hash collisions. Representative examples of simple sketches include CM Sketch [8], CU Sketch [11]. In contrast, complex sketches use advanced algorithms to separate the storage of elephant and mouse flows, thereby achieving higher memory efficiency and accuracy. Representative examples of complex sketches include HeavyGuardian [12], OneSketch [13], Elastic Sketch [14], and so on [15]. However, their hierarchical granularity remains relatively coarse, leaving room for further improvements in memory efficiency. For instance, Elastic Sketch uses a simple two-tier structure.

B. Hierarchical Sketch

Recent approaches exploit the skewness of network traffic to enhance memory efficiency and accuracy through hierarchical designs, as exemplified by Cold Filter [16], Tower Sketch

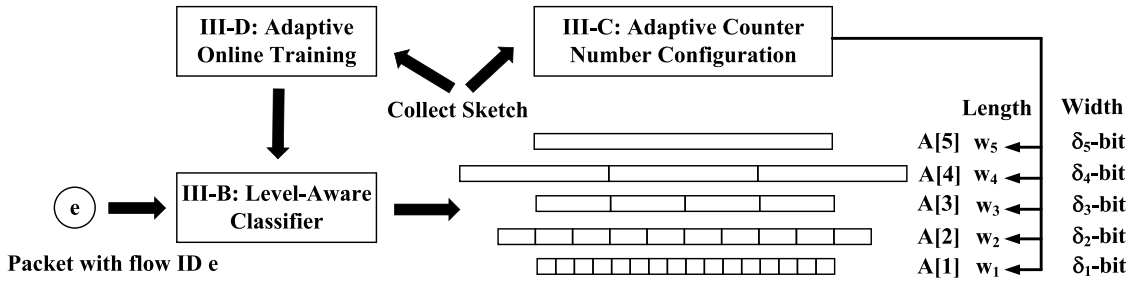


Fig. 2. Overview of LA-Sketch.

[9], FCM Sketch [10] and so on [17]. These works allocate counters of varying sizes and quantities across different levels, with lower-level arrays generally containing more counters of smaller size, and higher-level arrays containing fewer counters of larger size. While hierarchical designs improve memory efficiency and accuracy, they inevitably introduce additional challenges, including increased memory access overhead, severe hash collisions between elephant and mouse flows, and limited adaptability to dynamic network environments.

C. Learning-Enhanced Sketch

In recent years, several works have emerged that enhance sketches using machine learning [18], [19], [20], [21], [22]. Among these, the most relevant are those focusing on learning flow frequencies [18], [19] to improve accuracy by avoiding collisions between elephant and mouse flows. However, learning the exact frequency of each flow is both complex and unnecessary. In contrast, our work learns the level associated with each flow rather than its frequency and introduces new techniques, including an adaptive counter configuration method and an adaptive online training method.

III. LA-Sketch

In this section, we first provide an overview of LA-Sketch. Then, we present a detailed explanation of its three key components: the level-aware classifier, the adaptive counter configuration method, and the adaptive online training method.

A. Overview

As shown in Fig. 2, in addition to the hierarchical structure, LA-Sketch consists of three novel components: the level-aware classifier, the adaptive counter configuration method, and the adaptive online training method. The primary functions of these components are as follows:

1) **Adaptive Counter Configuration:** Before performing measurement tasks, for a given memory size, we need to configure the number of counters in each level of LA-Sketch. For a given network traffic distribution, the adaptive counter configuration method in LA-Sketch determines the optimal number of counters per level, which can minimize the overall hash collisions in LA-Sketch and optimize its performance.
2) **Level-Aware Classifier:** When inserting a packet with flow ID e, LA-Sketch employs its pre-trained level-aware classifier to directly map e to its corresponding level. At this level, the CM/CU [8], [11] insertion operation is executed. If overflow occurs, the packet is incrementally inserted into higher levels. This approach significantly reduces hash collisions between elephant and mouse flows and minimizes the number of memory accesses required across levels.
3) **Adaptive Online Training:** Network traffic characteristics evolve over time, leading to shifts where mouse flows may become elephant flows and vice versa. Moreover, a substantial number of flows emerge that were absent in previous periods. Such changes can degrade the performance of the level-aware classifier. To address this, LA-Sketch periodically queries flow frequency information to perform adaptive online training. This process leverages approximate query values from LA-Sketch, eliminating the need for massive traffic data collection. It is worth noting that the query values provided by sketches are approximations rather than exact values. For instance, when the memory allocated to LA-Sketch is relatively small, the average relative error (ARE) may be as high as several times. ARE is defined as the ratio of the absolute difference between the query value and the actual value to the actual value. For example, if the query value is 5 and the actual value is 1, the ARE is 4. Section III-D provides a detailed explanation of how LA-Sketch's classifier can maintain robust performance even when trained with highly approximate query values rather than accurate traffic data.

B. Data Structure and Operations

Data Structure: Similar to previous hierarchical sketches, in LA-Sketch, the lower-level arrays contain more counters with smaller sizes, while higher-level arrays have fewer counters with larger sizes. LA-Sketch consists of d arrays, $A[1]$, ..., $A[d]$. Each array $A[i]$ contains w_i counters and is associated with k hash functions $h_{ij}(\cdot)$ $(1 \leq j \leq k)$. The size of each counter in array $A[i]$ is δ_i bits. For instance, in the example shown in Fig. 2, $d = 5$, and the counter widths δ_1, δ_2, δ_3, δ_4, δ_5 are 2, 4, 8, 16, and 32, respectively.

CM Insertion: As shown in Alg. 1, when inserting a packet with flow ID e, LA-Sketch uses its level-aware classifier to map

Algorithm 1: Insertion

```
lev ← Classifier(e);
min_value ← A[lev].insert(e);
while min_value ≥ 2^{δ_lev} − 1 do
    lev ← lev + 1;
    min_value ← A[lev].insert(e);
end
```

Algorithm 2: Query

```
query_value ← 0;
lev ← Classifier(e);
min_value ← A[lev].query(e);
while min_value ≥ 2^{δ_lev} − 1 do
    query_value ← query_value + 2^{δ_lev} − 2;
    lev ← lev + 1;
    min_value ← A[lev].query(e);
end
query_value ← query_value + min_value;
return query_value;
```

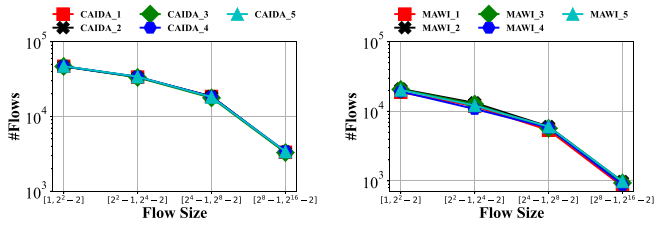
e to its corresponding level lev, and then simply increments the k hashed counters in array $A[lev]$ at level lev by 1. If overflow occurs, the packet is forwarded to the next higher-level array, repeating this process until at least one counter does not overflow. It is worth noting that if a counter overflows upon increment, it is marked as an overflow counter and its value is set to $2^{\delta_i} - 1$. That is, for a δ_i-bit counter, the maximum value it can record is $2^{\delta_i} - 2$.

CU Insertion: LA-Sketch can employ the CU insertion to improve accuracy. Instead of incrementing all k hashed counters, CU insertion increments only the smallest non-overflowed counter among them.

Query: As shown in Alg. 2, when querying a flow with flow ID e, LA-Sketch first employs the level-aware classifier to identify the corresponding level lev. The query then retrieves the minimum value among the k hashed counters in array $A[lev]$. If this minimum value is less than $2^{\delta_i} - 1$, the query process terminates. Otherwise, LA-Sketch accumulates the effective count, $2^{\delta_i} - 2$, from array $A[lev]$ and proceeds to query the next higher-level arrays. This process continues, accumulating effective counts, until the minimum value of the k hashed counters at a level is less than $2^{\delta_i} - 1$.

Level-Aware Classifier: The corresponding level for each flow can be derived from historical traffic data, which is easily obtained. For instance, a CM Sketch can be used to query flow frequency information, as demonstrated in our experiments in Section VI-D. Section III-D explains the feasibility of using approximate sketch query values for this purpose. The level-aware classifier is trained using flow IDs as features and their corresponding levels as labels. Based on the learned features, this classifier can intelligently determine the level for each flow. The level-aware classifier employs a Multi-Layer Perceptron (MLP) to model the relationship between flows and their levels. The MLP incorporates batch normalization, employs the ReLU activation function, and uses the Adam optimizer. Cross-entropy is chosen as the loss function due to its widespread application in classification tasks. The model's parameters include a random seed of 42, a learning rate of 10^{-3}, and an input layer vector length determined by the flow ID. For example, if the flow ID corresponds to the source IP, the input layer vector length is 32. In other words, we use its 32-bit binary embedding representation as input, such as $(0, 0, 1, 0, \ldots, 0, 0, 1)$. We also experimented with directly using its 32-bit integer encoding as input, but the performance is significantly worse than the binary embedding representation. We believe this is because the 32-bit binary embedding maximizes the differentiation between IPs and enhances the features of IP prefixes. The output layer vector length matches the number of levels, while the number and size of hidden layers are adapted to the complexity of the data. Although this paper employs an MLP as an example, other machine learning models, such as LSTM [23], can also be used. However, the focus of this work is not on the choice of machine learning models but rather on the functionality of the classifier. Furthermore, we advise against adopting the large language models (LLMs) for two reasons: 1) One of the key advantages of sketch lies in its memory efficiency. When a lightweight model such as an MLP is employed, LA-Sketch maintains superior memory efficiency compared to existing sketches, even accounting for parameter storage overhead. However, adopting an LLM would fundamentally compromise this efficiency due to its massive parameter count; 2) Network traffic measurement imposes stringent throughput requirements. Compared to LLMs, lightweight models such as MLPs incur substantially lower computational overhead, enabling the processing of a greater number of packets per unit time—and thus achieving higher throughput—under equivalent GPU resources.

Discussion: 1) Compared to FCM Sketch, the level-aware classifier in LA-Sketch effectively eliminates hash collisions between elephant and mouse flows, as well as excessive memory accesses caused by inserting flows sequentially from the lowest level. Compared to Tower Sketch, LA-Sketch addresses both hash collisions and higher estimation errors for elephant flows. Some may argue that the multiple hashing mechanisms in FCM Sketch and Tower Sketch already mitigate hash collisions between elephant and mouse flows. However, we emphasize that FCM Sketch indiscriminately inserts all flows from the lowest level upwards, and Tower Sketch hashes all flows to every level, which inevitably leads to hash collisions between elephant and mouse flows. Multiple hashing only alleviates these collisions, whereas the level-aware classifier in LA-Sketch fundamentally prevents elephant and mouse flows from being hashed to the same level. 2) In the experiments detailed in Section VI-B, the memory overhead introduced by the level-aware classifier is taken into account. Despite this, LA-Sketch consistently outperforms hierarchical sketches such as FCM Sketch and Tower Sketch, demonstrating superior accuracy.



(a) CAIDA. (b) MAWI.

Fig. 3. The flow size distributions of five consecutive periods from CAIDA [24] and MAWI [25] datasets.

C. Adaptive Counter Configuration Method

Our Observation: For a given memory size M, how to set the length w_i of each array $A[i]$ is an open challenge. Existing solutions, such as Tower Sketch and FCM Sketch, allocate counters based on predefined ratios. Tower Sketch allocates equal memory to each array, resulting in a $2:1$ counter ratio between adjacent levels. FCM Sketch employs a fixed counter ratio (e.g., $16:1$ or $8:1$), testing various configurations to identify the optimal setup. While such approaches may yield good performance under specific network traffic distributions, they lack adaptability to diverse traffic distributions and fail to consistently optimize performance. This leads a critical question: for a given network traffic distribution, is there an optimal configuration? To investigate, we analyze network traffic data sampled over five consecutive measurement periods, each consisting of approximately 5 million packets, from CAIDA [24] and MAWI [25] datasets. As shown in Fig. 3, we illustrate the flow size distribution curves for these five periods. It is evident that the flow size distribution remains relatively stable for specific regional data centers or backbone networks. Furthermore, considering that LA-Sketch can directly map flows to their corresponding levels, the problem can be reformulated: given a fixed memory size M and N_i flows at each level, is there an optimal counter number configuration to minimize hash collisions and maximize performance?

Adaptive Counter Configuration: With the goal of minimizing the overall hash collisions in LA-Sketch, we formulate the problem as an optimization task and solve it using the method of Lagrange multipliers, leading to the following result:

Theorem 1: Consider an LA-Sketch comprising d arrays $A[1], A[2], \ldots, A[d]$, where each counter in array $A[i]$ occupies δ_i bits and N_i flows are mapped to array $A[i]$. To minimize the overall hash collisions, the counter counts w_i and w_j of arrays $A[i]$ and $A[j]$ must satisfy:

$$\frac{w_i}{w_j} = \frac{N_i}{N_j}\sqrt{\frac{\delta_j}{\delta_i}}. \tag{1}$$

In practice, however, we adopt a modified formulation:

$$\frac{w_i}{w_j} = \frac{N_i}{N_j}. \tag{2}$$

This modification is motivated by two considerations. First, while Equation (1) minimizes the overall hash collisions, it does so at the expense of increasing collisions for elephant flows — that is, Equation (1) allocates more memory to lower levels and less to higher levels. This is because lower levels have smaller δ_i, making it more cost-effective to allocate additional memory there to reduce collisions. Nevertheless, elephant flows are generally of greater importance, as they typically require lower estimation error than mouse flows. Therefore, we decouple the memory allocation from the counter size δ_i. Second, the modified Equation (2) promotes a uniform ARE across levels. Specifically, as noted in prior work [26], most sketches impose unequal error bounds on elephant and mouse flows. In contrast, LA-Sketch maintains identical collision probabilities across levels because the ratio of counters between any two levels matches the ratio of flows mapped to them. Within each level, collisions occur among flows of the same order of magnitude, further guaranteeing equal ARE. For example, in the two lowest levels, level-1 employs 2-bit counters and level-2 employs 4-bit counters. If level-1 receives twice as many flows as level-2, the adaptive counter configuration method allocates level-1 twice as many counters. Under the Poisson collision model [27], [28], [29], the resulting collision probabilities in level-1 and level-2 are identical. Since flows within each level share the same magnitude, the standard ARE formula yields equal values across levels. Experimental validation is provided in Section VI-F. The significance of uniform ARE lies in ensuring that flows of different sizes experience equal errors. This property prevents a sketch from achieving good accuracy only for flows within a specific size range, which would otherwise reflect the estimation performance of only a subset of flows rather than the overall accuracy across all flows.

N_i can be derived from historical traffic. Thus, given a fixed memory size M, the corresponding counter count w_i for each array $A[i]$ can be determined by solving the following system of equations.

$$\begin{cases} M = \sum_{i=1}^{d} w_i \times \delta_i \\ \frac{w_i}{w_j} = \frac{N_i}{N_j} \quad \forall i, j \in [1, 2, \ldots, d] \end{cases} \tag{3}$$

Proof: We provide the proof of Theorem 1.

Step 1: Prior Knowledge.

1) In array $A[i]$, when N_i flows are randomly hashed into w_i buckets, let X denote the number of flows in a given bucket. X follows a binomial distribution with parameters N_i and $\frac{1}{w_i}$. For large N_i, X can be approximated by a Poisson distribution with $\lambda = \frac{N_i}{w_i}$. This has been proven in the previous paper SeqHash [30].
2) If X flows are hashed into the same bucket, each flow collides with the remaining $X-1$ flows, resulting in a total of $X(X-1)$ hash collisions for that bucket. It is worth noting that if you believe repeated hash collisions should not be considered, multiplying by $\frac{1}{2}$ is acceptable and does not affect the final result of Theorem 1.

Let X represent the number of flows in any given bucket. Based on the two prior knowledge above, the expected number of collisions in a single bucket of array $A[i]$ is:

$$E(X(X-1)) = E(X^2) - E(X) = \frac{N_i^2}{w_i^2}. \tag{4}$$

Thus, the total number of collisions in all w_i buckets of array $A[i]$ is:

$$\frac{N_i^2}{w_i} \tag{5}$$

Step 2: Minimizing Total Hash Collisions.

The objective is to minimize the total hash collisions across the d arrays of LA-Sketch:

$$\min \sum_{i=1}^{d} \frac{N_i^2}{w_i} \tag{6}$$

Assuming:

$$M = \sum_{i=1}^{d} w_i \times \delta_i \tag{7}$$

We apply the method of Lagrange multipliers. Introducing the Lagrange multiplier λ, the Lagrangian function is:

$$\mathcal{L} = \sum_{i=1}^{d} \frac{N_i^2}{w_i} + \lambda \left(\sum_{i=1}^{d} w_i \delta_i - M \right) \tag{8}$$

Taking the partial derivative of $\mathcal{L}$ with respect to each w_i and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial w_i} = -\frac{N_i^2}{w_i^2} + \lambda \delta_i = 0 \tag{9}$$

Solving for w_i yields:

$$w_i = \frac{N_i}{\sqrt{\lambda \delta_i}} \tag{10}$$

Finally, it follows that:

$$\frac{w_i}{w_j} = \frac{N_i}{N_j} \sqrt{\frac{\delta_j}{\delta_i}} \tag{11}$$

Discussion: In our prior conference version [1], Equation (7) was formulated as $W = \sum_{i=1}^{d} w_i$, i.e., the total number of counters was fixed. In the present version, we instead fix the total memory size. This revision is motivated by the fact that counters across different levels of LA-Sketch have varying sizes; consequently, fixing the total memory size is arguably more appropriate.

D. Adaptive Online Training Method

Our Observation: Network traffic characteristics tend to remain similar across adjacent time periods, enabling the level-aware classifier in LA-Sketch to accurately map flows to their corresponding levels. However, when the time interval becomes sufficiently long, these characteristics gradually diverge, leading to a decline in the classifier's performance. To address this issue, retraining the classifier with more recent traffic data becomes necessary. Incorporating traffic data from periods closer to the current time significantly improves LA-Sketch's performance. However, frequent data collection incurs substantial overhead. Thus, the key challenge is to enable adaptive online training while minimizing the overhead of massive traffic data collection.

Adaptive Online Training Method: The adaptive online training method leverages LA-Sketch itself to estimate flow sizes using its query values, eliminating the need for additional traffic data. This approach reduces overhead while ensuring the classifier adapts effectively to evolving traffic patterns. It is worth noting that we always use flow keys and their corresponding levels from the most recent period, while discarding historical traffic from earlier periods. This is because network traffic is inherently dynamic, and the most recent flows better reflect the characteristics of upcoming traffic. In contrast, outdated traffic may introduce noise or even degrade model performance if used for training. Furthermore, while the new model is being trained, the existing model continues to perform flow classification. Once the new model has been fully trained, it can be deployed immediately through a seamless model switch, thereby avoiding service interruption.

Analysis: The feasibility of using LA-Sketch's approximate query values for online training is justified as follows:

1) Since LA-Sketch never underestimates flow sizes, all elephant flows are accurately identified and correctly mapped to higher levels in subsequent periods.
2) Owing to its high memory efficiency, most mouse flows are also correctly identified and mapped to lower levels in subsequent periods.
3) In cases where some mouse flows collide with elephant flows and are overestimated, the adaptive counter configuration method mitigates this effect. If many such misclassifications occur, additional counters are allocated to the affected level, reducing collision probability and enabling correct estimation in subsequent periods.
4) Although persistent collisions between specific mouse and elephant flows are rare, periodic reseeding of the hash function is recommended to ensure correct identification of the affected mouse flows.

IV. Measurement Tasks

In this section, we elaborate on how LA-Sketch performs five representative measurement tasks. The explanation uses an end-host running LA-Sketch as an example.

Flow Size Estimation: estimating the size of any given flow. LA-Sketch returns the direct flow size estimate through the query algorithm in Alg. 2.

Heavy-Hitter Detection: reporting flows whose sizes are larger than a threshold Δ_h. Similar to previous hierarchical sketches, we maintain a small hash table to track heavy-hitters by recording their flow IDs. For each incoming packet, we insert it into LA-Sketch and query its estimated flow size $\hat{n}_j$. If $\hat{n}_j > \Delta_h$ and the flow is not already in the hash table, we add it to the table. To retrieve all heavy-hitters, we report all flow IDs stored in the hash table.

Heavy Change Detection: reporting flows whose sizes drastically change beyond a predefined threshold Δ_c in two adjacent time windows. We construct an LA-Sketch for each time window and use the hash table described above to maintain flows with size greater than Δ_c. For each flow recorded in the two hash tables, we calculate the difference in flow size by

querying the two LA-Sketch. If the difference exceeds Δ_c, the flow is reported as a significant change.

Flow Size Distribution Estimation: estimating the distribution of flow sizes. Similar to the design of Elastic Sketch [14], for each incoming packet, we insert it into LA-Sketch and query its flow size. For the flows in the lower-level arrays ($A[1] - A[3]$) of LA-Sketch, we maintain a distribution array $(n_0, n_1, \ldots, n_{270})$, where n_i represents the number of flows in LA-Sketch with a queried value of i. For flows in the higher-level arrays ($A[4] - A[5]$), the basic MRAC [31] algorithm is applied to each counter array in LA-Sketch. Finally, we combine the above results.

Entropy Estimation: estimating the entropy of flow sizes. For each incoming packet, we insert it into LA-Sketch and query its flow size. Based on the queried value before and after the insertion, we calculate the entropy in real-time.

V. Generalize to Other Sketches

In this section, we provide a detailed discussion on how the three key components of LA-Sketch can be integrated into existing hierarchical sketches.

A. Applying LA-Sketch to Tower Sketch [9]

Applying LA-Sketch: As shown in Fig. 2, the difference lies in replacing the data structure of LA-Sketch with that of Tower Sketch. The data structure of Tower Sketch is described in Section I-A and is not repeated here. When inserting a packet with flow ID e, the level-aware classifier directly maps e to its corresponding level level-l and performs insertions at level-l and all higher levels. This contrasts with the original Tower Sketch, which inserts indiscriminately into all levels. To preserve the property that each packet undergoes five insertions (corresponding to the five levels in Tower Sketch) for mitigating hash collision errors, l insertions are performed at level-l. For example, a flow mapped to level-2 executes two insertions exclusively at level-2. The adaptive counter configuration method accounts for these repeated insertions when determining the number of counters.

Analysis: We denote the Tower Sketch enhanced with LA-Sketch as Tower_LA. Tower_LA achieves lower estimation error than the original Tower Sketch for three main reasons: 1) The level-aware classifier directly maps each flow to its corresponding level. On the one hand, this reduces hash collisions between elephant and mouse flows. On the other hand, it significantly decreases the invalid counts in lower-level arrays, thereby reducing the collisions of lower-level flows and improving accuracy. Specifically, the original Tower Sketch inserts each packet into all levels. Since higher-level flows have higher frequencies while lower-level counters have fewer bits, higher-level flows consume a large portion of counters in lower-level arrays. For example, in the original Tower Sketch, every flow beyond level-1 consumes one level-1 counter, which overflows and becomes unusable. 2) The above issue also causes higher-level flows to retain only n (where $n \leq 5$) valid counters, because their insertions into lower-level counters are invalid due to overflow. This further increases their error, as a larger number of valid hash counters generally leads to lower estimation error. In contrast, Tower_LA performs l insertions only at the corresponding level-l, thereby avoiding this issue. 3) The adaptive counter configuration method in Tower_LA further reduces overall hash collisions, thereby improving accuracy.

B. Applying LA-Sketch to FCM Sketch [10]

Applying LA-Sketch: Similarly, as shown in Fig. 2, the difference lies in replacing the data structure of LA-Sketch with that of FCM Sketch. The data structure of FCM Sketch is described in Section I-A and is not repeated here. When inserting a packet with flow ID e, the level-aware classifier directly maps e to its corresponding level level-l and performs insertions starting from level-l, rather than starting from the lowest level and gradually overflowing to higher levels as in the original FCM Sketch. The adaptive counter configuration method is then applied to reduce overall hash collisions.

Analysis: We denote the FCM Sketch enhanced with LA-Sketch as FCM_LA. FCM_LA achieves lower estimation error than the original FCM Sketch for two reasons: 1) The level-aware classifier directly maps each flow to its corresponding level, thereby avoiding hash collisions between elephant and mouse flows caused by insertions starting from the lowest level. This prevents misidentifying mouse flows as elephant flows. Since FCM Sketch contains only three levels, the issue of invalid counts at lower levels, as observed in Tower Sketch, can be neglected. 2) The adaptive counter configuration method further reduces overall hash collisions, thereby improving accuracy.

VI. Experimental Results

In this section, we conduct extensive experiments on two real-world network traces across five measurement tasks to evaluate LA-Sketch.

A. Experimental Setup

Dataset: We use two real-world network traces to evaluate LA-Sketch, which have been widely adopted in prior studies [32], [33], [34], [35], [36], [37].

- **CAIDA:** We use anonymous IP tracking collected from CAIDA in 2018 [24]. Each trace contains approximately 2.7 million packets. In the case of aggregation by source IP, there are around $70,000$ flows. We consider ten consecutive traces, where each trace represents the network traffic data for one period.
- **MAWI:** We use the MAWI dataset [25], comprising real Internet traffic traces collected by the MAWI Working Group of the WIDE Project. Each trace contains approximately 8 million packets. In the case of aggregation by source IP, there are around $50,000$ flows. We consider ten consecutive traces, where each trace represents the network traffic data for one period.

Experimental Settings: We introduce the common experimental settings. LA-Sketch is designed with five levels, as shown in Fig. 2, with counter sizes of 2, 4, 8, 16, and 32 bits from the lowest to the highest level. The level-aware classifier

employs a 4-layer MLP with hidden layer dimensions of 128. Other model parameters are consistent with those described in Section III-B. The model is stored in half-precision, incurring a storage overhead of approximately 85 KB. As no significant performance degradation is observed relative to the full-precision counterpart, the half-precision model is adopted to optimize memory efficiency. The experiments consider model storage overhead, e.g., assuming a memory allocation of 300 KB, only 215 KB is allocated to the data structure part of LA-Sketch. It is worth noting that only model parameters require persistent storage, as runtime memory is released after execution. All sketches utilize five hash functions for packet insertion. We adopt the Bob hash function, which has been widely used in prior works [38], [39]. For heavy-hitter detection and heavy change detection tasks, the threshold is set to $\Delta_h = 500$. Other threshold values may also be adopted, depending on specific application requirements.

Abbreviations: The following are some abbreviations and their meanings.

- **LA-Sketch:** refers to the standard LA-Sketch. It utilizes the accurate traffic data from the previous period to train the level-aware classifier and applies it to predict traffic levels in the subsequent period. In addition, it employs the accurate traffic data from the previous period, together with the adaptive counter configuration method described in Section III-C, to determine the number of counters assigned to each level.
- **ILA-Sketch:** refers to the idealized version of LA-Sketch. It utilizes the accurate traffic data from the current period to train the level-aware classifier and applies it to predict traffic levels within the same period. In addition, it employs the accurate traffic data from the current period, together with the adaptive counter configuration method described in Section III-C, to determine the number of counters assigned to each level. This configuration represents the theoretical optimal performance of LA-Sketch. While this is unattainable in the networking domain due to the impossibility of predicting future traffic, it is feasible in database applications where the data requiring persistent storage is already known.
- **OLA-Sketch:** refers to the adaptive online training variant of LA-Sketch. It utilizes the approximate sketch query values from the previous period to train the level-aware classifier and applies it to predict traffic levels in the subsequent period. In addition, it employs the approximate sketch query values from the previous period, together with the adaptive counter configuration method described in Section III-C, to determine the number of counters assigned to each level.

B. Experimental Results on Accuracy

In this section, we compare the accuracy of LA-Sketch with the widely used CM Sketch [8] and two state-of-the-art hierarchical sketches, Tower Sketch [9] and FCM Sketch [10].

1) Experimental Settings: For fairness, we make the following settings. All sketches utilize 5 hash functions, matching the number of hash operations required by Tower Sketch due to its hierarchical design. The CM insertion algorithm is adopted for all sketches, as FCM Sketch supports only CM insertion. Secondly, none of the sketches use the EM algorithm for flow size distribution estimation. For FCM Sketch, the counter ratio per level is set to 8, as this configuration yields optimal performance in subsequent experiments. The other settings for Tower Sketch and FCM Sketch are consistent with their papers.

2) Accuracy on the CAIDA Dataset: The experimental results are as follows.

Flow Size Estimation [Fig. 4(a) and (b)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the AAE by 25.3, 4.68, and 1.24 times respectively; compared to CM and Tower, LA on average reduces the AAE by 17.7 and 3.26 times, respectively. Compared to CM, Tower, and FCM, ILA on average reduces the ARE by 219, 6.38, and 10.3 times respectively; compared to CM, Tower, and FCM, LA on average reduces the ARE by 46.6, 1.35, and 2.17 times, respectively.

Heavy-Hitter Detection [Fig. 4(c) and (d)]: We find that all sketches have an F1 score over 95%, but ILA and LA achieve lower errors compared to CM and Tower. Since FCM allocates a high proportion of memory to higher levels, it achieves the lowest AAE.

Heavy Change Detection [Fig. 4(e) and (f)]: Similarly, we find that all sketches have high F1 scores, but ILA and LA achieve lower errors compared to CM and Tower. Since FCM allocates a high proportion of memory to higher levels, it achieves the lowest AAE.

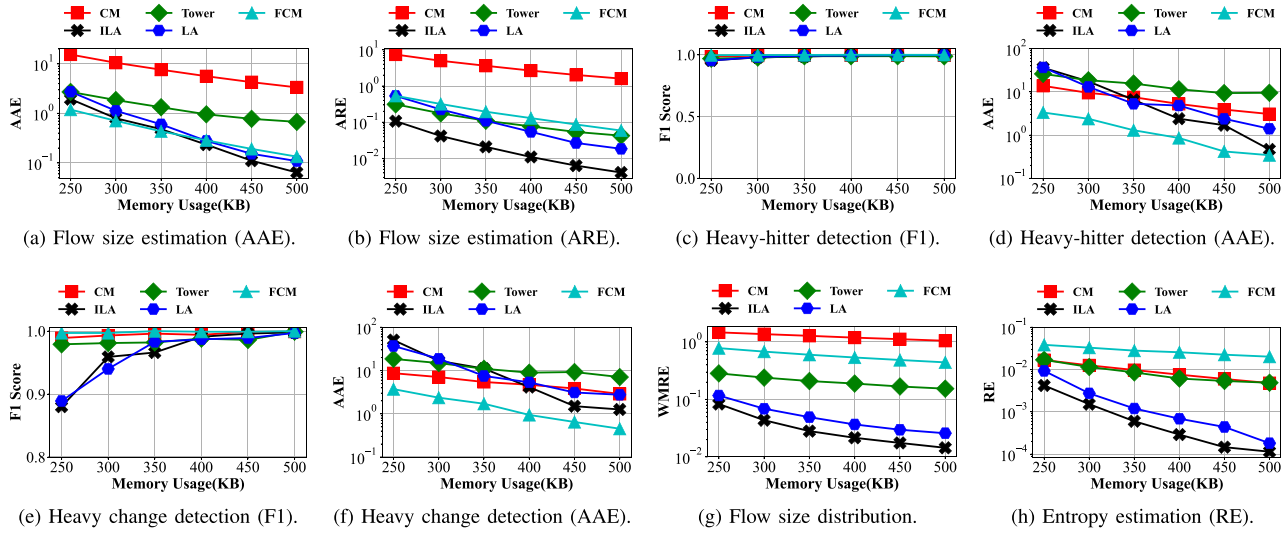
Flow Size Distribution Estimation [Fig. 4(g)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the WMRE by 47.8, 7.58, and 21.6 times respectively; compared to CM, Tower, and FCM, LA on average reduces the WMRE by 28.2, 4.49, and 12.8 times, respectively.

Entropy Estimation [Fig. 4(h)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the RE by 23, 21.1, and 83 times respectively; compared to CM, Tower, and FCM, LA on average reduces the RE by 10.9, 10.2, and 40 times, respectively.

3) Accuracy on the MAWI Dataset: The experimental results are as follows.

Flow Size Estimation [Fig. 5(a) and (b)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the AAE by 46.6, 9.54, and 1.68 times respectively; compared to CM, Tower, and FCM, LA on average reduces the AAE by 33.5, 7.16, and 1.24 times, respectively. Compared to CM, Tower, and FCM, ILA on average reduces the ARE by 391, 9.54, and 13.8 times respectively; compared to CM, Tower, and FCM, LA on average reduces the ARE by 96.8, 2.37, and 3.36 times, respectively.

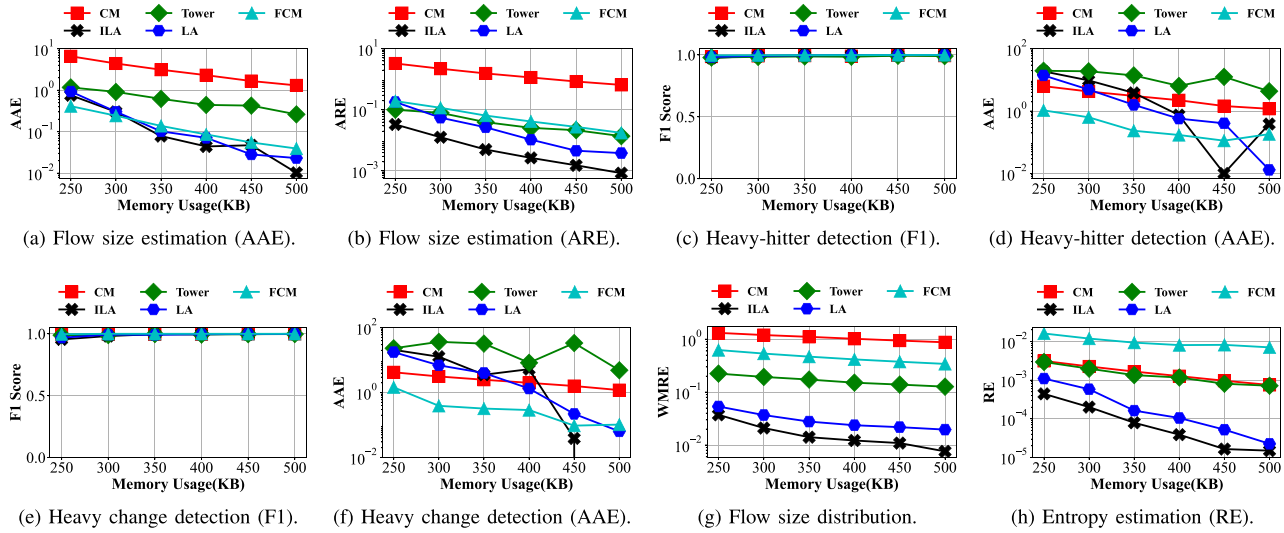
Heavy-Hitter Detection [Fig. 5(c) and (d)]: We find that all sketches have an F1 score over 95%, but ILA and LA achieve lower errors compared to CM and Tower. Since FCM allocates



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 4. Experimental results on accuracy using the CAIDA dataset.



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 5. Experimental results on accuracy using the MAWI dataset.

a high proportion of memory to higher levels, it achieves the lowest AAE.

Heavy Change Detection [Fig. 5(e) and (f)]: Similarly, we find that all sketches have high F1 scores, but ILA and LA achieve lower errors compared to CM and Tower. Since FCM allocates a high proportion of memory to higher levels, it achieves the lowest AAE.

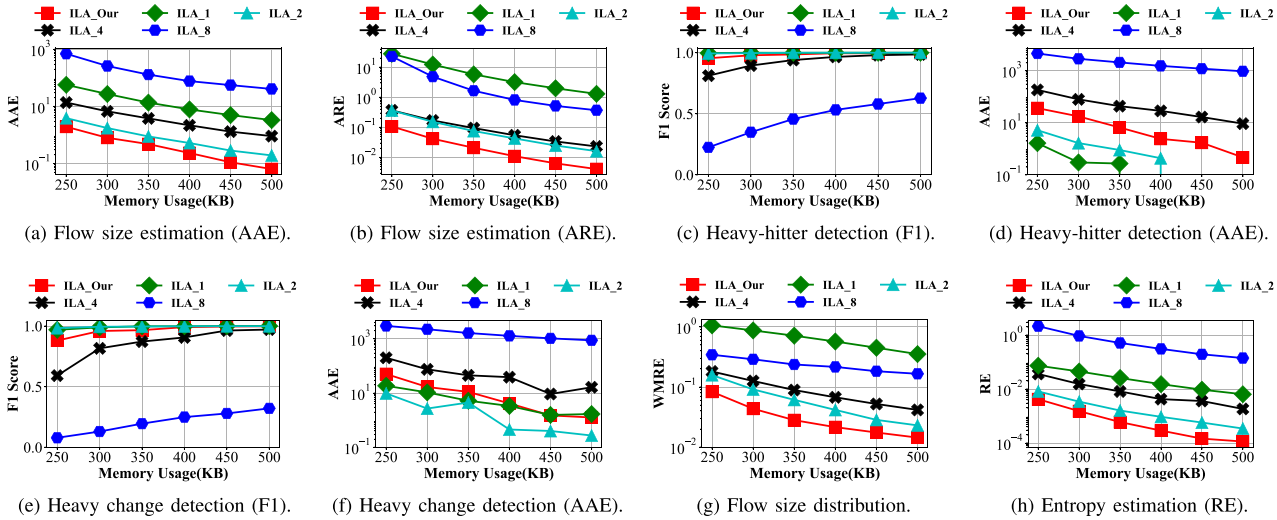
Flow Size Distribution Estimation [Fig. 5(g)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the WMRE by 76.1, 11.6, and 31.5 times respectively; compared to CM, Tower, and FCM, LA on average reduces the WMRE by 38, 5.81, and 15.9 times, respectively.

Entropy Estimation [Fig. 5(h)]: We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the RE by 30.1, 26.7, and 231 times respectively; compared to CM, Tower, and FCM, LA on average reduces the RE by 13.5, 12.1, and 106 times, respectively.

Analysis: In comparison with existing hierarchical sketches, LA-Sketch demonstrates superior accuracy across most measurement tasks. This improvement stems from its ability to map flows directly to their corresponding levels, thereby minimizing hash collisions between elephant and mouse flows. Furthermore, the proposed counter configuration method enhances LA-Sketch's performance, as corroborated by the experimental results presented below.

C. Experimental Results on the Counter Configuration Method

In this section, we evaluate the effectiveness of the proposed counter configuration method using the CAIDA dataset.



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 6. Experimental results on the counter configuration method using the CAIDA dataset.

1) Experimental Settings: The counter ratio per level in LA-Sketch is varied across values of 1, 2, 4, and 8 to compare against the proposed counter configuration method. To eliminate the potential impact of classification errors, we use ILA-Sketch for the experiments, which bypasses the classifier.

2) Accuracy: The experimental results are as follows.

Flow Size Estimation [Fig. 6(a) and (b)]: We find that ILA_Our achieves lower errors. Compared to ILA_1, ILA_2, ILA_4, and ILA_8, ILA_our on average reduces the AAE by 37, 2.31, 9.8, and 404 times, and the ARE by 292, 3.74, 4.66, and 109 times, respectively.

Heavy-Hitter Detection [Fig. 6(c) and (d)]: Compared to ILA_4 and ILA_8, ILA_Our achieves higher F1 scores and lower errors. Compared to ILA_1 and ILA_2, ILA_Our performs worse, as they have more counters in the higher level arrays.

Heavy Change Detection [Fig. 6(e) and (f)]: Compared to ILA_4 and ILA_8, ILA_Our achieves higher F1 scores and lower errors. Compared to ILA_1 and ILA_2, ILA_Our performs worse, as they have more counters in the higher level arrays.

Flow Size Distribution Estimation [Fig. 6(g)]: We find that ILA_Our achieves lower errors. Compared to ILA_1, ILA_2, ILA_4, and ILA_8, ILA_Our on average reduces the WMRE by 22.5, 1.89, 2.87, and 8.54 times, respectively.

Entropy Estimation [Fig. 6(h)]: We find that ILA_Our achieves lower errors. Compared to ILA_1, ILA_2, ILA_4, and ILA_8, ILA_Our on average reduces the RE by 44, 2.84, 14.6, and 933 times, respectively.

Analysis: In summary, the proposed counter configuration method significantly improves accuracy across most measurement tasks, confirming its effectiveness. By leveraging this method, LA-Sketch achieves optimized performance.

D. Experimental Results on the Adaptive Online Training Method

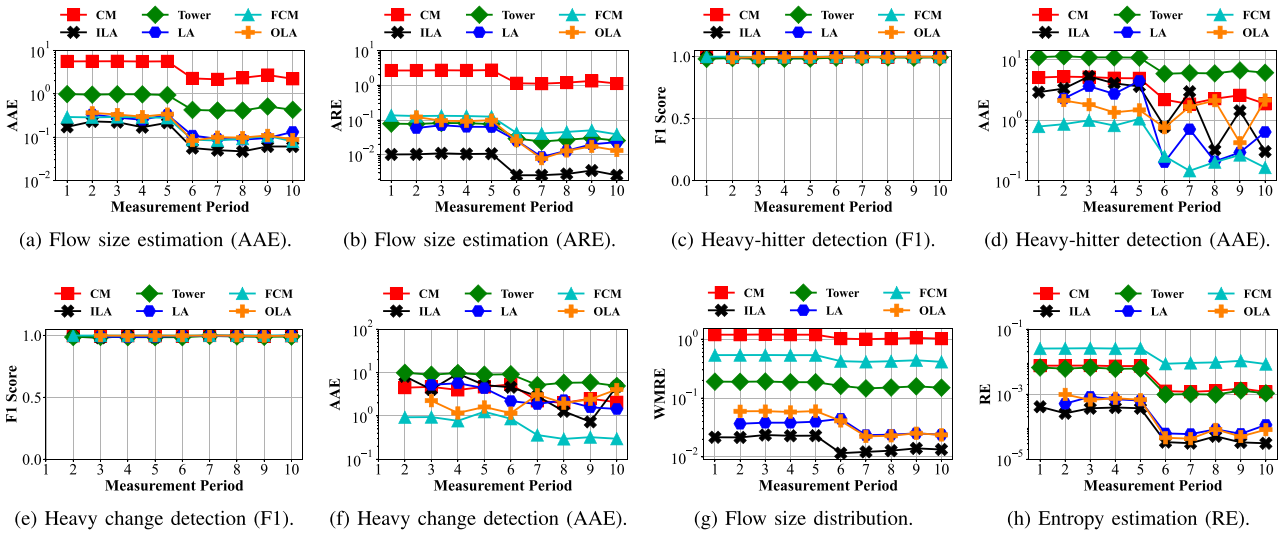
In this section, we evaluate the effectiveness of the proposed adaptive online training method.

1) Experimental Settings: The memory allocated to LA-Sketch is fixed at 400 KB. The first five periods use the CAIDA dataset, while the last five periods use the MAWI dataset, to evaluate the performance of LA-Sketch under abrupt traffic changes (from period 5 to period 6). It is worth noting that the traffic patterns and distribution characteristics of the CAIDA and MAWI datasets differ substantially, which enables us to demonstrate the effectiveness of adaptive online training method. Furthermore, even within the same dataset, the traffic patterns and distribution characteristics vary significantly between adjacent periods. Due to the absence of historical traffic data for the initial period, neither LA-Sketch nor OLA-Sketch includes results for this period in the figures. In subsequent periods, LA-Sketch trains using the accurate network traffic data from the preceding period, while OLA-Sketch utilizes query values from the prior sketch for training. Notably, during the first period, OLA-Sketch relies on query values from the CM Sketch for training, as historical traffic data is unavailable, precluding the use of LA-Sketch to train the level-aware classifier. Furthermore, at the beginning of each period, OLA-Sketch reconfigures the number of counters allocated to each level based on the flow size distribution derived from sketch query values of the preceding period, thereby enabling adaptive adjustment to shifts in network traffic characteristics.

2) Accuracy: The experimental results are as follows.

As shown in Fig. 7, the performance curves of OLA-Sketch and LA-Sketch are closely aligned, demonstrating that the adaptive online training method achieves accuracy comparable to the traditional online training method across most measurement tasks, which verifies the effectiveness of the adaptive online training method. Moreover, when traffic changes abruptly (from period 5 to period 6), OLA-Sketch is able to adapt and achieve satisfactory performance after a single period.

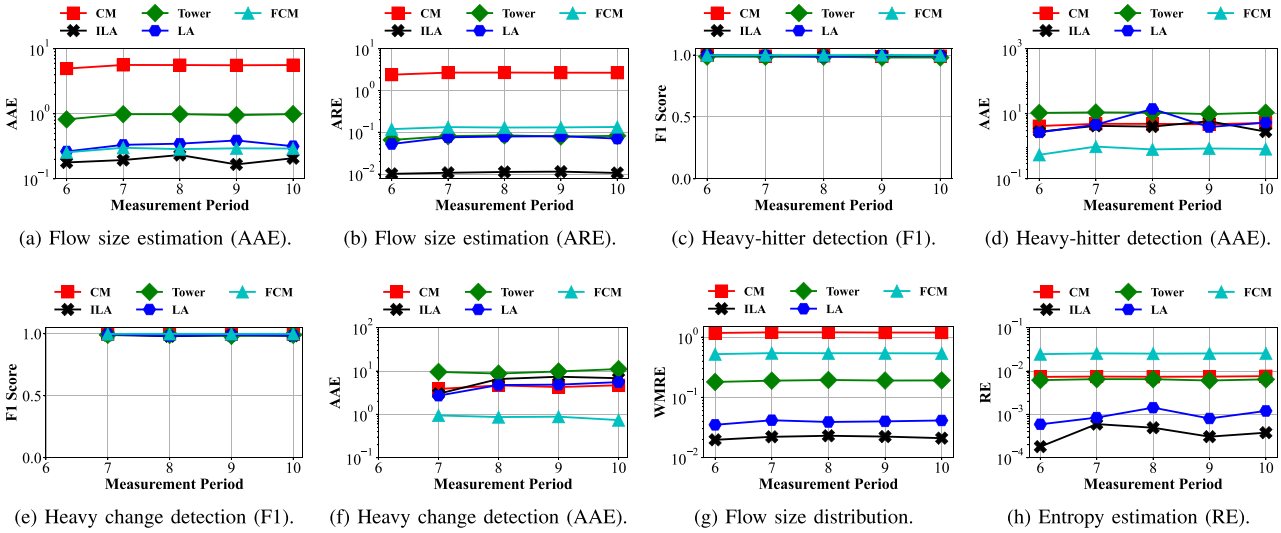
Analysis: In summary, the adaptive online training method performs comparably to conventional online training method that uses exact values. By employing adaptive online training, the classifier in LA-Sketch dynamically adapts to evolving



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 7. Experimental results on the adaptive online training method.



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 8. Experimental results with model training latency considered.

network traffic characteristics without requiring massive traffic data collection, maintaining high performance.

E. Experimental Results With Model Training Latency Considered

In this section, we evaluate the impact of model training latency on performance using the CAIDA dataset.

1) Experimental Settings: The memory allocated to LA-Sketch is fixed at 400 KB. We use the level-aware classifier trained on the first period to predict level for each flow in periods 6 through 10. The skipped intermediate periods represent potential delays introduced by model training.

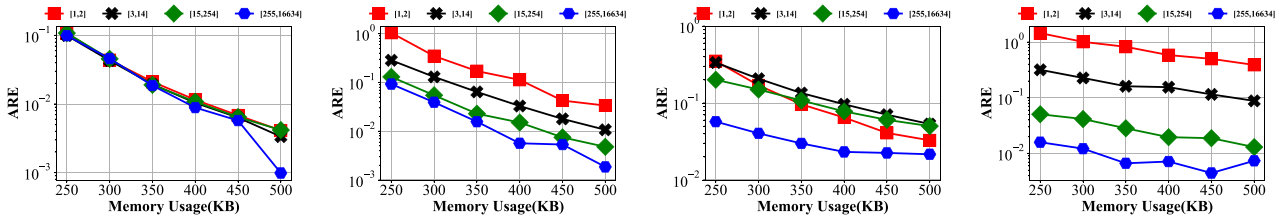
2) Accuracy: The experimental results are as follows.

As shown in Fig. 8, during period 6 to 10, we find that LA-Sketch exhibits lower error than CM, Tower, and FCM. This finding demonstrates that, even when accounting for model training latency, LA-Sketch still outperforms existing sketches.

F. Experimental Results on Error Analysis Across Levels

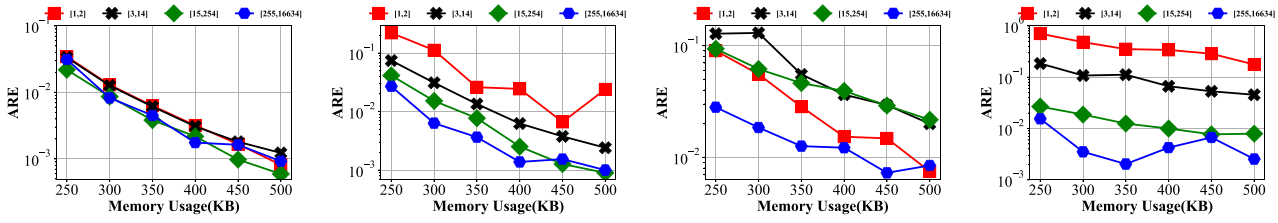
In this section, we evaluate the ARE of flows at different levels in LA-Sketch, demonstrating that LA-Sketch achieves nearly uniform ARE across almost all flows.

1) Experimental Settings: The experimental configuration of LA-Sketch follows that described in Section VI-B. The parameter settings of Tower Sketch and FCM Sketch are consistent with their respective papers. To analyze ARE variation across flow sizes, flows are divided into four groups defined by powers of two: $[1, 2]$, $[3, 14]$, $[15, 254]$, and $[255, 65534]$. The ARE is computed separately for each group. This partitioning strategy is motivated by the architecture of LA-Sketch, in which



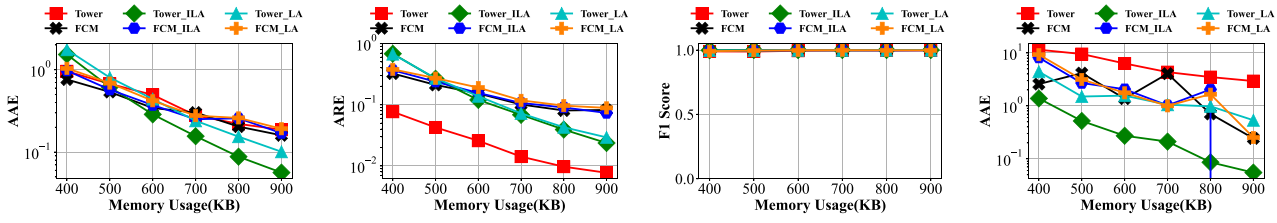
(a) ILA. (b) LA. (c) Tower. (d) FCM.

Fig. 9. Experimental results for ARE across different levels using the CAIDA dataset.



(a) ILA. (b) LA. (c) Tower. (d) FCM.

Fig. 10. Experimental results for ARE across different levels using the MAWI dataset.



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

Fig. 11. Experimental results of Tower Sketch and FCM Sketch enhanced with LA-Sketch using the CAIDA dataset.

counter sizes increase by a factor of two across successive levels, ensuring that each group faithfully captures the estimation error characteristic of its corresponding level.

2) Accuracy: The experimental results are as follows.

As shown in Figs. 9 and 10, ILA yields nearly identical ARE across levels, demonstrating the effectiveness of the adaptive counter configuration method. Since the flows in future periods differ substantially from those in the training phase, LA does not achieve perfectly uniform ARE across levels. Nevertheless, compared with Tower Sketch and FCM Sketch, it yields smaller variations in ARE across levels. Moreover, Tower Sketch exhibits less fluctuation than FCM Sketch.

G. Experimental Results Generalized to Other Sketches

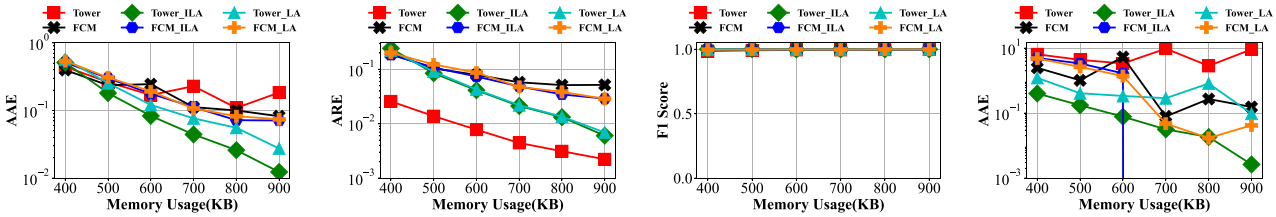
In this section, we evaluate the results of LA-Sketch when applied to two state-of-the-art hierarchical sketches: Tower Sketch [9] and FCM Sketch [10]. We present only the experimental results of flow size estimation and heavy-hitter detection, as the experimental results of other tasks are similar.

1) Experimental Settings: The experiments consider model storage overhead, e.g., assuming a memory allocation of 300 KB, only 215 KB is allocated to the data structure part of Tower_LA and FCM_LA. The adaptive counter configuration method follows the procedure described in Section V. Apart from this, all other parameters of the data structures are set consistently with their respective papers.

2) Accuracy: The experimental results are as follows.

Flow Size Estimation (Figs. 11 and 12): Tower_ILA and Tower_LA achieve lower AAE but higher ARE than Tower Sketch. This arises because Tower Sketch allocates longer counter arrays to lower levels, thereby reducing the estimation error for flows at lower levels. Given the highly skewed nature of network traffic, lower-level flows dominate, resulting in a smaller ARE for Tower Sketch; however, its AAE is correspondingly larger. In contrast, FCM_ILA and FCM_LA almost always exhibit the same error as FCM Sketch, since FCM employs only three counter levels (8-bit, 16-bit, and 32-bit). Due to the highly skewed nature of network traffic, the majority of flows fall into level-1, which limits the effectiveness of the LA-Sketch mechanism.

Heavy-Hitter Detection (Figs. 11 and 12): Tower_ILA and Tower_LA achieve the same F1 score as Tower Sketch but yield lower estimation errors. This is because in Tower Sketch, high-level elephant flows are assigned fewer effective counters,



(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

Fig. 12. Experimental results of Tower Sketch and FCM Sketch enhanced with LA-Sketch using the MAWI dataset.

resulting in larger errors. Tower_ILA and Tower_LA mitigate this issue by performing l insertions at level-l. FCM_ILA and FCM_LA almost always exhibit the same F1-score and estimation error as FCM Sketch. Similarly, due to the highly skewed nature of network traffic, the majority of flows fall into level-1, which limits the effectiveness of the LA-Sketch.

Analysis: In summary, integrating LA-Sketch into existing hierarchical sketches provides either improved or comparable performance, demonstrating the effectiveness of enhancement.

VII. Conclusion

In this paper, we propose a Level-Aware Sketch (LA-Sketch), a novel hierarchical sketching solution designed for efficient traffic measurement in software switches. Unlike prior hierarchical sketches, LA-Sketch leverages a level-aware classifier to intelligently map each flow to its corresponding level, significantly reducing hash collisions between elephant and mouse flows while minimizing sequential memory access overhead. Additionally, we introduce an adaptive counter number configuration method that minimizes overall hash collisions, optimizing LA-Sketch's performance. Finally, to adapt to the continuously changing network traffic characteristics, we develop an adaptive online training method that eliminates the necessity for massive traffic data collection by using only sketch query values. Extensive evaluations on two real-world network traces across five measurement tasks demonstrate that LA-Sketch outperforms state-of-the-art hierarchical sketches.

VIII. Limitations and Future Work

The machine learning model in LA-Sketch can impact throughput of the software switch. Fortunately, the parallel processing capabilities of GPUs enable simultaneous prediction for thousands of packets, thereby mitigating the performance overhead introduced by the model. Furthermore, we adopt the simplest MLP rather than more sophisticated models. Nevertheless, we acknowledge that the throughput issues arising from the use of machine learning constitute an inherent limitation of this work. In the future, we will seek to explore more cost-effective approaches.

References

[1] Y. Liu, K. Guo, F. Li, J. Shen, and X. Wang, "LA-Sketch: An adaptive level-aware sketch for efficient network traffic measurement," in *Proc. IEEE/ACM 33nd Int. Symp. Qual. Service (IWQoS)*. Piscataway, NJ, USA: IEEE Press, 2025, pp. 1–10.

[2] H. Zheng et al., "μMon: Empowering microsecond-level network monitoring with wavelets," in *Proc. ACM SIGCOMM 2024 Conf.*, 2024, pp. 274–290.

[3] K. Yang et al., "ChameleMon: Shifting measurement attention as network state changes," in *Proc. ACM SIGCOMM 2023 Conf.*, 2023, pp. 881–903.

[4] H. Zheng et al., "FlyMon: Enabling on-the-fly task reconfiguration for network measurement," in *Proc. ACM SIGCOMM 2022 Conf.*, 2022, pp. 486–502.

[5] H. Huang, J. Yu, Y. Du, J. Liu, H. Dai, and Y.-E. Sun, "Memory-efficient and flexible detection of heavy hitters in high-speed networks," *Proc. ACM Manage. Data*, vol. 1, no. 3, pp. 1–24, 2023.

[6] K. Guo, F. Li, Y. Zhang, H. Wan, J. Shen, and X. Wang, "MEC-Sketch: Memory-efficient per-flow cardinality measurement in high-speed networks," in *Proc. IEEE 33nd Int. Conf. Netw. Protocols (ICNP)*. Piscataway, NJ, USA: IEEE Press, 2025, pp. 1–11.

[7] Z. Cheng, G. Gao, H. Huang, Y.-E. Sun, Y. Du, and H. Wang, "BurstDetector: Real-time and accurate across-period burst detection in high-speed networks," in *Proc. INFOCOM or IEEE Conf. Comput. Commun.* Piscataway, NJ, USA: IEEE Press, 2024, pp. 2338–2347.

[8] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[9] K. Yang et al., "SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*. Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–12.

[10] C. H. Song, P. G. Kannan, B. K. H. Low, and M. C. Chan, "FCM-sketch: Generic network measurements with data plane support," in *Proc. 16th Int. Conf. Emerg. Netw. EXperiments Technologies*, 2020, pp. 78–92.

[11] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. 2002 Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2002, pp. 323–336.

[12] T. Yang, J. Gong, H. Zhang, L. Zou, L. Shi, and X. Li, "HeavyGuardian: Separate and guard hot items in data streams," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining*, 2018, pp. 2584–2593.

[13] Z. Fan et al., "OneSketch: A generic and accurate sketch for data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12887–12901, Dec. 2023.

[14] T. Yang et al., "Elastic sketch: Adaptive and fast network-wide measurements," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2018, pp. 561–575.

[15] Y. Li et al., "LadderFilter: Filtering infrequent items with small memory and time overhead," *Proc. ACM Manage. Data*, vol. 1, no. 1, pp. 1–21, 2023.

[16] Y. Zhou et al., "Cold filter: A meta-framework for faster and more accurate stream processing," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 741–756.

[17] Y. Li et al., "Pyramid family: Generic frameworks for accurate and fast flow size measurement," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 586–600, Apr. 2021.

[18] C.-Y. Hsu, P. Indyk, D. Katabi, and A. Vakilian, "Learning-based frequency estimation algorithms," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[19] E. Du, F. Wang, and M. Mitzenmacher, "Putting the 'learning' into learning-augmented algorithms for frequency estimation," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2021, pp. 2860–2869.

[20] H. Wang, H. Lin, Z. Zhong, T. Yang, and M. Shahzad, "Enhanced machine learning sketches for network measurements," *IEEE Trans. Comput.*, vol. 72, no. 4, pp. 957–970, Apr. 2022.

[21] F. Li, Y. Lv, Y. Yan, C. Gao, X. Wang, and J. Cao, "Learning-based sketch for adaptive and high-performance network measurement," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2571–2585, Jun. 2024.
[22] Y. Fu, D. Li, S. Shen, Y. Zhang, and K. Chen, "Clustering-preserving network flow sketching," in *Proc. INFOCOM or IEEE Conf. Comput. Commun.* Piscataway, NJ, USA: IEEE Press, 2020, pp. 1309–1318.
[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[24] CAIDA, "Anonymized Internet Traces 2018," 2018. Accessed: Mar. 6, 2026. [Online]. Available: https://catalog.caida.org/dataset/passive_2018_pcap
[25] MAWI Working Group of the WIDE Project, "MAWI Dataset," 2021. Accessed: Mar. 6, 2026. [Online]. Available: https://mawi.wide.ad.jp/mawi/samplepoint-F/2021/202109011400.html
[26] Q. Huang et al., "Toward nearly-zero-error sketching via compressive sensing," in *Proc. 18th USENIX Symp. Netw. Syst. Des. Implementation (NSDI),* 2021, pp. 1027–1044.
[27] R. Miao et al., "SketchConf: A framework for automatic sketch configuration," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE).* Piscataway, NJ, USA: IEEE Press, 2023, pp. 2022–2035.
[28] K. Guo, F. Li, Y. Liu, J. Shen, and X. Wang, "RA-Sketch: A unified framework for rapid and accurate sketch configurations," in *Proc. IEEE 33nd Int. Conf. Netw. Protocols (ICNP).* Piscataway, NJ, USA: IEEE Press, 2025, pp. 1–11.
[29] F. Li, K. Guo, Y. Liu, J. Shen, X. Wang, and J. Cao, "A unified configuration framework for heterogeneous sketches," *IEEE Trans. Netw.*, vol. 34, pp. 5319–5332, 2026.
[30] T. Bu, J. Cao, A. Chen, and P. P. Lee, "Sequential hashing: A flexible approach for unveiling significant patterns in high speed networks," *Comput. Networks*, vol. 54, no. 18, pp. 3309–3326, 2010.
[31] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 177–188, 2004.
[32] H. Zhang, H. Huang, Y.-E. Sun, G. Gao, Z. Wang, and S. Chen, "Multi-information sampling and mixed estimation for multi-task spread measurement with supercube," *IEEE Trans. Netw.*, vol. 33, no. 4, pp. 1795–1810, Aug. 2025.
[33] F. Li, K. Guo, J. Shen, and X. Wang, "Effective network-wide traffic measurement: A lightweight distributed sketch deployment," in *Proc. INFOCOM or IEEE Conf. Comput. Commun.* Piscataway, NJ, USA: IEEE Press, 2024, pp. 181–190.
[34] K. Guo, F. Li, J. Shen, X. Wang, and J. Cao, "Distributed sketch deployment for software switches," *IEEE Trans. Comput.*, vol. 74, no. 4, pp. 1210–1223, Apr. 2024.
[35] K. Guo, F. Li, J. Shen, and X. Wang, "Advancing sketch-based network measurement: A general, fine-grained, bit-adaptive sliding window framework," in *Proc. IEEE/ACM 32nd Int. Symp. Qual. Service (IWQoS).* Piscataway, NJ, USA: IEEE Press, 2024, pp. 1–10.
[36] Y. Du, H. Huang, Y.-E. Sun, S. Chen, G. Gao, and X. Wu, "Self-adaptive sampling based per-flow traffic measurement," *IEEE/ACM Trans. Netw.*, vol. 31, no. 3, pp. 1010–1025, Jun. 2022.
[37] H. Huang et al., "Spread estimation with non-duplicate sampling in high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2073–2086, Oct. 2021.
[38] K. Guo, F. Li, J. Shen, H. Wan, S. Chen, and M. Hou, "One-sketch: A unified framework for per-flow cardinality measurement with flexible bias control," in *Proc. INFOCOM or IEEE Conf. Comput. Commun.*, 2026, pp. 1–10.
[39] F. Li et al., "FSA-Hash: Flow-size-aware sketch hashing for software switches," *IEEE Trans. Comput.*, vol. 74, no. 11, pp. 3736–3749, Nov. 2025.

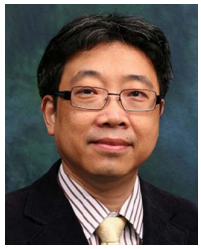
Fuliang Li (Member, IEEE) received the B.Sc. degree in computer science from the Northeastern University, China, in 2009, and the Ph.D. degree in computer science from Tsinghua University, China, in 2015. He is currently a Professor with the School of Computer Science and Engineering, Northeastern University. He has published more than 50 journals/conference papers. His research interests include network management and measurement, cloud computing, and network security.

Kejun Guo received the B.Sc. degree in computer science in 2023 from the Northeastern University, China, where he is currently working toward the Ph.D. degree with the School of Computer Science and Engineering. His research interests include data stream interests, network measurement, and network security. He is a recipient of the IEEE ICNP 2025 Best Paper Award and the IEEE/ACM IWQoS 2025 Best Student Paper Runner-up Award.

Yuting Liu received the B.Sc. degree in computer science in 2023 from the Northeastern University, China, where she is currently working toward the M.Sc. degree with the School of Computer Science and Engineering. Her research interest includes network measurement. She is a recipient of the IEEE/ACM IWQoS 2025 Best Student Paper Runner-up Award.

Jiaxing Shen (Member, IEEE) received the B.E. degree in software engineering from Jilin University in 2014, and the Ph.D. degree in computer science from Hong Kong Polytechnic University in 2019. He is an Assistant Professor with the Division of Artificial Intelligence, Lingnan University. In 2017, he was a Visiting Scholar with the Media Lab, Massachusetts Institute of Technology. His research interests include mobile computing, data mining, and IoT systems. His research has been published in top-tier journals such as IEEE TRANSACTIONS ON MOBILE COMPUTING, *ACM TOIS*, *ACM IMWUT*, and IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. He was awarded conference best paper twice including one from IEEE International Conference on Computer Communications (INFOCOM) 2020.

Xingwei Wang (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from the Northeastern University in 1989, 1992, and 1998, respectively. He is currently a Professor with the School of Computer Science and Engineering, Northeastern University. He has published more than 100 journal articles, books, book chapters, and refereed conference papers. His research interests include cloud computing, future internet, and others. He has received several best paper awards.



Jiannong Cao (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in 1986 and 1990, respectively. He is currently a Chair Professor with the Department of Computing, The Hong Kong Polytechnic University (PolyU), Hong Kong. He is also the Dean of Graduate School, the Director of Research Institute of Artificial Intelligent of Things, and the Internet and Mobile Computing Lab, and the Vice Director of the University's Research Facility in Big Data Analytics, PolyU. He has coauthored five books, coedited nine books, and authored or coauthored over 500 papers in major international journals and conference proceedings. His research interests include distributed systems and blockchain, wireless sensing and networking, big data and machine learning, and mobile cloud and edge computing.