LA-Sketch: An Adaptive Level-Aware Sketch for Efficient Network Traffic Measurement

Yuting Liu[†△], Kejun Guo[†△], Fuliang Li[†✉], Jiaxing Shen[‡], Xingwei Wang[†✉]
[†]Northeastern University, Shenyang 110819, China, [‡]Lingnan University, Hong Kong
Email: 2301921@stu.neu.edu.cn, kejunguo@163.com, lifuliang@cse.neu.edu.cn, jiaxingshen@LN.edu.hk, wangxw@mail.neu.edu.cn

***Abstract*—Network traffic measurement is critical for effective network management. Sketch has been proven to be a promising network traffic measurement solution. 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. 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.**

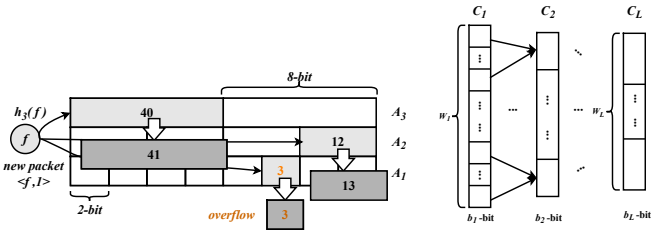
***Index Terms*—sketch, hierarchical designs, network traffic measurement.**

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 [1]–[8]. 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 [9] 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.



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

Fig. 1: Overview of Tower Sketch [10] and FCM Sketch [11].

Although the CM Sketch achieves high memory efficiency, its memory efficiency and accuracy are limited. To address this, recent sketch-based solutions have considered the skewed characteristics of network traffic and adopted hierarchical designs to improve both memory efficiency and accuracy. However, these solutions also introduce additional challenges. For example, Tower Sketch [10] and FCM Sketch [11] represent two state-of-the-art hierarchical sketches that exemplify these advancements and associated limitations.

Tower Sketch, as shown in Fig. 1a, 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 [9], [12]. 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.
- 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.

△ Yuting Liu and Kejun Guo contribute equally to this paper.

979-8-3315-4940-4/25/$31.00 © 2025 IEEE

- 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, as shown in Fig. 1b, 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. 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) Extensive experimental verification (§V): 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

In this section, we provide background knowledge on different types of sketches, previous work on hierarchical sketches, and previous work on learning-enhanced sketches.

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 flows and mouse flows, the accuracy of these sketches is poor due to hash collisions. Representative examples of simple sketches include CM Sketch [9], CU Sketch [12], CO Sketch [13]. 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 [14], OneSketch [15], Elastic Sketch [16] and so on. However, their hierarchical granularity remains relatively coarse, leaving room for further improvements in memory efficiency.

B. Hierarchical Sketch

Traditional sketch-based methods typically allocate fixed-length bit counters uniformly. Recent approaches exploit the skewness of network traffic to enhance memory efficiency and accuracy through hierarchical designs, as exemplified by Ladder Filter [17], Cold Filter [18], Tower Sketch [10], FCM Sketch [11] and so on. 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 [19]–[24]. Among these, the most relevant are those focusing on learning flow frequencies [19], [20], [25] 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.

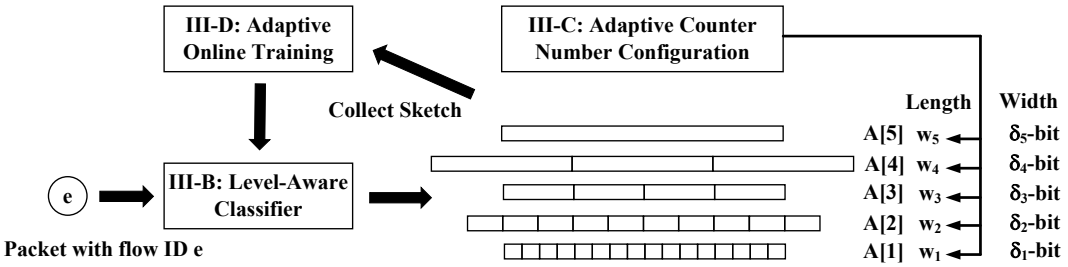


Fig. 2: Overview of LA-Sketch.

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 of LA-Sketch

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 [9], [12] 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. 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.

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
```

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, $A[1], \ldots, A[d]$. 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.

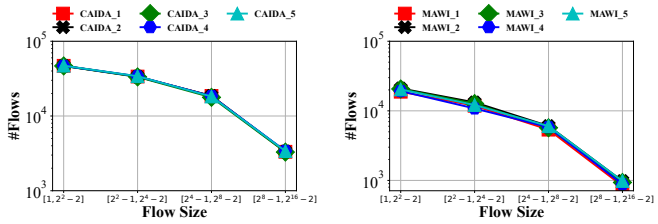
CM Insertion: As shown in Alg. 1, when inserting a packet with flow ID e, LA-Sketch uses its level-aware classifier to map e to its corresponding level lev, and then simply increments the k hashed counters in array $A[lev]$ at level lev by 1. If none of the k counters overflow, the insertion process terminates. 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

Algorithm 2: Query

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



(a) CAIDA. (b) MAWI.

Fig. 3: The flow size distributions of five consecutive periods from CAIDA [27] and MAWI [28] datasets.

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 V-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. 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 [26], 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.

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 V-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.

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 [27] and MAWI [28] traces. 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 consisting of d arrays, $A[1], A[2], \ldots, A[d]$, where N_i flows are mapped to array $A[i]$. To minimize total hash collisions, the number of counters w_i and w_j in arrays $A[i]$ and $A[j]$ must satisfy:

$$\frac{w_i}{w_j} = \frac{N_i}{N_j}$$

This indicates that the ratio of counter numbers between any two arrays at different levels should match the ratio of flows mapped to those levels. As shown in Fig. 3, flow size distributions for a specific data center or backbone network remain approximately constant, meaning N_i values can be treated as known. Thus, given a fixed memory size M, the memory size m_i for each array $A[i]$ can be determined by solving the following system of equations:

$$\begin{cases} M = \sum_{i=1}^{d} m_i \\ \frac{N_i}{N_j} = \frac{m_i}{2^i} \times \frac{2^j}{m_j} \quad \forall i, j \in [1, 2, \ldots, d] \end{cases}$$

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 [29].

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:

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

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

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

Step 2: Minimizing Total Hash Collisions.

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

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

Assuming $W = \sum_{i=1}^{d} w_i$, we apply the method of Lagrange multipliers. Introducing the Lagrange multiplier λ, the Lagrangian function is:

$$\mathcal{L}(w_1, w_2, \ldots, w_d, \lambda) = \sum_{i=1}^{d} \frac{N_i^2}{w_i} + \lambda \left(\sum_{i=1}^{d} w_i - W \right)$$

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

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

Solving for w_i yields:

$$w_i = \sqrt{\frac{N_i^2}{\lambda}} = \frac{N_i}{\sqrt{\lambda}}$$

Using the constraint $W = \sum_{i=1}^{d} w_i$, we have:

$$\sum_{i=1}^{d} \frac{N_i}{\sqrt{\lambda}} = W$$

Solving for $\sqrt{\lambda}$ gives:

$$\sqrt{\lambda} = \frac{\sum_{i=1}^{d} N_i}{W}$$

Substituting this back into the expression for w_i results in:

$$\frac{w_i}{W} = \frac{N_i}{\sum_{i=1}^{d} N_i}$$

This derivation generalizes to:

$$\frac{w_i}{\sum_{i=1}^{d} w_i} = \frac{N_i}{\sum_{i=1}^{d} N_i}$$

Finally, it follows that:

$$\frac{w_i}{w_j} = \frac{N_i}{N_j}$$

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, over time, 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.

Analysis: The feasibility of using LA-Sketch's approximate query values for online training is supported by the following points:

1) Accurate Mapping of Elephant Flows: In LA-Sketch, no flow is underestimated. As a result, elephant flows are consistently mapped to higher levels, ensuring their correct classification in subsequent periods.

2) Robust Mapping of Mouse Flows: Most mouse flows are correctly estimated as mouse flows. Although some mouse flows may be overestimated as elephant flows due to potential classification errors by the level-aware classifier or hash collisions, this leads to their temporary assignment to higher levels in the subsequent period. By periodically changing the hash seed, such mouse flows are reassigned to lower levels unless they again collide with elephant flows under the new hash configuration.

In summary, since most mouse flows are consistently identified as mouse flows and elephant flows are always recognized as elephant flows, the vast majority of flows are correctly mapped to their corresponding levels. Even if some misidentified mouse flows are mapped to higher levels, they can be corrected in subsequent periods through hash seed adjustments.

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. To perform these tasks, the end-host constructs an LA-Sketch and inserts each incoming packet, using its flow ID as the key.

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 [16], 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 [30] 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. Experimental Results

In this section, 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.

A. Experimental Setup

Dataset: Our evaluation utilizes two real-world datasets.

- CAIDA Dataset: We use anonymous IP tracking collected from CAIDA in 2018 [27]. 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 Dataset: We use the MAWI dataset [28], 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 Multi-Layer Perceptron (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. 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. For heavy hitter detection and heavy change detection tasks, the threshold is set to $\Delta_h = 500$.

Abbreviations: The following are some abbreviations and their meanings.

- LA-Sketch: refers to the standard LA-Sketch, which trains the level-aware classifier using accurate traffic data from the previous period and applies it to predict traffic levels in the subsequent period.
- ILA-Sketch: represents an idealized version of LA-Sketch, assuming perfect knowledge of the traffic levels in the current period. 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, which utilizes approximate sketch query values from the previous period for training.

Metrics: We evaluate the following metrics.

- *Average Absolute Error (AAE):* $\frac{1}{m}\sum_{i=1}^{n}|n_i - \hat{n}_i|$, where m is the number of flows, n_i and $\hat{n}_i$ are the actual and estimated flow sizes respectively.
- *Average Relative Error (ARE):* $\frac{1}{m}\sum_{i=1}^{m}\frac{|n_i-\hat{n}_i|}{n_i}$, where m is the number of flows, n_i and $\hat{n}_i$ are the actual and estimated flow sizes respectively.
- *$F1$ Score:* $\frac{2\cdot PR\cdot RR}{PR+RR}$, where PR (Precision Rate) refers to the ratio of the number of the correctly reported instances to the number of all reported instances, and RR (Recall Rate) refers to the ratio of the number of the correctly reported instances to the number of all correct instances.
- *Relative Error (RE):* $\frac{|True-Est|}{True}$, where $True$ and Est are the true and estimated values, respectively.
- *Weighted Mean Relative Error (WMRE):* $\frac{\sum_{i=1}^{z}|m_i-\hat{m}_i|}{\sum_{i=1}^{z}\left(\frac{m_i+\hat{m}_i}{2}\right)}$, where m_i and $\hat{m}_i$ are the true and estimated numbers of the flows of size i respectively, and z is the maximum flow size.

B. Experimental Results on Accuracy

In this section, we compare the accuracy of LA-Sketch with the widely used CM Sketch [9] and two state-of-the-art hierarchical sketches, Tower Sketch [10] and FCM Sketch [11].

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. 4a-4b): 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. 4c-4d): We find that all sketches have an F1 score over 95%, but ILA and LA achieve lower errors compared to CM and Tower.

Heavy Change Detection (Fig. 4e-4f): Similarly, we find that all sketches have high F1 scores, but ILA and LA achieve lower errors compared to CM and Tower.

Flow Size Distribution Estimation (Fig. 4g): 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. 4h): 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. 5a-5b): 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.

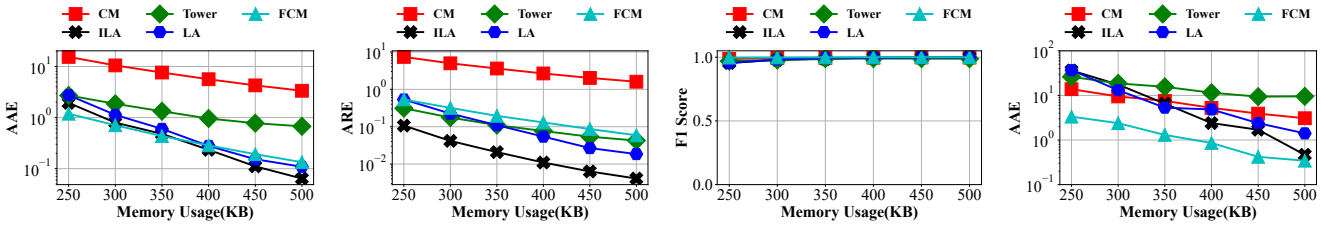
Heavy-Hitter Detection (Fig. 5c-5d): We find that all sketches have an F1 score over 95%, but ILA and LA achieve lower errors compared to CM and Tower.

Heavy Change Detection (Fig. 5e-5f): Similarly, we find that all sketches have high F1 scores, but ILA and LA achieve lower errors compared to CM and Tower.

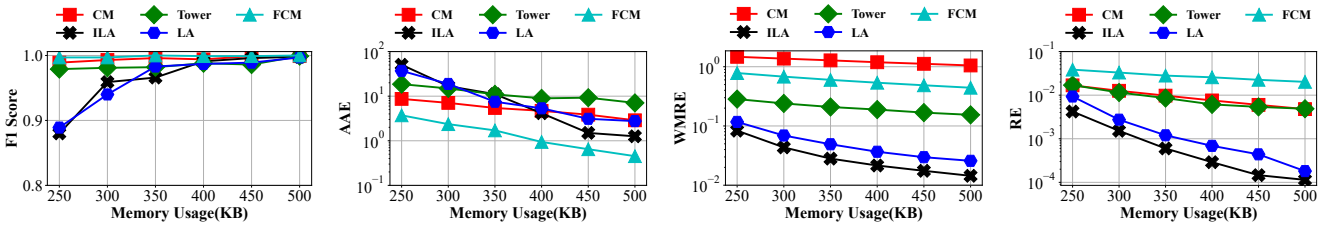
Flow Size Distribution Estimation (Fig. 5g): We find that ILA and LA achieve lower errors compared to CM, Tower, and FCM. Compared to CM, Tower, and FCM, ILA on average reduces the WMRE by 76.1, 11.6, and 31.5 times respectively; compared to CM, Tower, and FCM, LA on average reduces the WMRE by 38, 5.81,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

Entropy Estimation (Fig. 5h): 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.

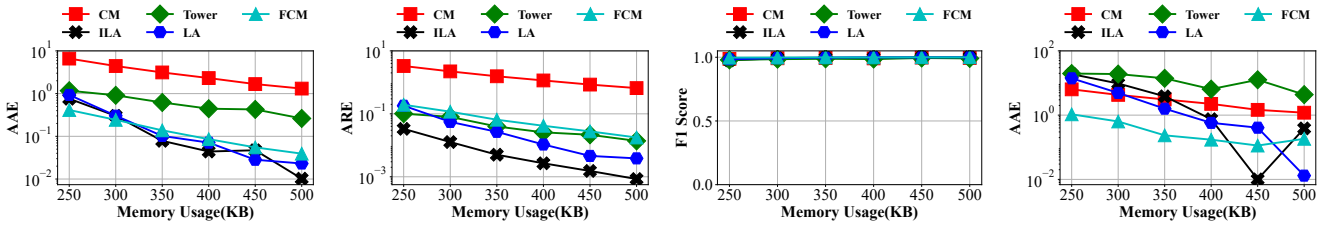


(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

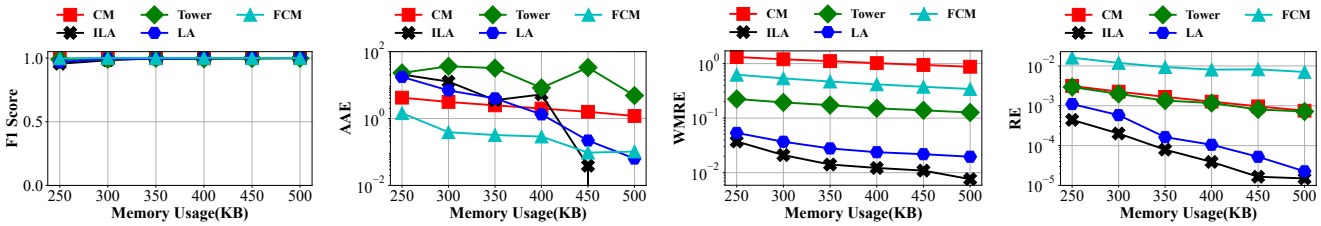


(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

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.

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.

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.

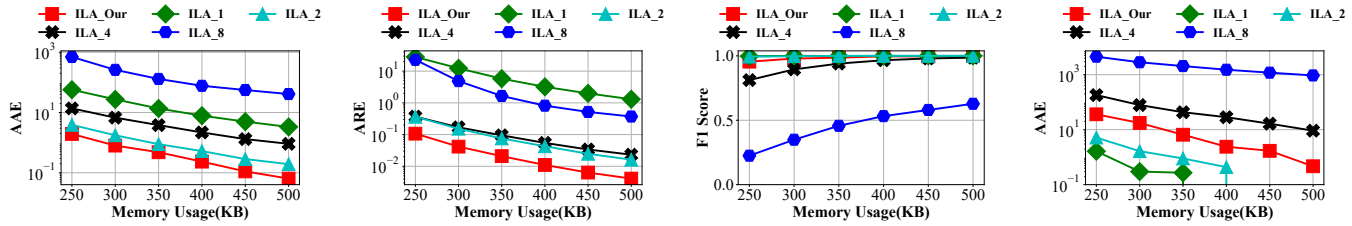
Flow Size Estimation (Fig. 6a-6b): We find that ILA_Our achieves lower errors. Compared to ILA_1, ILA_2, ILA_4, and ILA_8, ILA_our on average reduces the AAE by 37, 2.31, 9.8, and 404 times, and the ARE by 292, 3.74, 4.66, and 109 times, respectively.

Heavy-Hitter Detection (Fig. 6c-6d): 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.

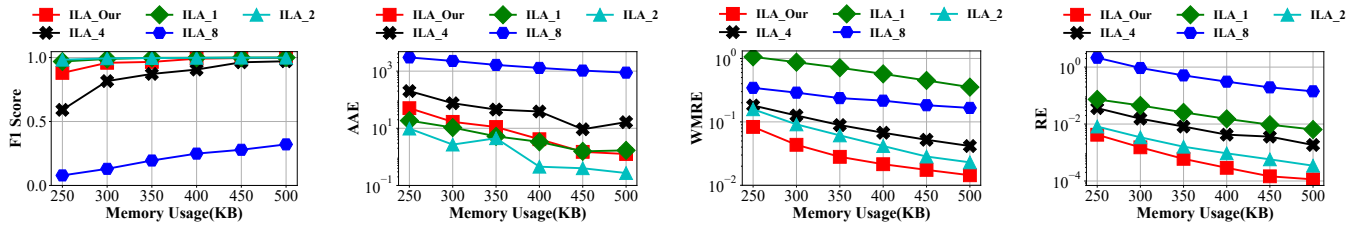
Heavy Change Detection (Fig. 6e-6f): Compared to ILA_4 and ILA_8, ILA_Our achieves higher F1 scores and lower errors. Compared to ILA_1 and ILA_2, ILA_Our performs worse, as they have more counters in the higher level arrays.

Flow Size Distribution Estimation (Fig. 6g): 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. 6h): 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.

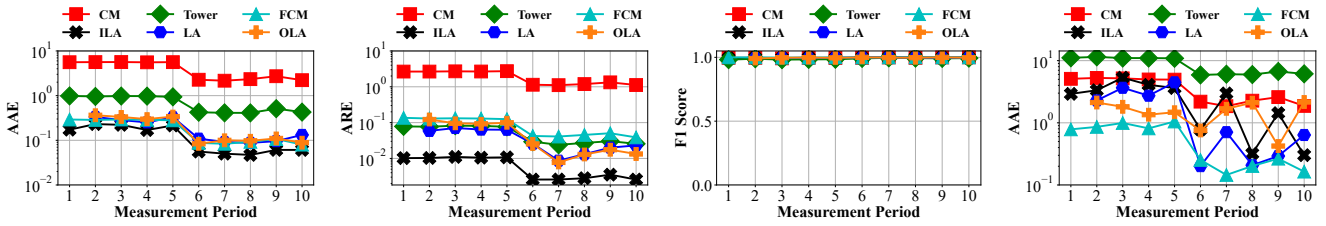


(a) Flow size estimation (AAE). (b) Flow size estimation (ARE). (c) Heavy-hitter detection (F1). (d) Heavy-hitter detection (AAE).

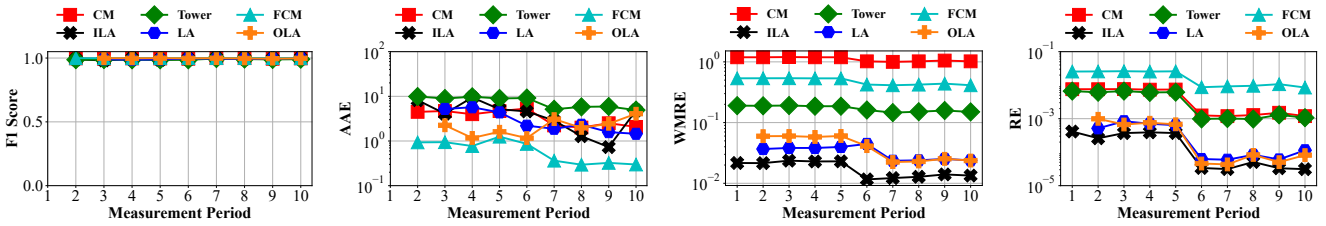


(e) Heavy change detection (F1). (f) Heavy change detection (AAE). (g) Flow size distribution. (h) Entropy estimation (RE).

Fig. 6: Experimental Results on the Counter Configuration Method using the CAIDA Dataset.



(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.

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 even within the same dataset, traffic 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.

2) Accuracy: The experimental results are as follows.

Due to space limitations, we omit detailed analysis for each task. 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. By employing adaptive online training, the classifier in LA-Sketch dynamically adapts to evolving network traffic characteristics without requiring massive traffic data collection, maintaining high performance.

VI. Conclusion

In this paper, we propose a Level-Aware Sketch (LA-Sketch), a novel hierarchical sketching solution. 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.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 62432003, U22B2005, and 92267206; the Liaoning Revitalization Talents Program under Grant No. XLYC2403086; and the financial support of Lingnan University (LU) under Grant No. DB23A9.

References

[1] H. Zheng, C. Huang, X. Han, J. Zheng, X. Wang, C. Tian, W. Dou, and G. Chen, "μmon: Empowering microsecond-level network monitoring with wavelets," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 274–290.

[2] K. Yang, Y. Wu, R. Miao, T. Yang, Z. Liu, Z. Xu, R. Qiu, Y. Zhao, H. Lv, Z. Ji *et al.*, "Chamelemon: Shifting measurement attention as network state changes," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 881–903.

[3] L. Tang, Q. Huang, and P. P. Lee, "Spreadsketch: Toward invertible and network-wide detection of superspreaders," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1608–1617.

[4] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: High precision congestion control," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 44–58.

[5] Q. Huang, X. Jin, P. P. Lee, R. Li, L. Tang, Y.-C. Chen, and G. Zhang, "Sketchvisor: Robust network measurement for software packet processing," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 113–126.

[6] K. Guo, F. Li, J. Shen, X. Wang, and J. Cao, "Distributed sketch deployment for software switches," *IEEE Transactions on Computers*, 2024.

[7] K. Guo, F. Li, J. Shen, and X. Wang, "Advancing sketch-based network measurement: A general, fine-grained, bit-adaptive sliding window framework," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[8] F. Li, K. Guo, J. Shen, and X. Wang, "Effective network-wide traffic measurement: A lightweight distributed sketch deployment," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 181–190.

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

[10] K. Yang, Y. Li, Z. Liu, T. Yang, Y. Zhou, J. He, T. Zhao, Z. Jia, Y. Yang *et al.*, "Sketchint: Empowering int with towersketch for per-flow per-switch measurement," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–12.

[11] C. H. Song, P. G. Kannan, B. K. H. Low, and M. C. Chan, "Fcm-sketch: generic network measurements with data plane support," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 78–92.

[12] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002, pp. 323–336.

[13] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 693–703.

[14] T. Yang, J. Gong, H. Zhang, L. Zou, L. Shi, and X. Li, "Heavyguardian: Separate and guard hot items in data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2584–2593.

[15] Z. Fan, R. Wang, Y. Cai, R. Zhang, T. Yang, Y. Wu, B. Cui, and S. Uhlig, "Onesketch: A generic and accurate sketch for data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 887–12 901, 2023.

[16] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 561–575.

[17] Y. Li, F. Wang, X. Yu, Y. Yang, K. Yang, T. Yang, Z. Ma, B. Cui, and S. Uhlig, "Ladderfilter: Filtering infrequent items with small memory and time overhead," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–21, 2023.

[18] Y. Zhou, T. Yang, J. Jiang, B. Cui, M. Yu, X. Li, and S. Uhlig, "Cold filter: A meta-framework for faster and more accurate stream processing," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 741–756.

[19] C.-Y. Hsu, P. Indyk, D. Katabi, and A. Vakilian, "Learning-based frequency estimation algorithms." in *International Conference on Learning Representations*, 2019.

[20] E. Du, F. Wang, and M. Mitzenmacher, "Putting the "learning into learning-augmented algorithms for frequency estimation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2860–2869.

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

[22] 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 Transactions on Networking*, no. 01, pp. 1–15, 2024.

[23] Y. Fu, D. Li, S. Shen, Y. Zhang, and K. Chen, "Clustering-preserving network flow sketching," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1309–1318.

[24] D. Bertsimas and V. Digalakis, "Frequency estimation in data streams: Learning the optimal hashing scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1541–1553, 2021.

[25] T. Li, S. Chen, and Y. Ling, "Per-flow traffic measurement through randomized counter sharing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1622–1634, 2012.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] https://catalog.caida.org/dataset/passive_2018_pcap, Anonymized Internet Traces 2018.

[28] https://mawi.wide.ad.jp/mawi/samplepoint-F/2021/202109011400.html.

[29] T. Bu, J. Cao, A. Chen, and P. P. Lee, "Sequential hashing: A flexible approach for unveiling significant patterns in high speed networks," *Computer Networks*, vol. 54, no. 18, pp. 3309–3326, 2010.

[30] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 177–188, 2004.