Unbiased Real-time Traffic Sketching

Yuhan Wu[†‡], Shiqi Jiang[†], Yifei Xu[†], Siyuan Dong[†], Kaicheng Yang[†], Peiqing Chen[†], Tong Yang[†‡]
[†] *National Key Laboratory for Multimedia Information Processing, Peking University, China,*
[‡] *Peng Cheng Laboratory, China.*

Abstract—In network measurement, sliding window measurement has the advantage of providing recent and timely measurement results. Recently, sketches have become the most popular method of conducting flow-level network measurements due to their favorable trade-off between small memory overhead and high measurement accuracy. However, it remains a challenge that no current sketches are able to support unbiased estimation toward flow size measurement, which can improve the performance of tasks including network diagnoses, delay measurement and heavy hitter detection. In this paper, we propose the first work that achieves unbiased flow size measurement in sliding windows, namely **Unbiased Cleaning sketch (UC sketch)**. The key technique of the UC sketch is *Unbiased Cleaning* which can remove outdated keys from the sliding windows in a balanced way. Besides, we significantly reduce the variance of flow size by two optimization techniques, namely *Linear Scaling* and *Column Randomizing*. To prove the result, we conduct rigorous mathematical analysis and reasonable experiments. All related source codes are open-sourced at Github.

Index Terms—Network Measurement, Sketches, Sliding Windows, Unbiased Estimation.

1 Introduction

1.1 Background and Motivation

Sketches are widely used in network measurement [1]–[9] due to their high accuracy and low memory requirements (usually less than 1 MB), making them particularly suitable for memory-constrained environments, such as the L2 cache in software switches or programmable switches with limited memory of several 10 MB [10]. Among all measurement tasks, flow size measurement, which involves counting the number of packets with the same key (e.g., 5-tuple), is critical for various system decision-making processes, including traffic engineering [11], [12] and load balancing [13]–[15]. There are two models for flow size estimation: 1) Fixed window model: it divides the time into fixed time windows and estimates sizes of flow that appears in each time window, and a lot of existing work falls in this category [16]–[19]. 2) Sliding window model: supports real-time queries on flows in a time window right before each query, and it is a more challenging task and has fewer works [20], [21] because it is challenging to keep track of a sliding window where there are constantly new packets arriving and outdated packets getting evicted. This paper focuses on the sliding window model. We conduct flow size measurement on the most recent packets (*e.g.*, the latest $10M$ packets or the packets in recent 1 second), and captures the latest characteristics of a data stream in real-time.

Unbiasedness is a widely acknowledged property in network measurement of practical importance. In theoretical analysis, the unbiased estimation can eliminate systematical errors [22] and provide the basis for further analysis, *e.g.*, deriving error bounds based on variance and Chebyshev inequality [23]. In many applications, unbiased flow size measurement serves well in network diagnoses [24], delay distribution estimation [25], and heavy hitter detection [4], [26]. Therefore, it is practically significant to achieve *unbiased flow size estimation under sliding windows*, which is abbreviated as *Unbiased Sliding* for convenience.

It is challenging to achieve *Unbiased Sliding* for two reasons. First, it is memory-consuming to keep all keys and then delete them in real time as the past packets move out of the sliding window. Consider a straightforward approach that uses a queue to record all keys and delete the tail record when adding a new record to the head. Another approach is to use a counter for tracking the flow size and a timestamp for the arrival time of the last packet of this flow, while we may scan all counters to remove those with outdated timestamps. Both approaches are memory-consuming. Second, unbiasedness must be formally proved, which turns out to be a challenging task. Among all existing works, only the Count sketch [27], the CMM sketch [28], and the CSM [29] sketch have been proven to be unbiased under the fixed window model. To the best of our knowledge, *no existing work has achieved Unbiased Sliding.*

1.2 Proposed Solution

This paper proposes the **Unbiased Cleaning sketch**, UC sketch for short, which is the first solution for unbiased flow size estimation under sliding windows. The key idea is to let time-dependent deviations (*i.e.*, estimation biases incurred by removing old keys too early or too late) cancel each other out. In the first step of our design, assuming that queries are uniformly distributed over time, we devise an *unbiased estimator* that neutralizes time-dependent deviations through randomized key removal. Our second step will remove the assumption with randomized window alignment that achieves unbiased estimation for any query arrival pattern.

One way to implement sliding window is to divide the time window into d segments and use $d{+}1$ counters for each flow, with one counter tracking the flow sizes in the current

Corresponding author: Tong Yang (yangtongemail@gmail.com). Co-primary authors: Yuhan Wu, Shiqi Jiang, and Yifei Xu.

time segment and d counters recording the heaviness of the flows in the previous d segments. At the end of the current time segment, the counter for the oldest time segment is zeroed out, which essentially removes the occurrences of the flow key during the oldest segment and thus frees up the counter for the next time segment. When a query about the key arrives, if we return the sum of the first d counters, it will be an under-estimation with a negative bias. If we return the sum of all $d+1$ counters, it will be an over-estimation with a positive bias. These $d+1$ counters together form an estimator. To reduce the overall number of counters, we may share a certain number of estimators for a much larger number of keys based on a hashing structure such as the Count sketch [27], which will make our analysis much more complicated—to illustrate our idea below, we will leave out estimator sharing.

To remove the negative/positive biases, when answering a query on the size of a flow, we will include the last counter with 50% probability at random, which essentially removes the occurrences of the key in the oldest time segment with 50% probability. We have proved that, under the assumption that the query arrival is uniformly random, the bias in our estimation is zero when we set the length of each time segment to be $t = \frac{W}{d+0.5}$, where W is the size of the sliding window.

To remove the above assumption of random query arrival, we adopt a data structure consisting of several arrays of estimators that are shared by all keys. These estimators have different segment beginning times, which are distributed uniformly at random based on a pseudo-random mechanism such as hashing; hence, the segment beginning time of each estimator is predictable. In other words, considering an arbitrary period of $[0, t)$, the estimators will each remove its last counter and use the counter for a new segment at an arbitrary instance during the period. Each key will be hashed to and recorded by multiple estimators. The query results from these estimators will differ depending on when they start their time segments, which causes negative or positive biases that follow a certain uniform distribution, allowing us to perform bias removal.

Furthermore, we propose two optimization techniques. One is called Linear Scaling which significantly reduces the variance of our unbiased estimation, as it produces a median-unbiased flow size estimate, rather than a mean-unbiased one. The other is called Column Randomizing which can further reduce the variance by eliminating errors due to hash collisions. Based on the unbiased flow size estimation, we applied Unbiased Cleaning sketch to three tasks, including heavy hitters, estimating subset sum, and estimating distributed sum. In this paper, we introduce related work in Section 2, present our algorithm in Section 3, conduct mathematical analysis in Section 4, and provide experiment results in Section 5.

2 Related Work

In this section, we show two kinds of algorithms for estimating the flow size in the sliding window, sketch-based algorithms and KV-based algorithms. The sketch-based algorithms save the flow size information through linear projections. The KV-based algorithms record Key-Value pairs $\langle flowkey, flowsize \rangle$ for flows.

2.1 Preliminaries

Data Stream: a data stream S is a sequence of keys (e.g., 5-tuple of packet headers), i.e., $S = \{e_1, e_2, e_3, \dots\}$. Each key appears once or more than once.

Sliding-window Model: A sliding window includes the keys which appear most recently in the data stream. A sliding window with size W could be time-based, i.e., including the keys appearing in the last W time units, or count-based, i.e., including the last W keys. The size of flow e, denoted as f_e, is defined as the number of times that key e appears in the sliding window. An estimation $\hat{f_e}$ of f_e is called unbiased only if its expected value is equal to f_e (i.e., $E(\hat{f_e}) = f_e$). As this paper focuses on estimating flow sizes in the sliding window, for other tasks in the sliding window, please refer to the literature [30]–[38].

2.2 Sketch-based Algorithms

Sketches [39]–[51] are a kind of probabilistic highly-compact data structure for inexact flow size estimation. Traditional sketches are used for flow size estimation in the whole data stream, and they do not support the sliding window, including CM [16], CU [18], Count [27], and Augmented [52].

In fixed-window estimation, there are two types of sketches that can achieve unbiased estimation: counter-based sketches (including Count, CMM and CSM) and ID-based sketches (including Waving [53], USS [54], and Coco [19]). We will introduce counter-based sketches using Count Sketch as an example. Count is a sketch that achieves unbiased flow size estimation. A Count sketch consists of multiple arrays, each containing many counters. Each array is associated with two hash functions, including a hash function $h(\cdot)$ that maps each key to a counter and a hash function $s(\cdot)$ that maps keys to a value $t \in \{+1, -1\}$. To insert a key, for each array, Count maps the key to a counter, namely mapped counters, by hash function $h(\cdot)$, and maps the key to a value t by $s(\cdot)$. Then, for each array, Count adds t to the mapped counter. The value of each counter in Count is called unbiased sum. To estimate the size of a flow, the sketch multiplies the value of each mapped counter by t, and answers the median (known as median-unbiased) of these products. Count is unbiased in fixed windows, but it cannot be applied to the sliding window model. CMM and CSM achieve unbiasedness through the mean, which is called mean-unbiased. While ID-based sketches, including Waving [53], USS [54], and Coco [19], can achieve unbiased estimation in fixed windows by storing keys and using probabilistic replacements, they differ significantly from Count. Hence, we will focus on a design based on Count.

For the sliding window [55]–[63], there are three typical sketches, the ECM sketch [64], the Proportional Windowed Count-Min (PROPORTIONAL) [65], and the splitter windowed count-min sketch (SPLITTER) [65]. The ECM sketch combines the CM sketches and exponential histograms [55]. The exponential histograms divide the sliding window into smaller windows with different sizes and answer the query by summarizing the answer from different smaller windows. The idea of Sliding sketch [66] and SHE [67] is very similar to that of ECM. PROPORTIONAL, which is based on the CM sketch, proportionally decreases the counter of the CM sketch to remove keys outside the sliding window, while SPLITTER, another CM-based sketch, records the changes of a cell, including the time and value.

2.3 KV-based Algorithms

There are two typical algorithms for both estimating flow size and finding heavy hitters in the sliding window, SWAMP [68] and WCSS [69]. However, both of them are not unbiased for any task. SWAMP supports multiple functions in sliding windows at the same time. It uses a cyclic queue to record all keys in the sliding window and uses a hash table technique, Tiny Table [70], to update and query information of keys in the cyclic queue. Instead of storing all keys in the sliding window, WCSS only stores keys of heavy flows. WCSS divides the window to multiple fixed time blocks and uses a novel structure called CSS to record the size of flows in each time block. But both of them are not unbiased.

TABLE 1: Main Notations Used in Section 3

Notation	Meaning
W	size of a sliding window
e_i	i_{th} key in a data stream S
T	current time
t	size of a time segment, which equals the period of the scanner
k	number of matrices in UC sketch
b	number of rows (estimators) in a matrix
d	Each estimator has $d+1$ counters (columns).
$A[p][q][r]$	r_{th} counter of q_{th} estimator in p_{th} matrix

3 The Unbiased Cleaning sketch

In this section, we present the basic version of the Unbiased Cleaning sketch (UC sketch) with its data structure and operations. Then, we present the optimized version with two optimization techniques, namely *Linear Scaling* and *Column Randomizing*. We list main notations used in this section in Table 1.

Fig. 1: Data Structure and Insertion Examples.

Algorithm 1: Insertion for the UC sketch

```
Function Insertion(e_i):
    for p = 1 to k do
        q ← h_p(e_i) mod b
        r ← R(T; p; q)
        A[p][q][r] ← A[p][q][r] + s_p(e_i)
    end
end
```

3.1 Data Structure of Basic UC sketch

The data structure (Figure 1) of the UC sketch has k matrices consisting of counters, $A[1], A[2], \dots, A[k]$. Each matrix $A[p]$ has b rows and $d+1$ columns. In matrix $A[p]$, the q^{th} row is denoted by $A[p][q]$, and the counter in row q and column r is denoted by $A[p][q][r]$. Each row of the matrices is an unbiased estimator. Each matrix $A[p]$ has two corresponding hash functions $h_p(\cdot)$ and $s_p(\cdot)$, where $h_p(\cdot)$ maps each key into an estimator in $A[p]$, and $s_p(\cdot)$ maps each key to a value in $\{+1, -1\}$. In addition, we have a scanner, which scans all estimators in the k matrices circularly and uniformly with period t. The scanner will clear the oldest counter in each estimator periodically. The size of the sliding window is denoted by W, and the current time is denoted by T. For each estimator, we divide the timeline into time segments of size t. Each estimator will record the frequency for at least the recent $d \cdot t$ time and at most $(d+1) \cdot t$ time. We set t as $\frac{W}{d+0.5}$, ensuring that each estimator records the average frequency for the most recent W time.

3.2 Operations of Basic UC sketch

Initialization: All counters are set to 0 initially. The initial position of the scanner is in estimator $A[1][1]$, and the scanning period of the scanner equals the size of the time segment. The time when estimator $A[p][q]$ being scanned for the first time (launch time) is $L(p;q) = \frac{b(p-1)+(q-1)}{bk} \cdot t$.

Insertion: To insert key e_i, for each matrix $A[p]$, we calculate $h_p(e_i)$ to get its mapped row (estimator) $A[p][h_p(e_i)]$. As the scanner scans all estimators periodically and uniformly, in each estimator $A[p][q]$, we can easily determine the column index of the most recently cleared counter $R(T;p;q)$ by the following formula:

$$R(T;p;q) = \left(\left\lfloor \frac{T - L(p;q)}{t} \right\rfloor \bmod (d+1) \right) + 1. \quad (1)$$

Then, For each mapped estimator $A[p][q]$, we add $s_p(e_i)$ to counter $A[p][q][R(T;p;q)]$, which is the most recently cleared counter.

Cleaning: We use a scanner to clear the counters which could record the out-dated keys. The scanner scans all estimators circularly with period t. Specifically, in each period, it starts from $A[1][1]$, passes $A[1][b]$, passes $A[k][1]$, and reaches $A[k][b]$. When the scanner detects a new estimator, the scanner will clear the oldest counter in the estimator, i.e., the counter which has not been cleared by the scanner for the longest time. Because the oldest counter must be the next counter of the most recently cleared counter, we can calculate the column index of the oldest counter $O(T;p;q)$ by the following formula:

$$O(T;p;q) = [R(T;p;q) \bmod (d+1)] + 1. \quad (2)$$

Query: To query the size of flow e_i, we first calculate $h_1(e_i), h_2(e_i), \ldots, h_k(e_i)$ to get its k mapped estimators. Then, For each mapped estimator $A[p][q]$, we sum up the values of all counters in it and multiplying the sum by $s_p(e_i)$ to calculate the estimated flow size sum_p by $sum_p = s_p(e_i) \cdot \sum_{i=1}^{d+1} A[p][q][i]$. Finally, we return the mean of $\{sum_1, sum_2, \ldots, sum_k\}$ as the size of flow e_i, regardless of whether it is negative or not.

Heavy Hitters Query: As is known to all, we can also implement the function of finding heavy hitters in sliding windows by adding an additional heap to the data structure.

Subset Sum Query: To query the total flow size of a subset, we query the size of each flow in the subset and add them up as the result.

Distributed Sum Query: To query the size of all flows in a distributed system, we query the size of each flow in each distributed node and add them up as the result.

3.3 Optimized Version

In this section, we propose the optimized version of our UC sketch by introducing two optimization techniques, namely Linear Scaling and Column Randomizing. Compared with the basic version, the optimized version improves the variance and achieves the median-unbiased estimation.

Linear Scaling: The Linear Scaling allows us to estimate the flow size unbiasedly by the median value, and therefore it significantly reduces the variance. When adopting the Linear Scaling, the only difference is the query operation. When querying a key e_i, for each mapped estimator $A[p][q]$, the estimator records the keys in a time range $[d, d+1)(\Delta t)$ randomly. We hope that the estimators record the keys in the sliding window, whose size equals $d + 0.5(\Delta t)$. To achieve that, we scale up or down the value of the oldest counter $A[p][q][O(T, p, q)]$, and get C, which is the value of the counter after scaling, by the following formula:

$$C = \left\{1.5 - \frac{T - L(p,q)}{\Delta t}\right\} \cdot A[p][q][O(T,p,q)]; \quad (3)$$

where $\{x\}$ denotes the fractional part of x. Then, we calculate sum_p by summing up C and the values of the other counters and multiplying the sum by $s_p(e_i)$. Finally, we return the median of $\{sum_1, sum_2, \ldots, sum_k\}$ as the size of flow e_i.

Column Randomizing: The Column Randomizing can make each column works more independently and therefore reduce the variance. For each column in each matrix, we assign it an independent hash function $s_{p,r}(\cdot)$, where $s_{p,r}$ denotes the hash function for the r_{th} column in matrix $A[p]$. To insert key e_i, for each mapped estimator $A[p][q]$, we add $s_{p,r}(e_i)$ to the most recently cleared counter $A[p][q][R(T,p,q)]$. To query the size of flow e_i, for each mapped estimator $A[p][q]$, we calculate sum_p by summing up the value of $A[p][q][r] \cdot s_{p,r}(e_i)$, where r values from 1 to $d+1$. Then, we return the median of $\{sum_1, sum_2, \ldots, sum_k\}$ as the size of flow e_i.

4 Mathematical Analysis

In this section, we provide theoretical analysis of our Unbiased Cleaning sketch.

4.1 Proof Sketch

Unbiasedness: For a flow e_i recorded in a matrix, we notice that the expectation of the time deviation is 0. Thus, under the assumption that the data stream is evenly distributed during the edge of the window, the expectation of error caused by time deviation is 0. On the other hand, the expectation of error caused by hash collision is 0. Therefore, each matrix provides an unbiased estimation and thus our Unbiased Cleaning sketch provides an unbiased flow size estimation. The detailed proof is shown in Section 4.2.

Variance: For further analysis, we assume the flow size is proportional to the duration. Then we can express the error of our estimation and thus calculate the variance of a matrix. The details are shown in Section4.3.

Error Bound: Given the variance, we can easily get the error bound of estimation by a matrix, based on which we can further give out the error bound of our Unbiased Cleaning sketch taking either the mean or the median of the estimated flow size of matrices. The details and the proof are provided in Section 4.4.

Robustness: We show that our algorithm guarantees the superiority of flows whose estimated size exceeds a certain percentage of the total in Section 4.5, which indicates that our Unbiased Cleaning sketch has a property of robustness.

Zipfian Distribution: We do an analysis of the case of Zipfian distribution in Section 4.6.

4.2 Proof of Unbiasedness

We assume that the data stream is evenly distributed during the edge of the window, i.e., the data stream is evenly distributed during $(T - (d+1)\Delta t, T - d\Delta t)$. Below we prove that for each flow e_i, its estimated flow size is unbiased.

Theorem 1. *Let $\hat{f}_i$ be the estimation of the size of flow e_i in a matrix, then $\hat{f}_i$ is unbiased, i.e., $E(\hat{f}_i) = f_i$.*

Proof. For a specific estimator in a matrix, let $F_1(T_1, T_2)$ be the flow sizes during time period $[T_1, T_2]$ whose $s(\cdot)$ is $+1$, and $F_{-1}(T_1, T_2)$ be the flow sizes whose $s(\cdot)$ is -1. If $T_1 > T_2$, then we take $F_1(T_1, T_2) = -F_1(T_2, T_1)$, and the same for $F_{-1}(T_1, T_2)$. Let T be the current time, and T_C be the last time before T that we clean a counter in this estimator.

For a given key e_i, without loss of generality, we take $s(e_i) = 1$. Then our estimation for e_i in matrix $A[p]$ $(p \in \{1, 2, \ldots, k\})$ is the result recorded in the estimator $A[p][h_p(e_i)]$ during period $(T_C - d\Delta t, T)$, i.e.,

$$\hat{f}_i = F_1(T_C - d\Delta t, T) - F_{-1}(T_C - d\Delta t, T). \quad (4)$$

Let $\hat{f}_i^{\,0}$ be the estimation of size of flow e_i if there is no time deviation, which means $\hat{f}_i^{\,0}$ is the result recorded during period $(T - (d+0.5)\Delta t, T)$, i.e.,

$$\hat{f}_i^{\,0} = F_1(T - (d+0.5)\Delta t, T) - F_{-1}((T - (d+0.5)\Delta t, T). \quad (5)$$

To prove $\hat{f}_i$ is unbiased, we divide $E(\hat{f}_i) - f_i$ into two parts as shown in Equation 6:

$$E(\hat{f}_i) - f_i = E(\hat{f}_i - \hat{f}_i^{\,0}) + E(\hat{f}_i^{\,0} - f_i). \quad (6)$$

Below we prove that $E(\hat{f}_i - \hat{f}_i^{\,0}) = 0$ and $E(\hat{f}_i^{\,0} - f_i) = 0$.

First, for the former term $(\hat{f}_i - \hat{f}_i^0)$, which is the error caused by time deviation, we have

$$\hat{f}_i - \hat{f}_i^0 = F_1(T_C - dt; T - (d+0.5)t) - F_{-1}(T_C - dt; T - (d+0.5)t). \quad (7)$$

Let T_1 be $T_C - dt$ and T_2 be $T - (d+0.5)t$, then we have $\hat{f}_i - \hat{f}_i^0 = F_1(T_1; T_2) - F_{-1}(T_1; T_2)$. For $T_1; T_2$, we have

$$T_2 - T_1 = (T - (d+0.5)t) - (T_C - dt) = T - 0.5t - T_C. \quad (8)$$

Because of the randomness of T_C, T_C follows a uniform distribution $U(T - t; T)$. Thus, $T_2 - T_1 = T - 0.5t - T_C$ follows a uniform distribution $U(-0.5t; 0.5t)$, which indicates that T_1 follows $U(T_2 - 0.5t; T_2 + 0.5t)$. Since the flow sizes are evenly distributed during the edge of the sliding window, i.e., $(T_2 - 0.5t; T_2 + 0.5t)$, we can get

$$E(F_1(T_1; T_2)) = \int_{-0.5t}^{0.5t} F_1(T_2 - x; T_2)\, dx = 0. \quad (9)$$

Similarly, we have $E(F_{-1}(T_1; T_2)) = 0$. Thus, for the first part in Equation 6, we have

$$E(\hat{f}_i - \hat{f}_i^0) = E(F_1(T_1; T_2)) - E(F_{-1}(T_1; T_2)) = 0. \quad (10)$$

On the other hand, for the latter term in Equation 6, i.e., $(\hat{f}_i^0 - f_i)$, which is the error caused by hash collisions, we have

$$\hat{f}_i^0 - f_i = \sum_{e_{i_j}} f_{i_j} \cdot s(e_{i_j}); \quad (11)$$

where e_{i_j} is key inserted in this estimator other than e_i and f_{i_j} is the size of flow e_{i_j} during period $(T - (d+0.5)t; T)$. $s(e_{i_j})$ has same chance to be 1 and -1 and is independent of f_{i_j}. Thus, for the second part in Equation 6, we have

$$E(\hat{f}_i^0 - f_i) = \sum_{e_{i_j}} [E(f_{i_j}) \cdot E(s(e_{i_j}))] = 0. \quad (12)$$

Therefore, by combining Equation 10 and Equation 12, we can get that

$$E(\hat{f}_i) - f_i = E(\hat{f}_i - \hat{f}_i^0) + E(\hat{f}_i^0 - f_i) = 0; \quad (13)$$

which indicates that the estimated size $\hat{f}_i$ is unbiased. □

We take the mean of estimation in k matrices as the estimated size of our Unbiased Cleaning sketch. We have

$$E(mean(\hat{f}_i)) = E(\frac{1}{k}\sum \hat{f}_i) = \frac{1}{k}\sum E(\hat{f}_i) = \frac{1}{k} \cdot k f_i = f_i; \quad (14)$$

here $\sum \hat{f}_i$ is to sum k matrices. We can get the theorem below.

Theorem 2. *The estimation of the flow size of e_i by our Unbiased Cleaning sketch, $mean(\hat{f}_i)$, is unbiased, i.e., $E(mean(\hat{f}_i)) = f_i$.*

Below we prove that the theorems still hold for two optimizations. Without loss of generality, we still take $s(e_i) = 1$ in each counter.

Column Randomizing: When adopting the option, we can prove that Theorem 1 and Theorem 2 still hold.

Proof. Let $\hat{f}_{i;1}, \cdots, \hat{f}_{i;d+1}$ be the estimated flow size during period $(T_C - dt; T_C - (d-1)t), \cdots, (T_C; T)$, which corresponds to $(d+1)$ counters. And let $f_{i;1}, \cdots, f_{i;d+1}$ be the real flow size of e_i during corresponding periods.

Same as the proving process of $E(\hat{f}_i^0 - f_i) = 0$ in Theorem 1, we can get that the estimation is unbiased if there is no time deviation. Thus, we have

$$E(\hat{f}_{i;j} - f_{i;j}) = 0. \quad (15)$$

Let $f_{i;0}$ be the flow size of e_i during period $(T - (d+0.5)t; T_C - (d-1)t)$, and $\hat{f}^0_{i;1}$ be the estimation given by a counter during corresponding period if there is no time deviation, i.e.,

$$\hat{f}^0_{i;1} = F_1(T - (d+0.5)t; T_C - (d-1)t) - F_{-1}(T - (d+0.5)t; T_C - (d-1)t). \quad (16)$$

Similar to the above, we have $E(\hat{f}^0_{i;1} - f_{i;0}) = 0$.

For $\hat{f}^0_{i;1}$ and $\hat{f}_{i;1}$, we have

$$\hat{f}_{i;1} - \hat{f}^0_{i;1} = F_1(T_C - dt; T - (d+0.5)t) - F_{-1}(T_C - dt; T - (d+0.5)t). \quad (17)$$

Same as the proving process of Equation 9 in Theorem 1, we can get that

$$E(F_1(T_C - dt; T - (d+0.5)t) - F_{-1}(T_C - dt; T - (d+0.5)t)) = 0. \quad (18)$$

Thus, we have $E(\hat{f}_{i;1} - \hat{f}^0_{i;1}) = 0$, from which and Equation 15 we can derive that

$$E(\hat{f}_{i;1} - f_{i;0}) = E(\hat{f}_{i;1} - \hat{f}^0_{i;1}) + E(\hat{f}^0_{i;1} - f_{i;0}) = 0. \quad (19)$$

For our estimation $\hat{f}_i$, by combining the results of Equation 15 and Equation 19, we can get that

$$\begin{aligned} E(\hat{f}_i) - f_i &= E(\sum_{j=1}^{d+1} \hat{f}_{i;j}) - (f_{i;0} + \sum_{j=2}^{d+1} f_{i;j}) \\ &= E(\hat{f}_{i;1} - f_{i;0}) + \sum_{j=2}^{d+1} E(\hat{f}_{i;j} - f_{i;j}) = 0. \end{aligned} \quad (20)$$

Therefore, we have $E(\hat{f}_i) - f_i = 0$, i.e., the estimated flow size $\hat{f}_i$ is unbiased. Same as the proof of Theorem 2, we can get that $mean(\hat{f}_i)$ is also unbiased. □

Linear Scaling: When adopting the option, we need to strengthen the assumption to that data stream is evenly distributed during $(T - (d+1)t; T - (d-1)t)$. We can prove that Theorem 1 and Theorem 2 still hold. Moreover, our estimation is still unbiased if we take the median of $\hat{f}_i$ in k matrices, i.e., $E(median(\hat{f}_i)) = f_i$.

Proof. We continue to use the mark in the case of Column Randomizing. Here our estimation for the size of flow e_i is

$$\begin{aligned} \hat{f}_i &= \hat{f}_{i;1} \cdot \frac{(T_C - (d-1)t) - (T - (d+0.5)t)}{t} + \sum_{j=2}^{d+1} \hat{f}_{i;j} \\ &= \hat{f}_{i;1} \cdot \frac{1.5t - (T - T_C)}{t} + \sum_{j=2}^{d+1} \hat{f}_{i;j}. \end{aligned}$$

Let $\hat{f}^0_{i;1}$ be $\hat{f}_{i;1} \cdot \frac{1.5t - (T - T_C)}{t}$. Since the data stream is evenly distributed during $(T - (d+1)t; T - (d-1)t)$, let

$v_1, v_2, \cdots, v_n$ be the flow size of $e_1, e_2, \cdots, e_n$ in unit time during $(T-(d+1)\Delta t, T-(d-1)\Delta t)$. Thus, we have

$$\begin{aligned} f_{i,0} &= v_i[(T_C-(d-1)\Delta t)-(T-(d+0.5)\Delta t)] \\ &= (1.5\Delta t-(T-T_C))v_i, \end{aligned} \quad (21)$$

and also

$$\hat{f}_{i,1} = \sum_{j=1}^{n} v_j \Delta t\, s(e_j) = \Delta t \sum_{j=1}^{n} v_j s(e_j). \quad (22)$$

Since T_C is independent of $s_j(\cdot)$, we can get the expectation of $(\hat{f}'_{i,1} - f_{i,0})$ that

$$\begin{aligned} E(\hat{f}'_{i,1} - f_{i,0}) &= E[(1.5\Delta t-(T-T_C))\sum_{j\neq i} v_j s(e_j)] \\ &= E(1.5\Delta t-(T-T_C))\sum_{j\neq i} v_j E(s(e_j)) = 0. \end{aligned} \quad (23)$$

Same as the proving process in the case of Column Randomizing, we have $E(\hat{f}_{i,j} - f_{i,j}) = 0$. For our estimation $\hat{f}_i$, We can get that

$$\begin{aligned} E(\hat{f}_i) - f_i &= E(\hat{f}'_{i,1} + \sum_{j=2}^{d+1}\hat{f}_{i,j}) - (f_{i,0} + \sum_{j=2}^{d+1} f_{i,j}) \\ &= E(\hat{f}'_{i,1} - f_{i,0}) + \sum_{j=2}^{m} E(\hat{f}_{i,j} - f_{i,j}) = 0. \end{aligned} \quad (24)$$

Therefore, we have $E(\hat{f}_i) - f_i = 0$, i.e., the estimated flow size $\hat{f}_i$ is unbiased. Same as the proof of Theorem 2, we can get that mean$(\hat{f}_i)$ is also unbiased. □

Moreover, when T_C is fixed and we only calculate the mathematical expectation about $s_j(\cdot)$, the above proving process still holds. Since $s_j(\cdot)$ has the same probability of being $+1$ and -1, we notice that $(\hat{f}_i - f_i)$ is symmetrical about 0 for any fixed T_C. Thus, when T_C is not fixed we still have $(\hat{f}_i - f_i)$ is symmetrical about 0. So median$(\hat{f}_i - f_i)$ is symmetrical about 0, which indicates that $E(\text{median}(\hat{f}_i)) - f_i = 0$. Therefore, $E(\text{median}(\hat{f}_i)) = f_i$, i.e., our estimation is still unbiased if we take the median of $\hat{f}_i$ in k matrices.

4.3 Variance

For further analysis, we assume that the increment of the flow size is stable over a period of time. In particular, we assume the flow size is proportional to duration. Let $v_1, v_2, \cdots, v_n$ be the flow size of $e_1, e_2, \cdots, e_n$ in unit time. We show the estimated variance of $\hat{f}_i$ below.

Theorem 3. For the sake of brevity, let $d' = d + 0.5$. The estimation of the variance of $\hat{f}_i$ satisfies that

$$Var(\hat{f}_i) = \frac{1}{b}\sum_{j\neq i} v_j^2 \cdot (d'^2 + \frac{1}{12})\Delta t^2 + \frac{1}{12}v_i^2\Delta t^2. \quad (25)$$

Proof. Let $e_{i_1}, e_{i_2}, \cdots, e_{i_l}$ be the keys inserted to the same estimator as e_i, and $v_{i_1}, v_{i_2}, \cdots, v_{i_l}$ be the flow size of $e_{i_1}, e_{i_2}, \cdots, e_{i_l}$ in unit time. Without loss of generality, we take $s(e_i) = 1$. As proved in 1, $E(\hat{f}_i) = f_i$. Thus, for the variance, we have

$$\begin{aligned} Var(\hat{f}_i) &= E(\hat{f}_i - E(\hat{f}_i))^2 = E(\hat{f}_i - f_i)^2 \\ &= E((\hat{f}_i - \hat{f}_i') + (\hat{f}_i' - f_i))^2. \end{aligned} \quad (26)$$

For the former term, i.e., $(\hat{f}_i - \hat{f}_i')$, we have $\hat{f}_i - \hat{f}_i' = F_1(T_1, T_2) - F'_1(T_1, T_2)$, so we can get that

$$\hat{f}_i - \hat{f}_i' = (\sum_{j=1}^{l} v_{i_j} s(e_{i_j}) + v_i) \cdot w\Delta t, \quad (27)$$

here $w\Delta t$ is the deviation of time, which satisfies a uniform distribution $U(-0.5\Delta t, 0.5\Delta t)$ due to our assumption, i.e., w satisfies a uniform distribution $U(-0.5, 0.5)$.

On the other hand, for the latter term in Equation 26, i.e., $(\hat{f}_i' - f_i)$, we have

$$\hat{f}_i' - f_i = \sum_{e_{i_j}} f^0_{i_j} s(e_{i_j}) = (\sum_{j=1}^{l} v_{i_j} s(e_{i_j})) \cdot d'\Delta t. \quad (28)$$

Therefore, for the variance $Var(\hat{f}_i)$, we can get the following results that

$$\begin{aligned} Var(\hat{f}_i) &= E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}) + v_i) \cdot w\Delta t + \sum_{j=1}^{l} v_{i_j} s(e_{i_j}) \cdot d'\Delta t]^2 \\ &= E[\sum_{j=1}^{l} v_{i_j} s(e_{i_j}) \cdot (w + d')\Delta t + v_i \cdot w\Delta t]^2 \\ &= E[\sum_{j=1}^{l} v_{i_j} s(e_{i_j}) \cdot (w + d')\Delta t]^2 + E[v_i \cdot w\Delta t]^2. \end{aligned} \quad (29)$$

In Equation 29, it's because $s(e_{i_j})$ has same chance to be 1 and -1 and is independent of other variables so that

$$E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j})) \cdot (w + d')\Delta t \cdot (v_i s(e_i) \cdot w\Delta t)] = 0. \quad (30)$$

Then we first calculate mathematical expectation in Equation 29 on w. Since w satisfies a uniform distribution $U(-0.5, 0.5)$, we have

$$\begin{aligned} Var(\hat{f}_i) &= E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2 \cdot (w + d')^2\Delta t^2] + E[v_i^2 \cdot w^2\Delta t^2] \\ &= \int_{-0.5}^{0.5} E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2\Delta t^2](d' + w)^2 + v_i^2\Delta t^2 w^2\, dw \\ &= E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2\Delta t^2] \cdot (d'^2 + \frac{1}{12}) + \frac{1}{12}v_i^2\Delta t^2. \end{aligned} \quad (31)$$

For mathematical expectation on $s(e_{i_j})$, since $s(e_{i_j})$ has same chance to be 1 and -1 and is independent from each other, cross terms in $E(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2$ are 0. Thus, we have $E(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2 = \sum_{j=1}^{l} v_{i_j}^2$. Therefore, we can get that

$$Var(\hat{f}_i) = \sum_{j=1}^{l} v_{i_j}^2 \cdot (d'^2 + \frac{1}{12})\Delta t^2 + \frac{1}{12}v_i^2\Delta t^2. \quad (32)$$

Moreover, since there are b estimators in each matrix and each key is hashed into one estimator, we have $E(\sum_{j=1}^{l} v_{i_j}^2) = \frac{1}{b}\sum_{j\neq i} v_j^2$. Therefore, the estimated variance satisfies

$$Var(\hat{f}_i) = \frac{1}{b}\sum_{j\neq i} v_j^2 \cdot (d'^2 + \frac{1}{12})\Delta t^2 + \frac{1}{12}v_i^2\Delta t^2. \quad (33)$$

□

Below we prove that the theorems still hold for two optimizations. Without loss of generality, we still take $s(e_i) = 1$ in each counter.

Column Randomizing: When adopting the option, $s(\ )$ in each column are independent. Thus, similar to the proving process above, we have

$$
\begin{aligned}
Var(\hat{f}_i) &= E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2] \quad ((wt)^2 + d^0 \quad t^2) + E[v_i^2 \quad w^2t^2] \\
&= \sum_{j=1}^{l} v_{i_j}^2 \quad (d^0 + \frac{1}{12})t^2 + \frac{1}{12}v_i^2t^2:
\end{aligned} \tag{34}
$$

Therefore, when adopting Column Randomizing, we have the variance $Var(\hat{f}_i)$ follows

$$Var(\hat{f}_i) = \frac{1}{b}\sum_{j \neq i} v_j^2 \quad (d^0 + \frac{1}{12})t^2 + \frac{1}{12}v_i^2t^2: \tag{35}$$

Linear Scaling: When adopting the option, we eliminate the error caused by the deviation of time. Thus, we can get that

$$Var(\hat{f}_i) = E[(\sum_{j=1}^{l} v_{i_j} s(e_{i_j}))^2 \quad d^{00}t^2] = \sum_{j=1}^{l} v_{i_j}^2 \quad d^{00}t^2: \tag{36}$$

Therefore, when adopting Linear Scaling, we have the variance $Var(\hat{f}_i)$ follows

$$Var(\hat{f}_i) = \frac{1}{b}\sum_{j \neq i} v_j^2 \quad d^{00}t^2: \tag{37}$$

4.4 Error Bound

In this section, we show the error bound of $\hat{f}_i$ in Theorem 4, and then we show the error bound of our Unbiased Cleaning sketch in Theorem 5 if we take the mean and in Theorem 6 if we take the median. In addition, for the median case, we also show a more accurate error bound written in summation form in Theorem 7.

Theorem 4. For a given that > 0, we have

$$Prf \ \hat{f}_i \quad f_i > \ g \leqslant \frac{1}{^2}Var(\hat{f}_i): \tag{38}$$

Proof. As proved in theorem 1, $E(\hat{f}_i) = f_i$. According to Chebyshev inequality, we can easily get that

$$
\begin{aligned}
Prf \ \hat{f}_i \quad f_i > \ g &\leqslant \frac{1}{^2}Var(\hat{f}_i) \\
&= \frac{1}{^2 b}\sum_{j \neq i} v_j^2 \quad (m^{00} + \frac{1}{12})t^2 + \frac{1}{12\ ^2}v_i^2t^2;
\end{aligned} \tag{39}
$$

from which we get the error bound of $\hat{f}_i$, i.e., the error bound of the estimated flow size by a matrix . □

If we take the mean, i.e., the estimation of the flow size e_i by our Unbiased Cleaning sketch is $mean(\hat{f}_i)$, we can derive an error bound for our Unbiased Cleaning sketch.

Theorem 5. For a given that > 0, we have

$$Prf \ mean(\hat{f}_i) \quad f_i > \ g \leqslant \frac{1}{^2 k}Var(\hat{f}_i): \tag{40}$$

Proof. For $mean(\hat{f}_i)$, we can get the variance that $Var(mean(\hat{f}_i)) = \frac{1}{k}Var(\hat{f}_i)$. Since $mean(\hat{f}_i)$ is unbiased, according to Chebyshev inequality, we have

$$
\begin{aligned}
Prf \ mean(\hat{f}_i) \quad f_i > \ g &\leqslant \frac{1}{^2}Var(mean(\hat{f}_i)) \\
&= \frac{1}{^2 k}Var(\hat{f}_i):
\end{aligned} \tag{41}
$$

□

If we take the median, i.e., the estimation of the flow size e_i by our Unbiased Cleaning sketch is $median(\hat{f}_i)$, let f_i be $\hat{f}_i \quad f_i$. We can also derive an error bound for our Unbiased Cleaning sketch. Here we just consider the case that the number of the arrays is odd. Otherwise, we can only use $k \quad 1$ arrays.

Theorem 6. For a given that $> \sqrt{2Var(\hat{f}_i)}$, we derive an error bound of our Unbiased Cleaning sketch that

$$
\begin{aligned}
&Prf \ median(\hat{f}_i) \quad f_i > \ g \\
&\leqslant \frac{(2r+1)!}{(r!)^2} \quad 1 \quad \left(\frac{Var(\hat{f}_i)}{^2}\right)^{r} \left(\frac{Var(\hat{f}_i)}{^2}\right)^{r+1}:
\end{aligned} \tag{42}
$$

Proof. Let $k = 2r + 1$. We can get a $\hat{f}_i$ from each array and they are i.i.d. Let $f(x)$ be the probability density function of $f_i = \hat{f}_i \quad f_i$, and $F(x)$ be the cumulative distribution function of f_i.

Let $\hat{f}_i$ from $(2r+1)$ arrays be $\hat{f}_i^{(1)}; \hat{f}_i^{(2)}; \quad ; \hat{f}_i^{(2r+1)}$ in order from small to large. Then we have $median(\hat{f}_i) = \hat{f}_i^{(r+1)}$. Let $f^{(r+1)}(x)$ be the probability density function of $f_i^{(r+1)} = median(\hat{f}_i) \quad f_i$, and $F^{(r+1)}(x)$ be the cumulative distribution function of $f_i^{(r+1)}$. By the theorem of order statistics, we can get that

$$f^{(r+1)}(x) = \frac{(2r+1)!}{(r!)^2}F(x)^r(1 \quad F(x))^r f(x): \tag{43}$$

We have $F(x) = 1 \quad Prf \ f_i > xg$ for $x > 0$ and $F(x) = Prf \ f_i \leqslant xg$ for $x \leqslant 0$. Let $P(x)$ be $Prf \ f_i > xg$ when $x > 0$ and $Prf \ f_i \leqslant xg$ when $x \leqslant 0$. Then we have $F(x)(1 \quad F(x)) = P(x)(1 \quad P(x))$. We can also get that $P(x) \leqslant Prfj \ f_i j > jxjg$.

As proved in Theorem 4, $Prfj \ f_i j > jxjg \leqslant \frac{1}{x^2}Var(\hat{f}_i)$. For x that $jxj > \sqrt{2Var(\hat{f}_i)}$, we have

$$P(x) \leqslant Prfj \ f_i j > jxjg \leqslant \frac{1}{x^2}Var(\hat{f}_i) \leqslant \frac{1}{2}: \tag{44}$$

from which we can derive that

$$
\begin{aligned}
F(x)(1 \quad F(x)) &\leqslant (1 \quad Prfj \ f_i j > xg)Prfj \ f_i j > xg \\
&\leqslant (1 \quad \frac{1}{x^2}Var(\hat{f}_i))(\frac{1}{x^2}Var(\hat{f}_i)):
\end{aligned} \tag{45}
$$

Therefore, for the median $f^{(r+1)}(x)$, we can get that

$$f^{(r+1)}(x) \leqslant \frac{(2r+1)!}{(r!)^2} \quad 1 \quad \left(\frac{Var(\hat{f}_i)}{x^2}\right)^{r} \left(\frac{Var(\hat{f}_i)}{x^2}\right)^{r} f(x): \tag{46}$$

For a given that $> \sqrt{2Var(\hat{f}_i)}$ and for x that $jxj >$, since $\frac{1}{x^2}Var(\hat{f}_i) \leqslant \frac{1}{^2}Var(\hat{f}_i) \leqslant \frac{1}{2}$, we have the inequality

$(1-\frac{1}{x^2}Var(\hat{f}_i))(\frac{1}{x^2}Var(\hat{f}_i)) \leqslant (1-\frac{1}{\varepsilon^2}Var(\hat{f}_i))(\frac{1}{\varepsilon^2}Var(\hat{f}_i))$. Thus, we can get that

$$\begin{aligned}
&\int_{\varepsilon}^{+\infty} f^{(r+1)}(x)\,dx \\
&\leqslant \int_{\varepsilon}^{+\infty} \frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{x^2}\right)^r\left(\frac{Var(\hat{f}_i)}{x^2}\right)^r f(x)\,dx \\
&\leqslant \int_{\varepsilon}^{+\infty} f(x)\,dx\,\frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r\left(\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r.
\end{aligned} \tag{47}$$

Similarly, we have

$$\begin{aligned}
&\int_{-\infty}^{-\varepsilon} f^{(r+1)}(x)\,dx \\
&\leqslant \int_{-\infty}^{-\varepsilon} f(x)\,dx\,\frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r\left(\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r.
\end{aligned} \tag{48}$$

From the above results in Equation 47 and Equation 48, we can get that $median(\hat{f}_i)$ follows

$$\begin{aligned}
&Pr\{|median(\hat{f}_i)-f_i|>\varepsilon\} \\
&= \int_{\varepsilon}^{+\infty} f^{(r+1)}(x)\,dx + \int_{-\infty}^{-\varepsilon} f^{(r+1)}(x)\,dx \\
&\leqslant Pr\{|\hat{f}_i-f_i|>\varepsilon\}\,\frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r\left(\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r \\
&\leqslant \frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r\left(\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^{r+1}.
\end{aligned} \tag{49}$$

Therefore, for a given ε that $\varepsilon > \sqrt{2Var(\hat{f}_i)}$, our Unbiased Cleaning sketch has an error bound that

$$\begin{aligned}
&Pr\{|median(\hat{f}_i)-f_i|>\varepsilon\} \\
&\leqslant \frac{(2r+1)!}{(r!)^2}\left(1-\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^r\left(\frac{Var(\hat{f}_i)}{\varepsilon^2}\right)^{r+1}.
\end{aligned} \tag{50}$$

□

Theorem 6 shows an error bound that can be directly calculated. In addition to that, we can derive a more accurate error bound written in summation form.

Theorem 7. Let $P = Var(\hat{f}_i)/\varepsilon^2$. For a given ε that $\varepsilon > \sqrt{2Var(\hat{f}_i)}$, we can derive an error bound for $median(\hat{f}_i)$ that

$$\begin{aligned}
&Pr\{|median(\hat{f}_i)-f_i|>\varepsilon\} \\
&\leqslant \sum_{k=r+1}^{2r+1}\binom{2r+1}{k}P^k(1-P)^{2r+1-k}
\end{aligned} \tag{51}$$

Proof. If the error of our estimation is greater than ε, i.e., $|median(\hat{f}_i)-f_i|>\varepsilon$, then there are at least $(r+1)$ arrays satisfy that $\hat{f}_i-f_i>\varepsilon$ or at least $(r+1)$ arrays satisfy that $\hat{f}_i-f_i\leqslant-\varepsilon$. Thus, at least $(r+1)$ arrays satisfy that $|\hat{f}_i-f_i|>\varepsilon$. According to theorem 4, we have

$$Pr\{|\hat{f}_i-f_i|>\varepsilon\}\leqslant P. \tag{52}$$

Let n_0 be the number of arrays that satisfy $|\hat{f}_i-f_i|>\varepsilon$. For a given ε that $\varepsilon > \sqrt{2Var(\hat{f}_i)}$, we have $P\leqslant\frac{1}{2}$. We can get that

$$\begin{aligned}
Pr\{n_0\geqslant r+1\} &= \sum_{k=r+1}^{2r+1} Pr\{n_0=k\} \\
&= \sum_{k=r+1}^{2r+1}\binom{2r+1}{k}Pr^k(1-Pr)^{2r+1-k} \\
&\leqslant \sum_{k=r+1}^{2r+1}\binom{2r+1}{k}P^k(1-P)^{2r+1-k}.
\end{aligned} \tag{53}$$

Here $Pr = Pr\{|\hat{f}_i-f_i|>\varepsilon\}$. Therefore, we can derive an error bound written in summation form that

$$\begin{aligned}
Pr\{|median(\hat{f}_i)-f_i|>\varepsilon\} &\leqslant Pr\{n_0\geqslant r+1\} \\
&\leqslant \sum_{k=r+1}^{2r+1}\binom{2r+1}{k}P^k(1-P)^{2r+1-k}.
\end{aligned} \tag{54}$$

□

4.5 Analysis of Robustness

In this section, we show that our Unbiased Cleaning sketch has a property of robustness, which guarantees superiority of estimation for keys whose flow size exceeds a certain percentage of the total.

Let f_i^0 be the exact flow size of key e_i recorded in the estimator in a matrix, i.e., the size of flow e_i during period $(T_c - dt, T)$. Without loss of generality, we take $f_1^0 \geqslant f_2^0 \geqslant \cdots \geqslant f_n^0$.

Theorem 8. If $f_1^0 > 0.5\sum_{i=1}^{n} f_i^0$, then e_1 is the key with largest estimated flow size by this matrix, i.e., $\hat{f}_1 = \max_{1\leqslant i\leqslant n}\hat{f}_i$.

Proof. Without loss of generality, we take $s(e_1)=1$. Below we prove that for any key e_i other than e_1, we always have $\hat{f}_i \leqslant \hat{f}_1$.

If e_i and e_1 are hashed in the same estimator, let $\hat{f}$ be the number recorded in this estimator. We have $\hat{f}_i = \hat{f}\cdot s(e_i)$ and $\hat{f}_1 = \hat{f}\cdot s(e_1) = \hat{f}$. Thus, the estimated flow size for e_i and e_1 satisfy $\hat{f}_i = \hat{f}_1$ or $\hat{f}_i = -\hat{f}_1$. Since $s(e_1)=1$, we have

$$\hat{f} = \sum_{j=1}^{n} f_j^0\cdot s(e_j) \geqslant f_1^0 - \sum_{j=2}^{n} f_j^0 > 0; \tag{55}$$

thus $\hat{f}_1 > 0$. Therefore, we can get that $\hat{f}_1 \geqslant \hat{f}_i$.

Next, we consider the case that they are in different estimators, i.e., $h(e_i)\neq h(e_1)$. Let $e_{i_1}, e_{i_2}, \cdots, e_{i_l}$ be the keys inserted to the same estimator as e_i, and $e_{1_1}, e_{1_2}, \cdots, e_{1_{l'}}$ be the keys inserted to the same estimator as e_1. Then we have the estimation of size of flow e_i that

$$\hat{f}_i = \left(f_i^0\cdot s(e_i) + \sum_{j=1}^{l} f_{i_j}^0\cdot s(e_{i_j})\right)\cdot s(e_i); \tag{56}$$

and the estimation of the flow size of e_1 that

$$\begin{aligned}
\hat{f}_1 &= \left(f_1^0\cdot s(e_1) + \sum_{j=1}^{l'} f_{1_j}^0\cdot s(e_{1_j})\right)\cdot s(e_1) \\
&= f_1^0 + \sum_{j=1}^{l'} f_{1_j}^0\cdot s(e_{1_j}).
\end{aligned} \tag{57}$$

For key e_j, it's inserted into at most one estimator. Thus, we can derive that

$$\hat{f}_1 - \hat{f}_i = f_1^0 + \sum_{j=1}^{l^0} f_{1_j}^0 - s(e_{1_j}) - (f_i^0 - s(e_i) + \sum_{j=1}^{l} f_{i_j}^0 - s(e_{i_j}))s(e_i)$$
$$\geq f_1 - \sum_{j=1}^{l^0} f_{1_j}^0 - f_i^0 - \sum_{j=1}^{l} f_{i_j}^0 \geq f_1^0 - \sum_{j=2}^{n} f_j^0 \geq 0. \quad (58)$$

Therefore, we always have $\hat{f}_i \leq \hat{f}_1$. In other words, e_1 is the key with the largest estimated flow size given by this matrix, i.e., $\hat{f}_1 = \max_{1 \leq i \leq n} \hat{f}_i$. □

According to analysis in theorem 1, we have $T_c - (m-1)t \in (T - mt, T - (m-1)t]$. Thus, if $\forall T_c - (m-1)t \in (T - mt, T - (m-1)t]$ we have $f_1^0 \geq 0.5 \sum_{i=1}^{n} f_i^0$, then we get $\hat{f}_1 = \max_{1 \leq i \leq n} \hat{f}_i$ in each matrix. We can derive the theorem below.

Theorem 9. If $\forall T_c - (m-1)t \in (T - mt, T - (m-1)t]$ we have $f_1^0 \geq 0.5 \sum_{i=1}^{n} f_i^0$, then e_1 is always the key with the largest flow size due to estimation of our Unbiased Cleaning sketch, either we take the median or the mean.

Proof. First, if we take the mean, since in each matrix we have $\hat{f}_1 = \max_{1 \leq i \leq n} \hat{f}_i$, then $\text{mean}(\hat{f}_1) = \max_{1 \leq i \leq n} \text{mean}(\hat{f}_i)$.

Second, if we take the median, we assume that there exists an key e_i that satisfies $\text{median}(\hat{f}_i) > \text{median}(\hat{f}_1)$, i.e., $\hat{f}_i^{(r+1)} > \hat{f}_1^{(r+1)}$. Thus, for $1 \leq l_1 \leq r+1$ and $r+1 \leq l_2 \leq 2r+1$, we have $\hat{f}_1^{(l_1)} \leq \hat{f}_1^{(r+1)} < \hat{f}_i^{(r+1)} \leq \hat{f}_i^{(l_2)}$. Since $\hat{f}_1^{(l_1)}$ and $\hat{f}_i^{(l_2)}$ are both from $(r+1)$ different matrices, there is at least a matrix which contains both $\hat{f}_1^{(l_1)}$ and $\hat{f}_i^{(l_2)}$. In this matrix, we have $\hat{f}_1 < \hat{f}_i$, which contradicts the conclusion that $\hat{f}_1 = \max_{1 \leq i \leq n} \hat{f}_i$ in each matrix. This indicates that our assumption is not true. Thus, We always have $\text{median}(\hat{f}_1) = \max_{1 \leq i \leq n} \text{median}(\hat{f}_i)$.

Therefore, under either cases we always have the conclusion that e_1 is the key with the largest flow size due to estimation of our Unbiased Cleaning sketch. □

Particularly, under the assumption that the flow size is proportional to duration over a period of time, if $v_1 \geq 0.5 \sum_{i=1}^{n} v_i$, then we have $\hat{f}_1 = \max_{1 \leq i \leq n} \hat{f}_i$ in each matrix. We can easily prove that $\text{mean}(\hat{f}_1) = \max_{1 \leq i \leq n} \text{mean}(\hat{f}_i)$ and $\text{median}(\hat{f}_1) = \max_{1 \leq i \leq n} \text{median}(\hat{f}_i)$, which indicates that e_1 is always the key with the largest estimated flow size due to our Unbiased Cleaning sketch. This inference still holds for two optimizations.

4.6 Analysis for Zipfian Distribution

In this section, we assume that the flow sizes complies with Zipfian distribution. Under the assumption that the flow size is proportional to duration, the flow size in unit time $v_1, v_2, \cdots, v_n$ also comply with Zipfian distribution. Let $v_1, v_2, \cdots, v_n$ satisfy $v_i = \frac{c}{i^z}$.

Below we show that Theorem 9 about the robustness is satisfied under the assumption of Zipfian distribution, and we also give out the magnitude of the variance.

Theorem 10. For Zipfian distribution with parameter z, i.e., $v_i = \frac{c}{i^z}$, if $z \geq 2$, then e_1 is always the key with the largest flow size due to estimation by our Unbiased Cleaning sketch.

Proof. In the case that $z \geq 2$, we have

$$\sum_{i=2}^{n} v_i \leq \sum_{i=2}^{n} \frac{c}{i^2} \leq \sum_{i=2}^{n} \frac{c}{(i-1) \cdot i} \leq c = v_1 \quad (59)$$

Therefore, we have $v_1 \geq 0.5 \sum_{i=1}^{n} v_i$. According to Theorem 9, e_1 is always the key with the largest flow size due to estimation by our Unbiased Cleaning sketch. □

Theorem 11. For Zipfian distribution with parameter z, i.e., $v_i = \frac{c}{i^z}$, the magnitude of the variance satisfies that

$$Var(\hat{f}_i) = \begin{cases} O(n^{1-2z}) & z < \frac{1}{2} \\ O(\log(n)) & z = \frac{1}{2} \\ O(1) & z > \frac{1}{2} \end{cases} \quad (60)$$

Proof. In Equation 33 we have $Var(\hat{f}_i) = O(\sum v_i^2)$. For the magnitude of $\sum v_i^2 = \sum_{i=1}^{n} \frac{c}{i^{2z}}$, we have

$$\sum_{i=1}^{n} \frac{c}{i^{2z}} \approx \int_1^n \frac{c}{x^{2z}} dx = \begin{cases} \frac{1}{1-2z}(n^{1-2z} - 1) & z \neq \frac{1}{2} \\ c \cdot \log(n) & z = \frac{1}{2} \end{cases} \quad (61)$$

Here z and c are fixed constant. We have the magnitude of $Var(\hat{f}_i)$ equals to the magnitude of $\sum v_i^2$. Therefore, we can get that the magnitude of $Var(\hat{f}_i)$ is $O(n^{1-2z})$ when $z < \frac{1}{2}$, $O(\log(n))$ when $z = \frac{1}{2}$, and $O(1)$ when $z > \frac{1}{2}$, as shown in Equation 60. □

5 Experimental Results

5.1 Experimental Setup

Implementation: We have implemented the Unbiased Cleaning sketch and all other algorithms in C++. The hash functions are the 32-bit Bob Hash [71] with random seeds.

Datasets: We use two kinds of datasets in the experiments: the CAIDA dataset and the synthetic Zipfian dataset. The CAIDA dataset is streams of anonymized IP traces collected in 2016 and 2018 by CAIDA [72]. CAIDA-2016 has 30 million packets from 0.6 million flows, while CAIDA-2018 has 27 million packets from 1.3 million flows. We use the pair of source IP address (4 bytes) and destination IP address (4 bytes) as an 8-byte key, which is a common flow identifier. The synthetic Zipfian dataset is generated by ourselves and follows the Zipfian distribution by using Web Polygraph [73], an open-source performance testing tool. Each Zipfian dataset has 32 million keys, the skewness of datasets varies from 0.3 to 3.0, and the length of each key is 4 bytes.

Computation Platform: We conducted all the experiments on a machine with two 6-core processors (12 threads, Intel Xeon CPU E5-2620 @2 GHz) and 64 GB DRAM memory. Each processor has three levels of cache memory: one 32KB L1 data caches and one 32KB L1 instruction cache for each core, one 256KB L2 cache for each core, and one 15MB L3 cache shared by all cores.

Metrics:

1) Average Relative Error (ARE): $\frac{1}{| \ |}\sum_{e_i \in \ } |f_i - \hat{f}_i|/f_i$, where f_i is the real flow size of e_i, $\hat{f}_i$ is its estimated flow size, and is the query set. Here, we randomly pick a checkpoint to stop inserting, query the actual flow sizes in the sliding window, and calculate ARE.

2) Average Absolute Error (AAE): $\frac{1}{| \ |}\sum_{e_i \in \ } |f_i - \hat{f}_i|$, where f_i is the real flow size of e_i, $\hat{f}_i$ is its estimated flow size, and is the query set. Here, we randomly pick a checkpoint to stop inserting, query the actual flow sizes in the sliding window and calculate AAE.

3) Throughput: Million insertions per second (Mips). All the experiments about throughput are repeated 10 times and the average throughput is reported.

5.2 Experimental Evaluation

In this section, we present the AAE, ARE, and throughput to evaluate the performance of the Unbiased Cleaning sketch. We additionally show the result of heavy flows because heavy flows are often what we care about in application. Here we define heavy flows as flows that account for over 0.5% of the total number of packets. We change the settings of the number of matrices, the number of counters per estimator, the memory size, the window size, and the data skewness to show how the performance of the UC sketch depends on parameters. The default values of above parameters are as follows: number of matrices $k = 2$, number of counters per estimator $m = 4$, memory size = 500KB, window size = 1×10^4 and the default dataset is the CAIDA-2016 dataset.

(a) ARE (b) AAE

Fig. 2: Impact of Number of Matrices.

Impact of Number of Matrices (Figure 2(a)-2(b)): We find that the ARE and AAE of all flows goes higher when the number of matrices (k) goes larger, while the ARE and AAE of the heavy flows goes lower when the number of matrices goes larger. The reason for this is that, under a fixed total memory, increasing the value of k leads to a larger average size of the counters, which can decrease the accuracy of light flows. However, heavy flows can reduce the error more effectively with a larger value of m, because their sizes are much larger than the average counter size. The category "Basic" refers to the UC Sketch without the LS or the CR optimizations. The category "OPT" refers to the UC Sketch with both the LS and the CR optimizations. Based on the default settings, we change the number of matrices from 1 to 6. The ARE of Basic is on average 0.19 time higher than the ARE of OPT on all flows. The AAE of Basic is on average 0.03 time higher than the AAE of OPT on all flows. The ARE and AAE of Basic and OPT are similar on heavy flows.

(a) Number of Counters per Estimator (b) Memory Sizes

(c) Window Sizes (d) Data Skewness

Fig. 3: Impact on Accuracy.

(a) Time-Based (b) Count-Based

Fig. 4: Comparison with Sliding Sketch.

(a) Impact of # Matrices (b) Impact of Optimizations

Fig. 5: Impact on Throughput.

Impact of Number of Counters per Row (Figure 3(a)): We find that the ARE goes higher when the number of counters per row (m) goes too large or too small. If m is too large, hash collisions can cause significant errors, while if m is too small, the window bias can cause significant errors. Based on the default settings, we change the number of counters per row from 1 to 6. The ARE of Basic is on average 0.07 time higher than the ARE of OPT on all flows. The ARE of Basic and OPT are similar on heavy flows.

(a) Subset Sum (b) Distributed Sum

Fig. 6: Performance of Sum Queries.

Impact of Memory Sizes (Figure 3(b)): We nd that the ARE goes lower when the memory size goes larger. Based on the default settings, we change the memory size from 100KB to 500KB. The ARE of Basic is on average 0.19 time higher than the ARE of OPT on all ows. The ARE of Basic is on average 0.18 time higher than the ARE of OPT on heavy ows.

Impact of Window Sizes (Figure 3(c)): We nd that, when the memory is xed, for all ows, the ARE is in nearly direct ratio to the window size as it goes larger. For heavy ows, the ARE also goes higher when the window size goes larger. Based on the default settings, we change the window size from 1 10^4 to 16 10^4. The ARE of Basic is on average 0.25 time higher than the ARE of OPT on all ows. The ARE of Basic and OPT are similar on heavy ows.

Impact of Data Skewness (Figure 3(d)): We nd that, for all ows, the ARE goes lower when the data skewness goes higher. For heavy ows, the ARE goes higher when the data skewness goes higher. Here, we use synthetic Zip an dataset to show the performance of Unbiased Cleaning sketch under different the data skewness. Based on the default settings, we change the data skewness from 0:6 to 3:0. The ARE of Basic is on average 0.27 time higher than the ARE of OPT on all ows. The ARE of Basic is on average 0.30 time higher than the ARE of OPT on heavy ows.

Comparison with Sliding Sketch (Figure 4(a)-4(b)): The experiment results show that our UC Sketch performs much better than Sliding Sketch on both the time-based sliding window and the count-based sliding window. We use the dataset CAIDA-2018 to conduct the time-based experiment, and CAIDA-2016 to conduct the count-based experiment. The ARE of Sliding Sketch is on average 3.92 higher than the ARE of UC Sketch on the time-based sliding window. The ARE of Sliding Sketch is on average 1.68 higher than the ARE of UC Sketch on the count-based sliding window.

Experiments on Throughput (Figure 5(a)-5(b)): We nd that the throughput goes lower when the number of matrices goes larger. The LS and CR optimizations bring little throughput loss. In Figure 5(a), we change the number of matrices from 1 to 21. The throughput is higher than 1:03Mips when the number of matrices is not larger than 13. In Figure 5(b), the LS optimization brings nearly no throughput loss, and the CR optimization brings less than 3% throughput loss.

Performance of Sum Queries (Figure 6(a)-6(b)): We applied our algorithm to subset sum query and distributed sum query. We set the subset size from 1 to 256, exponentially, and the number of distributed nodes from 1 to 16, exponentially. We nd that, the ARE goes lower when the subset size goes larger in subset sum query, and the ARE also goes lower when the number of sketches (nodes) goes larger in distributed sum query. Note that here in distributed sum query, a key is taken into account only when it appears in every sketch.

6 CONCLUSION

The sliding window model can capture the latest characteristics of network data streams in real-time. Achieving unbiased estimation in sliding windows is challenging and signi cant in network measurement. In this paper, we propose an algorithm called Unbiased Cleaning sketch. It is the rst work that achieves unbiased ow size estimation in sliding windows. The Unbiased Cleaning sketch is both mean-unbiased and median-unbiased. Its two optimization techniques, *Linear Scaling* and *Column Randomizing* drastically reduce the variance. By strict mathematical analysis, we prove the unbiasedness and show other properties of the Unbiased Cleaning sketch. The experiment results are consistent with the theory. All related source codes are open-sourced at Github [74].

ACKNOWLEDGMENT

We thank all anonymous reviewers for their help in improving this paper. This work is supported by multiple grants, including the Key-Area Research and Development Program of Guangdong Province 2020B0101390001 and the National Natural Science Foundation of China (NSFC) (No. U20A20179).

REFERENCES

[1] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network ow monitoring with univmon. In *Proc. of ACM SIGCOMM*, 2016.

[2] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. Elastic sketch: adaptive and fast network-wide measurements. In *SIGCOMM 2018*, pages 561–575. ACM, 2018.

[3] Hao Zheng, Chen Tian, Tong Yang, Huiping Lin, Chang Liu, Zhaochen Zhang, Wanchun Dou, and Guihai Chen. Flymon: enabling on-the- y task recon guration for network measurement. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 486–502, 2022.

[4] Peiqing Chen, Yuhan Wu, Tong Yang, Junchen Jiang, and Zaoxing Liu. Precise error estimation for sketch-based ow measurement. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 113–121, 2021.

[5] Jiaqian Liu, Haipeng Dai, Rui Xia, Meng Li, Ran Ben Basat, Rui Li, and Guihai Chen. Duet: A generic framework for nding special quadratic elements in data streams. In *Proceedings of the ACM Web Conference 2022*, pages 2989–2997, 2022.

[6] Hun Namkung, Zaoxing Liu, Daehyeok Kim, Vyas Sekar, Peter Steenkiste, G Liu, A Li, C Canel, AA Philip, R Ware, et al. Sketchlib: Enabling ef cient sketch-based monitoring on programmable switches. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 743–759, 2022.

[7] Haiting Zhu, Hao Zhang, Junmei Wan, Gaofeng He, and Lu Zhang. Introducing additional network measurements into active queue management. In *ICC 2022-IEEE International Conference on Communications*, pages 2303–2308. IEEE, 2022.

[8] Chaoyi Ma, Olufemi O Odegbile, Dimitrios Melissourgos, Haibo Wang, and Shiping Chen. From countmin to super kjoin sketches for ow spread estimation. *IEEE Transactions on Network Science and Engineering*, 2023.

[9] Rui Ding, Shibo Yang, Xiang Chen, and Qun Huang. Bitsense: Universal and nearly zero-error optimization for sketch counters with compressive sensing. In *Proc. of SIGCOMM*, 2023.

[10] Intel tofino. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html.

[11] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. MicroTE: Fine Grained Traffic Engineering for Data Centers. In *Proc. of ACM CoNEXT*, 2011.

[12] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. pFabric: Minimal Near-optimal Datacenter Transport. In *Proc. of ACM SIGCOMM*, 2013.

[13] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. CONGA: Distributed Congestion-aware Load Balancing for Datacenters. In *Proc. of ACM SIGCOMM*, 2014.

[14] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. Netcache: Balancing key-value stores with fast in-network caching. In *SOSP 2017*, pages 121–136. ACM, 2017.

[15] Lei Ying, R. Srikant, and Xiaohan Kang. The Power of Slightly More than One Sample in Randomized Load Balancing. In *Proc. of IEEE INFOCOM*, 2015.

[16] Graham Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1), 2005.

[17] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 2004.

[18] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *ACM SIGMCOMM CCR*, 32(4), 2002.

[19] Yinda Zhang, Zaoxing Liu, Ruixin Wang, Tong Yang, Jizhou Li, Ruijie Miao, Peng Liu, Ruwen Zhang, and Junchen Jiang. Cocosketch: high-performance sketch-based measurement over arbitrary partial key query. In *SIGCOMM 2021*, pages 207–222. ACM, 2021.

[20] Peiqing Chen, Dong Chen, Lingxiao Zheng, Jizhou Li, and Tong Yang. Out of many we are one: Measuring item batch with clock-sketch. In *Proceedings of the 2021 International Conference on Management of Data*, pages 261–273, 2021.

[21] Zheng Zhong, Shen Yan, Zikun Li, Decheng Tan, Tong Yang, and Bin Cui. Burstsketch: Finding bursts in data streams. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2375–2383, 2021.

[22] Mikhail Nikulin and Vassiliy Voinov. *Unbiased Estimators and Their Applications*, pages 1619–1621. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[23] P. Tchebichef. Des valeurs moyennes. *Journal de Mathématiques Pures et Appliquées. 2. 12*, page 177–184, 1867.

[24] Yao Zhao, Yan Chen, and David Bindel. Towards unbiased end-to-end network diagnosis. *ACM SIGCOMM Computer Communication Review*, 36(4):219–230, 2006.

[25] Yufei Zheng, Xiaoqi Chen, Mark Braverman, and Jennifer Rexford. Unbiased delay measurement in the data plane. In *Symposium on Algorithmic Principles of Computer Systems (APOCS)*, pages 15–30. SIAM, 2022.

[26] Yikai Zhao, Wenchen Han, Zheng Zhong, Yinda Zhang, Tong Yang, and Bin Cui. Double-anonymous sketch: Achieving fairness for finding global top-k frequent items. In *Proc. of ACM SIGMOD*, 2023.

[27] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proc. of ICALP*, 2002.

[28] Fan Deng and Davood Rafiei. New estimation algorithms for streaming data: Count-min can do more. *Webdocs. Cs. Ualberta. Ca*, 2007.

[29] Tao Li, Shigang Chen, and Yibei Ling. Per-flow traffic measurement through randomized counter sharing. *IEEE/ACM Transactions on Networking*, 20(5):1622–1634, 2012.

[30] Xiang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Zengfeng Huang. Skype: top-k spatial-keyword publish/subscribe over sliding window. *Proceedings of the VLDB Endowment*, 9(7):588–599, 2016.

[31] Kanat Tangwongsan, Martin Hirzel, and Scott Schneider. Optimal and general out-of-order sliding-window aggregation. *Proceedings of the VLDB Endowment*, 12(10):1167–1180, 2019.

[32] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 283–293. IEEE, 2007.

[33] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A Tucker. No pane, no gain: efficient evaluation of sliding-window aggregates over data streams. *Acm Sigmod Record*, 34(1):39–44, 2005.

[34] Cheqing Jin, Ke Yi, Lei Chen, Jeffrey Xu Yu, and Xuemin Lin. Sliding-window top-k queries on uncertain streams. *Proceedings of the VLDB Endowment*, 1(1):301–312, 2008.

[35] Kanat Tangwongsan, Martin Hirzel, Scott Schneider, and Kun-Lung Wu. General incremental sliding-window aggregation. *Proceedings of the VLDB Endowment*, 8(7):702–713, 2015.

[36] Yun Chi, Haixun Wang, Philip S Yu, and Richard R Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 59–66. IEEE, 2004.

[37] Yufei Tao and Dimitris Papadias. Maintaining sliding window skylines on data streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):377–391, 2006.

[38] Hua-Fu Li and Suh-Yin Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert systems with applications*, 36(2):1466–1477, 2009.

[39] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. Sketching linear classifiers over data streams. In *Proceedings of the 2018 International Conference on Management of Data*, pages 757–772, 2018.

[40] Yang Zhou, Tong Yang, Jie Jiang, Bin Cui, Minlan Yu, Xiaoming Li, and Steve Uhlig. Cold filter: A meta-framework for faster and more accurate stream processing. In *SIGMOD Conference*, 2018.

[41] Anshumali Shrivastava, Arnd Christian Konig, and Mikhail Bilenko. Time adaptive sketches (ada-sketches) for summarizing data streams. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1417–1432, 2016.

[42] Yanqing Peng, Jinwei Guo, Feifei Li, Weining Qian, and Aoying Zhou. Persistent bloom filter: Membership testing for the entire history. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1037–1052, 2018.

[43] Zhewei Wei, Ge Luo, Ke Yi, Xiaoyong Du, and Ji-Rong Wen. Persistent data sketching. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*, pages 795–810, 2015.

[44] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 61–72, 2002.

[45] Graham Cormode and Ke Yi. Tracking distributed aggregates over time-based sliding windows. In *International Conference on Scientific and Statistical Database Management*, pages 416–430. Springer, 2012.

[46] Ran Ben Basat, Gil Einziger, Michael Mitzenmacher, and Shay Vargaftik. Salsa: self-adjusting lean streaming analytics. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 864–875. IEEE, 2021.

[47] Hui Han, Zheng Yan, Xuyang Jing, and Witold Pedrycz. Applications of sketches in network traffic measurement: A survey. *Information Fusion*, 82:58–85, 2022.

[48] Haoyu Li, Qizhi Chen, Yixin Zhang, Tong Yang, and Bin Cui. Stingy sketch: a sketch framework for accurate and fast frequency estimation. *Proceedings of the VLDB Endowment*, 15(7):1426–1438, 2022.

[49] Sebastiano Miano, Xiaoqi Chen, Ran Ben Basat, and Gianni Antichi. Fast in-kernel traffic sketching in ebpf. *ACM SIGCOMM Computer Communication Review*, 53(1), 2023.

[50] Qun Huang, Xin Jin, Patrick P. C. Lee, Runhui Li, Lu Tang, Yi-Chao Chen, and Gong Zhang. Sketchvisor: Robust network measurement for software packet processing. In *SIGCOMM 2017*, pages 113–126. ACM, 2017.

[51] Qun Huang, Patrick P. C. Lee, and Yungang Bao. Sketchlearn: relieving user burdens in approximate measurement with automated statistical inference. In *SIGCOMM 2018*, pages 576–590. ACM, 2018.

[52] Pratanu Roy, Arijit Khan, and Gustavo Alonso. Augmented sketch: Faster and more accurate stream processing. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1449–1463, 2016.

[53] Jizhou Li, Zikun Li, Yifei Xu, Shiqi Jiang, Tong Yang, Bin Cui, Yafei Dai, and Gong Zhang. Wavingsketch: An unbiased and generic sketch for finding top-k items in data streams. In *Proceedings of the*