Unbiased Real-time Traffic Sketching

Yuhan Wu†, Shiqi Jiang†, Yifei Xu†, Siyuan Dong†, Kaicheng Yang†, Peiqing Chen ‡, Tong Yang‡
† 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 The key technique of the UC sketch is Unbiased Cleaning which can remove outdated keys from the sliding windows in a balanced way.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 anonymously.

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

1 INTRODUCTION

1.1 Background and Motivation

Sketches are widely used in network measurement [1]–[4] 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. 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 [5], [6] and load balancing [7]–[9]. 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 [10]–[13]. 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 [14], [15] 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 systematic errors [16] and provide the basis for further analysis, e.g., deriving error bounds based on variance and Chebyshev inequality [17]. In many applications, unbiased flow size measurement serves well in network diagnoses [18], delay distribution estimation [19], and heavy hitter detection [4], [20]. 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 [21], the CMM sketch [22], and the CSM [23] 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 time segment and $d$ counters recording the heaviness of the flows in the previous $d$ segments. At the end of the current
2 \textbf{Related Work} 

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

2.1 Preliminaries

Data Stream: a data stream \(S\) is a sequence of keys (e.g., 5-tuple of packet headers), \(i.e., S = \{e_1, e_2, e_3, \ldots \}\). 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 \[24\] \[32\].

2.2 Sketch-based Algorithms

Sketches \[10\], \[33\] \[39\] 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 \[10\], CU \[12\], Count \[21\], and Augmented \[40\].

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 \[41\], USS \[42\], and Coco \[13\]). 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 \(t \in s(\cdot)\). 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 \[41\], USS \[42\], and Coco \[13\], 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 \[43\] \[50\], there are three typical sketches, the ECM sketch \[51\], the Proportional Windowed Count-Min (PROPORTIONAL) \[52\], and the splitter windowed count-min sketch (SPLITTER) \[52\]. The ECM sketch combines the CM sketches and exponential histograms \[43\]. 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 \[53\] and SHE \[54\] 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 SPLINTER, 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 [55] and WCSS [56]. 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 [57], 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 the sliding window only stores keys of heavy flows. It 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

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

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.

![Insert Packet $e_i$](image)

**Unbiased Cleaning sketch**

- **Insert Packet $e_i$**
  - $h_i(e_i) = 3$, estimator $A[1][3]$
  - $R(T, 1, 3) = 2$, recent counter $A[1][3][2]$

**Function**

1. $h_i(e_i) = 3$
2. $R(T, 1, 3) = 2$
3. $A[1][3][2]$
4. $\text{A scanner moves periodically.}$
5. $\text{the most recently cleared counters in each estimator}$

**Fig. 1: Data Structure and Insertion Examples.**

3.1 Data Structure of Basic UC sketch

The data structure (Figure 1) of the UC sketch has $k$ matrices consisting of counters, $A[1], A[2], \ldots, A[k]$. 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 $d \times t$ time and at most $(d+1) \times t$ time. We set $t$ as $\frac{W}{\sum_{i=1}^{k} t}$, ensuring that each estimator records the average frequency for the most recent $W$ time.

#### Algorithm 1: Insertion for the UC sketch

1. **Function Insertion$(e_i)$:**
   2. **for** $p = 1$ to $k$ do
   3. $q \leftarrow h_p(e_i) \mod b$
   4. $r \leftarrow R(T, p, q)$
   5. $A[p][q][r] \leftarrow A[p][q][r] + s_p(e_i)$
   6. **end**

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 $L(p, q) = \frac{b(p-1)+(q-1)}{b} \cdot t$. **Insertion:** To insert key $e_i$, for each matrix $A[p]$, we calculate $h_p(e_i)$ to get its mapped row $\text{estimator } A[p][h_p(e_i)]$. 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:

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

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 outdated 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:

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

**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 \( \text{sum}_p \) by

\[
\text{sum}_p = s_p(e_i) \times \left( \sum_{i=1}^{d+1} A[p][q][i] \right).
\]

Finally, we return the mean of \( \{ \text{sum}_1, \text{sum}_2, \ldots, \text{sum}_k \} \) 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 \([d, d+1) \times t\) randomly. We hope that the estimators record the keys in the sliding window, whose size equals \( d + 0.5 \times t \). 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:

\[
C = \left[ 1.5 - \frac{T - L(p, q)}{t} \right] \times A[p][q][O(T, p, q)],
\]

where \( \{x\} \) denotes the fractional part of \( x \). Then, we calculate \( \text{sum}_p \) 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 \( \{ \text{sum}_1, \text{sum}_2, \ldots, \text{sum}_k \} \) 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 \( s_{p,r}() \), 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 \( \text{sum}_p \) by summing up the value of \( A[p][q][r] \times s_{p,r}(e_i) \), where \( r \) values from 1 to \( d + 1 \). Then, we return the median of \( \{ \text{sum}_1, \text{sum}_2, \ldots, \text{sum}_k \} \) 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 Section 4.3.

**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 \((T - (d + 1)t, T - dt)\). 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 \( s() \) is +1, and \( F_{-1}(T_1, T_2) \) be the flow sizes whose \( s() \) is −1. If \( T_1 \geq T_2 \), 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 \((T_C - dt, T)\), i.e.,

\[
\hat{f}_i = F_1(T_C - dt, T) - F_{-1}(T_C - dt, T).
\]

Let \( \hat{f}_i' \) be the estimation of size of flow \( e_i \) if there is no time deviation, which means \( \hat{f}_i' \) is the result recorded during period \( (T - (d + 0.5)t, T) \), i.e.,

\[
\hat{f}_i' = F_1(T - (d + 0.5)t, T) - F_{-1}(T - (d + 0.5)t, T).
\]

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

\[
E(\hat{f}_i) - \hat{f}_i = E(\hat{f}_i' - \hat{f}_i') + E(\hat{f}_i' - \hat{f}_i).
\]

Below we prove that \( E(\hat{f}_i - \hat{f}_i') = 0 \) and \( E(\hat{f}_i' - \hat{f}_i) = 0 \).
First, for the former term \((\hat{f}_i - \hat{f}_i')\), which is the error caused by time deviation, we have
\[
\hat{f}_i - \hat{f}_i' = F_1(T_C - dt, T - (d + 0.5)t) - F_1(T_C - dt, T - (d + 0.5)t).
\] (7)
Let \(T_1 = T_C - dt\) and \(T_2 = T - (d + 0.5)t\), then we have
\[
\hat{f}_i - \hat{f}_i' = F_1(T_1, T_2) - F_1(T_1, T_2).
\]
For \(T_1, T_2\), we have
\[
T_2 - T_1 = (T - (d + 0.5)t) - (T_C - dt) = -0.5t - T_C.
\] (8)
Because of the randomness of \(TC\), \(T_C\) follows a uniform distribution \(U(-0.5t, 0.5t)\), which indicates that \(T_1\) follows \(U(-0.5t, T_C + 0.5t)\). 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
\[
E(F_1(T_1, T_2)) = \int_{-0.5t}^{0.5t} F_1(T_2 - x, T_2) \, dx = 0.
\] (9)
Similarly, we have \(E(F_1(T_1, T_2)) = 0\). Thus, for the first part in Equation 6 we have
\[
E(\hat{f}_i - \hat{f}_i') = E(F_1(T_1, T_2)) - E(F_1(T_1, T_2)) = 0.
\] (10)
On the other hand, for the latter term in Equation 4 \(i.e., \hat{f}_i - f_i\), which is the error caused by hash collisions, we have
\[
\hat{f}_i - f_i = \sum_{e_i} f_i \cdot s(e_i),
\] (11)
where \(e_i\) is key inserted in this estimator other than \(e_i\) and \(f_i\) is the size of flow \(e_i\) during period \((T - (d + 0.5)t, T)\). \(s(e_i)\) has same chance to be 1 and -1 and is independent of \(f_i\). Thus, for the second part in Equation 6 we have
\[
E(\hat{f}_i' - f_i) = \sum_{e_i} [E(f_i) \cdot E(s(e_i))] = 0.
\] (12)
Therefore, by combining Equation 10 and Equation 12 we can get that
\[
E(\hat{f}_i) - f_i = E(\hat{f}_i - \hat{f}_i') + E(\hat{f}_i' - f_i) = 0,
\] (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
\[
E(\text{mean}(\hat{f}_i)) = E\left(\frac{1}{k} \sum \hat{f}_i\right) = \frac{1}{k} \sum E(\hat{f}_i) = \frac{1}{k} \cdot kf_i = f_i,
\] (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, \(\text{mean}(\hat{f}_i)\), is unbiased, \(i.e., \), \(E(\text{mean}(\hat{f}_i)) = f_i\).

Below we prove that the theorems 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 \(\hat{f}_{i,1}, \ldots, \hat{f}_{i,d+1}\) be the estimated flow size during period \((T_C - dt, T_C - (d + 1)t), \ldots, (T_C, T)\), which corresponds to \((d + 1)\) counters. And let \(\hat{f}_{i,1}, \ldots, \hat{f}_{i,d+1}\) be the real flow size of \(e_i\) during corresponding periods.

Same as the proving process of \(E(\hat{f}_i - f_i) = 0\) in Theorem 1 we can get that the estimation is unbiased if there is no time deviation. Thus, we have
\[
E(\hat{f}_i - f_i) = 0.
\] (15)
Let \(\hat{f}_{i,0}\) be the flow size of \(e_i\) during period \((T - (d + 0.5)t, T_C - (d - 1)t)\), and \(\hat{f}_{i,1}\) be the estimation given by a counter during corresponding period if there is no time deviation, \(i.e., \),
\[
\hat{f}_{i,1} = F_1(T - (d + 0.5)t, T_C - (d - 1)t) - F_1(T - (d + 0.5)t, T_C - (d - 1)t).
\] (16)
Similar to the above, we have \(E(\hat{f}_{i,1} - f_{i,0}) = 0\).

For \(\hat{f}_{i,1}\) and \(\hat{f}_{i,1}\), we have
\[
\hat{f}_{i,1} - f_{i,1} = F_1(T_C - dt, T - (d + 0.5)t) - F_1(T_C - dt, T - (d + 0.5)t).
\] (17)
Same as the proving process of Equation 9 in Theorem 1, we can get that
\[
E(F_1(T_C - dt, T - (d + 0.5)t)) - F_1(T_C - dt, T - (d + 0.5)t) = 0.
\] (18)
Thus, we have \(E(\hat{f}_{i,1} - \hat{f}_{i,1}) = 0\), from which and Equation 15 we can derive that
\[
E(\hat{f}_{i,1} - f_{i,0}) = E(\hat{f}_{i,1} - \hat{f}_{i,1}) + E(\hat{f}_{i,1} - f_{i,0}) = 0.
\] (19)
For our estimation \(\hat{f}_i\), by combining the results of Equation 15 and Equation 19 we can get that
\[
E(\hat{f}_i) - f_i = E\left(\sum_{j=1}^{d+1} \hat{f}_{i,j}\right) - (f_{i,0} + \sum_{j=2}^{d+1} \hat{f}_{i,j})
\]
\[
= E(\hat{f}_{i,1} - f_{i,0}) + \sum_{j=2}^{d+1} E(\hat{f}_{i,j} - f_{i,j}) = 0.
\] (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 \(\text{mean}(\hat{f}_i)\) 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, \(i.e., \), \(E(\text{median}(\hat{f}_i)) = f_i\).

**Proof.** We continue to use the mark in the case of Column Randomizing. Here our estimation for the size of flow \(e_i\) is
\[
\hat{f}_i = \hat{f}_{i,1} \cdot \frac{(T_C - (d - 1)t) - (T - (d + 0.5)t)}{t} + \sum_{j=2}^{d+1} \hat{f}_{i,j}
\]
\[
= \hat{f}_{i,1} \cdot \frac{1.5t - (T - T_C)}{t} + \sum_{j=2}^{d+1} \hat{f}_{i,j}.
\]
Let \(\hat{f}_{i,1}\) 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
where \( v_1, v_2, \ldots, v_n \) be the flow size of \( e_1, e_2, \ldots, e_n \) in unit time during \((T - (d + 1)t), T - (d - 1)t)\). Thus, we have
\[
\hat{f}_{i,0} = v_i[(T_C - (d - 1)t) - (T - (d + 0.5)t)] \\
= (1.5t - (T - T_C))v_i, 
\]
and also
\[
\hat{f}_{i,1} = \frac{1}{n} \sum_{j=1}^{n} v_j \cdot t \cdot s(e_j) = t \sum_{j=1}^{n} v_j s(e_j). 
\]

Since \( T_C \) is independent of \( s_j(\cdot) \), we can get the expectation of \((\hat{f}_{i,1} - f_i)\) that
\[
E(\hat{f}_{i,1} - f_i) = E(1.5t - (T - T_C)) \cdot \sum_{j \neq i} v_j s(e_j) \\
= E(1.5t - (T - T_C)) \cdot \sum_{j \neq i} v_j E(s(e_j)) = 0. 
\]

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
\[
E(\hat{f}_i - f_i) = E(\hat{f}_{i,1} - f_i) + \sum_{j=2}^{d+1} \hat{f}_{i,j} - f_{i,j} \\
= E(\hat{f}_{i,1} - f_i) + \sum_{j=2}^{d+1} E(\hat{f}_{i,j} - f_{i,j}) = 0. 
\]

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 \((\hat{f}_i - f_i) = E(\hat{f}_i - f_i) = 0 \). Therefore, \( E(mean(\hat{f}_i)) = f_i \), 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 \( v_1, v_2, \ldots, v_n \) be the flow size of \( e_1, e_2, \ldots, e_n \) 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
\[
\hat{Var}(\hat{f}_i) = \frac{1}{b} \sum_{j \neq i} v_j^2 \cdot (d'^2 + \frac{1}{12}) + \frac{1}{12} v_i^2 t^2. 
\]

**Proof.** Let \( e_{i_1}, e_{i_2}, \ldots, e_{i_b} \) be the keys inserted to the same estimator as \( e_i \) and \( v_{i_1}, v_{i_2}, \ldots, v_{i_b} \) be the flow size of \( e_{i_1}, e_{i_2}, \ldots, e_{i_b} \) in unit time. Without loss of generality, we take \( s(e_i) = 1 \). As proved in \( \text{[1]} \), \( E(f_i) = f_i \). Thus, for the variance, we have
\[
\hat{Var}(\hat{f}_i) = E(\hat{f}_i - E(\hat{f}_i))^2 = E(\hat{f}_i - f_i)^2 \\
= (\hat{f}_i - f_i) \cdot (\hat{f}_i - f_i), 
\]

For the former term, i.e., \((\hat{f}_i - f_i) \), we have \( \hat{f}_i - f_i = F_1(T_1, T_2) - F_2(T_1, T_2) \), so we can get that
\[
\hat{f}_i - f_i = \left( \sum_{j=1}^{l} v_i s(e_{ij}) + v_i \right) \cdot wt, 
\]
where \( wt \) is the deviation of time, which satisfies a uniform distribution \( U(-0.5t, 0.5t) \) 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 \( \text{[26]} \), i.e., \((\hat{f}_i - f_i) \), we have
\[
\hat{f}_i - f_i = \sum_{c_{ij}} f_{ij} s(e_{ij}) = (\sum_{j=1}^{l} v_i s(e_{ij})) \cdot d't. 
\]

Therefore, for the variance \( Var(\hat{f}_i) \), we can get the following results that
\[
\hat{Var}(\hat{f}_i) = E[\left( \sum_{j=1}^{l} v_j s(e_{ij}) + v_i \right) \cdot wt + \sum_{j=1}^{l} v_j s(e_{ij}) \cdot d't]^2 \\
= \sum_{j=1}^{l} v_j s(e_{ij}) \cdot (w + d't)^2 + \sum_{j=1}^{l} v_j s(e_{ij}) \cdot wt \cdot d't \\
= \sum_{j=1}^{l} v_j s(e_{ij}) \cdot (w + d't)^2 + \sum_{j=1}^{l} v_j w t \cdot d't. 
\]

In Equation \( \text{[29]} \), it’s because \( s(e_{ij}) \) has same chance to be 1 and -1 and is independent of other variables so that
\[
E[\left( \sum_{j=1}^{l} v_j s(e_{ij}) \right) \cdot (w + d't)^2 + \sum_{j=1}^{l} v_j s(e_{ij}) \cdot wt] = 0. 
\]

Then we first calculate mathematical expectation in Equation \( \text{[29]} \) on \( w \). Since \( w \) satisfies a uniform distribution \( U(-0.5, 0.5) \), we have
\[
\hat{Var}(\hat{f}_i) = E[\left( \sum_{j=1}^{l} v_j s(e_{ij}) \right) \cdot (w + d't)^2 + \sum_{j=1}^{l} v_j s(e_{ij}) \cdot wt] \\
= \int_{-0.5}^{0.5} E[\left( \sum_{j=1}^{l} v_j s(e_{ij}) \right) \cdot (d' + w)^2 + \sum_{j=1}^{l} v_j s(e_{ij}) \cdot wt] \cdot dw \\
= \int_{-0.5}^{0.5} \sum_{j=1}^{l} v_j s(e_{ij}) \cdot (d'^2 + \frac{1}{12}) + \frac{1}{12} v_i^2 t^2. 
\]

For mathematical expectation on \( s(e_{ij}) \), since \( s(e_{ij}) \) has same chance to be 1 and -1 and is independent from each other, cross terms in \( E(\sum_{j=1}^{l} v_j s(e_{ij})^2) \) are 0. Thus, we have \( E(\sum_{j=1}^{l} v_j s(e_{ij})^2) = \sum_{j=1}^{l} v_j^2 \). Therefore, we can get that
\[
\hat{Var}(\hat{f}_i) = \sum_{j=1}^{l} v_j^2 \cdot (d'^2 + \frac{1}{12})^2 + \frac{1}{12} v_i^2 t^2. 
\]

Moreover, since there are \( b \) estimators in each matrix and each key is hashed into one estimator, we have \( E(\sum_{j=1}^{l} v_j^2) = \frac{1}{b} \sum_{j \neq i} v_j^2 \). Therefore, the estimated variance satisfies
\[
\hat{Var}(\hat{f}_i) = \frac{1}{b} \sum_{j \neq i} v_j^2 \cdot (d'^2 + \frac{1}{12})^2 + \frac{1}{12} v_i^2 t^2. 
\]
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, \( s(\cdot) \) in each column are independent. Thus, similar to the proving process above, we have

\[
\text{Var}(\hat{f}_i) = E[(\sum_{j=1}^{t} v_{ij} s(e_{ij}))^2] \cdot ((\text{wt})^2 + d' \cdot t^2) + E[v_{i}^2 \cdot w^2 t^2] \\
= \sum_{j=1}^{t} v_{ij}^2 \cdot (d' + \frac{1}{12}) t^2 + \frac{1}{12} v_{i}^2 t^2. \tag{34}
\]

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

\[
\text{Var}(\hat{f}_i) = \frac{1}{b} \sum_{j \neq i} v_{ij}^2 \cdot d'^2 t^2. \tag{35}
\]

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

\[
\text{Var}(\hat{f}_i) = E[(\sum_{j=1}^{t} v_{ij} s(e_{ij}))^2] \cdot d'^2 t^2 = \sum_{j=1}^{t} v_{ij}^2 \cdot d'^2 t^2. \tag{36}
\]

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

\[
\text{Var}(\hat{f}_i) = \frac{1}{b} \sum_{j \neq i} v_{ij}^2 \cdot d'^2 t^2. \tag{37}
\]

### 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 \( \epsilon > 0 \), we have

\[
\Pr\{f_i - \hat{f}_i \geq \epsilon\} \leq \frac{1}{\epsilon^2} \text{Var}(\hat{f}_i). \tag{38}
\]

**Proof.** As proved in Theorem 2 \( E(\hat{f}_i) = f_i \). According to Chebyshev inequality, we can easily get that

\[
\Pr\{f_i - \hat{f}_i \geq \epsilon\} \leq \frac{1}{\epsilon^2} \text{Var}(\hat{f}_i) = \frac{1}{\epsilon^2 b} \sum_{j \neq i} v_{ij}^2 \cdot (m^2 + \frac{1}{12}) t^2 + \frac{1}{12} v_{i}^2 t^2, \tag{39}
\]

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 \( \text{mean}(\hat{f}_i) \), we can derive an error bound for our Unbiased Cleaning sketch.

**Theorem 5.** For a given \( \epsilon > 0 \), we have

\[
\Pr\{\text{mean}(\hat{f}_i) - f_i \geq \epsilon\} \leq \frac{1}{\epsilon^2 k} \text{Var}(\hat{f}_i). \tag{40}
\]

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

\[
\Pr\{\text{mean}(\hat{f}_i) - f_i \geq \epsilon\} \leq \frac{1}{\epsilon^2 k} \text{Var}(\hat{f}_i). \tag{41}
\]

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 \( \Delta f_i = f_i - \hat{f}_i \). 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 \( k-1 \) arrays.

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

\[
\Pr\{\text{median}(\hat{f}_i) - f_i \geq \epsilon\} \leq \frac{(2r + 1)!}{(r!)^2} \left(1 - \frac{\text{Var}(\hat{f}_i)}{\epsilon^2}\right)^r \left(\frac{\text{Var}(\hat{f}_i)}{\epsilon^2}\right)^{r+1}. \tag{42}
\]

**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 \( \Delta f_i = f_i - f_i \), and \( F(x) \) be the cumulative distribution function of \( \Delta f_i \).

Let \( \hat{f}_i \) from \( 2r + 1 \) arrays be \( \hat{f}_i^{(1)}, \hat{f}_i^{(2)}, \ldots, \hat{f}_i^{(2r+1)} \) 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 \( \Delta f_i^{(r+1)} = \text{median}(\hat{f}_i) - f_i \), and \( F^{(r+1)}(x) \) be the cumulative distribution function of \( \Delta f_i^{(r+1)} \). By the theorem of order statistics, we can get that

\[
f^{(r+1)}(x) = \frac{(2r + 1)!}{(r!)^2} F(x)^{r}(1 - F(x))^{r} f(x). \tag{43}
\]

We have \( F(x) = 1 - \Pr\{\Delta f_i \geq x\} \) for \( x \geq 0 \) and \( F(x) = \Pr\{\Delta f_i \leq x\} \) for \( x < 0 \). Let \( P(x) \) be \( \Pr\{\Delta f_i \geq x\} \) when \( x > 0 \) and \( \Pr\{\Delta f_i \leq x\} \) when \( x < 0 \). Then we have

\[
F(x)(1 - F(x)) = P(x)(1 - P(x)). \tag{44}
\]

As proved in Theorem 4, \( \Pr\{\Delta f_i \geq |x|\} \leq \frac{1}{x^2} \text{Var}(\hat{f}_i) \). For \( x \) that \( |x| \geq \sqrt{2\text{Var}(\hat{f}_i)} \), we have

\[
P(x) \leq \Pr\{\Delta f_i \geq |x|\} \leq \frac{1}{x^2} \text{Var}(\hat{f}_i) < \frac{1}{2}, \tag{44}
\]

from which we can derive that

\[
F(x) = 1 - F(x) \leq (1 - \Pr\{\Delta f_i \geq x\})\Pr\{\Delta f_i \geq x\} \leq (1 - \frac{1}{x^2} \text{Var}(\hat{f}_i))\left(\frac{1}{x^2} \text{Var}(\hat{f}_i)\right). \tag{45}
\]

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

\[
f^{(r+1)}(x) \leq \frac{(2r + 1)!}{(r!)^2} \left(1 - \frac{\text{Var}(\hat{f}_i)}{x^2}\right)^r \left(\frac{\text{Var}(\hat{f}_i)}{x^2}\right)^{r+1} f(x). \tag{46}
\]

For a given \( \epsilon \geq \sqrt{2\text{Var}(\hat{f}_i)} \) and for \( x \) that \( |x| \geq \epsilon \), since \( \frac{1}{x^2} \text{Var}(\hat{f}_i) \leq \frac{1}{\epsilon^2} \text{Var}(\hat{f}_i) \leq \frac{1}{2} \), we have the inequality
\[(1 - \frac{1}{k} \text{Var}(\hat{f}_i))(\frac{1}{k} \text{Var}(\hat{f}_i)) \leq (1 - \frac{1}{k} \text{Var}(\hat{f}_i))(\frac{1}{k} \text{Var}(\hat{f}_i)).\]

Thus, we can get that
\[
\int_{\epsilon}^{+\infty} f^{(r+1)}(x) \, dx \\
\leq \frac{2(2r+1)!}{(r+1)2} \left( 1 - \text{Var}(\hat{f}_i) \right) \left( \text{Var}(\hat{f}_i) \right) \epsilon^r f(x) \, dx \\
\leq \int_{\epsilon}^{+\infty} f(x) \, dx \cdot \frac{(2r+1)!}{(r+1)2} \left( 1 - \text{Var}(\hat{f}_i) \right) \left( \text{Var}(\hat{f}_i) \right) \epsilon^r.
\]

Similarly, we have
\[
\int_{-\infty}^{-\epsilon} f^{(r+1)}(x) \, dx \\
\leq \int_{-\infty}^{-\epsilon} f(x) \, dx \cdot \frac{(2r+1)!}{(r+1)2} \left( 1 - \text{Var}(\hat{f}_i) \right) \left( \text{Var}(\hat{f}_i) \right) \epsilon^r.
\]

From the above results in Equation 47 and Equation 48, we can get that median(\hat{f}_i) follows
\[
\text{Pr}\{\text{median}(\hat{f}_i) - f_i \geq \epsilon\} \\
= \int_{\epsilon}^{+\infty} f^{(r+1)}(x) \, dx + \int_{-\infty}^{-\epsilon} f^{(r+1)}(x) \, dx \\
\leq \frac{(2r+1)!}{(r+1)2} \left( 1 - \text{Var}(\hat{f}_i) \right) \left( \text{Var}(\hat{f}_i) \right) \epsilon^r.
\]

Therefore, for a given \( \epsilon \geq \sqrt{2 \text{Var}(\hat{f}_i)} \), our Unbiased Cleaning sketch has an error bound that
\[
\text{Pr}\{\text{median}(\hat{f}_i) - f_i \geq \epsilon\} \\
\leq \frac{(2r+1)!}{(r+1)2} \left( 1 - \text{Var}(\hat{f}_i) \right) \left( \text{Var}(\hat{f}_i) \right) \epsilon^r.
\]

Let \( n_0 \) be the number of arrays that satisfy \( |\hat{f}_i - f_i| \geq \epsilon \). For a given \( \epsilon \) that \( \epsilon \geq \sqrt{2 \text{Var}(\hat{f}_i)} \), we have \( P_r \leq \frac{1}{2} \). We can get that
\[
\text{Pr}\{n_0 \geq r + 1\} = \sum_{k=r+1}^{2r+1} \text{Pr}\{n_0 = k\} \\
= \sum_{k=r+1}^{2r+1} \binom{2r+1}{k} P_r^k (1 - P_r)^{2r+1-k} \leq \sum_{k=r+1}^{2r+1} \binom{2r+1}{k} P_r^k (1 - P_r)^{2r+1-k}.
\]

Here \( P_r = \text{Pr}\{|\hat{f}_i - f_i| \geq \epsilon\} \). Therefore, we can derive an error bound written in summation form that
\[
\text{Pr}\{\text{median}(\hat{f}_i) - f_i \geq \epsilon\} \leq \sum_{k=r+1}^{2r+1} \binom{2r+1}{k} P_r^k (1 - P_r)^{2r+1-k}.
\]

\[\Box\]

### 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 \( f'_i \) 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 \((T_e - dt, T_e)\). Without loss of generality, we take \( f_1' \geq f_2' \geq \cdots \geq f_n' \).

**Theorem 8.** If \( f_1' > 0.5 \sum_{i=1}^{n} f_i' \), then \( e_i \) is the key with largest estimated flow size by this matrix, i.e., \( \hat{f}_1 = \max_{1 \leq i \leq n} f_i' \).

**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 \( \hat{f}_i \leq f_i' \).

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 \( f_i = f \cdot s(e_i) = f \). Thus, the estimated flow size for \( e_i \) and \( e_1 \) satisfy \( \hat{f}_i = f_i' \) or \( f_i = -\hat{f}_i \). Since \( s(e_1) = 1 \), we have
\[
\hat{f} = \sum_{j=1}^{n} f_j' \cdot s(e_j) \geq f_1' - \sum_{j=2}^{n} f_j' \geq 0,
\]

thus \( \hat{f}_1 \geq 0 \). Therefore, we can get that \( \hat{f}_i \geq f_i' \).

Next, we consider the case that they are in different estimators, i.e., \( h(e_i) \neq h(e_j) \). Let \( e_{i_1}, e_{i_2}, \cdots, e_{i_l} \) be the keys inserted to the same estimator as \( e_i \) and \( e_{i_1}, e_{i_2}, \cdots, e_{i_l} \) be the keys inserted to the same estimator as \( e_1 \). Then we have the estimation of size of flow \( e_i \) that
\[
\hat{f}_i = (f_i' \cdot s(e_i) + \sum_{j=1}^{l} f_j' \cdot s(e_{i_j}) \cdot s(e_i)),
\]

and the estimation of the flow size of \( e_1 \) that
\[
\hat{f}_1 = (f_1' \cdot s(e_1) + \sum_{j=1}^{l} f_j' \cdot s(e_{i_j}) \cdot s(e_1)) \cdot s(e_1)
\]

\[\Box\]

**Theorem 7.** Let \( P_r = \text{Var}(\hat{f}_i)/\epsilon^2 \). For a given \( \epsilon \) that \( \epsilon \geq \sqrt{2 \text{Var}(\hat{f}_i)} \), we can derive an error bound for median(\hat{f}_i) that
\[
\text{Pr}\{\text{median}(\hat{f}_i) - f_i \geq \epsilon\} \\
\leq \sum_{k=r+1}^{2r+1} \binom{2r+1}{k} . P_r^k (1 - P_r)^{2r+1-k}
\]

**Proof.** If the error of our estimation is greater than \( \epsilon \), i.e., \( |\text{median}(\hat{f}_i) - f_i| \geq \epsilon \), then there are at least \( (r + 1) \) arrays satisfy that \( \hat{f}_i - f_i \geq \epsilon \) or at least \( (r + 1) \) arrays satisfy that \( \hat{f}_i - f_i \leq -\epsilon \). Thus, at least \( (r + 1) \) arrays satisfy that \( |\hat{f}_i - f_i| \geq \epsilon \). According to theorem \( 4 \), we have
\[
\text{Pr}\{|\hat{f}_i - f_i| \geq \epsilon\} \leq P_r.
\]
For key $e_j$, it’s inserted into at most one estimator. Thus, we can derive that
\[
\hat{f}_1 - \hat{f}_i = \hat{f}_1 + \sum_{j=1}^{l'} f_{ij} \cdot s(e_{ij}) - (\hat{f}_1' \cdot s(e_1) + \sum_{j=1}^{l'} f_{ij}' \cdot s(e_{ij})) s(e_1)
\geq \hat{f}_1 - \sum_{j=1}^{l'} f_{ij}' - \hat{f}_1' - \sum_{j=2}^{n} f_{ij}' \geq 0.
\]
(58)

Therefore, we always have $\hat{f}_1 \leq \hat{f}_i$. 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 $\forall T_c \in (T - nt, T - (m-1)t)$ we have $\hat{f}_1 \geq 0.5 \sum_{i=1}^{n} f_{ij}'$, 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 $\forall T_c \in (T - nt, T - (m-1)t)$ we have $\hat{f}_1 \geq 0.5 \sum_{i=1}^{n} f_{ij}'$, 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 $mean(\hat{f}_1) = \max_{1 \leq i \leq n} mean(\hat{f}_i)$.

Second, if we take the median, we assume that there exists an key $e_i$ that satisfies $median(\hat{f}_i) > median(\hat{f}_1)$, i.e., $\hat{f}_i = (r+1)$, $\hat{f}_1 (r+1) > (r+1)$.

Thus, for $1 \leq l_1 \leq r+1$ and $r+1 \leq l_2 \leq 2r+1$, we have $f_{ij} (l_1) > \hat{f}_{ij} (r+1) < f_{ij} (r+1)$. Since $f_{ij} (l_1)$ and $f_{ij} (l_2)$ are both from $(r+1)$ different matrices, there is at least a matrix which contains both $f_{ij} (l_1)$ and $f_{ij} (l_2)$. In this matrix, we have $\hat{f}_1 < \hat{f}_i$, which contradicts the conclusion that $\hat{f}_1 = \max_{1 \leq i \leq n} median(\hat{f}_i)$.

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.

**Theorem 10.** For Zipfian distribution with parameter $z$, i.e., $v_i = \frac{1}{z^i}$, 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
\[
\sum_{i=2}^{n} c_i \leq \sum_{i=2}^{n} \frac{c}{(i-1) \times i} \leq c = v_1
\]
(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., $v_i = \frac{1}{z^i}$, the magnitude of the variance satisfies
\[
Var(\hat{f}_i) = \begin{cases} O(n^{1-2z}) & z < \frac{1}{2} \\ O(\log(n)) & z = \frac{1}{2} \\ O(1) & z > \frac{1}{2} \end{cases}
\]
(60)

**Proof.** In Equation (53) we have $Var(\hat{f}_i) = O(\sum v_i^2)$. For the magnitude of $\sum v_i^2 = \sum_{i=1}^{n} v_i^2$, we have
\[
\sum_{i=1}^{n} v_i^2 \sim \int_{1}^{n} \frac{c}{x^{2z}} dx = \frac{1}{1 - 2z} \left( \frac{n^{1 - 2z}}{1 - 2z} - 1 \right) = \frac{1}{2z} \cdot \log(n) - z < \frac{1}{2}
\]
(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 [58] 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 [59]. 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 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 [60], 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 cache 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): \( \frac{1}{|\Psi|} \sum_{i \in \Psi} |f_i - \hat{f}_i| / f_i \), where \( f_i \) is the real flow size of \( e_i \), \( \hat{f}_i \) is its estimated flow size, and \( \Psi \) 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): \( \frac{1}{|\Psi|} \sum_{i \in \Psi} |f_i - \hat{f}_i| \), where \( f_i \) is the real flow size of \( e_i \), \( \hat{f}_i \) is its estimated flow size, and \( \Psi \) 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 = \( 1 \times 10^4 \) and the default dataset is the CAIDA-2016 dataset.

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.

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.
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 significant in network measurement. In this paper, we propose an algorithm called Unbiased Cleaning sketch. It is the first work that achieves unbiased flow 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 experimental results are consistent with the theory. All related source codes are open-sourced at Github [61].

Acknowledgment

We thank all anonymous reviewers for their help in improving this paper. This work is supported by Key-Area Research and Development Program of Guangdong Province 2020B0101390001, and National Natural Science Foundation of China (NSFC) (No. U20A20179, and 61832001).

References


Yuhang Wu, Zhuochen Fan, Qilong Shi, Yixin Zhang, Tong Yang, Cheng Chen, Zheng Zhong, Junlan Li, Ariel Shlit, and Yaofeng


Yuhan Wu
Yuhan Wu received his B.S. degree in the Department of Electrical Engineering and Computer Science at Peking University in 2021. Currently he is a CS Ph.D. student in School of Computer Science at Peking University, advised by Tong Yang. His research interests include sketches, network measurement, and key-value stores.

Shiqi Jiang
Shiqi Jiang received her B.S. degree in Statistics from Peking University in 2021. She is currently a master student at the Department of Mathematics, National University of Singapore. Her research interests include deep learning, dynamic systems and optimal control.

Yifei Xu
Yifei Xu received his B.S. degree from Peking University in 2021. He is currently pursuing a Ph.D. degree in computer science at University of California, Los Angeles. His research interests include networked systems, mobile computing, and network security. He is a member of the Wireless Networking Group.

Siyuan Dong
Siyuan Dong is an undergraduate majoring in computer science at School of EECS, Peking University, advised by Tong Yang. His research interests cover network measurements, sketches, network systems and machine learning.

Kaicheng Yang
Kaicheng Yang received his B.S. degree in Computer Science from Peking University in 2021. He is currently pursuing the Ph.D. degree in the School of Computer Science, Peking University, advised by Tong Yang. His research interests include network measurement and programmable data plane.

Peiqing Chen
Peiqing Chen received his B.S. degree in Computer Science from Peking University in 2021. He is currently pursuing the Ph.D. degree in the School of Electrical Computer Engineering, Boston University, advised by Prof. Zaoxing Liu. His research interests include network and data systems.

Tong Yang
Tong Yang received the PhD degree in computer science from Tsinghua University in 2013. He visited the Institute of Computing Technology, Chinese Academy of Sciences (CAS). He is currently an Associate Professor with School of Computer Science, Peking University. His research interests include network measurement, sketches, IP lookups, Bloom filters, and KV stores. He has published more than ten papers in SIGCOMM, SIGKDD, SIGMOD, NSDI, etc.