ExtendedSketch: Fusing Network Traffic for Super Host Identification With a Memory Efficient Sketch

Xuyang Jing, Zheng Yan, *Senior Member, IEEE*, Hui Han, and Witold Pedrycz, *Life Fellow, IEEE*

Abstract—Super host refers to the host that has a high cardinality or exhibits a big change in a network. Facing big-volume network traffic, sketches have been widely applied to identify super hosts in an efficient and accurate way. However, most sketches cannot flexibly balance memory usage and accuracy in host cardinality estimation. Setting an inappropriate counter size for a sketch could either lead to inaccurate host cardinality estimation or cause memory waste. In order to solve this issue, we propose a novel extensible and reversible sketch, named ExtendedSketch, to achieve accurate super host identification with high memory efficiency. The core idea of ExtendedSketch is to monitor low-cardinality hosts with small-sized counters while dynamically extending the size of counters when monitoring high-cardinality hosts by applying an adaptive extension strategy. Such the strategy can adaptively increase counter size according to network traffic status at runtime, which not only ensures the accuracy of high-cardinality host estimation but also avoids unnecessary memory consumption. We perform theoretical analysis and conduct a series of experimental evaluations on ExtendedSketch based on real world network traffic. Experimental results show that under same memory usage, compared to the state-of-the-art, ExtendedSketch achieves $1.4\sim7.5$ times smaller error rate in estimating host cardinality with $1.9\sim26.7$ times better accuracy on super host identification and $95\sim2^{15}$ times faster speed on abnormal address reconstruction. Its advance in accuracy and efficiency demonstrates the practical significance of ExtendedSketch for super host identification.

Index Terms—Network traffic measurement, host cardinality estimation, super host identification, sketch, memory efficiency

1 Introduction

Super host identification plays an important role in network management, which can be applied to detect network attacks (e.g., Distributed Denial of Service (DDoS) attacks [1], network scanning [2]), track hot-spot web content [3], monitor user activities [4]. Accurately estimating host cardinality is the premise of super host identification. The cardinality of a host is the number of different hosts it communicates with, e.g., Destination Cardinality (DC) is the number of different destination addresses that a host communicates with, while Source Cardinality (SC) is the number of different source addresses that a host is connected with. A super host is a host that exhibits abnormal behavior in terms of its cardinalities. For example, the host with high DC (SC) is known as a super spreader (a super receiver) and the host has heavy changes in cardinality between two adjacent time intervals is called as a super changer [5].

- *Xuyang Jing and Hui Han are with the State Key Lab of ISN, the School of Cyber Engineering, Xidian University, Xi'an, China. E-mail: xuyangjing91@163.com, 1299634339@qq.com.*
- *Zheng Yan is with the State Key Lab of ISN, the School of Cyber Engineering, Xidian University, Xi'an, China, and also with the Department of Communications and Networking, Aalto University, Finland. E-mail: zyan@xidian.edu.cn; zheng.yan@aalto.fi..*
- *Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada. E-mail: wpedrycz@ualberta.ca.*

Manuscript received 5 June 2021; revised 21 Aug. 2021; accepted 4 Sept. 2021. Date of publication 9 Sept. 2021; date of current version 11 Nov. 2022. (Corresponding author: Zheng Yan.) Digital Object Identifier no. 10.1109/TDSC.2021.3111328

With the continuous increase of networking speed, an online traffic processing module with fast traffic collection and efficient memory usage is highly expected [6], [7], [8]. For example, the switches in Software Defined Networks (SDN) can measure flow size/cardinality by using flow tables. Because of the number of flows mostly exceeds the size of flow tables, traditional flow measurement methods employed at switches are not flexible [9]. Sketches, which use data-oriented hashing to fuse data into a compact way, have been widely applied in many aspects of analyzing big-volume network traffic, e.g., flow size estimation [10], [11], host cardinality estimation [12], [13], network anomaly detection [14], [15], persistent spread measurement [16], [17], and flow statistics collection in SDN [18], [19]. The characteristics of sketches with regard to fast processing of huge network-wide traffic with expected accuracy and memory usage can solve the challenges of identifying super hosts in a high-speed network environment.

There are many research efforts on super host identification. The Online Streaming Module (OSM) combines a two-dimensional bit arrays with some hash functions to identify high-cardinality hosts [20]. The Random Aging Streaming Filter improves OSM by using a random aging algorithm [21]. However, both of them fail to support memory efficiency and recover abnormal addresses. The Double Connection Degree Sketch is proposed to estimate host cardinality and reconstruct addresses by using Chinese Remainder Theorem [22]. The Vector Bloom Filter employs a Bit-Extraction hash function to store cardinality information and uses string mergence to recover addresses [23]. A noisy group testing-based sketch is designed to identify high-cardinality and

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

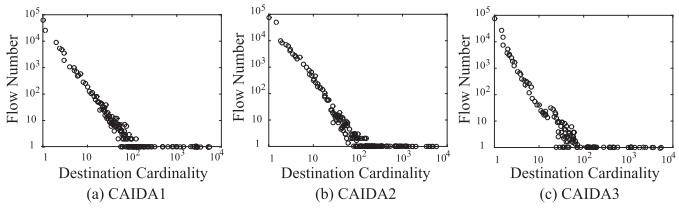


Fig. 1. Host cardinality distribution of three traffic datasets. Each dot denotes the number of flows having a certain host cardinality.

reconstruct abnormal hosts [24]. However, the memory usage and reversible calculation of them are inefficient when there is a large amount of abnormal addresses. SpreadSketch equips each bucket of Count-Min sketch with a multiresolution bitmap to estimation host cardinality and then perform super spreader identification [25]. But, it has some cardinality information loss. In summary, the main problem in the field of super host identification is inefficient memory usage and inaccurate estimation regarding host cardinality estimation caused by skewed network traffic. Accurate host cardinality estimation can be achieved by using a sketch with large size. But, it is unnecessary to use large-sized counters to monitor the hosts with low cardinality since it causes serious memory waste. Most of existing methods are unable to balance their accuracy and memory usage. The Compact Spread Estimator aims to solve this problem with the help of bit sharing [26]. But, memory sharing among multiple hosts incurs inaccurate cardinality estimation and makes it difficult to reconstruct original super host addresses.

Practically, we face a number of challenges to solve the above problems. First, the distribution of host cardinality in real network traffic is highly skewed and dynamically changed [4], [12], [26], [27], [28], as shown in Fig. 1. The number of small-cardinality hosts is dominant while the number of high-cardinality hosts constitutes a small percentage. Most of existing methods for estimating host cardinality allocate counters of the same size to monitor these two types of hosts. However, the counters used to monitor low-cardinality hosts should be much smaller than those used to monitor high-cardinality hosts. It is difficult to determine a suitable counter size since inappropriate counter size will lead to inaccuracy in estimating high-cardinality hosts and cause unnecessary memory consumption in monitoring low-cardinality hosts. Inaccurate cardinality estimation for high-cardinality hosts will directly affect the accuracy of super host identification. Second, host cardinality is not additive, which is different from flow size that has an additive property. For example, the destination cardinality of a host will only increase when it sends flows to a new host. This characteristic of host cardinality makes separate monitoring of high- and low-cardinality more difficult than monitoring large- and small-flow in skewed traffic (e.g., [11], [29]). The third issue is the sketches cannot store any information about original host addresses due to data compression [14], [30]. However, after detecting super hosts, it is necessary to take some mitigation approaches to release detected anomalies. Therefore, it becomes essential to innovate a novel sketch that can achieve memory efficiency, host cardinality estimation accuracy and abnormal address reconstruction efficiency at the same time. All of the above expectations in super host identification motivate our work.

In this paper, we propose a novel extensible and reversible sketch to solve the above challenges about identifying super hosts facing skewed network traffic, named ExtendedSketch. The extensibility makes ExtendedSketch feasible to be applied into analyzing skewed big-volume traffic with high memory efficiency and super host identification accuracy. At the beginning, ExtendedSketch allocates a number of same-sized counters to monitor cardinalities of all hosts. Each counter is a bit array that used to store communication information of each host. With traffic coming in, the counters that record high-cardinality hosts will be rapidly filled up while the counters that monitor low-cardinality hosts will not change much. ExtendedSketch will increase the size of filled counters by using an extension strategy with the purpose of dynamically enlarging estimation ability of counters. Such the extension strategy can transfer the recorded cardinality information from old counters to new counters as in a lossless manner as possible. In this way, ExtendedSketch not only solves the problem of unnecessary memory usage but also keeps accurate cardinality estimation. In addition to extensibility, ExtendedSketch also owns reversibility since it can reconstruct addresses of super hosts in an accurate and efficient way by applying Chinese Remainder Theorem. Thanks for the four operations (namely update, estimation, merge, and reversible calculation) defined on the basis of ExtendedSketch, we design an accurate and fast method for identifying super hosts. In particular, the main contributions of this paper can be summarized as below:

1) We propose an extensible, mergeable and reversible sketch to estimate host cardinality and further identify super hosts, named ExtendedSketch. It can achieve memory efficiency, fast traffic processing, and accurate reconstruction of addresses of super hosts at the same time. These advanced properties owned by ExtendedSketch have seldomly been investigated simultaneously in the literature.
2) We further propose an accurate and fast super host identification method based on ExtendedSketch.
3) We provide a detailed theoretical analysis on ExtendedSketch with regard to its space and time complexities and estimation operation.
4) We compare ExtendedSketch with several state-of-the-art methods with regard to accuracy, efficiency, and reversibility based on real world network traffic. The experimental results show that ExtendedSketch outperforms others as a whole.

The paper is organized as follows. Section 2 gives a brief review on related work. In Section 3, we give an overview of ExtendedSketch. It is a novel data structure for super host identification originally proposed in this paper: its structure is adaptively adjusted based on the distribution of host cardinality, which is totally different from any existing sketches. The formal analysis of ExtendedSketch is provided in Section 4, followed by performance evaluation results and additional discussion on its effectiveness in Section 5. Finally, conclusions are drawn in the last section.

2 Related Work

There are two categories of methods widely applied in super host identification: sampling-based and sketch-based.

TABLE 1
Comparison of ExtendedSketch With Other Sketches

Method	MA	DI	IE	RC
OSM[20]	No	NS	No	No
RASF[21]	No	NS	No	No
DCDS[22]	No	Yes	No	Yes
VBF[23]	No	Yes	No	Yes
FS[24]	No	Yes	No	Yes
SS[25]	No	Yes	No	Yes
ExtendedSketch	Yes	Yes	Yes	Yes

Yes: support requirement; No: cannot support requirement;
NS: not strong to support requirement;
MA: Memory Adaptability; DI: Distributed Identification;
IE: Identification Efficiency; RC: Reversible Calculation

TABLE 2
Notations

Symbol	Notion
H	The number of two-dimensional bit arrays;
ES_i	The i-th two-dimensional bit arrays
w	The number of rows of ES_i
p_i	The number of columns of ES_i
h_i	The i-th hash function
cd	The crowding degree
et	The extension times
ε	The predefined threshold
s/d	The source/destination address
$DC(s)$	The destination cardinality of s
v/v'	The number of zero-bit buckets

Some studies have pointed out that sampling-based methods exhibit poor accuracy and high memory consumption or storage cost [10], [31]. Sketch-based methods have been attracting much attention in recent years due to their special advantages over the sampling-based methods.

Zhao *et al.* [20] proposed the Online Streaming Module (OSM), which includes a two-dimensional bit arrays and some independent hash functions, to detect super spreaders/receivers. Each bucket in OSM stores a connection between a source and a destination. To improve the accuracy of OSM, Yoon *et al.* [21] designed a Random Aging Streaming Filters (RASF) to randomly reset some buckets when the two-dimensional bit arrays become full. However, these methods suffer from high memory overhead and fail to reconstruct original IP addresses.

To make the sketch reversible, Wang *et al.* [22] designed the Double Connection Degree Sketch (DCDS) that includes multiple two-dimensional bit arrays to detect super hosts. Qin *et al.* [32] proposed a bi-directional traffic model for abnormal host behavior detection based on DCDS. The original host addresses can be reconstructed based Chinese Remainder Theorem. However, their methods suffer from such problems as unnecessary memory usage and exhaustive combination of possible addresses, which increase computing overhead and false positives. Liu *et al.* [23] proposed Vector Bloom Filter (VBF) by employing Bit-Extraction hash functions. They divided addresses into eight parts and used a number of newly designed hash functions to store traffic information. A super host can be reconstructed based on the overlapping of hashed bit strings. Liu *et al.* [24] proposed a Fast Sketch (FS) inspired by considering high-cardinality host identification as a channel-coding problem. An abnormal address can be recovered by using a decoding algorithm to process error-correcting codes. However, the memory and reversible calculation overhead of these methods will become high when there is a large number of abnormal hosts. Tang *et al.* [25] designed SpreadSketch (SS), which is a combination of Count-min sketch with multiresolution bitmap, to perform network-wide super spreader identification. Original IP address is stored in each bucket of SS. But, SS faces under-estimation problem.

A tabulated comparison of ExtendedSketch with the popular sketches is presented in Table 1. We can see that ExtendedSketch satisfies the whole requirements in terms of memory adaptability, distributed identification, identification efficiency and reversible calculation while other sketches only satisfy some of the requirements. The main difference of ExtendedSketch from existing sketches is its extensible characteristic. Compared to other sketches that use a fixed-size structure, ExtendedSketch can adaptively adjust its memory usage according to the changes of network traffic. When there is no any high-cardinality host, ExtendedSketch accounts for a very small memory to monitor host cardinalities. When high-cardinality hosts appear, ExtendedSketch only increases the memory sizes of some counters where high-cardinality hosts locate. Its extensibility characteristic makes ExtendedSketch very memory efficient while achieving accurate host cardinality estimation at the same time. Therefore, ExtendedSketch can be applied into resource-constrained traffic measurement scenarios, e.g., ExtendedSketch is feasible to be deployed in SDN switches.

3 Overview of ExtendedSketch

In this section, we first describe the structure of ExtendedSketch, which is a novel memory efficient sketch that used for super host identification. Then, we define the operations of ExtendedSketch, namely update operation, estimation operation, merge operation and reversible calculation operation. At last, we give the accurate and fast methods of super host identification according to the traffic information recorded by ExtendedSketch. In Table 2, we list main notations used in this paper for the convenience of reading.

3.1 The Structure of ExtendedSketch

The structure of ExtendedSketch is shown in Fig. 2. It consists of two parts: a *core part* that is used to measure the host cardinality and an *extended part* that is applied to increase record capacity of *core part*.

The *core part* of ExtendedSketch has H two-dimensional bit arrays with the size of $w \times p_i$, $i = 1, 2, \ldots, H$, which is denoted as $ES = (ES_1, \ldots, ES_H)$. In ES_i, $ES_i[j][l] \in \{0, 1\}$ represents the value recorded in bucket (j, l), where $j \in \{0, 1, \ldots w-1\}$ and $l \in \{0, 1, \ldots, p_i - 1\}$. The column of ES_i is associated with a data-oriented hash function used to get the column index, namely $h_i(x) \equiv x \bmod p_i$, where $p_1, p_2, \ldots, p_H$ are selected as pair-wise coprime numbers around an integer P according to the Chinese Remainder Theorem [14]. Each column has an additional information triple *(cd, et, flag)*, where *cd* is the crowding degree that represents the proportion of "1" in a column, *et* is the extension times that records

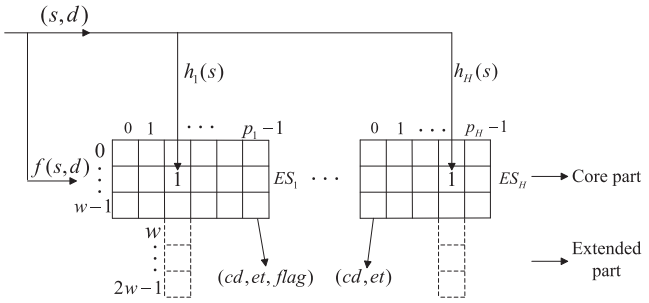


Fig. 2. The structure of ExtendedSketch that includes a core part and an extended part. In the ExtendedSketch, $h_i(s)$ and $f(s,d)$ are used to obtain column and row index. Each column is associated with an information triple *(cd, et, flag)*.

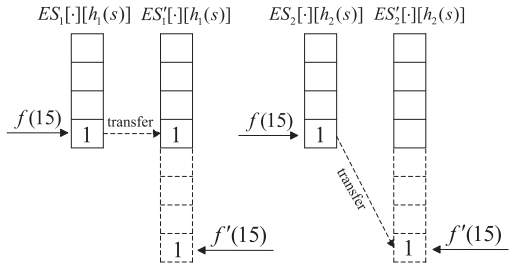


Fig. 3. The column extension process of ExtendedSketch.

the times of a column has been extended, and *flag* is a list that records the next location (column number) in ES_{i+1} that a given address locates (there is no *flag* in ES_H). A *flag* is selected as a set in order to reduce false negative rate caused by hash collisions. If a *flag* is a numeric value, it will be replaced by the location of another source address when hash collision occurs [14], [33]. The row hash functions that used to get row index of *core part* are the same at the beginning, namely $f(x) \equiv x \bmod w$. They will be changed after adding *extended part*.

The *extended part* is the extension of some columns of *core part*. The crowding degrees of these columns are larger than a predefined threshold. It means that such columns will be fully filled with a large number of different destination addresses. Accordingly, the modules of row hash function associated with extended columns will be enlarged based on extension times. For example, as described by the dotted line in Fig. 2, we extend a column when its crowding degree is large. The associated row hash function of that column becomes $f'(x) \equiv x \bmod (2w)$, while the row hash functions of other columns stay the same as initial one.

In this paper, we select the *key* as the combination of source address and destination address for row hashing and the *key* as source address for column hashing with the purpose of measuring destination cardinality. They could be changed according to different measurement purposes and tasks. For example, we can estimate source cardinality by setting the *key* of row hashing to the combination of source address and destination address and the *key* of column hashing to destination addresses.

3.2 Operations of ExtendedSketch

There are four operations of ExtendedSketch designed to measure host cardinality and identify super host in the context of skewed network-wide traffic.

3.2.1 Update Operation

Update operation is used to record host cardinality information. Initially, ExtendedSketch only has *core part* and all its bits are zero. The additional information triple is also initialized, namely we select $cd = et = 0$ as their initialized values and set *flag* as an empty list. Suppose ε is a predefined threshold, t is a variable number, s/d is a source/destination address. Given a flow (s,d), there are two cases for column $ES_i[\cdot][h_i(s)](1 \le i \le H)$ update:

(a) $cd < \varepsilon$ and $et = t$: we set $ES_i[f'(s,d)][h_i(s)] = 1$, where $f'(s,d) \equiv (s,d) \bmod (2^t \times w)$, as shown in Fig. 2. Then, cd is updated and $h_{i+1}(s)(1 \le i \le H-1)$ is added to *flag*.

(b) $cd \ge \varepsilon$ and $et = t$: this condition means that $ES_i[\cdot][h_i(s)]$ has been filled up now and needs to be further extended. However, the hash collisions (e.g., some different source addresses are hashed into a same column) can also make *cd* become large. To solve this problem, the column extension will carry out under the condition that the crowding degrees of all columns that s locates over $ES = (ES_1, \ldots, ES_H)$ are greater than or equal to ε. In column extension, we first enlarge the length of $h_i(s)$ from $2^t \times w$ to $2^{t+1} \times w$ and set $et \mathrel{+}= 1$. Next, we should transfer the recorded information from the old column to the extended column, which needs to be carefully performed in order to preserve the cardinality information.

Herein, due to the randomness of hashing operation [34], we introduce a transfer strategy to guarantee the lossless transmission of cardinality information based on the following lemma.

Lemma 1. *Given two arbitrary integers i and a, if* $i \bmod a = b$, *then* $i \bmod (2a) = b$ *or* $i \bmod (2a) = b + a$.

Therefore, in column extension, the buckets where (s,d) locates in old column $(ES_i[\cdot][h_i(s)], 1 \le i \le H)$ will be transferred to the extended column $(ES'_i[\cdot][h_i(s)])$ with the following strategy:

- for $i = 1, 3, 5, \ldots$, set $ES'_i[f'(s,d)][h_i(s)] = ES_i[f'(s,d)][h_i(s)]$.
- for $i = 2, 4, 6, \ldots$, set $ES'_i[f'(s,d) + 2^t \times w][h_i(s)] = ES_i[f'(s,d)][h_i(s)]$.

where $f'(s,d) \equiv (s,d) \bmod (2^t \times w)$. After transferring cardinality information, the bucket that (s,d) locates in extended column is updated as the steps described in condition (a).

Fig. 3 explicitly shows an example of extension process. Suppose w is 4 (row hash function is $f(x) \equiv x \bmod 4$ at the beginning), the stored value is 15 and H is 2. After column extension, the new column has 8 buckets and the row hash function becomes $f'(x) \equiv x \bmod 8$. We then transfer information from the old column to the extended column using the transfer strategy, as shown the dotted arrows in Fig. 3. However, the extension strategy may cause some estimation problems. For example, storing 15 in $ES'_1[\cdot][h_i(s)]$ will increase an extra one-bit and lead to over-estimation problem. We will eliminate the estimation problems as much as possible by using a specialized cardinality estimation method. The update operation of ExtendedSketch is described in Algorithm 1.

Algorithm 1. Update Operation of ExtendedSketch

Input: Initialized $ES = (ES_1, \ldots, ES_H)$;
Output: ExtendedSketch that records traffic information;
1: New arriving flow (s, d);
2: **for** $i = 1$ to H do:
3: $h_i(s) \equiv s \bmod p_i$;
4: **if** $cd_{h_i(s)} \geq \varepsilon$ and extension condition is satisfied:
5: extend column $h_i(s)$;
6: transfer host cardinality information;
7: update the bucket that (s, d) locates;
8: update $cd_{h_i(s)}$, et and *flag*;
9: **else**:
10: $ES_i[(s, d) \bmod (2^t \times w)][h_i(s)] = 1$;
11: update $cd_{h_i(s)}$ and *flag*;
12: **end for**

3.2.2 Estimation Operation

For estimating the destination cardinality of a given source address s, we first find H columns in ES that s locates, denoted as $ES_i(s) = ES_i[\cdot][h_i(s)](1 \leq i \leq H)$. There are two cases over $ES_i(s)(1 \leq i \leq H)$:

a) $et = 0$: it means that $ES_i(s)(1 \leq i \leq H)$ has not been extended. In order to eliminate the over-estimated problem caused by hash collisions, we construct an estimation bitmap by conducting bitwise-AND operation on each buckets of $ES_i(s)(1 \leq i \leq H)$, namely $ES(s)$=$ES_1(s) \otimes \cdots \otimes ES_H(s)$. The destination cardinality of s can be estimated by using probabilistic counting algorithm [35]

$$DC(s) = -w \ln(v/w), \tag{1}$$

where v is the number of zero-bit buckets in $ES(s)$.

b) $et = t(\geq 1)$: in this case, the column $ES_i(s)(1 \leq i \leq H)$ has been extended t times and their length becomes to $2^t \times w$. Taking bitwise-AND operation on H extended columns is not suitable since this leads to cardinality information loss. For example, in Fig. 3, we will loss the information of 15 when using bitwise-AND operation if it does not come again after column extension. Moreover, the transfer strategy may bring an extra one-bit value, which increases the cardinality estimation results calculated by the probabilistic counting algorithm, e.g., the column $ES'_1[\cdot][h_i(s)]$ in Fig. 3. With the purpose of solving the estimation problems caused by t times of column extension, we first calculate $DC_i(s)$ from $ES_i(s)(1 \leq i \leq H)$ as

$$DC_i(s) = -w' \ln(v'/w') + \lambda w \sum_{u=1}^{t} 2^{u-1}, \tag{2}$$

where $w' = 2^t \times w$ is the length of extended $ES_i(s)$, v' is the number of zero-bit buckets in $ES_i(s)$, $\lambda = \ln(2-\varepsilon)^2/4(1-\varepsilon)$. The first item in Formula (2) realizes the probabilistic counting algorithm [35], while the second item stands for compensation for estimation error caused by t times of column extension. The theoretical analysis of estimation error is provided in Section 4. Finally, the estimated cardinality of s is the minimum value in the set $\{DC_1(s), \ldots, DC_H(s)\}$. Algorithm 2 describes the estimation operation of ExtendedSketch.

Algorithm 2. Estimation Operation of ExtendedSketch

Input: $ES = (ES_1, \ldots, ES_H)$, a source address s;
Output: $DC(s)$;
1: Find the columns that s locates over ES, namely $ES_i(s) = ES_i[\cdot][h_i(s)](1 \leq i \leq H)$;
2: **if** $et_{ES_i(s)} = 0(1 \leq i \leq H)$:
3: $ES(s)$=$ES_1(s) \otimes \cdots \otimes ES_H(s)$;
4: count the number of zero-bit buckets v in $ES(s)$;
5: $DC(s) = -w \ln(v/w)$;
6: **else**:
7: **for** $i = 1$ to H do:
8: $w' = 2^{et_{ES_i(s)}} \times w$;
9: calculate the number of zero-bit buckets v';
10: $C_i(s) = -w' \ln(v'/w') + \lambda w \sum_{u=1}^{et_{ES_i(s)}} 2^{u-1}$;
11: **end for**
12: $DC(s) = \min\{DC_1(s), \ldots, DC_H(s)\}$;

3.2.3 Merge Operation

ExtendedSketch is applicable in distributed collection of network traffic, which is highly needed in network-wide traffic measurement [36]. Give a set of T ExtendedSketches with same H and $p_i(1 \leq i \leq H)$, namely $(ES^1, \ldots, ES^T)$, where $ES^t(1 \leq t \leq T)$ is t-th ExtendedSketch. A merge operation is performed as below:

$$ES_i[j][l] = ES_i^1[j][l] \oplus \cdots \oplus ES_i^T[j][l], \tag{3}$$

where $ES_i^t[j][l](1 \leq t \leq T)$ is the value of bucket (j, l) of ith bit arrays in ES^t, $\oplus$ is the bitwise-OR operator.

Notably, we should normalize column length over $(ES^1, \ldots, ES^T)$ before doing merge operation. A column with a short length should be extended to the length of a long column. For example, if et of $ES_1^1[\cdot][l]$ is 2 and et of $ES_1^2[\cdot][l]$ is 3, $ES_1^1[\cdot][l]$ should be extended for one time to reach the same length as $ES_1^2[\cdot][l]$.

3.2.4 Reversible Calculation Operation

When applying ExtendedSketch to detect super hosts, abnormal addresses can be accurately and effectively recovered by using the reversible calculation operation. Suppose there are some super spreaders and a number of abnormal columns are detected over ES. From ES_1 to ES_H, we use the *flag* indicator of abnormal columns, which stores column number of next two-dimensional bit arrays, to generate column links that reveal all locations of source addresses in ES. Given a super spreader s, the location link $\{c_1, \ldots, c_H\}$ will be successfully generated by matching the *flag* indicator, where $c_i(1 \leq i \leq H)$ is calculated by $h_i(s) \equiv c_i \bmod p_i$. Based on Chinese Remainder Theorem, the decimal representation of s can be uniquely determined with $s \equiv \sum_{i=1}^{H} Q_i Q'_i c_i \bmod p$, where $p = p_1 p_2 \ldots p_H$ and it is greater than or equal to the size of source address space, $Q_i = p/p_i$, $Q_i Q'_i \equiv 1 \bmod p_i$, $i = (1, 2, \ldots, H)$ [14], [33], [37]. By using the *flag*-based column combination, the computational burden of reversible calculation and the

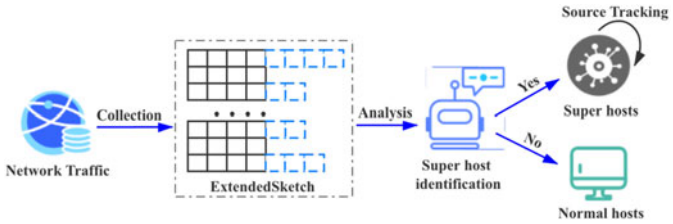


Fig. 4. The process of super host identification.

false positive rate caused by unexpected column combinations will be highly reduced.

3.3 Super Host Identification

A super host refers to the host that holds high cardinality (super spreader) or has a big change in cardinality between two temporal intervals (super changer). In ExtendedSketch, when choosing a small initial value of w, the extension times of the columns that a super host locates are larger than the columns that other hosts locate. Based on this principle, we propose an efficient super host identification method. The process of super host identification is shown in Fig. 4. We first use the ExtendedSketch to collect network traffic and then apply the proposed identification method to find out super hosts. The original IP addresses of super hosts can be easily constructed by using reversible calculation operation of ExtendedSketch.

3.3.1 Super Spreader Identification

To detect super spreaders, we first identify abnormal columns whose extension times are not equal to zero ($et \neq 0$) in the set of $(ES_1, \ldots, ES_H)$. After that, a number of H lists that store the numbers of abnormal column are obtained. By using *flag*-based column combination, the link information of a host can be easily generated. The host is considered as a super spreader if its cardinality calculated based on estimation operation of ExtendedSketch is larger than a predefined threshold.

3.3.2 Super Changer Identification

For detecting super changers, we need to identify changed columns between ExtendedSketches of two temporal intervals. Therefore, we check the abnormal columns (where $et \neq 0$) of the two ExtendedSketches and then calculate cardinality difference of the abnormal columns between the previous and current temporal intervals. The host is considered as a super changer if its cardinality difference is larger than a predefined threshold.

Compared with other identification methods [20], [21], [22], [23], [25], our method has the following advantages:

(a) high efficiency: it can directly identify super hosts by checking the abnormal columns with extension times. The proposed method conducts identification with no need of traversing all the columns to estimate host cardinality;
(b) high accuracy: the *flag*-based column combination can reduce false positive caused by hash collisions and wrong column combination. The reason is that the link information that shows the locations of a super host stored in ExtendedSketch can be successfully generated since it will simultaneously exhibit abnormal behavior over $(ES_1, \ldots, ES_H)$.

4 Theoretical Analysis

In this section, we provide a formal analysis of ExtendedSketch. The analysis is completed under the condition that the number of arrays is H, the number of columns in each array is P, the counter size in each bucket is K, the counter size in *core part* of ExtendedSketch is w (much smaller that K), the domain size of IP addresses is $|\mathbf{E}|$, the number of super hosts is N, the average column extension times of N super hosts in ExtendedSketch is a.

4.1 Analysis on Space and Time Complexities

Table 3 provides the comparison of ExtendedSketch with other sketches regarding to space and time complexities. We assume that the counters contained in bit arrays have $O(1)$ time complexity.

Memory Usage. In *core part* of ExtendedSketch, there are nearly wPH buckets. The extended part has $(2^a - 1)wHN$ buckets. Each bucket in ExtendedSketch holds one-bit value. Thus, the memory usage of ExtendedSketch is $O(wH(P + 2^aN))$. Because w is far smaller than P and a is always a small integer, ExtendedSketch has the smallest memory usage than other sketches, as indicated in Table 3.

Update Time. Updating a flow needs H+1 hash operations and then accesses H buckets over ES. Thus, the total update time complexity of ExtendedSketch is $O(H)$, which is the same as DCDS and SS.

Identification Time. ExtendedSketch traverses all additional information triples to find out abnormal columns that super hosts locate. Then, it accesses H columns to calculate the cardinality of each super host. Thus, the identification time of ExtendedSketch is $O(NH)$, which is the lowest since N is far smallest than P.

TABLE 3
Theoretical Comparison of ExtendedSketch With Other Sketches

Method	MU	UT	IT	RCT
DCDS [22]	$O(KHP)$	$O(H)$	$O(PH)$	$O(N^H)$
VBF [23]	$O(KP\log\log\|\mathbf{E}\|)$	$O(\log\log\|\mathbf{E}\|)$	$O(P\log\log\|\mathbf{E}\|)$	$O(N^{\log\log\|\mathbf{E}\|})$
FS [24]	$O(KH\log\frac{\|\mathbf{E}\|}{P})$	$O(H\log\frac{\|\mathbf{E}\|}{P})$	$O(P\log\frac{\|\mathbf{E}\|}{P})$	$O(\log^{O(1)}\frac{\|\mathbf{E}\|}{P}+H)$
SS [25]	$O((K+\log\|\mathbf{E}\|+\log\log\|\mathbf{E}\|)HP)$	$O(H)$	$O(PH)$	Record IP addresses
ExtendedSketch	$O(wH(P+2^aN))$	$O(H)$	$O(NH)$	$O(H)$

MU: Memory Usage; UT: Update Time; IT: Identification Time; RCT: Reversible Calculation Time.

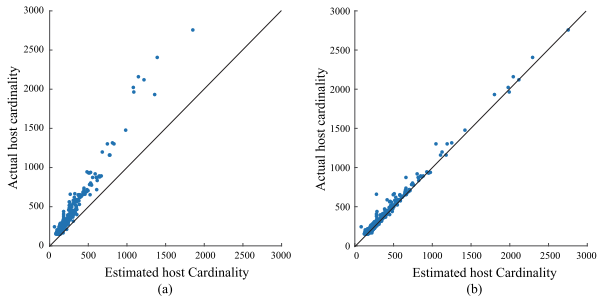


Fig. 5. Comparison of two estimation operations of ExtendedSketch based on CAIDA1 dataset. (a) Estimation without error compensation; (b) Estimation with error compensation.

Reversible Calculation Time. For each super host, ExtendedSketch reversely computes its original address based on Chinese Remainder Theorem. The complexity of *flag*-based reversible calculation operation depends on the number of bit arrays H. Therefore, the reversible calculation complexity of ExtendedSketch is $O(H)$, which is much smaller than DCDS, VBF and FS. Although SS finds out abnormal addresses without reversible calculation, recording IP addresses leads to extra memory usage and only part of the addresses can be recorded.

4.2 Analysis of Estimation Operation of ExtendedSketch

The estimation operation of ExtendedSketch is based on the probabilistic counting algorithm, which is widely used in many schemes for host cardinality estimation, e.g., [20], [22], [23]. Therefore, the analysis processes of Formula (1) with regard to mean and variance are similar to [22] and [23], which we omit here. In this subsection, we focus on the analysis of Formula (2). We first study the estimation error caused by column extension, as discussed in Theorem 1.

Theorem 1. *For a column with a number of t times column extension, its estimation error in terms of host cardinality calculation is* $\ln\frac{(2-\varepsilon)^2}{4(1-\varepsilon)} w \sum_{u=1}^{t} 2^{u-1}$.

Proof. Column $ES_i(s)$ with an initialized length w will be extended when the number of bits in it reaches $\varepsilon\omega$. At this moment, the cardinality of s estimated from $ES_i(s)$ is $-w\ln(1-\varepsilon)$. The extended $ES_i(s)$, denoted as $ES_i(s)[1]$, has $2w$ buckets after the first time column extension. Since the number of bits in $ES_i(s)[1]$ is the same as in $ES_i(s)$, the estimated host cardinality calculated from $ES_i(s)[1]$ is $-2w\ln\frac{(2-\varepsilon)}{2}$. Hence, the estimation error caused by the first time extension is $w\ln\frac{(2-\varepsilon)^2}{4(1-\varepsilon)}$. In a similar way, $ES_i(s)[1]$ will be further extended to $ES_i(s)[2]$ if the number of bits in $ES_i(s)[1]$ is $2\varepsilon\omega$. The estimation error between $ES_i(s)[1]$ and $ES_i(s)[2]$ is $2w\ln\frac{(2-\varepsilon)^2}{4(1-\varepsilon)}$. The extension process will continue as long as the extension condition is satisfied. Finally, the total estimation error of $ES_i(s)$ caused by t times column extension is the sum of estimation error in each extension namely $\ln\frac{(2-\varepsilon)^2}{4(1-\varepsilon)} w \sum_{u=1}^{t} 2^{u-1}$. □

Therefore, as indicated in Formula (2), the estimated host cardinality of an extended column contains the cardinality estimated by the probabilistic counting algorithm and the cardinality that is lost during column extension process.

Fig. 5 compares the accuracy of estimation operation of ExtendedSketch with and without error compensation, where x-coordinate is the estimated host cardinality and y-coordinate is the actual cardinality (we select the host whose cardinality is in the scope of $150\sim3000$ as examples). The diagonal line in the Fig. 5 is used to evaluate the estimation results. The closer a point is located to the line, the closer the estimated cardinality to its actual value is. As we can see from the figure, the performance of cardinality estimation of ExtendedSketch is greatly improved by compensating the estimation error caused by column extension.

5 Experimental Study and Performance Evaluation

We conducted a series of experiments to evaluate the performance of ExtendedSketch by comparing with other state-of-the-art sketches based on real world traffic data. In this section, we first discuss the experimental settings. Next, some evaluations of super host identification are reported. All experiments were implemented with Python programming language on a MacOS with an Intel Core i5 CPU @ 2.3GHZ and 8.0GB RAM. We have released the source code of ExtendedSketch at GitHub [38].

5.1 Experimental Setting

5.1.1 Evaluation Metrics

We consider the following evaluation metrics:

- Precision Rate (PR): the ratio of true super hosts identified over all super hosts detected;
- Recall Rate (RR): the ratio of true super hosts identified over all true super hosts;
- F1 score: $2 \times PR \times RR/(PR + RR)$;
- Average Relative Error (ARE): the relative error of host cardinality, namely $\frac{1}{n}\sum_{i=1}^{n}|\hat{d}_i - d_i|/d_i$, where $\hat{d}_i$ is the estimated cardinality, d_i is the true cardinality and n is the number of hosts;
- Throughput: the number of flows processed per second;
- Time consumption: the time spent on super host identification and reversible calculation of super host addresses.

5.1.2 Datasets Description

In our experiments, we used three 10-minutes traffic traces selected from CAIDA equinix-nyc and equinix-chicago [39], referred to CAIDA1, CAIDA2, and CAIDA3. Each trace contains ten temporal intervals with one-minute-long. All traces are converted from packet-level into flow-level by using CICFlowMeter [40]. Table 4 shows the detailed information of the three traffic traces. In this table, #Flows denotes the numbers of different flows; #SIP denotes the numbers of different source addresses; #SS is the numbers of super spreaders; T_SS is the threshold that used to identify super spreaders. We first set the threshold in each temporal interval as 0.03% of the total destination cardinality. T_SS is the average value of these thresholds. #SC is the number of super changers; T_SC is the threshold that used to identify super changers. We set the threshold as 0.02% of

TABLE 4
Traffic Traces Used in Experiments (M: Million)

Trace	#Flows	#SIP	#SS	T_SS	#SC	T_SC
CAIDA1	16.31M	0.76M	199	379	127	193
CAIDA2	14.31M	0.73M	145	313	65	142
CAIDA3	13.62M	1.95M	42	182	12	72

the total cardinality changes between two adjacent temporal intervals. T_SC is the average of these values.

5.1.3 Comparative Analysis

We compared ExtendedSketch with several existing sketches that are used to identify super host, including Random Aging Streaming Filters (RASF) [21], Double Connection Degree Sketch (DCDS) [22], Vector Bloom Filter (VBF) [23] and SpreadSketch (SS) [25]. All of them have a good performance in super host identification. We set the number of two-dimensional bit arrays as 4 in ExtendedSketch and DCDS and as 5 in VBF. All comparison work was operated under the same memory size by tuning the number of columns and rows. For ExtendedSketch, we allocated 90% of available memory usage for *core part* and 10% of available memory usage for *extended part*.

5.2 Performance Evaluation

5.2.1 Host Cardinality Estimation

Fig. 6 shows ARE values of all sketches on three traffic traces versus memory usage. The experimental results show that ExtendedSketch achieves more accurate cardinality estimation for super hosts than other sketches with a small memory. When allocating 0.25MB memory to all sketches, the ARE of ExtendedSketch is about 1.4, 4.78, 4.82, and 7.22 times lower than SS, RASF, DCDS, and VBF on CAIDA1 (similar observations on other traffic traces). For RASF, DCDS, and VBF, small memory limits the length of columns that source addresses are hashed into and further negatively impacts the maximum estimation capability of the probabilistic counting algorithm used to estimate host cardinality. Such a situation provides accurate estimation on low-cardinality hosts but inaccurate to monitor high-cardinality hosts. That is to say, these three sketches require long enough columns to complete accurate estimation for both high- and low- cardinality

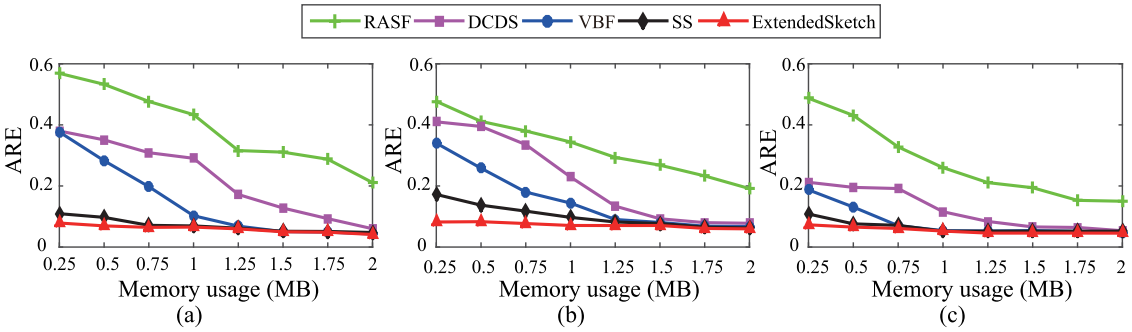


Fig. 6. Experimental results of host cardinality estimation. (a) CAIDA1; (b) CAIDA2; (c) CAIDA3.

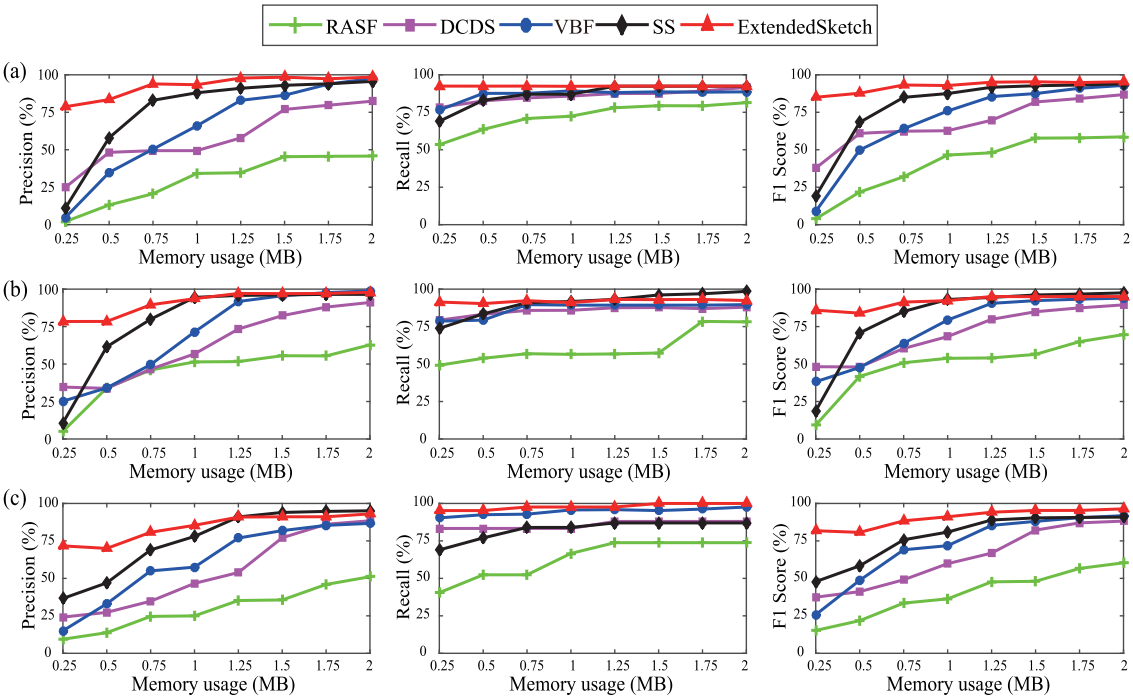


Fig. 7. Experimental results of super spreader identification. (a) CAIDA1; (b) CAIDA2; (c) CAIDA3.

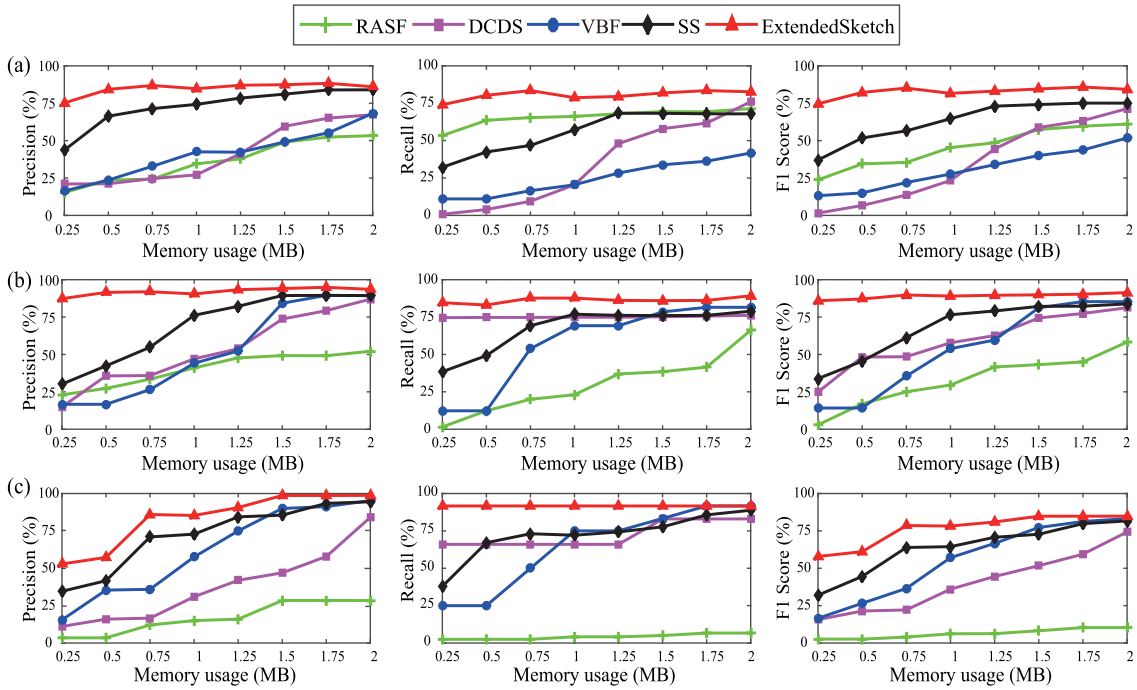


Fig. 8. Experimental results of super changer identification. (a) CAIDA1; (b) CAIDA2; (c) CAIDA3.

hosts. For SS, its limited length of columns causes much hash collisions, so that some host cardinality information is lost. As we can see from Fig. 6, the values of AREs of RASF, DCDS, VBF, and SS continuously decrease as the memory size increases. In contrast, ExtendedSketch has no strict requirement on the length of columns since it can dynamically enlarge the length according to the cardinality distribution of network traffic.

In summary, through comparison, we can see that ExtendedSketch has the capability of high-cardinality host monitoring and achieves memory efficiency at the same time.

5.2.2 Super Host Identification

Figs. 7 and 8 show the results of super host identification of all compared sketches in terms of all traffic traces versus the memory usage. We obtain some observations from these figures.

First, the results clearly show that ExtendedSketch performs much better in super spreader and super changer identification than other sketches when the memory is tight (from 0.25MB to 1.25MB). For example, when allocating 0.25MB to all sketches, the F1 score of ExtendedSketch for super spreader identification is about 2.62, 4.09, 8.79, and 16.8 times higher than DCDS, SS, VBF and RASF on CAIDA1, respectively. Notably, similar results are achieved regarding other traffic traces. It always obtains higher precision, recall and F1 score than all benchmark sketches at the same time over all traffic traces. Thanks to the extension strategy, ExtendedSketch can dynamically increase the size of each counter according to the cardinality of each host. This property allows us to allocate a large number of columns (P) and a small number of rows (w) under a given memory size. Therefore, the probability of hash collisions that different source addresses are hashed into one column is reduced. Moreover, the *flag*-based reversible calculation method further decreases false positives caused by wrong column combination. All these advantages make ExtendedSketch outperform in super host identification.

Second, RASF, DCDS, VBF and SS have low precision in both super spreader and super changer detection under a small memory, which means that they report many benign hosts as abnormal. With insufficient number of columns, hash collisions occurred with high probability. Such a situation makes RASF, DCDS, VBF and SS return a lot of false positives, which reduces the precision of their identification. Moreover, wrong column combination in DCDS and unexpected IP merging in VBF also generate false positives. For RASF without enough memory support, it needs to execute an aging algorithm frequently, which decreases the accuracy of host cardinality estimation and further highly degrades super host identification.

Third, although ExtendedSketch competes SS and VBF with regard to recall in super spreader identification, the latter two have much lower precision than ExtendedSketch, especially when memory size is limited.

In summary, from the comparison results, we can conclude that ExtendedSketch can accurately identify super hosts by using a tight memory size.

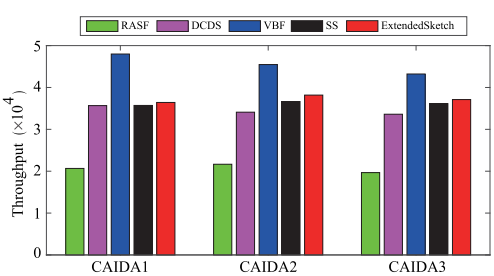


Fig. 9. Throughput of all sketches.

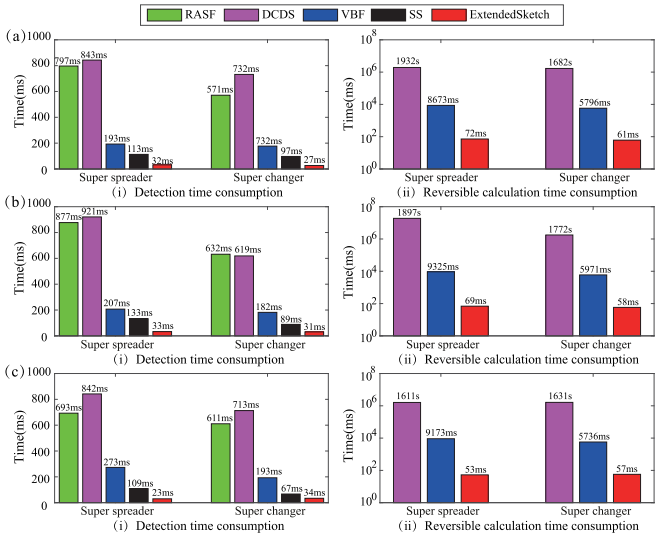


Fig. 10. Time consumption of all sketches. (a) CAIDA1; (b) CAIDA2; (c) CAIDA3.

5.2.3 Effectiveness

Fig. 9 demonstrates the throughput of all compared sketches under 1MB memory size. VBF has the highest throughput since it updates a flow by using two module hash calculations and one Bit-Extraction calculation, while other sketches need multiple hash calculations to find the column and the row that a source address locates. RASF has the lowest throughput since it needs to traverse all buckets to count the percentage of bits when updating each flow. From Fig. 9, we can see that ExtendedSketch achieves better throughput than SS and DCDS. Although ExtendedSketch has a lower throughput than VBF, it is more efficient in terms of memory consumption and more accurate for super host identification than VBF and other sketches, as illustrated in Figs. 6, 7, and 8.

Fig. 10 shows the time consumption of super host detection and reversible calculation of super host addresses. We fix the memory size of all sketches as 1MB and provide the average results over all temporal intervals on all traffic traces. Notably, we only provide reversible calculation time of DCDS, VBF and ExtendedSketch since RASF is irreversible and SS stores original addresses. As illustrated in Fig. 10, both detection time and reversible calculation time of ExtendedSketch are much lower than other sketches. In detection stage, ExtendedSketch only checks the additional information associated with columns and

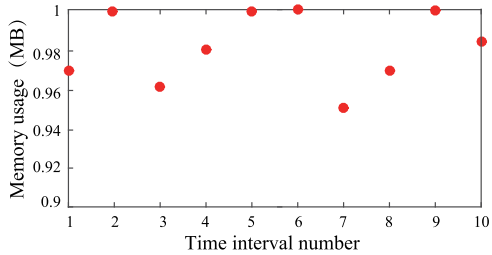


Fig. 11. Memory usage of ExtendedSketch on CAIDA1 when allocating 1MB memory size.

finds out abnormal columns with large extension times to perform further judge, while other sketches traverse each column or bucket when conducting an estimation operation. For reversible calculation, ExtendedSketch is about 120.4 and 26833.3 (95 and 27573.8) times faster than VBF and DCDS with regard to reconstructing addresses of super spreaders (super changers). The reason is DCDS generates large amount of redundant column combinations, which highly slows down its reversible calculation. VBF needs to operate several string mergences to obtain an original address. In ExtendedSketch, the *flag*-based reversible calculation method can exactly guide column combination so as to reduce the computational burden of reversible calculation.

In short, ExtendedSketch has low time overhead when detecting super hosts and reconstructing original IP addresses of super hosts.

5.2.4 Memory Efficiency

Table 5 shows the memory usage of all compared sketches when achieving same performance on CAIDA1 (the results are similar for other traces). The results obviously indicate that ExtendedSketch obtains good performance on host cardinality measurement and super host identification with only a small amount of memory.

Fig. 11 shows the memory usage of ExtendedSketch on CAIDA1 (similar results on other traces). We allocate memory size as 1MB and record memory usage at end of each time interval. Therefore, the memory usage of *core part* of ExtendedSketch is 0.9MB and the memory usage of *extended part* is up to 0.1MB. As indicated in the figure, ExtendedSketch has the ability of adaptively adjusting memory usage according to the distribution of network traffic. Such the property makes ExtendedSketch can achieve high performance while using memory as small as possible.

TABLE 5
Memory Usage of All Sketches When Achieving Same Performance on CAIDA1

Methods	ARE=0.08	Pre_SS=90%	Rec_SS=95%	F1_SS=90%	Pre_SC=90%	Rec_SC=80%	F1_SC=80%
RASF	3.25MB	3.75MB	2.75MB	3.25MB	3.75MB	3.00MB	3.00MB
DCDS	2.00MB	2.75MB	2.50MB	2.75MB	3.00MB	2.25MB	2.50MB
VBF	1.25MB	1.75MB	1.75MB	2.25MB	2.50MB	2.75MB	2.75MB
SS	0.75MB	1.25MB	1.25MB	1.50MB	1.75MB	2.50MB	2.25MB
ExtendedSketch	**0.25MB**	**0.75MB**	**0.25MB**	**0.75MB**	**0.75MB**	**0.50MB**	**0.75MB**

Pre_SS/Rec_SS/F1_SS : the precision/recall/F1 Score of super spreader identification
Pre_SC/Rec_SC/F1_SC : the precision/recall/F1 Score of super changer identification

6 Conclusion

In this paper, we proposed ExtendedSketch, a novel extensible and reversible data structure, for host cardinality estimation. It aims to achieve accurate super host identification and memory efficiency at the same time, which has been seldomly investigated in the past literature. We evaluated the performance of ExtendedSketch based on three real traffic traces. The experimental results show that, under the same memory allocation, ExtendedSketch highly outperforms other sketches in terms of host cardinality estimation accuracy, super host identification accuracy and efficiency, and abnormal addresses reconstruction efficiency. That is to say, ExtendedSketch requires only a limited amount of memory to achieve high performance, which save resources and computation consumption. All these advantages of ExtendedSketch are highly expected in practical super host identification. Regarding future work, we will integrate ExtendedSketch into a decentralized network trust evaluation framework in order to support trustworthy networking in 6G.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China under Grant 6210070253, the National Natural Science Foundation of China under Grant 62072351, the Academy of Finland under Grants 308087 and 335262, the Shaanxi Innovation Team project under grant 2018TD-007, and the 111 project under grant B16037, as well as Huawei Technologies Group Co., Ltd.

References

[1] X. Jing, Z. Yan, and W. Pedrycz, "Security data collection and data analytics in the internet: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 586–618, First Quarter 2019.

[2] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1496–1519, Third Quarter 2014.

[3] S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proc. 16th Int. Conf. Extending Database Technol.*, pp. 683–692, 2013.

[4] P. Wang, P. Jia, X. Zhang, J. Tao, X. Guan, and D. Towsley, "Utilizing dynamic properties of sharing bits and registers to estimate user cardinalities over time," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 1094–1105.

[5] G. Cormode and S. Muthukrishnan, "What's new: Finding significant differences in network data streams," *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1219–1232, Dec. 2005.

[6] S. G. Chen, M. Chen, and Q. J. Xiao, "Traffic measurement for big network data," in *Wireless Networks*, Cham, Switzerland: Springer, 2016.

[7] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, "Networking for big data: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 531–549, First Quarter 2017.

[8] T. Yang *et al.*, "A generic technique for sketches to adapt to different counting ranges," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2017–2025.

[9] G. Zhao, L. Huang, Z. Yu, H. Xu, and P. Wang, "On the effect of flow table size and controller capacity on SDN network throughput," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[10] T. Yang, S. Gao, Z. Sun, Y. Wang, Y. Shen, and X. Li, "Diamond sketch: Accurate per-flow measurement for big streaming data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2650–2662, Dec. 2019.

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

[12] Q. Xiao *et al.*, "Cardinality estimation for elephant flows: A compact solution based on virtual register sharing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3738–3752, Dec. 2017.

[13] T. Qin, Z. Liu, P. Wang, S. Li, X. Guan, and L. Gao, "Symmetry degree measurement and its applications to anomaly detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1040–1055, 2020, doi: 10.1109/TIFS.2019.2933731.

[14] X. Y. Jing, Z. Yan, X. Q. Liang, and W. Pedrycz, "Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch," *Inf. Fusion*, vol. 51, pp. 100–113, 2018.

[15] C. Wang, T. T. N. Miu, X. Luo, and J. Wang, "SkyShield: A sketch-based defense system against application layer ddos attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 559–573, Mar. 2018.

[16] H. Dai, M. Shahzad, A. X. Liu, M. Li, Y. Zhong, and G. Chen, "Identifying and estimating persistent items in data streams," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2429–2442, Dec. 2018.

[17] H. Huang *et al.*, "You can drop but you can't hide: K-persistent spread estimation in high-speed networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2018, pp. 1889–1897.

[18] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpensKetch," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 29–42.

[19] H. Xu, Z. Yu, C. Qian, X.-Y. Li, Z. Liu, and L. Huang, "Minimizing flow statistics collection cost using wildcard-based requests in SDNs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3587–3601, Dec. 2017.

[20] Q. Zhao, J. Xu, and A. Kumar, "Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1840–1852, Oct. 2006.

[21] M. Yoon and S. Chen, "Detecting stealthy spreaders by random aging streaming filters," *IEICE Trans. Commun.*, vol. E94-B, no. 8, pp. 2274–2281, 2011.

[22] P. Wang, X. Guan, T. Qin, and Q. Huang, "A data streaming method for monitoring host connection degrees of high-speed links," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 1086–1098, Sep. 2011.

[23] W. Liu, W. Qu, J. Gong, and K. Li, "Detection of superpoints using a vector bloom filter," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 3, pp. 514–527, Mar. 2016.

[24] Y. Liu, W. Chen, and Y. Guan, "Identifying high-cardinality hosts from network-wide traffic measurements," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 547–558, Sep./Oct. 2016.

[25] L. Tang, Q. Huang, and P. P. Lee, "SpreadSketch: Toward invertible and network-wide detection of superspreaders," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2020, pp. 1608–1617.

[26] M. Yoon, T. Li, S. Chen, and J.-K. Peir, "Fit a compact spread estimator in small high-speed memory," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1253–1264, Oct. 2011.

[27] Q. Xiao, S. Chen, Y. Zhou, and J. Luo, "Estimating cardinality for arbitrarily large data stream with improved memory efficiency," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 433–446, Apr. 2020.

[28] J. Xu, W. Ding, X. Y. Hu, and Q. S. Gong, "VATE: A trade-off between memory and preserving time for high accurate cardinality estimation under sliding time window," *Comput. Commun.*, vol. 138, pp. 20–31, 2019.

[29] T. Yang *et al.*, "HeavyKeeper: An accurate algorithm for finding top-k elephant flows," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1845–1858, Oct. 2019.

[30] R. Schweller *et al.*, "Reversible sketches: Enabling monitoring and analysis over high-speed data streams," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1059–1072, Oct. 2007.

[31] C. Hu *et al.*, "Discount counting for fast flow statistics on flow size and flow volume," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 970–981, Jun. 2014.

[32] T. Qin, X. H. Guan, W. Li, P. H. Wang, and M. Zhu, "A new connection degree calculation and measurement method for large scale network monitoring," *J. Netw. Comput. Appl.*, vol. 41, pp. 15–26, 2014.

[33] X. Y. Jing, J. J. Zhao, Q. H. Zheng, Z. Yan, and W. Pedrycz, "A reversible sketch-based method for detecting and mitigating amplification attacks," *J. Netw. Comput. Appl.*, vol. 142, pp. 15–24, 2019.

[34] L. H. Chi and X. Q. Zhu, "Hashing techniques," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 1–36, 2017.

[35] K. Y. Whang, B. T. Vander-zanden, and H. M. Taylor, "A linear-time probabilistic counting algorithm for database applications," *ACM Trans. Database Syst.*, vol. 15, no. 2, pp. 208–229, 1990.

[36] J. Xu, "READ: A three-communicating-stage distributed super points detections algorithm," 2019, *arXiv:1907.08057*.

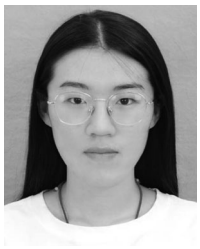
[37] X. Jing, H. Han, Z. Yan, and W. Pedrycz, "SuperSketch: A multi-dimensional reversible data structure for super host identification," *IEEE Trans. Dependable Secure Comput.*, 2021, doi: 10.1109/TDSC.2021.3072295.
[38] The source codes of extendedsketch. 2021. [Online]. Available: https://github.com/JasonXYJing/ExtendedSketch
[39] The caida anonymized internet traces. 2015. [Online]. Available: https://www.caida.org/catalog/datasets/monitors/passive-equinix-chicago/
[40] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISS*, 2017, pp. 253–262.



Xuyang Jing received the PhD degree in cyberspace security from Xidian University, Xi'an, China, in 2020. He is currently a teacher with the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include network security analysis, network traffic classification, and sketches.



Zheng Yan (Senior Member, IEEE) received the DSc degree in technology from the Helsinki University of Technology, Espoo, Finland, in 2007. She is currently a professor with the School of Cyber Engineering, Xidian University, Xi'an, China and a visiting professor and Finnish Academy research fellow with the Aalto University, Helsinki, Finland. Her research interests include trust, security, privacy, and security related data analytics. She is an area editor or an associate editor of the *IEEE Internet of Things Journal*, *Information Fusion*, *Information Sciences*, *IEEE Access*, and *Journal of Network and Computer Applications*. She served as a general chair or program chair for numerous international conferences, including IEEE TrustCom 2015 and IFIP Networking 2021. She is a Founding Steering Committee co-chair of IEEE Blockchain conference. She received several awards in recent years, including the Distinguished Inventor Award of Nokia, Aalto ELEC Impact Award, Best Journal Paper Award issued by IEEE Communication Society Technical Committee on Big Data and the outstanding associate editor of 2017 and 2018 for *IEEE Access*, etc.



Hui Han is currently working toward the master's degree in cyberspace security at Xidian University, Xi'an, China. Her research interest include network traffic measurement.



Witold Pedrycz (Life Fellow, IEEE) received the MSc, PhD, and DSci degrees from the Silesian University of Technology, Gliwice, Poland. He is currently a professor and the Canada research chair in computational intelligence with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He is a foreign member of the Polish Academy of Sciences. He has authored 15 research monographs covering various aspects of computational intelligence, data mining, and software engineering. His current research interests include computational intelligence, fuzzy modeling, and granular computing, knowledge discovery and data mining, fuzzy control, pattern recognition, knowledge-based neural networks, relational computing, and software engineering. He has published numerous papers in the above areas. He was a fellow of the Royal Society of Canada, Ottawa, ON, Canada, in 2012, and is currently a member of number of editorial boards of other international journals. He received a prestigious Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society, in 2007, and also received the IEEE Canada Computer Engineering Medal, a Cajastur Prize for soft computing from the European Center for Soft Computing, a Killam Prize, and a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society. He is a member of numerous program committees of the IEEE conferences in the area of fuzzy sets and neurocomputing. He is intensively involved in editorial activities. He currently serves on the advisory board of the *IEEE Transactions on Fuzzy Systems*. He is the editor-in-chief of *Information Sciences*, *WIREs Data Mining and Knowledge Discovery* (Wiley), and *International Journal of Granular Computing* (Springer).

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.