A Novel Data Streaming Method Detecting Superpoints

Weijiang Liu*†, Wenyu Qu†, Gong Jian‡ and Li Keqiu *
*School of Electronic and Information Engineering, Dalian University of Technology, Dalian , China
†School of Information Science and Technology, Dalian Maritime University, Dalian, China
‡School of Computer Science and Engineering, Southeast University , Nanjing , China
Email: wjliu@dlmu.edu.cn, eunice.qu@gmail.com, jgong@njnet.edu.cn, likeqiu@gmail.com

***Abstract*—Internet attacks such as distributed denial-of-service (DDoS) attacks and worm attacks are increasing in severity. Identifying realtime attack and mitigation of Internet traffic is an important and challenging problem for network administrators. A compromised host doing fast scanning for worm propagation can make a very high number of connections to distinct destinations within a short time. We call such a host a superpoint, which is the source that connect to a large number of distinct destinations. Detecting superpoints can be utilized for traffic engineering and anomaly detection. We propose a novel data streaming method for detecting superpoints and prove guarantees on their accuracy and memory requirements. The core of this method is a novel data structure called Vector Bloom Filter (VBF). A VBF is a variant of standard Bloom Filter (BF). The VBF consists of 6 hash functions, 4 hash functions of which projectively select some consecutive bits from original strings as function values. We obtain the information of superpoints using the overlapping of hash bit strings of the VBF. The theoretical analysis and experiment results show that our schemes can precisely and efficiently detect superpoints.**

I. Introduction

With the rapid growth of Internet, network security has become one of the major challenges of communication networking. Security breaches come in many forms, such as distributed denial-of-service (DDoS) attacks and worm attacks. The approach typically taken to manage those breaches has been to accept the loss when it occurs, and in parallel to develop and deploy methods to reduce the likelihood of loss. Network security monitoring is a kind of approach that defend against and mitigate such large-scale Internet attacks. Firewalls and intrusion detection/prevention systems (IDS/IPS) are the cornerstone of a networks security infrastructure [1].

Recently, worms on the Internet have become a pressing issue following a series of events (such as Slammer, Blaster and Nachi). A compromised host can send packets to unusually high number of distinct destinations for worm propagation in a short time interval. The Slammer worm caused some infected hosts to send up to 30,000 scans a second [2]. We call such a host a superpoint. To mitigate worm spreading, a network monitoring point need quickly identify superpoints and take appropriate action. To control the consumption of resources in measurement, we propose a data streaming method to identify superpoints without maintaining the identifier information of source node.

Our data structure is a variant of Bloom Filters, consisting of 5 $n \times m$ 2-dimentional bit arrays and 6 hash functions. There is the similar data structure in [3], but we develop a new method that constructs host information, identify superpoints and estimate cardinality of superpoints. For each packet coming, 5 bits selected in the bit arrays are set to one. The row indexes for these 5 bits are calculated by the 5 hash functions with inputs as source IP address, and the column indexes are obtained by the same hash function with the input as flow label. Four hash functions of the five hash functions are used to reconstruct the host address–the identifier of superpoint, the other one filters the candidate of superpoint out. The theoretical analysis and experiment results show that our schemes can precisely and efficiently detect superpoints.

Note that we cannot detect a malicious host that spoofs IP addresses and contacts many destinations since VBF relies on the soure IP address field. Some IP traceback systems provide a means to find the real attack sources [4],[5].

The main contributions of our work include:

- We design four hash functions, each of which maps a superpoint ID into the part bit strings of the ID. This is the key to reconstruct host address (superpoint id).
- We give the method for reconstructing the information of superpoints. Although the VBF do not preserve any host address information, we can reconstruct the address information of host using the overlapping of hash bit strings of the VBF.
- We introduce bitwise-and in estimating the cardinality of superpoints. Using bitwise-and requires less bit operating number than using bitwise-or. Furthermore, the estimate is more steady.

The rest of this paper is organized as follows. In the next section, we present a brief look at related work with a discussion about the context of our work. In Section III, we review some elementary concepts on superpoints and counting Bloom filter. In Section IV, we propose our data streaming model that describes how to construct an VBF and identify superpoints. Furthermore, we present the analysis of VBF performance. In Section V, we evaluate their performance by using some public traces to experiment. We conclude in Section VI.

II. Related Work

This problem of detecting superpoints has been studied in recent years. Snort [6] and Flowscan [7] maintaining per-

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

flow state require large quantities of DRAM for operation, so they are unfeasible for monitoring on high speed links. Venkataraman [8] proposes two flow sampling techniques for detecting superspreaders. But its accuracy is limited by sampling rate. Qi Zhao [3] proposes two algorithms to solve the problems using data streaming algorithm and sampling technology. But his paper focused on estimating the number of flows in source or destination IP, and didn't give a method to keep the source/destination IP records. Noriaki [9] proposes an adaptive method of identifying super points by flow sampling, and Guang Cheng [10] proposes an algorithm based on adaptive sampling. Their common feature is constructing hash link list for storing the information of hosts (IP) sampled in memory. The number of memory accesses is drastically determined by the list length. The maximum list length will become a bottleneck on high speed links. Authors in [11] uses the remainder theorem to reconstruct host addresses, but computational time of large prime is not ignorable. Jin cao et al in [12] proposes an online sampling approach for identifying high cardinality host. Estimation accuracy is subject to sampling rate.

III. Some Elementary Concepts

A. Definition of superpoints

There are at least a few definitions for the term flow depending on the context of research. In this study, we employ the one adopted from [13] which stems from the packet train model by Jain and Routhier [14].

Definition 1. A flow is defined as a stream of packets subject to flow specification and timeout.

In most cases, we call flow specification as flow identifier. When a packet arrives, the specific rules of flow specification determine which active flow this packet belongs to, or if no active flow is found that matches the description of this packet, a new flow is created. In our experiments, a flow is a stream of packets subject to timeout and having the same source and destination IP addresses.

Definition 2. For a measurement period Φ and an arbitrary given threshold m^*, a superpoint is a source (destination) that connects to at least m^* distinct destinations (sources) within a measurement period Φ.

There are two kinds of superpoint: one is a host that connects to a larger number of distinct destinations (i.e. superspreader in [8], super source), the other is a host that is contacted by a large number of sources (super destination). Finding super destination is the dual of finding super source. Without loss of generality, we only describe our algorithm for identifying super source. The algorithm can be easily applied to find super destination by replacing source IP by destination IP. In this paper, all IP address are 32-bit, i.e., IPV4 address. IP address is the inherent identifier of a superpoint. In general, we define a superpoint as a host that contributes more than $\epsilon\%$ of all flows. In this paper, we select ϵ=0.1 in experiment respectively.

TABLE I
The hash functions of the VBF

function name	output
h_1	first byte, high 4 bits of second byte of SIP
h_2	second byte, high 4 bits of third byte of SIP
h_3	third byte, high 4 bits of fourth byte of SIP
h_4	fourth byte, high 4 bits of first byte of SIP
h_5	generate uniform output of 12 bits string
f	generate uniform output over $\{0, \ldots, m-1\}$

B. Bloom Filter

A standard bloom filter (BF) for representing a set $S = \{x_1, x_2, \ldots, x_n\}$ of n elements is described by an array of m bits, initially all set to 0. A BF uses k independent hash functions $h_1, \ldots, h_k$ with range $\{1, \ldots, m\}$. For each element $x \in S$, the bits $h_i(x)$ are set to 1 for $1 \leq i \leq k$. A location can be set to 1 multiple times, but only the first change has an effect. To check if an item y is in S, we check whether all $h_i(y)$ are set to 1. If not, then clearly y is not a member of S. If all $h_i(y)$ are set to 1, we assume that y is in S, although we are wrong with some probability. Hence a BF may yield a false positive, where it suggests that an element y is in S even though it is not.

IV. Data streaming model

In this section we present a model for detecting superpoints. We begin with a discussion of constructing a Vector Bloom Filter (VBF), and then describe how use VBF to detect superpoints. We also analyze the complexity and accuracy of the model.

A. Hash functions of a VBF

A vector bloom filter (VBF) is a variant of bloom filter which consists of $k + l + 1$ hash functions. The first k hash functions which projectively select some consecutive bits from original strings as function values, are used to reconstruct host addresses. The middle l hash functions which can ensure the sources hashed into value range uniformly, are used to filter some host candidates out. The last functions is used to generate column index of bit for all flows insert. The vector means that each entry in VBF is not a single bit but instead a bit vector. Instead of having one array of size n shared by the $k+l$ hash functions, each hash function has a range of n consecutive vector locations disjoint from all others. In this work, k is set to 4, and l is set to 1. The meaning of these hash functions is showed in Table I.

Let IP address $A = s_1s_2s_3s_4s_5s_6s_7s_8$ be a superpoint identifier, where $s_i (1 \leq i \leq 8)$ all are 4 bits strings. An IP address is a 32-bit string. Then

$h_1(A) = s_1s_2s_3, h_2(A) = s_3s_4s_5, h_3(A) = s_5s_6s_7$, and $h_4(A) = s_7s_8s_1$.

Now, we introduce string functions: $tail$ and $head$. For w a string of length n bits, $tail(w)$ chooses low 4 bits of w and $head(w)$ gets high 4 bits of w. For example, if $w = s_1s_2s_3s_4$, then $tail(w) = s_4$, $head(w) = s_1$.

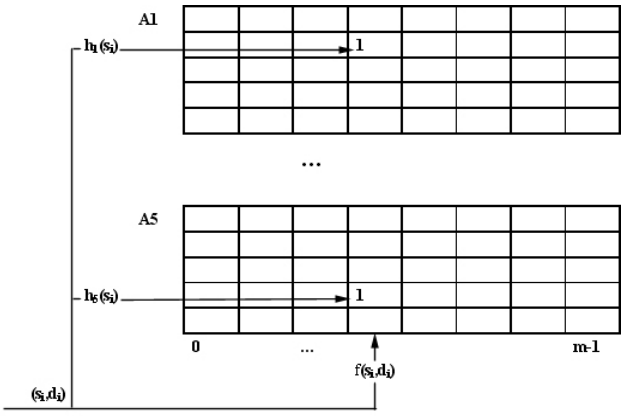

Fig. 1. Update procedure for a VBF

Definition 3 Let H_1, H_2 be hash functions. If $tail(H_1(B)) = head(H_2(B))$ for any IP identifier B, we say that H_1 is overlap-related to H_2, denoted as $H_1 \rightarrow H_2$.

$H_1 \rightarrow H_2$ means that the hashed values of the same original string by H_1 and H_2 have the same value at the overlapping bit positions.

Property 1. There is an overlap-relation loop in the VBF, i.e., $h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4 \rightarrow h_1$.

Property 1 can be induced from the definition of hash functions of the VBF. It suggests the overlap relationship between these hash functions. These relationships are the key to reconstruct original string (IP address). The length of overlap string is a trade-off between space efficiency and reconstructing reliability. The longer the overlap string is, the larger the space overhead is, but the higher the reliability is. When all original strings are not displayed, it is a key point to judge whether two different hash strings are mapped from the same original string. The overlapped bits among different hash strings deployed in this paper really benefits the judgement. We claim that two hash values obtained by two overlay-related hash functions are mapped from the same original string if they have the same value at the overlapping bit positions.

B. Data structure and algorithm

The data structure used in the VBF is denoted as $\mathbf{A} = (A_1, A_2, \cdots, A_5)$. $A_i (1 \leq i \leq 5)$ is a $4096 \times m$ bit array $A_i[j][k] (0 \leq j < 4096, 0 \leq k < m)$ associated with a function h_i. All $A_i (1 \leq i \leq 5)$ share a hash function $f : \{0, 1, \cdots, 2^{32} - 1\} \rightarrow \{0, 1, \cdots, m - 1\}$. When a packet $p_i = (s_i, d_i))$ arrives, each A_k is updated by setting the bit in its row $h_k(s_i)$ and column $f(s_i, d_i)$ as illustrated in Fig.1. The bits in $\mathbf{A}$ are set to all 0s at the beginning of measurement. The algorithm of updating $\mathbf{A}$ is shown in Fig.2.

Let w be a superpoint identifier, $|w|$ denote the cardinality of w, i.e., the number of all flows generated by w. Note that $A_i[j]$ is the jth row of A_i(viewed as bit vectors), let $C(A_i[j])$ denote the number of bits in $A_i[j]$ that are 0s. Assume that no other source is hashed to the same rows as w is hashed. Thus,

1. **Initialize**
2. $A_k[j, l] = 0, \quad k = 1, \ldots, 5 \quad j = 0, \ldots, 4047$
   $l = 0, \ldots, m - 1$
3. **Update**
4. Upon the arrival of a packet (s_i, d_i)
5. $col = f(s_i, d_i)$
6. for k=1 to 5
7. $row = h_k(s_i)$
8. $A_k[row, col] = 1$
9. endfor

Fig. 2. Algorithm of update for VBF

$C(A_1[h_1(w)]) = C(A_2[h_2(w)]) = \cdots = C(A_5[h_5(w)])$, simply denoted as $C(w)$ irrespective of arrays. A fairly accurate estimate of $|w|$ in [15] is

$$|w| = mln\frac{m}{C(w)} \tag{1}$$

Property 2. $max\{C(A_i[h_i(w)])|1 \leq i \leq 5\} \leq C(A_1[h_1(w)]\&A_2[h_2(w)]\& \cdots \&A_5[h_5(w)]) \leq C(w)$.

Property 2 is obtained from the hash collision. According to cardinality threshold m^*, we can give a zero bit number threshold Z^*. How to obtain Z^* according to m^* will be presented in this section. Hence, by Equation (1) and Property 2, at the end of each measurement epoch, we only collect the hash strings whose corresponding vectors contain zero bit number less than Z^*. Denote the string collected as $(u, V[u])$ where u is a string of 12 bits and $V(u)$ is bit vector $A_i[u]$ of row u. Thus we get five sets H_1, H_2, H_3, H_4 and H_5 composed of $(u, V[u])$ from the five different hash spaces, respectively.

Our algorithm for merging string is shown in Fig.3. An algorithm for generating IP address is shown in Fig.4. Finally, our algorithm for detecting superpoints is given in Fig.5. Note that we use bitwise-and as the instead of bitwise-or in our algorithm, because bitwise-and have some good features:

We rewrite Formula (4) in [3] as follows

$$\begin{aligned}\hat{F}_s = \quad & \textstyle\sum_{1 \leq i \leq k} D_{T_i} - \sum_{1 \leq i_1 < i_2 \leq k} D_{T_{i_1} \bigcup T_{i_2}} \\ & + \textstyle\sum_{1 \leq i_1 < i_2 < i_3 \leq k} D_{T_{i_1} \bigcup T_{i_2} \bigcup T_{i_3}} \\ & + \cdots + (-1)^{k-1} D_{T_1 \bigcup T_2 \cdots T_k}\end{aligned} \tag{2}$$

where $\hat{F}_s$ is an estimator for $|T_1 \bigcap T_2 \bigcap \cdots \bigcap T_k|$. Although the above estimator is almost unbiased, it is not a good estimator of because computation complexity and variance. Intersection of k sets needs almost 2^k bitwise-or operations, but log_2^k operations are enough if using bitwise-and.

Consider the basic case $k = 2$. T_1 is associated with bit vector A_1, T_2 is associated with bit vector A_2, and there are m bits in each vector. Let $C(A_1)$ denote zero bit number of A_1,$C(A_2)$ denote zero bit number of A_2. Assume that the case all bits are zero or one does not happen in operation process of bitwise-or and bitwise-and. We have the following property:

```
Merge_String(Input1,Input2,Output)
1. Output=Φ
2. for each (u, V[u]) ∈ Input1
3.   for each (w, V[w]) ∈ Input2
4.     if tail(u) == head(w)
5.       v = merge(u, w);
6.       V[v] = V[u]&V[w];// & bitwise-and operator
7.       if C(V[v]) ≤ Z*
8.         adding (v, V[v]) into Output
9.       endif
10.    endif
11.  endfor
12. endfor
```

Fig. 3. Algorithm for merging string

```
Generate_IP(Input1,Input2,Output)
1. Output=Φ
2. for each (u, V[u]) ∈ Input1
3.   for each (w, V[w]) ∈ Input2
4.     if tail(u) == head(w) and tail(w) == head(u)
5.       v = merge(u, w);
6.       V[v] = V[u]&V[w];//& bitwise-and
7.       if C(V[v]) ≤ Z*
8.         adding (v, V[v]) into Output
9.       endif
10.    endif
11.  endfor
12. endfor
```

Fig. 4. Algorithm for generating IP

Property 3. $mln\frac{m}{C(A_i\&A_2)} \geq \hat{F_s}$, where & is bitwise-and operator.

Proof: Let $C(A_1) = a, C(A_2) = b$, and $C(A_1\&A_2) = c$. Hence, $C(A_1|A_2) = a + b - c$, where | is bitwise-or operator.

$$\begin{aligned}\hat{F_s} &= mln(\tfrac{m}{a}) + mln(\tfrac{m}{b}) - mln(\tfrac{m}{a+b-c}) \\ &= mlnm + mln(\tfrac{a+b-c}{ab})\end{aligned}$$

but $mln\frac{m}{C(A_i\&A_2)} = mlnm - mln(c)$, thus $mln\frac{m}{C(A_i\&A_2)} - \hat{F_s} = -mln(c) - mln(\frac{a+b-c}{ab}) = mln(\frac{ab}{(a+b-c)c})$. Since $(a-c)(b-c) \geq 0 \Leftrightarrow \frac{ab}{(a+b-c)c} \geq 1$. Therefore the property is obtained.

For example, let $m = 512$, $C(A_1) = 112, C(A_2) = 112, C(A_1\&A_2) = 212$, then $C(A_1|A_2) = 12$. We have $\hat{F_s} = -365, mln\frac{m}{C(A_i\&A_2)} = 1922$.

Although the estimator obtained by using bitwise-and is not unbiased, it has some good features. For example, according to property 2,3, it always is larger than the actual value and it is steady. Therefore, we manage to correct the deviation so that it may become a fair estimator. Now, we analyze the deviation. Without considering the estimate deviation of Equation (1). For a superpoint w, value $C(A_1[h_1(w)]\&A_2[h_2(w)]\&\cdots\&A_5[h_5(w)])$ comes from two parts: one is $C(w)$ from w, which is determinate; the other is dedicated by other flows except w, which is random. In general, the first four hash functions in VBF are not perfectly random. No random function slightly affects value because many vectors are saturated by ones. However, hash functions h_3 and h_4 can be thought as a random function in the view of performance because they contain the active low 12 bits of IP address. Moreover, h_5 may be perfectly random. Therefore, we think there are two random functions in the five functions. Now, we analyze approximately. Assume there are s flows in a measurement epoch. Averagely, each vector in a hash space receive $\frac{s}{4096}$ flows. According to Equation (1), they will occupy $t = m(1-e^{-\frac{s}{4096m}})$ bit positions in one hash space(set 1). If the corresponding bit positions of some vector in the other hash space is occupied by another flows, the deviation happens (note that we only think there are two hash spaces). Averagely, the number of flow incoming into the t bit positions is $\delta_b = \frac{t}{m} * \frac{s}{4096} = \frac{st}{4096m}$.

Let $m^* = mln(\frac{m}{x})$, where m^* is the threshold. Solving the formula, we obtain $x = me^{-\frac{m^*}{m}}$. So we set

$$Z^* = me^{-\frac{m^*}{m}} - \delta_b, \delta = mln(\frac{m}{Z^*}) - m^*. \tag{3}$$

To determine Z^* and δ, we need to estimate total flows number. Since h_5 is random, we use A_5 to estimate as follows

$$s = \sum_{i=0}^{4095} mln(\frac{m}{C(A_5[i])}) \tag{4}$$

where $C(A_5[i])$ is replaced with 1 if it is equal to 0.

Now, we use a packet trace from the MAWI working group of the WIDE project (MAWI) [18]. We will describe the trace in Section V in detail. The trace contains 2453203 packets from 157020 flows. We select ϵ=0.1 for the definition of a superpoint. Then the threshold value is $157020 * 0.1\% = 157$. Under this definition, we have 75 superpoints in this trace. In our algorithm for this trace $m = 512$, we get:

$$\delta_b = 2.53 \approx 3, Z^* = 374,\ \delta = 4.$$

C. Complexity Analysis

The above scheme has low storage (SRAM) complexity and allows for high speed links.

Memory (SRAM) consumption. Define M_{vbf} bytes as the required memory size of VBF. We have

$$M_{vbf} = 5 * 4096 * m/8 = 2560m$$

where m is the number of bit in a vector. Thus a moderate amount of SRAM can support very high link speeds. Assume that a flow consists of 10 packets, i.e., the average flow size of 10 packets [16], 640KB ($m = 256$) SRAM is can support a measurement epoch which is longer than 6 seconds for a link with 10 million packets per second. However, 3MB SRAM can support 30 seconds for OC-192. In [17] 72Mbits SRAM is in production and available today.

Streaming speed. In our algorithm, the processing time is determined by the Bloom filter. The calculation of the hash values in VBF can be executed in parallel on hardware, so we

Detect_Superpoint
1. **Merge**
2. Merge_String(H_1, H_2, H_{12})
3. Merge_String(H_{12}, H_3, H_{123})
4. Generate_IP(H_{123}, H_4, IP)
5. **Filter**
6. Output=Φ
7. for each $(v, V[v]) \in IP$
8. $V[v] = V[u] \& A_5[h_5(v)]$;
9. if $C(V[v]) \leq Z^*$
10. adding $(v, V[v])$ into Output
11. endif
12. endfor
13.**Estimate**
14. for each $(u, V[u]) \in$ Output
15. u is identified as a superpoint with cardinality
16. $Cardinality(u) = mln(\frac{m}{C(V[u])}) - \delta$
17. Endfor

Fig. 5. Algorithm for detecting superpoints

can ignore the time needed to obtain hash values. Moreover, CPU calculation time is much shorter than memory access time, so we use the required number of memory accesses to measure the required processing time. In parallel, VBF requires one write to SRAM . Using less than 5ns SRAM In [17], on an OC768 link the packet time of minimum length (40bytes) is 8ns, so the VBF can support 40Gbps link.

V. Evaluation and Experiments

In this section, we use VBF to identify superpoints of some real traces and compare the experimental results with the actual values of the traces. We also compare the VBF with other algorithms.

A. Traffic trace

In order to make the experimental data representative, we use packet header traces gathered from as different locations of the Internet as possible. They include the MAWI Working Group of the WIDE Project(MAWI) [18], Jiangsu provincial network border of China Education and Research Network (CERNET) [19], and NLANR [20]. The trace form MAWI was collected on a trans-Pacific line (150Mbps link), on March 30, 2009 at 00:00 am. The IPv6 packets of MAWI are filtered out in following experiments.

The CERNET were collected at Jiangsu provincial network border of China Education and Research Network (CERNET) on April 17, 2004. The backbone capacity is 1000Mbps; mean traffic per day is 587 Mbps.

We also use a pair of unidirectional traces from NLANR: IPKS0 and IPKS1, collected simultaneously on both directions of an OC192 link on June 1, 2004. The link uses Packet-over-SONET connecting Indianapolis (IPLS) to Kansas City (KSCY). Table II summarizes all the traces used in the evaluation. We will use all traces to evaluate our detecting algorithm.

TABLE II
Traces used in our experiments. Note that source is source IP and flow label is 2-tuple $< srcIP, dstIP >$.

Trace	♯ of sources	♯ of flows	♯ of packets	♯ of superpoints	threshold (0.1%)
IPKS0	13639	52564	1453219	87	52
IPKS1	16031	41043	1453219	79	41
MAWI	47778	157020	2453203	75	157
CERNET	30281	91195	4032981	156	91

B. Evaluation metric

We use FNR and FPR to evaluate our scheme in identifying accuracy:

$$FNR = \frac{s^-}{s}, FPR = \frac{s^+}{s}$$

where s is the actual number of superpoints, s^- is the number of superpoints being incorrectly not identified, and s^+ is the number of non-superpoints being incorrectly identified.

We adopt the Weighted Mean Relative Difference (WMRD) as our evaluation metric for estimated cardinality. Suppose the number of flows of superpoint i is n_i and our estimation of this number is $\hat{n}_i$. The value of WMRD is given by:

$$WMRD = \frac{\sum_i |n_i - \hat{n}_i|}{\sum_i (\frac{n_i + \hat{n}_i}{2})}.$$

In computing, if a non-superpoint is incorrectly identified, its n_i is zero; similarly, if a superpoint is not identified incorrectly, its $\hat{n}_i$ is 0.

C. Accuracy of the VBF

In this section, we evaluate the accuracy of the VBF in estimating the cardinalities of superpoints and in detecting superpoints. Here the VBF will be compared with the flow sampled algorithm (Sampled) in [8], and the Bitmap algorithm (Bitmap) in [3]. For Sampled, sample rate p is set 1/8 and the estimated cardinality is given by scaling cardinality sampled by 8. For Bitmap, we set the size of 2D array A to 2MB($512rows \times 16384columns$). Figure 6 compares the cardinalities of the superpoints estimated using three algorithms with their actual cardinalities in traces IPKS0, IPKS1, CERNET, and MAVI respectively, where the X_{axis} is the actual cardinalities of the superpoints, and the Y_{axis} is the estimated cardinalities of the superpoints. The diagonal line is used as a standard line to compare the estimation performance. If a point is nearer to the diagonal line, then the estimated cardinality is closer to the actual value. Figure 6 shows that nearly all points by VBF are on or closer to the diagonal line. It is seen that in estimating cardinality VBF is much better than Sampled and Bitmap. In Table III three methods are compared quantitatively. For Bitmap, the bit vectors become almost full when the cardinality value is close to 3194($mlnm$). To be fair, in Table 3, we only consider superpoints whose cardinality is between the threshold and 3000 for Bitmap. In Table III, FNR, FPR, and WMRD of VBF are relatively small that shows VBF can efficiently and accurately identify the superpoints in the traces.

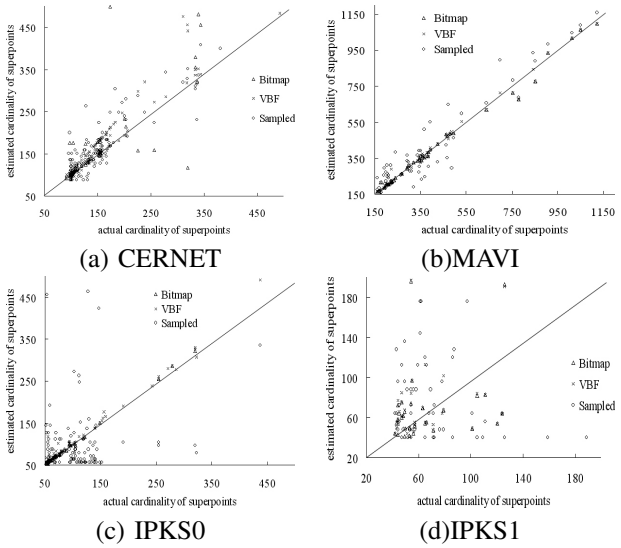

Fig. 6. Comparison of Accuracy in estimating cardinalities of VBF, Sampled and Bitmap.

TABLE III
EVALUATION METRIC BY THE THREE ALGORITHMS FOR DISTINCT TRACES

Trace	Algorithms	FNR	FPR	WMRD
IPKS0	VBF	0.05	0.29	0.18
	Sampled	0.05	0.51	0.48
	Bitmap	0.35	0.06	0.56
IPKS1	VBF	0.04	0.05	0.12
	Sampled	0.06	0.75	0.53
	Bitmap	0.27	0.03	0.86
MAWI	VBF	0	0.04	0.08
	Sampled	0.02	0	0.12
	Bitmap	0.40	0.11	0.60
CERNET	VBF	0.01	0.01	0.23
	Sampled	0.08	0.20	0.23
	Bitmap	0.50	0.25	1.40

VI. CONCLUSIONS

In this work, we have described how to construct a Vector Bloom Filter (VBF) and designed the corresponding algorithms to identify superpoints. The VBF can be used for high speed links with high accuracy. Although our current scheme only focus on detecting superpoints instead of exacting detailed flow statistics, we believe that detecting superpoints is an important problem in many network security and measurement applications. The advantages of our schemes include: 1)No need to maintain host identity in processing. 2)The Computation complexity of the hash functions is low that is due to its simply selecting some consecutive bits. 3) Can support 40Gbps links because parallel hash functions diminish access SRAM time.The theoretical analysis shows that false negative rate originated by hash collision is very low. The experimental results demonstrate that our schemes can identify superpoints precisely and efficiently.

For future work, we would like to combine flow sampling method with VBF for enhancing scalability.

ACKNOWLEDGMENT

This work is supported in part by 973 Program of China under Grant No.2009CB320505, National Natural Science Foundation of China under Grant No.90818002, 60973115 and China Postdoctoral Science Foundation funded project under Grant No.20080430182.

REFERENCES

[1] A.El-Atawy,E.Al-Shaer,T.Tran and R.Boutaba, Adaptive early packet filtering for protecting firewalls against DoS attacks. IEEE INFOCOM 2009, Brazil, April 2009. pp 2437-2445
[2] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. Security and Privacy Magazine. 2(4) ,pp 33-39, 2003
[3] Qi Zhao, Abhishek Kumar, and Jun Xu, Joint Data Streaming and Sampling Techniques for Detection of Super Sources and Destinations. IMC 2005, pp 77-90.
[4] MT.Goodrich, Probabilistic packet marking for large-scale IP traceback, IEEE/ACM Transactions on Networking 16(1), pp 15-24(2008).
[5] Yang Xiang, Wanlei Zhou, and Minyi Guo, Flexible Deterministic Packet Marking: An IP Traceback System to Find the Real Source of Attacks, IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 4, pp 567-580, Apr. 2009, doi:10.1109/TPDS.2008.132.
[6] M.Roesch. Snort-lightweight intrusion detection for network. In proc. USENIX Systems Administration Conference, pp 228-238, 1999.
[7] D. Plonka. Flowscan: A network traffic flow reporting and visualization tool. In USENIX LISA, pp 305-318,Dec. 2000.
[8] S. Venkataraman, D. Song, P. Gibbons, and A. Blum. New streaming algorithms for fast detection of superspreaders. In Proc. NDSS, pp 149-166, 2005.
[9] Noriaki Kamiyama, Tatsuya Mori, Ryoichi Kawahara: Simple and Adaptive Identification of Superspreaders by Flow Sampling. In:INFOCOM 2007. pp 2481-2485
[10] Guang Cheng, Jiang Gong, Wei Ding, Hua Wu and ShiQing Qiang. Detecting superpoint algorithm based on adaptive samling,Science China Ser E-Inf Sci, 2008, 38 (10) pp 1679-1696.
[11] Xiaohong Guan, Pinghui Wang,Tao Qin, A New Data Streaming Method for Locating Hosts with Large connection Degree. IEEE Globecom 2009, November 2009, pp 1-6.
[12] Jin Cao, Yu Jin, Aiyou Chen, Tian Bu. and Zhi-Li Zhang, Identifying High Cardinality Internet Hosts,In IEEE INFOCOM 2009, pp 810-818, April 2009.
[13] Claffy, K.C., Braun,H.W., and Polyzos, G.C.: A parameterizable methodology for Internet traffic flow profiling. IEEE JSAC**13**(1995) pp 1481-1494.
[14] Jain,R. and Routhier, S.A.: Packet trains-measurements and a new model for computer network traffic. IEEE JSAC **4**(1986) pp 986-995.
[15] K.Y. Whang, B.T. Vander-zanden, and H.M. Taylor. A linear-time probabilistic counting algorithm for database applications. ACM Transaction on Database Systems, 15(2),pp 208-229, June 1990.
[16] A. Kumar, M. Sung, J. Xu, and J.Wang. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In Proc. ACM SIGMETRICS, pp 177-188,2004
[17] Cypress Semiconductor Corporation, http://www.cypress.com/, 2010,Oct.
[18] WIDE, http://tracer.csl.sony.co.jp/mawi/samplepoint-F/20090330/200903300000.html, 2010,Oct.
[19] JSLAB, http://ntds.njnet.edu.cn/data/index.php,2010,Oct.
[20] NLANR, ftp://wits.cs.waikato.ac.nz/pma/long/ipls/3/,2010,Oct.