Computer Networks 56 (2012) 2773–2787

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Virtual indexing based methods for estimating node connection degrees

Pinghui Wang[a], Xiaohong Guan[a,b], Don Towsley[c], Jing Tao[a,*]

[a] MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China
[b] Department of Automation and NLIST Lab, Tsinghua University, Beijing 100083, China
[c] Department of Computer Science, University of Massachusetts, Amherst, MA 01003, United States

ARTICLE INFO

Article history:
Received 22 September 2011
Received in revised form 5 February 2012
Accepted 28 March 2012
Available online 4 April 2012

Keywords:
Data streaming
Traffic monitoring
Super host detection

ABSTRACT

It is difficult to accurately measure node connection degrees for a high speed network, since there is a massive amount of traffic to be processed. In this paper, we present a new virtual indexing method for estimating node connection degrees for high speed links. It is based on the virtual connection degree sketch (*VCDS*) where a compact sketch of network traffic is built by generating multiple virtual bitmaps for each network node. Each virtual bitmap consists of a fixed number of bits selected randomly from a shared bit array by a new method for recording the traffic flows of the corresponding node. The shared bit array is efficiently utilized by all nodes since every bit is shared by the virtual bitmaps of multiple nodes. To reduce the "noise" contaminated in a node's virtual bitmaps due to sharing, we propose a new method to generate the "filtered" bitmap used to estimate node connection degree. Furthermore, we apply *VCDS* to detect super nodes often associated with traffic anomalies. Since *VCDS* need a large amount of extra memory to store node addresses, we also propose a new data structure, the reversible virtual connection degree sketch, which identifies super node addresses analytically without the need of extra memory space but at a small increase in estimation error. Furthermore we combine the *VCDS* and *RVCDS* based methods with a uniform flow sampling technique to reduce memory complexities. Experiments are performed based on the actual network traffic and testing results show that the new methods are more memory efficient and more accurate than existing methods.

1. Introduction

In-degree/out-degree, defined as the number of distinct sources/destinations that a network node (e.g. router, host, or server) connects to, is one of most important traffic metrics for extracting network traffic characteristics and can provide insights into many network measurement and monitoring applications, such as firewalls and intrusion detection devices [1,2]. Identifying super nodes defined as nodes with out-degrees or in-degrees larger than a predefined threshold, is very useful for detecting network anomalies in high speed links [3]. For example, detecting nodes with large out-degrees is used to detect port scans and worm propagations, since a node launching a port scan or infected by worm generally exhibits a large number of connections to distinct destinations. These nodes are clearly considered as super sources. Since distributed denial of service (DDoS) attacks are generated by a large number of nodes flooding packets to a destination, the problem of detecting DDoS attacks can be viewed as that of identifying super destinations.

The node connection degree is generally measured in terms of flows, which refer to the sets of all packets with the same source and destination addresses. To obtain the total number of flows generated by a node, one needs to build a hash table that keeps track of existing flows to avoid duplicating flow records for packets from the same flow. A large number of flows must be processed and

* Corresponding author. Tel.: +86 029 8267 3443; fax: +86 029 8266 4603.
E-mail address: jtao@sei.xjtu.edu.cn (J. Tao).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.03.025

stored in order to detect super nodes in a large network, especially if they are conducting stealthy slow attacks where a large time window is needed. Clearly it is not practical to obtain node connection degrees by building per-node hash tables since there would be too many active nodes and flows. In addition, applications measuring node connection degrees in realtime have to be implemented with very fast and expensive SRAM, since DRAM may not support the required high access rates. Therefore, it is important to make node connection measurement and super node detection memory efficient.

Flow sampling provides a possibly efficient way to analyze and process massive packet streams [4,5], since only a relatively small subset of network traffic needs to be stored and processed. Recently, data streaming methods have been widely applied to approximately measure statistics of network traffic for high speed links, such as the total number of active flows [6], per-flow traffic [7], heavy hitter [8–10], heavy change [11–13], flow size distribution [14,15], super nodes [16–18], and traffic entropy estimation [19,20]. To reduce data dimensionality, these methods build a probabilistic summary of network traffic using a small and fast memory. Each and every incoming packet is processed in real-time using a few operations. Estan and Varghese [6] proposed a family of bitmap algorithms for estimating the total number of distinct flows on high speed links.

To measure node connection degrees, Yoon et al. [21] build a virtual bitmap for each node by taking bits randomly from a shared bit array using a group of hash functions. To detect super nodes, Li et al. [18] proposed a more memory efficient method combined the method in [21] and a uniform flow sampling technique. However, there are still several unresolved issues. First, each bit in the shared bit array is shared by all of the nodes, so that the bits in a node's virtual bitmap may not be set by the flows of this node but others. This introduces "noise" into a node's virtual bitmap that affects estimation of its connection degree. In particular, the virtual bitmap for a node with a small number of connections is sensitive to this "noise", since it contains more bits contaminated by other nodes. Second, since the number of distinct bits in the virtual bitmap selected by the methods in [18,21] varies due to hash collisions, there is no guarantee on the quality of the connection degree estimate with many hash collisions. Third, a large amount of memory is needed by the methods in [18,21] to store node addresses being monitored, and the reversing procedures developed in [12,9,17] for identifying super nodes or heavy hitters cannot be directly applied to the data structures in [21,18]. Fourth, the accuracy analysis for node connection degree estimators in [18,21] is based on the assumption that each flow independently and uniformly hashes into the shared bit array. However our analysis in Section 3 shows that a node's flows do not hash into the shared bit array independently. Aiming to address the above challenges, we develop the following methods in this paper.

(1) We present a new virtual indexing method to accurately estimate node connection degrees over high speed links. We develop a very compact sketch of network traffic called, *virtual connection degree sketch* (*VCDS*) is developed and applied to estimate the connection degree of each node based on its multiple virtual bitmaps. Each virtual bitmap consists of a fixed number of bits selected randomly from a small shared bit array using a new virtual bitmap generation method. We present an algorithm to correct for the "mutual affection or coupling" in the shared bit array by generating the "filtered" bitmap calculated based on the node's multiple virtual bitmaps.
(2) We develop two *VCDS* based super node detection algorithms. The first algorithm directly uses a *VCDS* to build a data digest of node connection degrees. The node address hash table for storing addresses of observed nodes is needed since a *VCDS* alone cannot identify super nodes. The second algorithm is based on a new data structure, *reversible virtual connection degree sketch* (*RVCDS*). It uses a group of reversible hash functions developed in our previous work [22] based on the Chinese remainder theorem in number theory to obtain the super node addresses. Although a *RVCDS* does not preserve any node address information, we can analytically reconstruct super node addresses based on properties of the group of reversible hash functions. Compared to the first method, which stores all node addresses, *RVCDS* is more memory efficient. However the first method finds super nodes from node addresses that appear in the monitored traffic, whereas *RVCDS* identifies super nodes from the entire node address space and may report more false super nodes. Both algorithms are highly efficient since their update/query complexities are low, and our experimental results show that they have almost the same accuracy when a small number of super nodes exist. Furthermore we combine *VCDS* and *RVCDS* with a uniform flow sampling technique to reduce memory complexity.

This paper is organized as follows. The data streaming model is introduced and the problem is formulated in Section 2. In Section 3 the new data streaming method *VCDS* is

Table 1
Table of notations.

F	Number of flows
$\boldsymbol{S}$	Source space
N	Size of source space $\boldsymbol{S}$
n	Number of appeared sources
O_s	Out-degree of node s
ϕ	Threshold for super node detection
H	Number of virtual bitmaps that a node has
$\{h_1,\ldots,h_R\}$	A group of reversible hash functions
A	$1 \times m$ Bit array
m	Size of bit array A
$\mathbb{F}_i(s)$	$\mathbb{F}_i(s) = \{f_{i,0}(s),\ldots,f_{i,L-1}(s)\}$
$B_i(s)$	$B_i(s) = (A[f_{i,0}(s)],\ldots,A[f_{i,L-1}(s)])$
L	Number of bits in a virtual bitmap
a, b, m_i, p	Parameters of hash function h_i
M	$M = m_1 \cdots m_R$
τ	Flow sampling rate

described in details. Section 4 presents several methods to detect super nodes. The performance evaluation and testing results are presented in Section 5. Section 6 summarizes related work. Concluding remarks then follow. A list of notations used is shown in Table 1.

2. Data streaming model

Let $\mathbb{P} = p_1, p_2, \ldots$ be an input packet stream arriving sequentially at a network monitor. Here $p_t = (s_t, d_t)$ represents the tth packet, where $s_t \in \boldsymbol{S}$ and $d_t \in \boldsymbol{D}$ are its source and destination respectively, and $\boldsymbol{S}$ and $\boldsymbol{D}$ are the source and destination spaces. The out-degree of node s is defined as $O_s = |D(s)|$, the number of elements in set $D(s)$, where $D(s) = \{d | (s,d) \in \mathbb{P}, d \in \boldsymbol{D}\}$ is the set of destinations associated with s. The in-degree of node $d \in \boldsymbol{D}$ is defined as $I_d = |S(d)|$, where $S(d) = \{s | (s,d) \in \mathbb{P}, s \in \boldsymbol{S}\}$ is the set of sources associated with d. We define the sets of super sources/destinations as nodes whose out-degrees/in-degrees are larger than ϕF, where $0 < \phi < 1$ is a threshold, and F is the total number of flows. The algorithms presented in the following sections are used to measure node out-degrees and detect super sources, and can also be applied to measure node in-degrees and detect super destinations.

3. Virtual connection degree sketch

In this section, we present a new data structure *VCDS* to build a very compact sketch of network traffic, which is used to estimate the connection degree of each node based on the associated virtual bitmaps. Each virtual bitmap consists of a fixed number of bits selected randomly from a shared bit array by a virtual bitmap generation method to be described below. The shared bit array is efficiently utilized by all nodes since every bit is shared by the virtual bitmaps of multiple nodes. To reduce the "noise" contaminating in each node's virtual bitmaps because of sharing, a new method is proposed to generate the "filtered" bitmap used to estimate the node connection degree.

3.1. Data structure

As shown in Fig. 1, a *VCDS* consists of a bit array $A[k]$ $(0 \leqslant k \leqslant m-1)$ associated with H independent groups of hash functions $\mathbb{F}_i(s) = \{f_{i,0}(s), \ldots, f_{i,L-1}(s)\}$ $(1 \leqslant i \leqslant H)$, each containing L hash functions that map the source space $\{0, \ldots, N-1\}$ to $\{0, \ldots, m-1\}$. Here N is the size of source space $\boldsymbol{S}$.

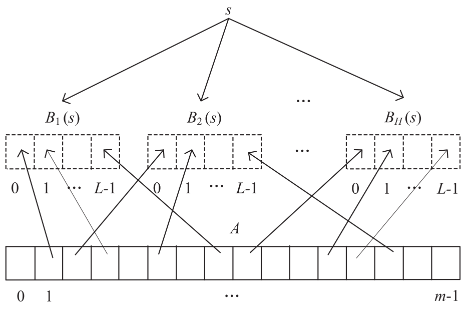


Fig. 1. Architecture of *VCDS*.

Each source s has H corresponding virtual bitmaps $B_i(s)$ $(1 \leqslant i \leqslant H)$ where $B_i(s)$ is defined as a bit array consisting of L bits selected randomly from A by the group of hash functions $\mathbb{F}_i(s)$, that is

$$B_i(s) = (A[f_{i,0}(s)], \ldots, A[f_{i,L-1}(s)]).$$

$B_i(s)$ can be viewed as a direct bitmap as proposed in [23] occupied only by source s. The length of the direct bitmap is constant and is a critical parameter in estimating the out-degree of s. However the number of distinct bits in $B_i(s)$ is smaller than L, since $A[f_{i,0}(s)], \ldots, A[f_{i,L-1}(s)]$ are selected from A and can suffer hash collisions. There is no guarantee on the quality of the estimate of the out-degree of a node whose virtual bitmap is generated with many collisions. For example, when $m = 10^6$ and $L = 10^4$, the number of distinct bits selected from A by the group of hash functions $\mathbb{F}_i(s)$ is smaller than L with a probability of $1 - 10^{-22}$ and its expectation is 9950. This problem also exists but is not noticed in [21,18]. To address this issue, we propose a virtual bitmap generating method by designing $f_{i,j}(s)$ $(1 \leqslant i \leqslant H,\ 0 \leqslant j \leqslant L-1)$ based on the double hashing scheme [24]

$$f_{i,j}(s) = \psi_{i,1}(s) + j\psi_{i,2}(s) \mod m,$$

where $\psi_{i,1}$ is a hash function that maps the source space uniformly to the range $\{0, 1, \ldots, m-1\}$, $\psi_{i,2}$ is a hash function that maps the source space uniformly to the range $\{1, 2, \ldots, m-1\}$, and m is a prime. It is easily validated that each $f_{i,j}$ also maps the source space uniformly to the range $\{0, 1, \ldots, m-1\}$. The following theorem shows that each virtual bitmap is hashed into L different bits in A.

Theorem 1. *For a source s, L different bits are selected from A by each group of hash functions $\mathbb{F}_i(s)$ $(1 \leqslant i \leqslant H)$, that is,*

$$f_{i,j_1}(s) \neq f_{i,j_2}(s), \quad 1 \leqslant j_1 < j_2 \leqslant L-1.$$

For any two distinct bits in A, the probability that they are selected by $B_i(s)$ satisfies the following theorem.

Theorem 2. *There are $m(m-1)$ distinct virtual bitmaps. For each virtual bitmap $B_i(s)$ $(1 \leqslant i \leqslant H,\ s \in S)$, we have the following probabilities for any distinct elements of A, $A[k_1]$ and $A[k_2]$ $(0 \leqslant k_1, k_2 \leqslant m-1$, and $k_1 \neq k_2)$.*

$$(1)\ P(k_1 \in \mathbb{F}_i(s)) = \frac{L}{m},$$

$$(2)\ P(k_1 \in \mathbb{F}_i(s) \text{ and } k_2 \in \mathbb{F}_i(s)) = \frac{L(L-1)}{m(m-1)},$$

$$(3)\ P(k_1 \in \mathbb{F}_i(s) \text{ and } k_2 \notin \mathbb{F}_i(s)) = \frac{L(m-L)}{m(m-1)},$$

$$(4)\ P(k_1 \notin \mathbb{F}_i(s) \text{ and } k_2 \notin \mathbb{F}_i(s)) = 1 - \frac{2L}{m} + \frac{L(L-1)}{m(m-1)}.$$

The proofs are given in Appendix.

3.2. Update procedure

Each bit in A is initially set to zero. When a packet $p = (s,d)$ arrives, we set the $g(d)$th bit in each virtual bitmap $B_i(s)$ $(1 \leqslant i \leqslant H)$ to one. Here g is a uniform hash function with range $\{0,1,\ldots,L-1\}$. As the $g(d)$th position in $B_i(s)$, corresponds to $A[f_{i,g(d)}(s)]$, we only need to set H bits for each incoming packet as follows:

$$A[f_{i,g(d)}(s)] = 1, \qquad 1 \leqslant i \leqslant H.$$

3.3. Connection degree estimator

The bits in $B_i(s)$ $(1 \leqslant i \leqslant H)$ that the flows of source s hash into using hash functions $\mathbb{F}_i(s)$ are denoted as the bits used by s in the following part. They are set to one to store the flow information of s, so each $B_i(s)$ can be used to estimate the out-degree of s similar to the direct bitmap proposed in [23]. Since each bit in $B_i(s)$ is selected randomly from A and also shared by other sources, the other bits in $B_i(s)$ not used by s might also be set to one by flows belonging to other sources. This introduces "noise" into the estimation of the out-degree of s. Therefore the more bits in $B_i(s)$ are not used by s, the more "noise" is generated. The size of the virtual bitmap, L, is usually set to several thousands to guarantee a high accuracy in estimating the out-degree of a source associated with a huge number of flows. It will generate a large number of bits containing "noise" especially for the source associated with a small number of flows. In what follows, we introduce a new "filtered" bitmap generation method to reduce the "noise" generated by other sources. The "filtered" bitmap B_s defined as a bit vector is computed from $B_1(s)$, $B_2(s)$, $\ldots$, and $B_H(s)$ as follows:

$$B_s = B_1(s) \otimes B_2(s) \otimes \ldots \otimes B_H(s),$$

where $\otimes$ is the bit-AND operation. For any flow (s,d) of s, the $g(d)$th bit in each $B_i(s)$ $(1 \leqslant i \leqslant H)$ is set to one, so the $g(d)$th bit in B_s is still one.

For any bit x in the set of bits in A excluding bits in $B_i(s)$ used by source s, we define α_s to be the probability that x is not set to one. For each virtual bitmap $B_i(s)$ $(1 \leqslant i \leqslant H)$ of source s, the probability that the other $nH-1$ virtual bitmaps generated by the n sources that appear in the monitored network all differ from $B_i(s)$ is

$$P_{dif} = \left(1 - \frac{1}{m(m-1)}\right)^{nH-1} > 1 - \frac{nH-1}{m(m-1)}. \qquad (1)$$

Note that P_{dif} is very close to 1. For example, when $m > n$, $m = 10^6$, and $H = 1$, $P_{dif} > 1 - 10^{-6}$. For simplicity, the following analysis assumes that these $nH-1$ virtual bitmaps all differ from $B_i(s)$. Let $\varphi(j)$ be the probability that x is not set to one by updates associated with $B_l(j)$ $(1 \leqslant l \leqslant H)$ that is a virtual bitmap of any source j and differs from $B_i(s)$. Based on Theorem 2, $B_l(j)$ contains x with probability L/m. When $B_l(j)$ contains x, x is set to one by updates associated with $B_l(j)$ with probability $1-(1-\frac{1}{L})^{O_j}$. Thus, we have

$$\varphi(j) = \frac{m-L}{m} + \frac{L}{m}\left(1-\frac{1}{L}\right)^{O_j}.$$

Therefore $\alpha_s = \frac{\Phi^H}{\varphi(s)}$, where $\Phi = \prod_{j\in \mathbf{S}} \varphi(j)$. For any given bit in filtered bitmap B_s not used by s, it is zero if and only if at least one of its associated bits in $B_i(s)$ $(1 \leqslant i \leqslant H)$ is not set to one, which occurs with probability $p_s = 1-(1-\alpha_s)^H$. Therefore it is a noise bit when all corresponding bits in $B_i(s)$ $(1 \leqslant i \leqslant H)$ are ones with probability

$$q_s = (1-\alpha_s)^H.$$

Let m and L are given. We want to optimize H to minimize the "noise" contaminated in the "filtered" bitmap B_s. There are two competing forces: for each bit in B_s not used by s, using a larger H gives us a greater chance of finding a zero bit in its H corresponding bits in $B_i(s)$; but using more bitmaps results in more bits in A being set to one, which increases the probability that a node's virtual bitmap is contaminated with "noise". When $m \gg L$, we have $q_s \approx \lambda(H) = (1-\Phi^H)^H$. The impact of H on the "noise" is described in the following Theorem.

Theorem 3. *$\lambda(H) = (1-\Phi^H)^H$ decreases with $H \in \left(0, -\frac{1}{\ln \Phi}\right)$, increases with $H \in \left(-\frac{1}{\ln \Phi}, +\infty\right)$, and obtains the minimum at $H = -\frac{1}{\ln \Phi}$.*

The proof is given in Appendix. In what follows a new method is proposed to estimate the out-degree of source s. For any given bit in filtered bitmap B_s not used by s, it is zero if and only if at least one of its associated bits in $B_i(s)$ $(1 \leqslant i \leqslant H)$ is not set to one, which occurs with probability $p_s = 1-(1-\alpha_s)^H$. Denote the number of bits in B_s not used by s as U_s, then the expectation of U_{B_s}, the number of zero bits in B_s is

$$E(U_{B_s}) = E(U_s p_s) = p_s E(U_s). \qquad (2)$$

Since the probability that a given bit in B_s is not used by source s is $(1-\frac{1}{L})^{O_s}$, the expectation of U_s is

$$E(U_s) = L\left(1-\frac{1}{L}\right)^{O_s}. \qquad (3)$$

Let $\overline{\mathbb{B}}_i(s) = \left\{y_i^{(k)}\right\}$ $(0 \leqslant k \leqslant m-L-1)$ be the set of bits in A excluding all L bits in $B_i(s)$. The probability that any bit $y_i^{(k)}$ is not set to one is α_s, so the expectation of $U'_{B_i(s)}$, the number of zero bits in $\overline{\mathbb{B}}_i(s)$ is

$$E(U'_{B_i(s)}) = (m-L)\alpha_s.$$

Since $p_s = 1-(1-\alpha_s)^H$, we have

$$p_s = 1 - \left(1 - \frac{E(U'_{B_i(s)})}{m-L}\right)^H. \qquad (4)$$

From (2)–(4), we have

$$O_s = \frac{\ln\frac{E(U_{B_s})}{L} - \ln\left(1-\left(1-\frac{E\left(U'_{B_i(s)}\right)}{m-L}\right)^H\right)}{\ln\left(1-\frac{1}{L}\right)}. \qquad (5)$$

Replacing $E(U_{B_s})$ and $E(U'_{B_i(s)})$ in (5) by the instantaneous values, U_{B_s} and $U'_{B_i(s)}$, we have the following estimators $O_s^{(i)}$ $(1 \leqslant i \leqslant H)$:

$$O_s^{(i)} = -L\ln\frac{U_{B_s}}{L} + L\ln\left(1-\left(1-\frac{U-U_{B_i(s)}}{m-L}\right)^H\right).$$

Finally we estimate the out-degree of source s as

$$O_s^{est} = median_{1\leqslant i\leqslant H} O_s^{(i)}.$$

3.4. Parameter configuration

Since the total complexity for updating each packet is $O(H)$, H should be small. The space complexity of *VCDS* is $O(m)$. If the out-degree of source s is much larger than $L\ln L + O(L)$, we will obtain all ones in its corresponding virtual bitmaps with high probability due to the result of the "coupon collector's problem" [25]. In this case, the out-degree of source s cannot be estimated accurately, and we only know that it is not smaller than $L\ln L$. To address this issue, we can directly use a larger L or use a sampled *VCDS* method presented in Section 4.3, which combines a flow sampling technique and *VCDS*.

4. Super node detection

In this section we first present a super node detection method which directly uses *VCDS* to build a data digest of node connection degrees, and a node address hash table to store addresses of observed nodes. The node address hash table is needed since *VCDS* cannot be used directly to identify super nodes. That is, for a given node, we can test whether or not it is a super node based on testing its connection degree estimated by *VCDS* is larger than the threshold. However, it is not possible to identify super nodes only by inspecting *VCDS* without storing links to node addresses. We then present a new data structure *RVCDS* to detect super nodes. It uses a group of reversible hash functions that takes advantage of the remainder characteristics from number theory to obtain the super node addresses. Although *RVCDS* does not preserve any node address information, we can analytically reconstruct super node addresses purely based on the characteristics of the group of reversible hash functions. Moreover the *VCDS* and *RVCDS* based super node detection methods are extended and incorporated with a uniform flow sampling technique to reduce memory.

4.1. VCDS based method

A block diagram of the *VCDS* based super node detection method is shown in Fig. 2. The F estimation module is used to estimate F, the total number of flows using the method

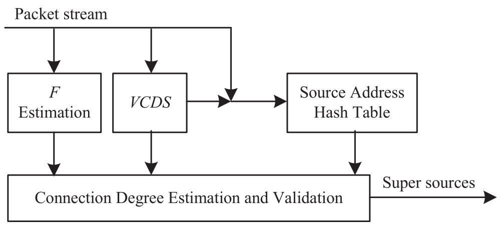

Packet stream
F Estimation
VCDS
Source Address Hash Table
Connection Degree Estimation and Validation
Super sources

Fig. 2. *VCDS* based super source detection method.

reported in [6]. The *VCDS* module encodes out-degree information for every source into a very compact data structure, and the source address hash table aims to store all source addresses that appear in the monitored network. Finally each collected source address is tested to determine whether or not it is a super source based on its estimated out-degree as calculated by *VCDS*. These modules are described in the following subsections.

4.1.1. Update procedure

We use a hash table to store all source addresses similar to the method proposed in [21]. For each packet $p = (s,d)$, s is inserted into the hash table if and only if at least one associated bit $A[f_{i,g(d)}(s)]$ $(1 \leqslant i \leqslant H)$ is set from zero to one. Note that the source of a flow is inserted into the hash table at most once since all of its packets hash into the same positions in A, and once they are set to one, the source address is never updated in the hash table. Hence it does not cost much to collect source addresses.

4.1.2. Super node detection

Finally each node in the source address hash table is tested to determine whether or not it is a super source based on its estimated out-degree as calculated by *VCDS*. If its estimated out-degree is larger than $\phi\widehat{F}$, where $\widehat{F}$ is an estimate of F obtained from the F estimation module, it is reported as a super source. For each coming packet, we determine whether or not its associated flow has appeared by examining the flow's H associated bits in A, which can be viewed as a Bloom filter [26]. The source address of a flow might not be collected in the source address hash table since its first packet might be wrongly identified as a packet associated with an appeared flow. Thus, there is a chance that we miss a super source when all of its flows are wrongly determined.

4.1.3. Parameter configuration

For sources with small out-degrees, Theorem 3 shows that the accuracies of their estimates are greatly improved by using a small value of H slightly larger than one. However there is less "noise" in the virtual bitmaps of a super source, and we simply set H to one, so as to make the processing of each packet more efficient. The space complexity of the *VCDS* based method consists of the space for the source address hash table and bit array A. The space complexity of the source address hash table is $O(n)$, where n is the number of source addresses that appear in the traffic stream. The space complexity of *VCDS* is $O(m)$.

4.2. RVCDS based method

The *VCDS* based super source detection system requires a large source address hash table particularly for a large network with a large number of existing sources. In what follows we propose a new data structure, *RVCDS*, which identifies super sources analytically using a group of reversible hash functions. It requires no extra memory space to store the source addresses that appear in the monitored network. A block diagram of the system is shown in Fig. 3. The F estimation module is the same as that

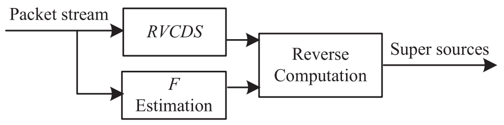


Fig. 3. *RVCDS* based super source detection method.

presented in Fig. 2, which is used to estimate *F*, the total number of flows. The other two modules are described later in this section.

4.2.1. RVCDS

A *RVCDS* consists of a bit array $A[k]$ $(0 \leqslant k \leqslant m-1)$ associated with a group of reversible hash functions $\{h_1,\ldots,h_R\}$ and H independent groups of hash functions $\mathbb{F}_i(s)$ $(1 \leqslant i \leqslant H)$. Here $\mathbb{F}_i(s)$ is the same as those introduced in Section 3. h_i maps the source space uniformly to the range $D_i = \{0,1,\ldots,m_i\}$ $(1 \leqslant i \leqslant R)$. As shown in Fig. 4, each source s has two types of virtual bitmaps: $B_i(h_i(s))$ $(1 \leqslant i \leqslant R)$ and $B_i(s)$ $(R+1 \leqslant i \leqslant H)$, where $B_i(s)$ $(R+1 \leqslant i \leqslant H)$ is the same as that introduced in Section 3. To generate virtual bitmap $B_i(h_i(s))$ $(1 \leqslant i \leqslant R)$, we first calculate a reverse index number $h_i(s)$ for source s, which is used in the reverse computation operation discussed later. $h_i(s)$ is then used as input to the corresponding group of hash functions $\mathbb{F}_i(s)$ used to generate the virtual bitmap $B_i(h_i(s))$, that is,

$$B_i(h_i(s)) = (A[f_{i,0}(h_i(s))],\ldots,A[f_{i,L-1}(h_i(s))]).$$

h_i $(1 \leqslant i \leqslant R)$ used in *RVCDS* is selected randomly from a class of 2-universal hash functions $\mathbb{H}_{m_i} = \{h_{a,b,m_i}(x)\}$ defined below. $h_{a,b,m_i}(x)$ is defined as

$$h_{a,b,m_i}(x) = g_{a,b,p}(x) \mod m_i, \qquad 1 \leqslant i \leqslant R \tag{6}$$

where $g_{a,b,p}(x) = ax + b \mod p$, $a \in \{1,2,\ldots,p-1\}$ and $b \in \{0,1,\ldots,p-1\}$, with p selected as a prime larger than the source space size N. This construction ensures that sources hash uniformly into different reverse indexes. The proof can be found in [24]. m_1, …, m_R are selected as pair-wise co-prime integers to make *RVCDS* reversible as will be discussed later.

4.2.2. Update procedure

For each incoming packet (s,d), we set the $g(d)$th bit in $B_i(h_i(s))$ $(1 \leqslant i \leqslant R)$ and $B_i(s)$ $(R+1 \leqslant i \leqslant H)$ to one. As the $g(d)$th position in $B_i(s)$ $(R+1 \leqslant i \leqslant H)$ corresponds to $A[f_{i,g(d)}(s)]$, and the $g(d)$th position in $B_i(h_i(s))$ $(1 \leqslant i \leqslant R)$ corresponds to $A[f_{i,g(d)}(h_i(s))]$, we only need to set the following H bits (similar to *VCDS*):

$$A[f_{i,g(d)}(h_i(s))] = 1, \qquad 1 \leqslant i \leqslant R;$$
$$A[f_{i,g(d)}(s)] = 1, \qquad R+1 \leqslant i \leqslant H.$$

4.2.3. Connection degree estimator

The "filtered" bitmap B_s for each source s is a bit vector computed from $B_1(h_1(s)),\ldots,B_R(h_R(s)), B_{R+1}(s),\ldots$, and $B_H(s)$ as

$$B_s = B_1(h_1(s)) \otimes \ldots \otimes B_R(h_R(s)) \otimes B_{R+1}(s) \otimes \ldots \otimes B_H(s).$$

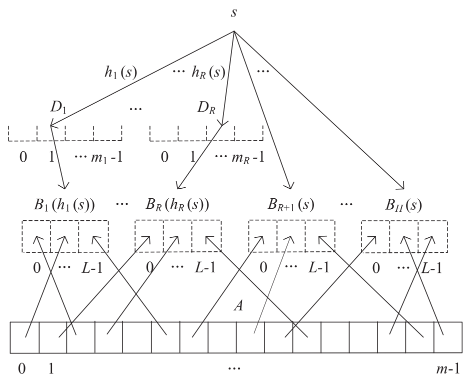


Fig. 4. Architecture of *RVCDS*.

For any bit x in $B_i(h_i(s))$ not used by source s, we define $\varphi(j)$ to be the probability that x is not set to one by updates associated with virtual bitmap $B_i(h_i(j))$ $(j \neq s)$ that is different from $B_i(h_i(s))$. Based on Theorem 2, we have that $B_i(h_i(j))$ contains x with probability L/m. When $B_i(h_i(j))$ contains x, x is set to one by flows of source j through $B_i(h_i(j))$ with probability $1-(1-\frac{1}{L})^{O_j}$. Thus, we have

$$\varphi(j) = \frac{m-L}{m} + \frac{L}{m}\left(1-\frac{1}{L}\right)^{O_j}. \tag{7}$$

The case $h_i(j) = h_i(s)$ occurs with probability $\frac{1}{m_i}$, and $B_i(h_i(j))$ is the same as $B_i(h_i(s))$. If this is not the case, $B_i(h_i(j))$ is the same as $B_i(h_i(s))$ with probability $\frac{1}{m(m-1)}$, since each virtual bitmap is selected uniformly and independently from a total of $m(m-1)$ distinct virtual bitmaps. Therefore, $B_i(h_i(j))$ is the same as $B_i(h_i(s))$ with probability $\frac{1}{m_i} + \frac{1}{m(m-1)}\left(1-\frac{1}{m_i}\right) \approx \frac{1}{m_i}$. When $B_i(h_i(j))$ is the same as $B_i(h_i(s))$, no flow of j sets x to one with probability $(1-\frac{1}{L})^{O_j}$. Define $\theta_i(j)$ to be the probability that x is not set to one by updates associated with $B_i(h_i(j))$. From (7), we have

$$\theta_i(j) = \left(1-\frac{1}{m_i}\right)\varphi(j) + \frac{1}{m_i}\left(1-\frac{1}{L}\right)^{O_j}. \tag{8}$$

We see that $\theta_i(j)$ decreases with $\frac{1}{m_i}$. When m_i goes to infinity, all existing sources are hashed by h_i into different reverse index numbers without collisions, $\theta_i(j) = \varphi(j)$, and *RVCDS* is the same as *VCDS*. Therefore *RVCDS* is less accurate than *VCDS* for estimating node connection degrees. From (8), we have $\left|1-\frac{\theta_i(j)}{\varphi(j)}\right| \leqslant \frac{1}{m_i}$, so $\theta_i(j) \approx \varphi(j)$ when m_i is reasonable large. The estimate for the out-degree of source s is similar to that of *VCDS*

$$O_s^{est} = median_{1 \leqslant i \leqslant H} O_s^{(i)},$$

where $O_s^{(i)}$ $(1 \leqslant i \leqslant R)$ is

$$O_s^{(i)} = -L\ln\frac{U_{B_s}}{L} + L\ln\left(1-\left(1-\frac{U-U_{B_i(h_i(s))}}{m-L}\right)^H\right)$$

and $O_s^{(i)}$ $(R < i \leqslant H)$ is

$$O_s^{(i)} = -L\ln\frac{U_{B_s}}{L} + L\ln\left(1-\left(1-\frac{U-U_{B_i(s)}}{m-L}\right)^H\right)$$

where U is the number of zero bits in A, U_{B_s} is the number of zero bits in B_s, $U_{B_i(s)}$ $(1 \leqslant i \leqslant R)$ is the number of zero bits in $B_i(s)$, and $U_{B_i(h_i(s))}$ $(R < i \leqslant H)$ is the number of zero bits in $B_i(h_i(s))$.

4.2.4. Reverse computation

As shown in Fig. 5, we first calculate super source candidates using the group of reversible hash functions $\{h_1,\ldots,h_R\}$. To reduce false super sources, each super source candidate is then validated by its estimated out-degree as calculated by *RVCDS*.

For a source s, denote the probability that a given bit in the set of bits in A excluding the bits in $B_i(h_i(s))$ used by s is not set to one as γ_i. Let U_s be the number of bits in $B_i(h_i(s))$ not used by s, and $U_{B_i(h_i(s))}$ be the number of zero bits in $B_i(h_i(s))$. Then the expectation of $U_{B_i(h_i(s))}$ is

$$E(U_{B_i(h_i(s))}) = \gamma_i E(U_s). \tag{9}$$

Let $\overline{\mathbb{B}}_i(h_i(s))$ be the set of bits in A excluding all L bits in $B_i(h_i(s))$, and $U'_{B_i(h_i(s))}$ be the number of zero bits in $\overline{\mathbb{B}}_i(h_i(s))$. Then we have

$$E\left(U'_{B_i(h_i(s))}\right) = (m-L)\gamma_i. \tag{10}$$

Since $E(U_s) = L(1-\frac{1}{L})^{O_s} \approx Le^{-\frac{O_s}{L}}$ and $E(U_{B_i(h_i(s))}) = E(U) - E\left(U'_{B_i(h_i(s))}\right)$, we have the following equation from (9) and (10):

$$E(U_{B_i(h_i(s))}) = \frac{E(U)Le^{-\frac{O_s}{L}}}{m-L+Le^{-\frac{O_s}{L}}}. \tag{11}$$

Replacing $E(U)$ by the instantaneous value U in (11), we have the following equation when $O_s = \frac{\widehat{\phi F}}{2}$, where $\widehat{F}$ is an estimate of F.

$$E(U_{B_i(h_i(s))}) = U_\phi = \frac{ULe^{-\frac{\widehat{\phi F}}{2L}}}{m-L+Le^{-\frac{\widehat{\phi F}}{2L}}}.$$

If source s is a super source whose out-degree O_s is larger than ϕF, then it is highly probable that $U_{B_i(h_i(s))}$ is smaller than the threshold U_ϕ. Based on the above analysis, we first detect heavy index number set $\mathbb{I}_i$ defined as

$$\mathbb{I}_i = \{k | U_{B_i(k)} < U_\phi, 0 \leqslant k \leqslant m_i - 1\}, \quad 1 \leqslant i \leqslant R.$$

Each hash function h_i maps all sources uniformly into the range $\{0,1,\ldots,m_i-1\}$, so super sources can be identified by finding source addresses in the source space that hash into $\mathbb{I}_i$ using function h_i. The super source candidate identification problem is formulated as: With known inputs: Hash functions $\{h_i\}_{i=1}^R$, h_i: $\{0,1,\ldots,N-1\} \to \{0,1,\ldots,m_i-1\}$, find all $x \in \{0,1,\ldots,N-1\}$ such that $h_i(x) \in \mathbb{I}_i$ for all $i \in \{1,2,\ldots,R\}$.

We first consider the simplest scenario. Assume each $\mathbb{I}_i$ contains only one index, denoted as c_i. Based on the hash functions defined in (6), the super source candidate identification problem converts to the problem of simultaneous congruence: find x such that

$$\begin{cases} g_{a,b,p}(x) \equiv c_i \mod m_i, & 1 \leqslant i \leqslant R \\ g_{a,b,p}(x) = ax+b \mod p \\ 0 \leqslant x \leqslant N-1 \end{cases} \tag{12}$$

If we denote $y = g_{a,b,p}(x)$, we have

$$y \equiv c_i \mod m_i, \quad 1 \leqslant i \leqslant R. \tag{13}$$

Based on the Chinese remainder theorem [27], the solutions of the congruencies (13) are easily obtained as

$$y \equiv \sum_{i=1}^{R} M_i M_i^{-1} c_i \mod M \tag{14}$$

where $M = m_1 \cdots m_R$ and $M_i = \frac{M}{m_i}$. M_i^{-1} is determined from $M_i M_i^{-1} \equiv 1 \mod m_i$. Since $y \in \{0,\ldots,p-1\}$, the solutions of (14) are

$$y = kM + \mu, \quad 0 \leqslant k \leqslant K \tag{15}$$

where μ is the unique solution of (13) smaller than M and $K = \lfloor\frac{p-\mu}{M}\rfloor$. Since $y = g_{a,b,p}(x) = ax + b \mod p$, (15) transforms to

$$ax + b = kM + \mu \mod p, \quad 0 \leqslant k \leqslant K. \tag{16}$$

From the Euler–Fermat theorem [27], we know $a^{\chi(p)} \equiv 1 \mod p$, where $\chi(p)$ is the Euler totient function defined as the number of positive integers not larger than p that are co-prime to p. Since p is a prime, $\chi(p) = p - 1$. Therefore (16) transforms to

$$x = a^{p-2}(kM + \mu - b) \mod p, \quad 0 \leqslant k \leqslant K.$$

Thus, the solutions of (15) are obtained as

$$\boldsymbol{X}_2 = \{x | x = a^{p-2}(kM+\mu-b) \mod p, \quad 0 \leqslant k \leqslant K\}$$

and the solutions of (12) are

$$\boldsymbol{X}_1 = \{x | x \in \boldsymbol{X}_2, \quad 0 \leqslant x \leqslant N-1\}.$$

If $M \geqslant N$, there is at most one solution in the source space. Otherwise, the number of solutions is at most $1 + \lfloor p/M \rfloor$. For the general reverse scenario, denote the set of all possible combinations of R heavy index numbers as $\{P_1,\ldots,P_Z\}$, where $Z = |\mathbb{I}_1| \cdots |\mathbb{I}_R|$. For each P_j $(1 \leqslant j \leqslant Z)$ consisting of R heavy index numbers selected separately from $\mathbb{I}_i$ $(1 \leqslant i \leqslant R)$, we obtain solution set S_j based on the method for solving the simplest scenario problem (12). Then the entire super source candidate set is the union of all sets S_j. Suppose two super sources exist, and each $\mathbb{I}_i$ contains two indices, then the number of super source candidates generated is at least 2^R. To reduce false positives, each super source candidate is validated based on its estimated out-degree as calculated by *RVCDS*.

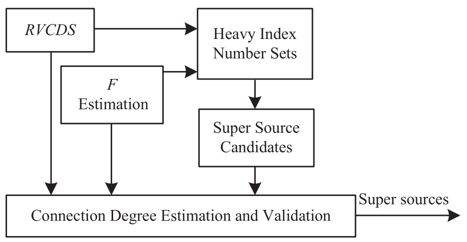


Fig. 5. Architecture of reverse computation.

4.2.5. Parameter configuration

RVCDS only needs to set H bits for each incoming packet so the update complexity is $O(H)$. The computational complexity for detecting heavy index sets $\mathbb{I}_i$ $(1 \leqslant i \leqslant R)$ is $O\left(\sum_{i=1}^{R} m_i\right)$, and 1 + $\lfloor p/M \rfloor$ source addresses need to be calculated for each possible combination of the heavy index numbers, therefore the computational complexity for generating the super sources is $O\left(\sum_{i=1}^{R} m_i + (1 + \lfloor p/M \rfloor)E^R\right)$, where $E = \max\{|\mathbb{I}_1|, \ldots, |\mathbb{I}_R|\}$. Since the computational complexity for generating super sources is exponentially related to the parameter R, a small value of R should be selected. In our experiments we set $R = 2$ and $H = 3$ to make the update complexity low. m_1 and m_2 are set relatively larger than the total number of sources that appear in the monitored network so as to ensure that most sources hash into distinct reverse index numbers without collisions.

4.2.6. Comparison with VCDS

Compared to *VCDS*, *RVCDS* is less accurate for estimating node out-degree degrees. For super source detection, *RVCDS* finds source addresses in the entire source space $\boldsymbol{S}$ that simultaneously hash into all heavy index number sets, but the *VCDS* based method collects source addresses from source addresses that appear in the monitored network, a small subset of $\boldsymbol{S}$. Therefore *RVCDS* may generate more false super sources. However, *RVCDS* is more memory efficient for detecting super sources, since the *VCDS* based method needs a large amount memory to store addresses of observed sources. Experimental results in Section 5 will show that the *RVCDS* and *VCDS* based methods have nearly the same accuracy when a small number of super sources exist.

4.3. Sampled methods

When m, the length of the bit array A is set smaller than the total number of sources that appear in the monitored network, almost all bits in A are set to one. Then *VCDS* and *RVCDS* both fail to estimate node connection degrees. When the number of sources is large, this can result in a large bit array A. In this section we propose two other methods for reducing the size of A. They apply flow sampling to *VCDS* and *RVCDS* similar to the method proposed in [18]. First uniform flow sampling is used to filter most of the nodes that have a very small number of flows. Each incoming packet (s,d) uniformly hashes into a number in the range $[0,Z)$. If the hash result is larger than τZ, the packet is directly discarded, where $0 < \tau < 1$ is the sampling rate. Otherwise it is added to *VCDS*/*RVCDS* as described previously. Finally each flow is sampled independently with probability τ. Since only the sampled flows are added to *VCDS*/*RVCDS*, a smaller bit array A can be used compared to the previous two methods. In what follows we present the estimators of the node out-degree for these sampling methods.

4.3.1. Sampled VCDS

For any bit x in the set of bits in A excluding bits in $B_i(s)$ used by source s, we define $\tilde{\alpha}_s$ to be the probability that x is not set to one. Let $\tilde{\varphi}(j)$ to be the probability that x is not set to one by updates associated with $B_l(j)$ $(1 \leqslant l \leqslant H)$ that is a virtual bitmap of any source j and differs from $B_i(s)$. Based on Theorem 2, we have that $B_l(j)$ contains x with probability L/m. When $B_l(j)$ contains x, x is set to one by updates associated with $B_l(j)$ with probability $1 - \left(1 - \frac{\tau}{L}\right)^{O_j}$. Thus, we have

$$\tilde{\varphi}(j) = \frac{m-L}{m} + \frac{L}{m}\left(1 - \frac{\tau}{L}\right)^{O_j}.$$

Therefore $\tilde{\alpha}_s = \frac{\widetilde{\Phi}^H}{\tilde{\varphi}(s)}$, where $\widetilde{\Phi} = \prod_{j \in \boldsymbol{S}} \tilde{\varphi}(j)$. For any given bit in filtered bitmap B_s not used by s, it is zero if and only if at least one of its associated bits in $B_i(s)$ $(1 \leqslant i \leqslant H)$ is not set to one, which occurs with probability $\tilde{p}_s = 1 - (1 - \tilde{\alpha}_s)^H$. Similar to the analysis in Section 3, we can estimate the out-degree of source s as

$$O_s^{est} = median_{1 \leqslant i \leqslant H} O_s^{(i)},$$

where $O_s^{(i)}$ is defined as

$$O_s^{(i)} = -\frac{L}{\tau}\ln\frac{U_{B_s}}{L} + \frac{L}{\tau}\ln\left(1 - \left(1 - \frac{U - U_{B_i(s)}}{m-L}\right)^H\right).$$

4.3.2. Sampled RVCDS

For *RVCDS* with a uniform flow sampling probability τ, the out-degree of source s is estimated similarly as follows

$$O_s^{est} = median_{1 \leqslant i \leqslant H} O_s^{(i)},$$

where $O_s^{(i)}$ $(1 \leqslant i \leqslant R)$ is defined as

$$O_s^{(i)} = -\frac{L}{\tau}\ln\frac{U_{B_s}}{L} + \frac{L}{\tau}\ln\left(1 - \left(1 - \frac{U - U_{B_i(h_i(s))}}{m-L}\right)^H\right)$$

and $O_s^{(i)}$ $(R < i \leqslant H)$ is defined as

$$O_s^{(i)} = -\frac{L}{\tau}\ln\frac{U_{B_s}}{L} + \frac{L}{\tau}\ln\left(1 - \left(1 - \frac{U - U_{B_i(s)}}{m-L}\right)^H\right).$$

For the reverse computation, the threshold U_ϕ used for detecting the heavy index number set $\mathbb{I}_i$ $(i = 1,\ldots,R)$ is defined as

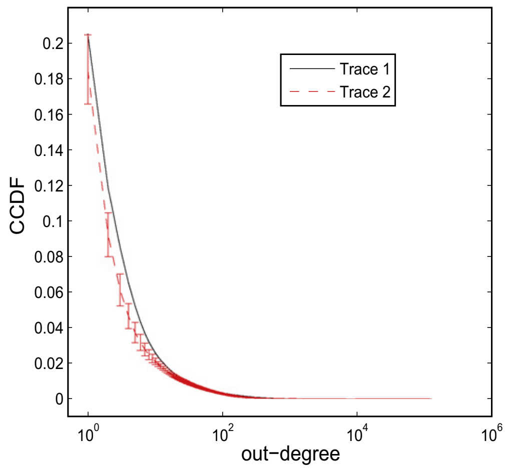


Fig. 6. CCDF of node out-degree distributions.

$$U_\phi = \frac{ULe^{-\frac{\phi\tau\widehat{F}}{2L}}}{m - L + Le^{-\frac{\phi\tau\widehat{F}}{2L}}}.$$

5. Experiments

We use two packet header traces obtained from the CERNET (China Education and Research Network) backbones, gathered at the Northwest Regional Center. Trace 1 is collected at a 10 Gbps link by using TCPDUMP for about ten minutes. It consists of 2.1×10^8 packet headers, and has 2.1×10^5 distinct sources and 8.2×10^5 distinct flows. Trace 2 is collected at an egress router with a bandwidth of 1.5 Gbps for about eleven hours. It consists of 5.8×10^8 packet headers, and has 1.7×10^6 distinct sources and 7.0×10^6 distinct flows. We split trace 2 into 30 min pieces segments, referred to as snapshots, and evaluate the performance of our methods based on trace 1 and each trace 2 snapshot. Fig. 6 shows the complementary cumulative distribution function (CCDF) of node out-degree distribution for trace 1, and the average and variance of the node out-degree distributions for the trace 2 snapshots. In the following experiments, we further compare our methods to the joint method (*JM*) [16], the compact spread estimator (*CSE*) [21], and the efficient scan detection estimator (*ESD*) [18].

5.1. Node out-degree estimation performance

The following experiments are used to evaluate the performance of estimating node out-degrees provided by

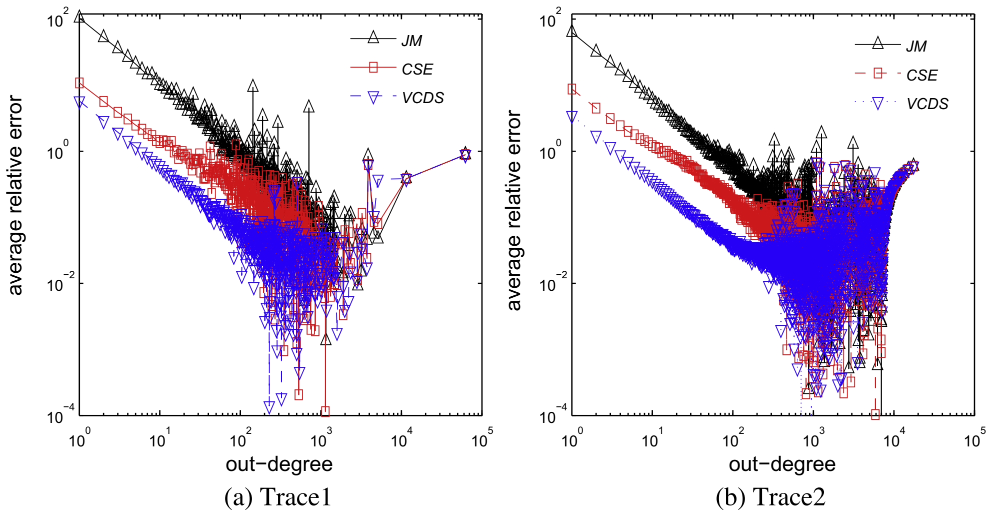


Fig. 7. Compared results with *CSE* and *JM*.

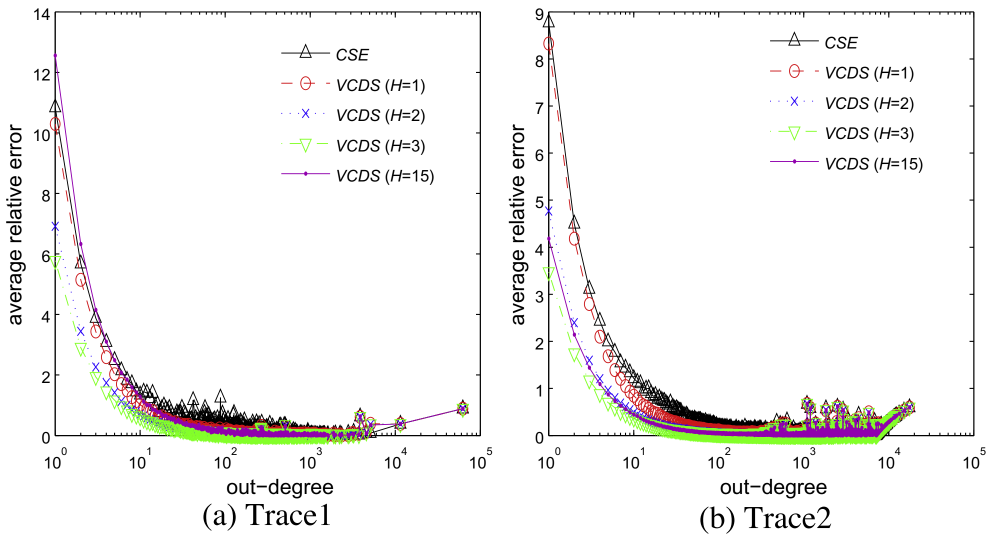


Fig. 8. Average relative error for different *H*.

VCDS. The relative out-degree estimation error of node s is defined as $\frac{|O_s^{est}-O_s|}{O_s}$, where $|O_s^{est}|$ is its estimated out-degree and O_s is its actual out-degree.

Fig. 7 shows the relative errors of node out-degree estimates for *VCDS* with parameters $m = 4 \times 10^6$, $H = 3$ and $L = 1 \times 10^3$ compared to *CSE* and *JM* under the same memory usage. It shows that *VCDS* outperforms *CSE* and *JM*.

Fig. 8 shows the relative errors of node out-degree estimates for different H, where $m = 4 \times 10^6$ and $L = 1 \times 10^3$. The accuracy of *VCDS* first increases with H and then decreases with H, which is consistent with Theorem 3. Based on the comparison of *CSE* and *VCDS* with $H = 1$ under the same space and computational complexities, we observe that the new virtual bitmap generating method is more accurate.

The impact of L is shown in Fig. 9, where $H = 3$ and $m = 4 \times 10^6$. For a node with a small out-degree, the error of its estimated out-degree increases with L, since a larger value of L introduces more unused bits in the node's virtual bitmaps, which might be contaminated. However, a smaller value of L generates a larger estimation error when the out-degree is large similar to the direct bitmap algorithm [23].

Fig. 10 shows the average relative node out-degree estimation errors for different m, where $H = 3$ and $L = 1 \times 10^3$. We observe that the accuracy of estimated out-degrees is improved by increasing m, which reduces the "noise", especially for nodes with small out-degrees. For a node with a small out-degree, the number of unused bits in its virtual bitmap is larger than that of a node with a large

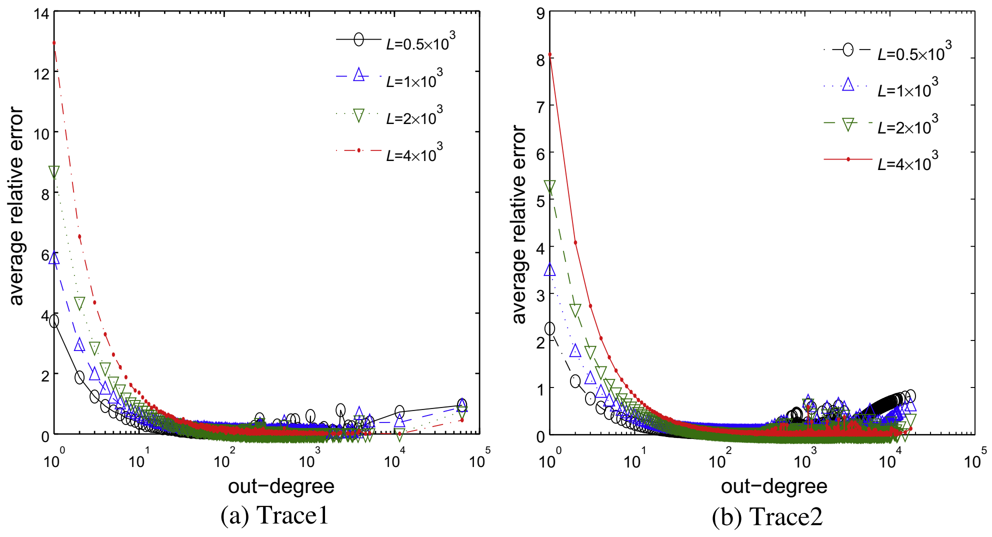


Fig. 9. Average relative error for different L.

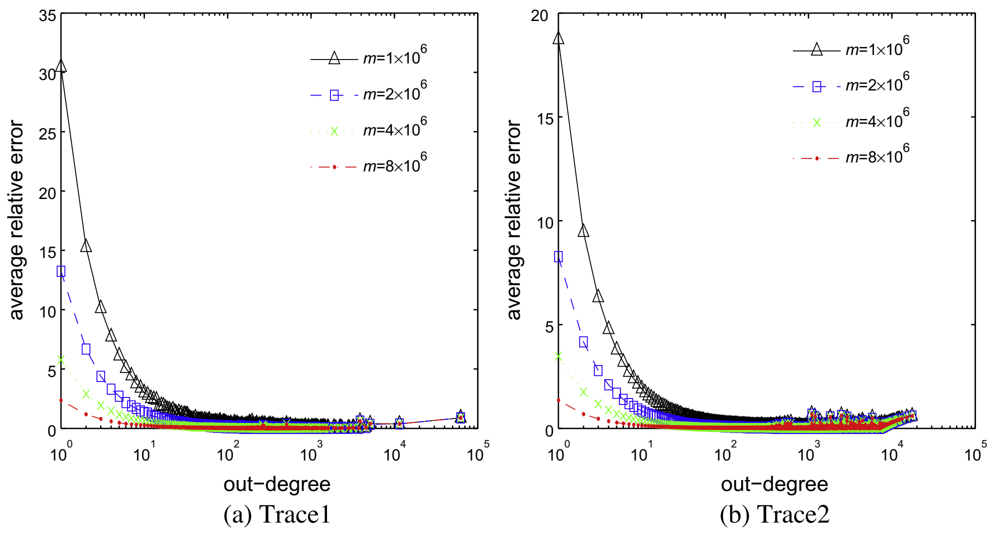


Fig. 10. Average relative error for different m.

Table 2
Number of super sources for different ϕ.

ϕ ($\times 10^{-4}$)	6	8	10	12	14	16
Trace 1	137	94	67	52	38	31
Trace 2	112	78	63	53	46	41

Table 3
Memory usage of *VCDS* based super source detection for different *m*.

m	2×10^5 (kB)	4×10^5 (kB)	8×10^5 (kB)
Bit array *A*	25	50	100
(Trace 1) Hash table	467	596	709
(Trace 2) Hash table	345	448	536

out-degree. Therefore the estimated out-degree of a node with a small out-degree is more sensitive to the "noise".

5.2. Super source detection performance

We use two metrics, false positive percentage and false negative percentage to evaluate the performance of identifying super sources by our methods. Assume A_s is the super source set identified by directly examining all appeared sources, and $\widehat{A}_s$ is the super source set identified by our methods. For different threshold ϕ, A_s, the number of super sources, is shown as Table 2. The false positive percentage and false negative percentage are defined as $\frac{|\widehat{A}_s \setminus A_s|}{|\widehat{A}_s|}$ and $\frac{|A_s \setminus \widehat{A}_s|}{|A_s|}$ respectively.

Fig. 11 shows the results of *VCDS* based super source detection for different values of *m*, with $L = 1 \times 10^3$. We observe that accuracy improves by increasing *m*. The total memory used to store collected source addresses is shown in Table 3. We found that the number of collected node addresses increases with *m*.

Fig. 12 compares the performance of *RVCDS* based super source detection to *VCDS*, *CSE*, and *JM* based super source detection, with $m = 3 \times 10^6$ and $L = 1 \times 10^3$. We observe that *RVCDS* is less accurate than *VCDS*, and more accurate than *JM*. When $\phi > 0.8$, *RVCDS* almost has the same accuracy compared to *VCDS* and *CSE*, and is the most memory efficient as shown in Table 4.

Fig. 13 shows the results of sampled *VCDS* and sampled *RVCDS* for detecting super sources, when $m = 10^5$, $\tau = 0.1$, and $L = 1 \times 10^3$. Compared to *ESD*, sampled *VCDS* exhibits a lower false positive percentage. Compared to sampled *VCDS*, sampled *RVCDS* exhibits slightly higher false positive percentages. Both *VCDS* and *RVCDS* fail to measure node out-degrees and detect super sources when $m = 10^5$, since all bits in their bit arrays are almost all set to one. As shown in Table 5, sampled *RVCDS* needs much less

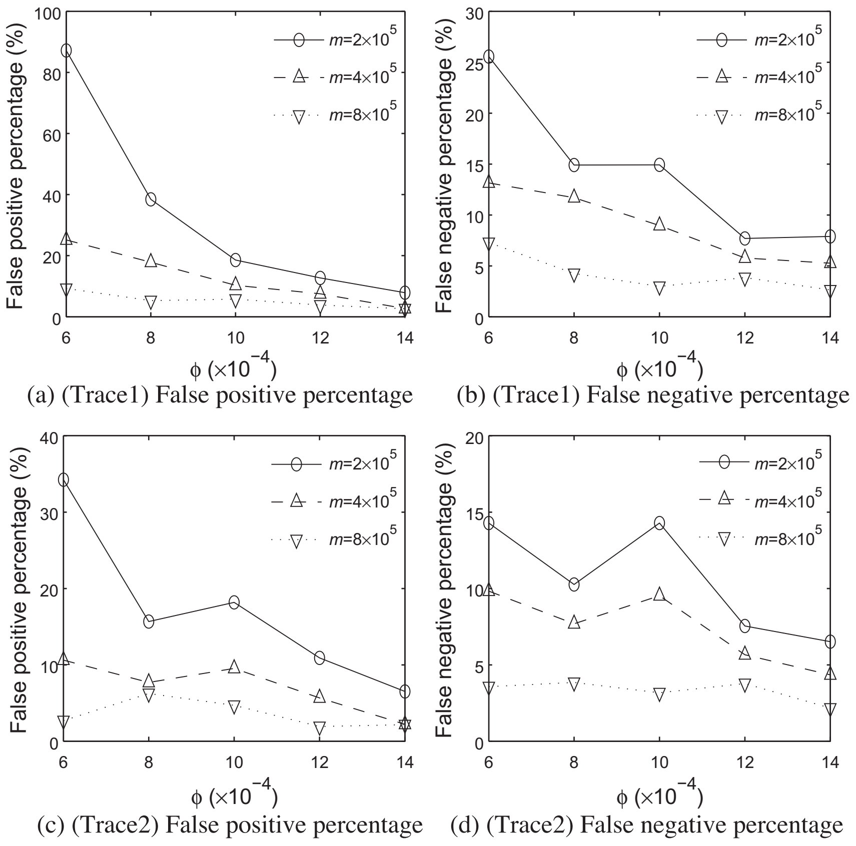
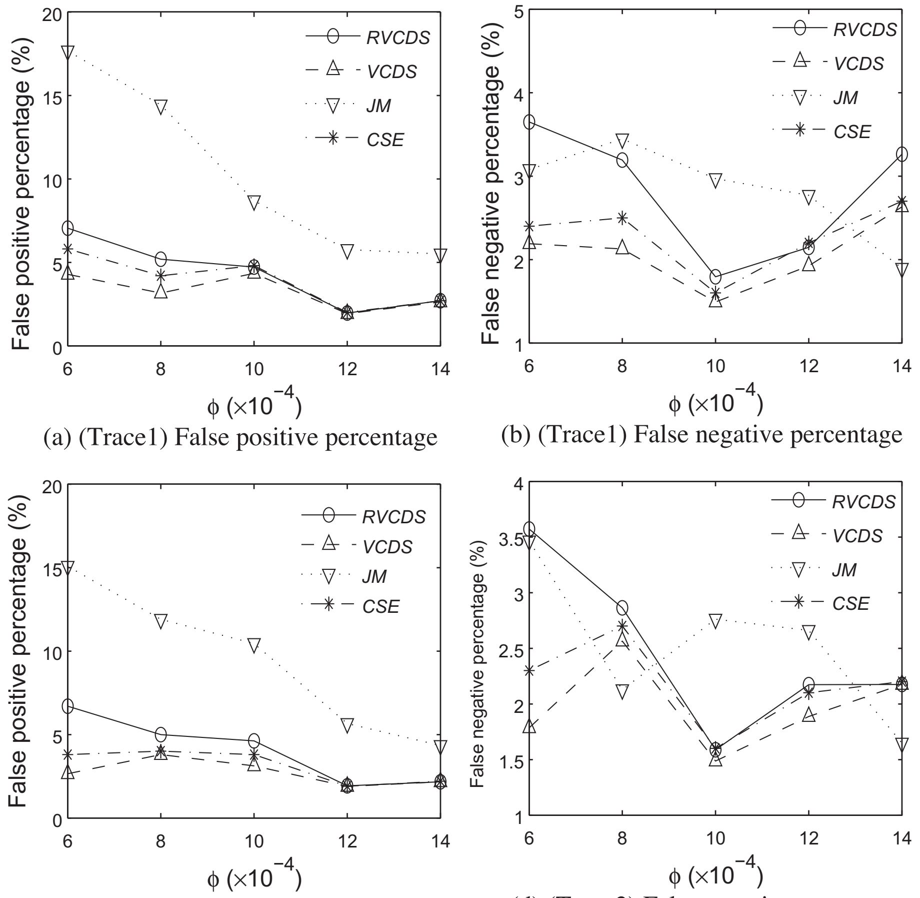


Fig. 11. Accuracy of *VCDS* based super source detection for different *m*.



(a) (Trace1) False positive percentage (b) (Trace1) False negative percentage

(c) (Trace2) False positive percentage (d) (Trace2) False negative percentage

Fig. 12. Comparison of accuracies of *RVCDS*, *VCDS*, *CSE*, and *JM* based super source detection.

Table 4
Comparison of memory usages of *RVCDS*, *VCDS*, *CSE*, and *JM* based super source detection.

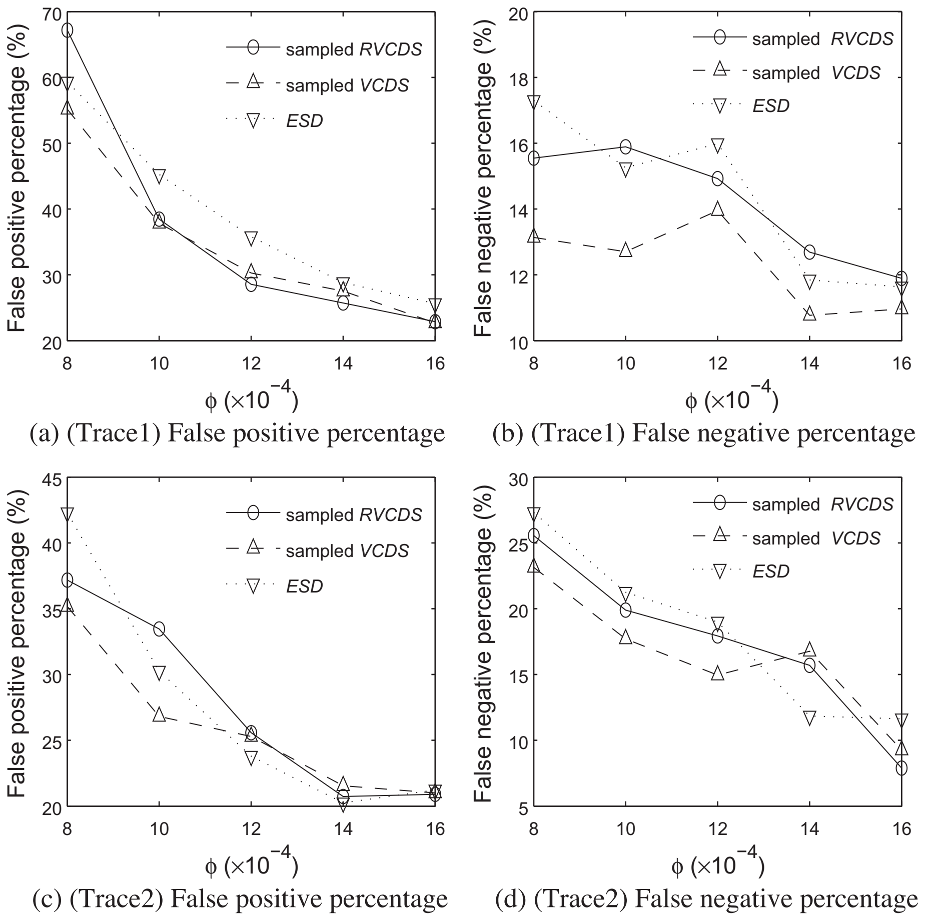
	RVCDS (MB)	*VCDS* (MB)	*CSE* (MB)	*JM* (MB)
Trace 1	0.375	1.17	1.17	1.18
Trace 2	0.375	1.00	1.00	1.01

memory compared to sampled *VCDS* and *ESD*, since it needs no extra memory to store source addresses sampled by the uniform flow sampling.

6. Related work

Estan and Varghese [6] proposed a family of bitmap algorithms for estimating the total number of distinct flows on high speed links. To estimate connection degree, a bitmap for each node needs to be constructed, which may not be scalable to high speed links carrying flows associated with a large number of nodes. Several sampling methods are proposed to detect super nodes [4,5], which greatly reduce the required memory space and the computational complexity by storing and processing only a small subset of network traffic. The accuracies of these methods greatly depend on the sampling rate.

Zhao et al. [16] proposed a data stream method to measure node connection degrees. This method is a variant of a Bloom filter [26] and consists of a two-dimensional bit array. Each node is associated with multiple columns in the bit array that are randomly selected using a group of hash functions, and a random bit in each of its associated columns is set to one for updating each packet of this node. The corresponding columns for one particular node can then be used to estimate its connection degree, since each column can be viewed as a direct bitmap as proposed in [23]. Wang et al. [17] modified this data structure and proposed an efficient reserve method to detect super nodes and nodes with significant connection degree changes. The direct bitmap method indicates that the number of rows in the bit array in [16,17] should be set in the order of thousands to perform the task of estimating connection degrees of nodes with thousands of flows. However, most nodes have a small set of flows and only a very small number of nodes have thousands of flows. This implies that most columns in the bit array are assigned to nodes with small connection degrees and contain mostly zeros. To reduce this space inefficiency, Yoon et al. [21] built a virtual bitmap for each node by taking bits randomly from a shared bit array using a group of hash functions. Each



(a) (Trace1) False positive percentage (b) (Trace1) False negative percentage (c) (Trace2) False positive percentage (d) (Trace2) False negative percentage

Fig. 13. Accuracy of sampled *VCDS* and sampled *RVCDS* compared to *ESD*.

Table 5
Memory usage of sampled *VCDS* and sampled *RVCDS* compared to *ESD*.

	Sampled *RVCDS* (kB)	Sampled *VCDS* (kB)	*ESD* (kB)
Trace 1	12.5	111	111
Trace 2	12.5	88.0	88.1

node's virtual bitmap is used to estimate its connection degree similar to what is proposed in [23] using a direct bitmap. Li et al. [18] proposed a super node detection method that incorporates *CSE* [21] and a uniform flow sampling technique. Additional memory is needed for a hash table to store node addresses for identification of super nodes. A super node in the hash table is determined by examining its estimated connection degree. The node address for each sampled flow is stored if and only if the associated bit in the shared bit array is set from zero to one. Therefore additional computation is required for updating each packet, and a super node might also be missed.

7. Conclusions and future work

In this paper we present a virtual indexing method *VCDS* for estimating node connection degrees. With this method we can generate a very compact sketch of network traffic, and accurately estimate the connection degree of each node based on the associated virtual bitmaps. Meanwhile several memory efficient methods are developed for detecting super nodes. The *VCDS* based method is accurate and computationally efficient. However it needs extra memory space to store node addresses that appear in the monitored network. The *RVCDS* based method can identify super node addresses analytically and needs no extra memory space but with a little compromised accuracy. The combined sampled *VCDS* and sampled *RVCDS* with a uniform flow sampling technique to filter nodes with a small number of flows are more memory efficient. The experiments based on the actual network traffic show that the new methods are more memory efficient and accurate than the existing methods.

Acknowledgments

The research presented in this paper is supported in part by National Natural Science Foundation (60574087, 61103241, 61103240), and the Fundamental Research Funds for Central Universities.

Appendix A

A.1. Proof of Theorem 1

The proof is by contradiction. Assume to the contrary that there exists $f_{i,j_1}(s) = f_{i,j_2}(s)$, for $j_1 \neq j_2$ then

$$(j_2 - j_1)\psi_{i,2}(s) \equiv 0 \mod m.$$

Since $1 \leqslant j_2 - j_1 \leqslant L - 1 \ll m$ and m is prime, we have $j_2 - j_1 \neq 0 \mod m$ and $\psi_{i,2}(s) \equiv 0 \mod m$. Note $\psi_{i,2}(s) \leqslant m - 1$, so we have $\psi_{i,2}(s) = 0$. This contradicts with the definition of $\psi_{i,2}(s) \in \{1, 2, \ldots, m-1\}$. □

A.2. Proof of Theorem 2

When $k_1 \in \mathbb{F}_i(s)$ and $A[k_1]$ corresponds the jth $(0 \leqslant j \leqslant L-1)$ element in $B_i(s)$, we have the following equation from the definition of $f_{i,j}(s)$

$$k_1 = \psi_{i,1}(s) + j\psi_{i,2}(s) \mod m.$$

For each pair of $\psi_{i,2}(s) \in \{1, 2, \ldots, m-1\}$ and $j \in \{0, 1, \ldots, L-1\}$, there is one and only one virtual bitmap containing $A[k_1]$, since $\psi_{i,1}(s)$ is determined as

$$\psi_{i,1}(s) = k_1 - j\psi_{i,2}(s) \mod m.$$

Therefore the total number of distinct virtual bitmaps containing $A[k_1]$ is $L(m-1)$.

When $k_1, k_2 \in \mathbb{F}_i(s)$, we have

$$\begin{cases} k_1 = \psi_{i,1}(s) + j_1\psi_{i,2}(s) \mod m \\ k_2 = \psi_{i,1}(s) + j_2\psi_{i,2}(s) \mod m \end{cases}$$

where $0 \leqslant j_1, j_2 \leqslant L-1$, and $j_1 \neq j_2$. Thus

$$\begin{cases} (j_1 - j_2)\psi_{i,1}(s) = j_1k_2 - j_2k_1 \mod m \\ (j_1 - j_2)\psi_{i,2}(s) = k_1 - k_2 \mod m. \end{cases}$$

Since $j_1 - j_2 \neq 0 \mod m$, we have $(j_1 - j_2)^{m-1} \equiv 1 \mod m$ from the Euler-Fermat theorem [27]. Then $\psi_{i,1}(s)$ and $\psi_{i,2}(s)$ are determined as

$$\begin{cases} \psi_{i,1}(s) = (j_1 - j_2)^{m-2}(j_1k_2 - j_2k_1) \mod m \\ \psi_{i,2}(s) = (j_1 - j_2)^{m-2}(k_1 - k_2) \mod m. \end{cases}$$

It indicates that there is one and only one virtual bitmap containing both $A[k_1]$ and $A[k_2]$ for each pair of j_1 and j_2. Therefore the total number of distinct virtual bitmaps containing both $A[k_1]$ and $A[k_2]$ is $L(L-1)$, the number of distinct pairs of j_1 and j_2.

Thus, there are $L(m-1) - L(L-1) = L(m-L)$ distinct virtual bitmaps containing $A[k_1]$ but not $A[k_2]$, and the total number of distinct virtual bitmaps not containing $A[k_1]$ or $A[k_2]$ either is $m(m-1) - L(L-1) - 2L(m-L)$.

Based on the above analysis, Theorem 2 is proved since each virtual bitmap $B_i(s)$ is selected uniformly from a total of $m(m-1)$ distinct virtual bitmaps. □

A.3. Proof of Theorem 3

Define $y = \ln\lambda(H)$. Then its first derivative is

$$\frac{dy}{dH} = \ln(1 - \Phi^H) - \frac{H\Phi^H \ln\Phi}{1 - \Phi^H},$$

and second derivative is

$$\frac{d^2y}{dH^2} = \frac{\Phi^H(2 - 2\Phi^H - H\ln\Phi)\ln\Phi}{(1 - \Phi^H)^2},$$

It can be easily shown that $\frac{dy}{dH} < 0$ when $H \in (0, -\frac{1}{\ln\Phi})$ and $\frac{dy}{dH} > 0$ when $H \in (-\frac{1}{\ln\Phi}, +\infty)$, therefore function $\lambda(H)$ decreases with $H \in (0, -\frac{1}{\ln\Phi})$ and increases with $H \in (-\frac{1}{\ln\Phi}, +\infty)$. Meanwhile we have $\frac{dy}{dH}\Big|_{H=-\frac{1}{\ln\Phi}} = 0$ and $\frac{d^2y}{dH^2}\Big|_{H=-\frac{1}{\ln\Phi}} = 0$, so the optimal value of $\lambda(H)$ is obtained at $H = -\frac{1}{\ln\Phi}$. □

References

[1] M. Roesch, Snort-lightweight intrusion detection for networks, in: Proceedings of the USENIX LISA Conference on System Administration, 1999, pp. 229–238.

[2] D. Plonka, Flowscan: a network traffic flow reporting and visualization tool, in: Proceedings of USENIX LISA, 2000, pp. 305–317.

[3] C. Estan, G. Varghese, New directions in traffic measurement and accounting, in: Proceedings of ACM SIGCOMM, 2002, pp. 323–336.

[4] S. Venkataraman, D. Song, P.B. Gibbons, A. Blum, New streaming algorithms for fast detection of superspreaders, in: Proceedings of NDSS, 2005, pp. 149–166.

[5] J. Cao, Y. Jin, A. Chen, T. Bu, Z. Zhang, Identifying high cardinality Internet hosts, in: Proceedings of IEEE INFOCOM, 2009, pp. 810–818.

[6] C. Estan, G. Varghese, M. Fisk, Bitmap algorithms for counting active flows on high speed links, in: Proceedings of ACM SIGCOMM IMC, 2003, pp. 182–209.

[7] A. Kumar, J. Xu, J. Wang, O. Spatschek, L. Li, Space-code Bloom filter for efficient per-flow traffic measurement, in: Proceedings of IEEE INFOCOM, 2004, pp. 1762–1773.

[8] Y. Zhang, S. Singh, S. Sen, N.G. Duffield, C. Lund, Online identification of hierarchical heavy hitters: algorithms, evaluation, and application, in: Proceedings of ACM SIGCOMM IMC, 2004, pp. 101–114.

[9] W. Feng, Y. Huang, X. Li, Reversible sketch data structure, Journal of Tsinghua University (Sci & Tech) 48 (2008) 1621–1622.

[10] J. Mai, A. Sridharan, H. Zang, C.-N. Chuah, Fast filtered sampling: catching mice and elephants with one net, Elsevier Computer Networks 54 (2010) 1885–1898.

[11] B. Krishnamurthy, S. Sen, Y. Zhang, Y. Chen, Sketch-based change detection: methods, evaluation, and applications, in: Proceedings of ACM SIGCOMM IMC, 2003, pp. 234–247.

[12] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, E. Parsons, Y. Zhang, P. Dinda, M. yang Kao, G. Memik, Reversible sketches: enabling monitoring and analysis over high-speed data streams, IEEE/ACM Transactions on Networking 15 (2007) 1059–1072.

[13] G. Cormode, S. Muthukrishnan, What's hot and what's not: tracking most frequent items dynamically, in: Proceedings of ACM PODC, 2003, pp. 296–306.

[14] A. Kumar, M. Sung, J. Xu, J. Wang, Data streaming algorithms for efficient and accurate estimation of flow size distribution, in: Proceedings of ACM SIGMETRICS, 2004, pp. 177–188.

[15] B. Ribeiro, T. Ye, D. Towsley, A resource-minimalist flow size histogram estimator, in: Proceedings of ACM SIGCOMM IMC, 2008, pp. 285–290.

[16] Q. Zhao, A. Kumar, J. Xu, Joint data streaming and sampling techniques for detection of super sources and destinations, in: Proceedings of ACM SIGCOMM IMC, 2005, pp. 77–90.

[17] P. Wang, X. Guan, T. Qin, Q. Huang, A data streaming method for monitoring host connection degrees of high-speed links, IEEE Transactions on Information Forensics and Security 6 (2011) 1086–1098.

[18] T. Li, S. Chen, W. Luo, M. Zhang, Scan detection in high-speed networks based on optimal dynamic bit sharing, in: Proceedings of IEEE INFOCOM, 2011, pp. 3200–3208.

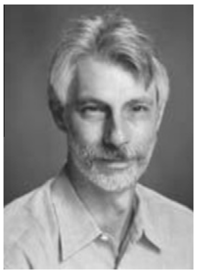
[19] A. Lall, V. Sekar, M. Ogihara, J. Xu, H. Zhang, Data streaming algorithms for estimating entropy of network traffic, in: Proceedings of ACM SIGMETRICS, 2006, pp. 145–156.

[20] H. Zhao, A. Lall, O. Spatscheck, J. Wang, J. Xu, A data streaming algorithm for estimating entropies of OD flows, in: Proceedings of ACM SIGCOMM IMC, 2007, pp. 279–290.

[21] M. Yoon, T. Li, S. Chen, J.K. Peir, Fit a spread estimator in small memory, in: Proceedings of IEEE INFOCOM, 2009, pp. 504–512.
[22] X. Guan, P. Wang, T. Qin, A new data streaming method for locating hosts with large connection degree, in: Proceedings of IEEE GLOBECOM, 2009, pp. 6421–6426.
[23] K. Whang, B.T. Vander-zanden, H.M. Taylor, A linear-time probabilistic counting algorithm for database applications, IEEE Transaction of Database Systems 15 (1990) 208–229.
[24] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press, Cambridge, MA, 2001.
[25] T. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, New York, NY, 1995.
[26] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (1970) 422–426.
[27] W. Stallings, Cryptography and Network Security: Principles and Practice, fourth ed., Pearson Education, Upper Saddle River, NJ, 1998.

Pinghui Wang received the B.S. and M.S. degrees in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2006, 2008 respectively. He is currently a Ph.D. candidate with the Systems Engineering Institute and SKLMS Laboratory, Xi'an Jiaotong University under the supervision of Prof. Xiaohong Guan. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement.

Xiaohong Guan received the B.S. and M.S. degrees in automatic control from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, US, in 1993. From 1993 to 1995, he was a consulting engineer at PG&E. From 1985 to 1988, he was with the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China. From January 1999 to February 2000, he was with the Division of Engineering and Applied Science, Harvard University, Cambridge, MA. Since 1995, he has been with the Systems Engineering Institute, Xi'an Jiaotong University, and was appointed Cheung Kong Professor of Systems Engineering in 1999, and dean of the School of Electronic and Information Engineering in 2008. Since 2001 he has been the director of the Center for Intelligent and Networked Systems, Tsinghua University, and served as head of the Department of Automation, 2003–2008. He is an Editor of *IEEE Transactions on Power Systems* and an Associate Editor of *Automata*. His research interests include allocation and scheduling of complex networked resources, network security, and sensor networks. He has been elected Fellow of IEEE.



Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from University of Texas. From 1976 to 1985 he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, NY; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs-Research, Florham Park, NJ; and Microsoft Research Lab, Cambridge, UK. His research interests include networks and performance evaluation. He currently serves as Editor-in-Chief of IEEE/ACM Transactions on Networking and on the editorial boards of Journal of the ACM, and IEEE Journal on Selected Areas in Communications, and has previously served on numerous other editorial boards. He was Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE 92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has received the 2007 IEEE Koji Kobayashi Award, the 2007 ACM SIGMETRICS Achievement Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous best conference/workshop paper awards. Last, he has been elected Fellow of both the ACM and IEEE.

Jing Tao received the B.S and M.S degrees in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2001, 2006 respectively. He is currently a teacher in Xi'an Jiaotong University and on-the-job Ph.D. candidate with the Systems Engineering Institute and SKLMS Laboratory, Xi'an Jiaotong University under the supervision of Prof. Xiaohong Guan. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and botnet.