ICIC Express Letters
Volume **11**, Number **3**, March **2017**
 ISSN 1881-803X
pp. 551–558

A NOVEL DETECTOR FOR PERSISTENT SPREADS OVER DATA CENTER BASED ON BLOOM FILTER

Lin Han[1], Weijiang Liu[1], Zhiyang Li[1], Wenyu Qu[2], Mingqian Bai[1] and Yegang Du[3]

[1]School of Information Science and Technology
Dalian Maritime University
No. 1, Linghai Road, Dalian 116026, P. R. China
xiaolinlin930211@icloud.com; { wjliu; zyli }@dlmu.edu.cn

[2]School of Computer Software
Tianjin University
No. 92, Weijin Road, Tianjin 300072, P. R. China
wenyu.qu@tju.edu.cn

[3]School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
duyegang@gmail.com

Received July 2016; accepted October 2016

Abstract. *Data center networks are vulnerable and easily plagued by long-term stealthy malicious attacks, which cannot be recognized by measurement based on host cardinality. Thus, this paper presents a novel detector for persistent spreads based on Bloom filter. In our design, multi-stage filter structure is proposed which can achieve high operation speed and low memory consumption because only filtered sips are calculated. Due to the nature of Bloom filter, false negative ratio (FNR) equals 0 all the time. The ideas and mechanisms are illustrated using different traces collected from real networks. Extensive experimental results based on real traces show that the proposed detector has better accuracy than other existing approaches.*
Keywords: Data center, Persistent spreads, Bloom filter, Cardinality estimation

1. **Introduction.** With the rapid development of cloud computing, events of security happen constantly and data centers are, therefore, point of vulnerability to many network attacks. Data center networks are easily plagued by them. These attacks result in a certain influence on the data center and pose a critical threat to user defined operations of data center networks. Hence, the growing importance of network security cannot be overstated. Accordingly, defense mechanisms are needed. As depicted in [1], among four defense mechanisms (precaution, detection, attack source track back and response), detection is the vital step in fighting against attacks. That is to say, these attack incidents need to be detected as soon as possible once they happen.

The objective of attacks is not to overwhelm the target, but to degrade its performance using a small fraction of attacking machines. These machines send a lot of fake requests for disrupting the legitimate connectivity and services, resulting in victim server resource exhaustion. To address this problem, Ishibashi et al. proposed the concept of host cardinality in [2], since then a lot of researches have focused on detection based on host cardinality. Existing measurements based on host cardinality [3-6] make great sense in practical network monitoring and management, especially for detecting Flow-Scan, DDOS attacks [7], Coral-Reef [8], worm, etc. Compact spread estimator (CSE) [9] utilizes virtual vectors to estimate host cardinality. CSE [9] is composed of two modules: one for storing source destination pairs in virtual vectors, and the other for estimating spreads. A virtual

vector consists of s bits randomly selected from a bit array. Two virtual vectors may share one or more common bits, and each source has its own virtual vector.

Measurements based on host cardinality deal with attacks caused by massive connections in a short period of time, but they are not good at detecting long-term stealthy malicious spreaders. For example, if an attacker keeps flow numbers within the threshold, this attack cannot be detected by measurements based on host cardinality. Yoon and Chen [10] present random aging streaming filter (RAS) to detect stealthy spreaders.

Recently, Xiao et al. [11] propose a new data structure called multi-virtual bitmap (MVBitmap) to detect low-rate stealthy attacks by measuring persistent spreads. In each time period, a physical bitmap is used to store the information of flows, and each destination is allocated with a virtual bitmap to record its elements in the time period. When the measurement terminates, the sequence of physical bitmaps M_1, $M_2, \ldots, M_t$ is used to estimate persistent spreads. For each destination, t virtual bitmaps B_1, $B_2, \ldots, B_t$ can be extracted from these physical bitmaps. Meanwhile, an algorithm for estimating the persistent spread of a destination is proposed. The number of different original hosts that contact a destination host persistently within a predefined t measurement period is called persistent spread of the destination host. If the persistent spread of a destination host is more than 0, then this host is called a persistent destination host. Also, an original host is called persistent element if it persistently contacts a persistent destination host in all t measurement periods.

Wang et al. [12] propose virtual connection degree sketch (VCDS) to measure host connection degrees in high-speed networks using H virtual vectors. In order to remove noise, a filtered bitmap is generated by bit-ANDing multiple virtual vectors. Accuracy can be guaranteed because VCDS uses much more virtual vectors.

In this paper, we present a novel detector for persistent spreads based on Bloom filter. The main contributions of this paper are summarized as follows.

(1) A novel detector based on Bloom filter is designed for persistent hosts to detect long-term stealthy malicious spreaders. Due to the nature of Bloom filter, false negative ratio is 0.

(2) A structure composed of multiple Bloom filters is designed. Since the detector only permits persistent traffic to pass the filter structure, traffic processed by packet updating module is significantly reduced in the tth measurement period. Therefore, the module only has a small memory requirement for recording persistent hosts and estimating persistent spreads.

(3) This system can be easily implemented and fit in small but fast memory space to keep up with high-speed links. Extensive experimental results based on real traces show that the proposed detector is more accurate than other existing approaches.

The rest of this paper is organized as follows. In Section 2, we describe the design of the proposed detector. Section 3 presents the analysis of algorithm complexity. In Section 4, we evaluate its performance by using public traces in experiments. This paper is concluded in Section 5.

2. Detector Structure.

In this section, we briefly introduce the architecture of our detector, and propose multi-stage filter structure.

2.1. Detector architecture.

Data center networks (DCNs) consist of intra DCNs and inter DCNs [13]. In an intra DCN, flow controller seizes IP packets from data center and extracts flow IDs by screening IP headers. Figure 1 illustrates the architecture of our detector. As shown in Figure 1, flow filter is the most important part of the architecture. We will describe it in the next subsection. Flow filter and packet updating modules form online updating. In packet updating module, all persistent hosts are stored in array flow, and we may use LinkedList, VCDS, and CSE to store flow information, respectively.

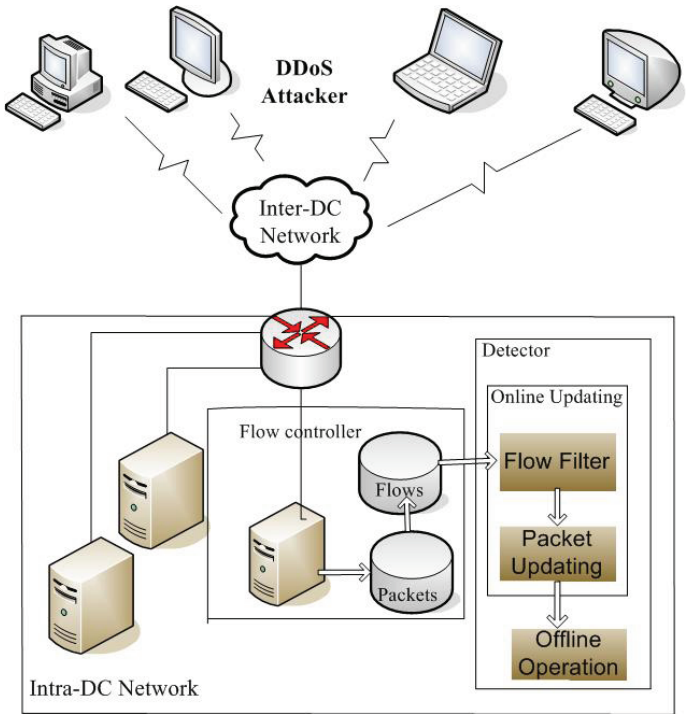


FIGURE 1. Detector architecture

In the final part offline module, we use VCDS and CSE to calculate persistent spreads respectively. Certainly, in specific applications one of these three methods may be chosen to implement according to the requirement for resources and measurement accuracy.

2.2. **Multi-stage filter structure.** In this subsection, we present multi-stage filter structure. Flow filter consists of $(t-1)$ Bloom filters, which are all arrays of m bits and are denoted as $BF_1[m]$, $BF_2[m]$, ..., $BF_{t-1}[m]$, respectively. The same three $h_1(sip, dip)$, $h_2(sip, dip)$, $h_3(sip, dip)$ are used to map $< sip, dip >$ into BF. In other words, when a packet $< sip, dip >$ arrives, the hash function $h_i(sip, dip)$ decides which bit will be set in array BF. Only those flows that have passed previous filters can update the current Bloom filter. Assuming that indexed bits are i, j, l, we set $BF_1[i] = BF_1[j] = BF_1[l] = 1$. During the t measurement period, the whole filter algorithm is outlined as Algorithm 1.

3. **Analysis.** Since there are enough resources for offline statistics, we mainly analyze the complexity of online module consisting of flow filter and packet updating.

3.1. **Space complexity.** The space consumption of the algorithm (mainly refers to the SRAM space consumption) consists of three aspects:

(1) Filter module based on Bloom filter;
(2) Data structure storing network flow information;
(3) $Flow[]$ representing the storage of different $dips$.

Since the number of persistent hosts is very small (at most hundreds) and false positive rate is low, the number of dips that can pass the filters is also very small. Therefore, the memory consumption of $Flow[]$ may be neglected.

In flow filter module, we use a Bloom filter with the array of m bits for each $(t-1)$ measurement, so the memory consumption in filter module is $(t-1) \times m$ bits. In the packet updating process, we analyze the memory consumption of linked list, VCDS and CSE respectively.

Algorithm 1 Filter Algorithm

```
Initialize
 BF_k[i] := 0; i = 0, 1, ..., m − 1; 1 ≤ k < t
End Initialize
Start Filter:
During the first measurement period:
 Upon the arrival of packet < sip, dip >
  i := h_1(sip, dip);
  j := h_2(sip, dip);
  l := h_3(sip, dip);
  BF_1[i] := 1;
  BF_1[j] := 1;
  BF_1[l] := 1;
During the k (2 ≤ k < t) measurement period:
 Upon the arrival of packet < sip, dip >
  i := h_1(sip, dip);
  j := h_2(sip, dip);
  l := h_3(sip, dip);
if BF_{k−1}[i] == 1&&BF_{k−1}[j] == 1&&BF_{k−1}[l] == 1 then
      BF_k[i] := BF_k[j] := BF_k[l] := 1;
end if
During the t measurement period:
 Upon the arrival of packet < sip, dip >
  i := h_1(sip, dip);
  j := h_2(sip, dip);
  l := h_3(sip, dip);
if BF_{t−1}[i] == 1&&BF_{t−1}[j] == 1&&BF_{t−1}[l] == 1 then
      Pass the packet to packet updating module;
end if
End Filter
end
```

A. Linked List

To store flow information, we assign storage space of size p to store the persistent IPs *sips* and *dips* in the updating process. So the online updating memory consumption is p, and memory consumption combining linked list and Bloom filter is $(t-1) \times m + p$ bits.

B. VCDS

Memory consumption in VCDS is mainly from $A = \{A_0, A_1, \ldots, A_n\}$ which is described by an array of M bits. The memory consumption is q bits, and memory consumption combining VCDS algorithm and Bloom filter is $(t-1) \times m + q$ bits.

C. CSE

Memory consumption mainly comes from bitmap B with the storage of flow information. Each unit occupies only one bit space, and memory consumption is n bits. Memory consumption combining CSE and Bloom filter is $(t-1) \times m + n$ bits.

3.2. **Time complexity.** Time consumption mainly concentrates on flow filter module and packet updating module. For each packet, processing time depends on the number of memory accesses. During $(t-1)$ measurement periods, Bloom filter launches parallel hash operations using flow IDs, and this time complexity is $O(1)$. Secondly, packet updating module is required to store flow information. During the process, VCDS and CSE need to hash flow IDs, so this time complexity is $O(1)$. For the linked list, we construct a linked

list for each *dip* as possible. Therefore, it is the key to insert an *src* into the linked list of *dip*. If *src* has occurred in the linked list, we discard *src*. In order to search *src* in the linked list, the time complexity is $O(k)$, where k is the persistent spread of *dip*. Finally, in order to store filtered *dips*, it requires to traverse the linked list storing *dips*. This operating time complexity is $O(r)$, where r is the number of persistent hosts. In general, k and r are a few tens in the order of magnitude.

4. **Experiment.**

4.1. **Evaluation metrics.**

In this section, data for experiments are obtained from actual network traffic in real-world Internet. Experiments are implemented on the data set which is from daily trace including MAWI [14] and NLANR [15]. All experiments have been performed on a stimulated cluster running Ubuntu server 64 bit. All algorithms mentioned in Section 3 are conducted with WEKA [16] API. First, we set up the experimental parameters. In this paper, Bloom filter size m is set to 65536. To make comparison, measurement period t is chosen as 4, 6, 8 respectively. We set the number linked lists for *dips* to 65536 in the experiment. For VCDS, a bit array A of size M is set to 8000, the size of bitmap is set to $L = 103$ and H is set to 2. Besides, the size of the bitmap B in CSE is described by the array $n = 160000$, and the size of the virtual vector is set to $s = 256$.

4.2. **Performance analysis of the detector.**

In order to evaluate the performance of the proposed detector, we investigate false negative ratio (FNR) and false positive ratio (FPR) [17]:

$$FNR = \frac{s^-}{s}$$

$$FPR = \frac{s^+}{s}$$

where s is the number of actual persistent hosts, s^- is the number of actual persistent hosts which are not identified, and s^+ is the number of non-persistent hosts being incorrectly identified.

According to FNR and FPR, the accuracy of the algorithm is evaluated. Measurement periods are set to t in advance, and corresponding detection results are shown in Table 1. It can be seen that our detector works well. Note that FPR of our detector only depends on filter module, independent of the three methods: linked list, VCDS, and CSE. In Table 2, we compare our detector with the method based on MVBitmap. In most cases, FPR of our detector in MAWI is lower than that of MVBitmap. The decrease of FPR reflects improved accuracy.

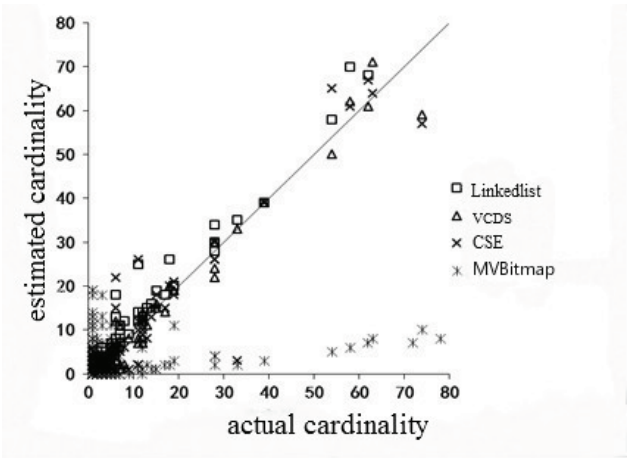
In this paper, we use two traces to verify the ability of our detector and the performance can be further evaluated by comparing actual persistent spreads and measured spreads. Figure 2 and Figure 3 compare three algorithms and MVBitmap using MAWI and NLANR. The x-coordinate represents actual persistent spreads, while y-coordinate represents estimated ones. The diagonal line is a baseline being used for evaluation. As

Table 1. The experimental results of FNR and FPR of our detector

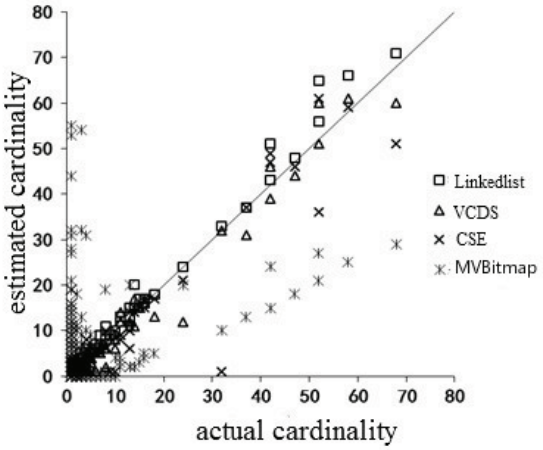
Trace	t	FNR	FPR
MAWI	4	0	28.9%
	6	0	13.9%
	8	0	9.9%
NLANR	4	0	6.4%
	6	0	5.9%
	8	0	4.7%

TABLE 2. Comparisons of FNR and FPR

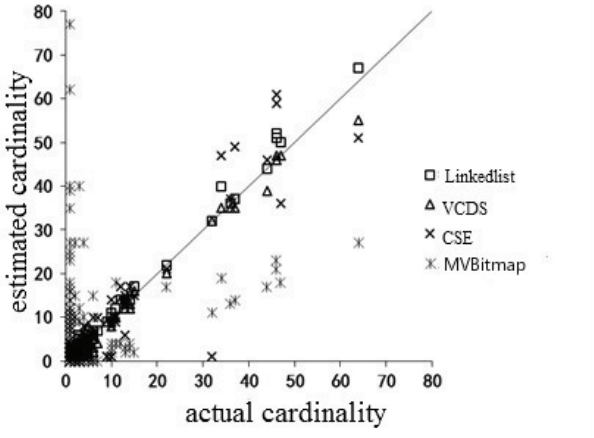
Trace	Algorithms	t	FNR	FPR
MAWI	our detector	4	0	28.9%
		6	0	13.9%
		8	0	9.9%
	MVBitmap	4	0	31.5%
		6	0	10.9%
		8	0	8.6%
NLANR	our detector	4	0	6.4%
		6	0	5.9%
		8	0	9.9%
	MVBitmap	4	0	11.6%
		6	0	11.6%
		8	0	10.4%



(a) $t = 4$



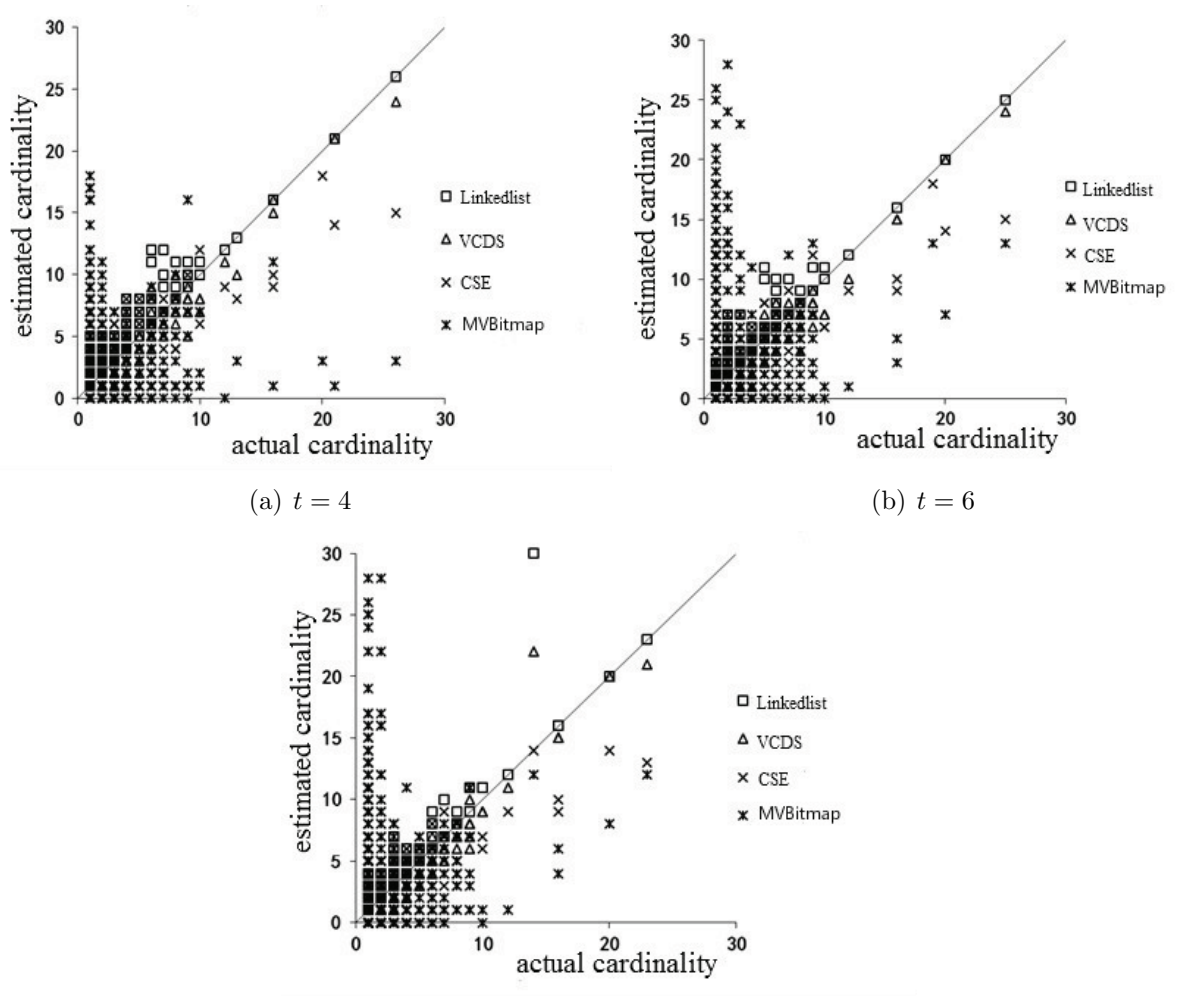
(b) $t = 6$



(c) $t = 8$

FIGURE 2. Comparisons of estimates of our detector with actual persistent spreads, (a) $t = 4$, (b) $t = 6$, (c) $t = 8$

Figure 2 and Figure 3 shown, most of points are near to the diagonal. This means that our detector is much better than MVBitmap in estimating persistent spreads. According to FPR, FNR, time and space complexity, Table 3 shows the comparisons between three algorithms in terms of accuracy, space consumption and time complexity. The result in



(a) $t = 4$

(b) $t = 6$

(c) $t = 8$

FIGURE 3. Comparisons of our detector with MVBitmap, (a) $t = 4$, (b) $t = 6$, (c) $t = 8$

TABLE 3. The experimental results of three methods in our detector

Algorithms	Accuracy	Space Consumption	Time Complexity
LinkedList	High	High	High
VCDS	Middle	Middle	Middle
CSE	Low	Low	Low

Table 3 is of a certain significance to further research. For further research, it is the best choice to choose LinkedList when high accuracy is necessary. If space consumption and time complexity are needed to be guaranteed, CSE is obviously the best way. VCDS can be seen as a best trade-off solution among accuracy, space consumption, and time complexity.

5. **Conclusions.** Data center networks are vulnerable to long-term stealthy malicious attacks which disrupt the legitimate services and result in victim server resource exhaustion. However, existing detection measurement based on host cardinality cannot recognize these attacks. Thus, in this paper, we design a novel detector for persistent spreads based on Bloom filter. At the same time, we propose a multi-stage filter structure. The theoretical analysis reveals that FNR equals 0 all the time and computation complexity is

low. The experimental results show that our detector can precisely and efficiently detect persistent spreads. Future research will be concentrated on improving data structure to consume lower memory during cardinality estimation.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China 61370187, 61370198, 61370199, the Fundamental Research Funds for the Central Universities 3132016029, 3132016032. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] J. H. Jun, H. Oh and S. H. Kim, DDoS flooding attack detection through a step-by-step investigation, *IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, Nesea, Perth, Australia, pp.1-5, 2011.

[2] K. Ishibashi, T. Mori, R. Kawahara, Y. Hirokawa, A. Kobayashi, K. Yamamoto and H. Sakamoto, Estimating top n hosts in cardinality using small memory resources, *Proc. of the 22nd International Conference on Data Engineering Workshops*, Atlanta, GA, USA, pp.29-35, 2006.

[3] S. Venkataraman, D. Song, P. B. Gibbons and A. Blum, New streaming algorithms for fast detection of superspreaders, *Network and Distributed System Security Symposium*, San Diego, California, USA, pp.149-166, 2005.

[4] Y. Xiang, K. Li and W. Zhou, Low-rate DDoS attacks detection and traceback by using new information metrics, *IEEE Trans. Inf. Forensics Secur.*, vol.6, no.2, pp.426-437, 2011.

[5] M. Roesch, Snort-lightweight intrusion detection for networks, *Proc. of the 13th USENIX Conference on System Administration*, Seattle, Washington, pp.229-238, 1999.

[6] D. Plonka, Flowscan: A network traffic flow reporting and visualization tool, *Proc. of the 14th System Administration Conference*, New Orleans, LA, pp.305-317, 2000.

[7] B. Krishnamurthy, S. Sen, Y. Zhang and Y. Chen, Sketch-based change detection: Methods, evaluation, and applications, *Proc. of the 3rd ACM SIGCOMM Conference on Internet Measurement*, Miami Beach, Florida, USA, pp.234-247, 2003.

[8] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch and K. Claffy, The architecture of CoralReef: An Internet traffic monitoring software suite, *Workshop on Passive and Active Measurements*, 2001.

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

[10] M. K. Yoon and S. Chen, Detecting stealthy spreaders by random aging streaming filters, *IEICE Trans. Commun.*, vol.94-B, no.8, pp.2274-2281, 2011.

[11] Q. Xiao, Y. Qiao and M. Zhen, Estimating the persistent spreads in high-speed networks, *IEEE International Conference on Network Protocols*, pp.131-142, 2014.

[12] P. Wang, X. Guan, W. Gong and D. Towsley, A new virtual indexing method for measuring host connection degrees, *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, China, pp.156-160, 2011.

[13] C. Guo, L. Yuan and D. Xiang, Pingmesh: A large-scale system for data center network latency measurement and analysis, *ACM Conference on Special Interest Group on Data Communication*, pp.139-152, 2015.

[14] MAWI, *WIDE Measurement and Analysis on the WIDE Internet Working Group*, http://Tracer.csl.sony.co.jp/mawi/samplepoint-F/20090330/200903300000.html.

[15] NLANR, *Passive Measurement and Analysis*, ftp://wits.cs.waikato.ac.nz/pma/long/ipls/3/, 2015.

[16] *WEKA*, http://www.cs.waikato.ac.nz/ml/weka, 2013.

[17] W. Liu, W. Qu, J. Gong and K. Li, Detection of superpoints using a vector bloom filter, *IEEE Trans. Information Forensics & Security*, vol.11, no.3, pp.514-527, 2016.