KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.1 pp.15~22 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.1.15

Effect of Sampling for Multi-set Cardinality Estimation

DinhNguyen Dao[†] · DaeHun Nyang[††] · KyungHee Lee[†††]

ABSTRACT

Estimating the number of distinct values is really well-known problems in network data measurement and many effective algorithms are suggested. Recent works have built upon technique called Linear Counting to solve the estimation problem for massive sets or spreaders in small memory. Sampling is used to reduce the measurement data, and it is assumed that sampling gives bad effect on the accuracy. In this paper, however, we show that the sampling on multi-set estimation sometimes gives better results for CSE with sampling than for MCSE that examines all the packets without sampling in terms of accuracy and estimation range. To prove this, we presented mathematical analysis, conducted experiment with real data, and compared the results of CSE, MCSE, and CSES.

Keywords : Traffic Measurement, Spreader, Estimation, Sampling

멀티셋의 크기 추정 기법에서 샘플링의 효과

DinhNguyen Dao[†] · 양 대 헌[††] · 이 경 희[†††]

요 약

멀티셋에서 중복을 제외한 서로 다른 원소의 수를 추정하는 것은 네트워크 트래픽 측정 분야에서 매우 잘 알려진 문제이며, 많은 알고리즘들이 제안되었다. 최근에는 선형 카운팅 기법(Linear Counting)에 기반해서 매우 작은 메모리만을 이용해서 멀티셋의 크기를 추정하는 알고리즘이 개발되었다. 너무 많은 데이터를 처리하기 어려운 경우 전체 데이터를 처리하지 않고, 패킷의 일부를 샘플링해서 사용하는데, 이 샘플링은 일반적으로 정확도에 부정적인 영향을 주는 것으로 알려져있다. 하지만, 이 논문에서는 멀티셋의 크기를 추정하는데 있어서 CSE를 이용하는 경우 샘플링이 정확도와 측정 범위의 측면에서 오히려 전수조사를 하는 MCSE보다 더 좋은 결과를 낼 수 있음을 보였다. 이를 입증하기 위해 수학적 분석, 실제 데이터를 이용한 실험을 수행하고, CSE, MCSE 그리고 CSES를 비교하였다.

키워드 : 트래픽 측정, 멀티셋 크기, 근사치 추정, 샘플링

1. Introduction

Among various kinds of network traffic measurements, counting items with a distinct value plays a vital role in many applications. The values can be a combination of source address, destination address, a flow or a resource address. In this paper, we focus on the spreader which has contact with numerous destination addresses. High spreader can be used by a worm, or by malware. Thus, it is important to find out the spreads or the number of distinct values. Certainly, the spread estimation algorithm can be extended to other fields that need to count the distinct value.

There have been many research works for spreader estimation such as [1-5], but these were for counting one flow. Because the number of spreads can be more than a million, the required amount of memory is still too large to be fit in a small memory. Other approaches such as [6-9] listed all the super spreaders of which spreads are greater than a defined threshold. It skips most of other small ones. Yoon [10] and Li [11] have a decent approach for this problem by adapting Linear Counting [12]. Instead of storing in an independent vector for each spreader, all the data is shared in a large memory by a random mapping process. However, this approach has its

† 준 회 원 : 인하대학교 컴퓨터정보공학과 석사과정
†† 정 회 원 : 인하대학교 컴퓨터정보공학과 교수
††† 종신회원 : 수원대학교 전기공학과 부교수
Manuscript Received : July 17, 2014
First Revision : September 12, 2014; Second Revision : September 29, 2014
Accepted : September 30, 2014
* Corresponding Author : KyungHee Lee(khlee@suwon.ac.kr)

drawback of small estimation range, and thus, sampling approach comes as a new solution.

Sampling means that we examine only a portion of packets instead of all packets, which might result in loss of information, and therefore, probably loss in the accuracy as widely accepted [13]. Nevertheless, by analyzing comparatively algorithms such as Compact Spread Estimator (CSE), Multiple Compact Spread Estimator (MCSE), and Compact Spread Estimator with sampling (CSES), we show that sampling is beneficial for estimation of high spreaders. Consequently, CSES is recommended to be used for high spreaders instead of using complicated and slow MCSE. In this paper, we show that the sampling on multi-set estimation sometimes gives better results in terms of accuracy and estimation range. To prove this, we presented mathematical analysis, conducted experiment with real data, and compared the results of CSE, MCSE, and CSES.

The remainder of this paper is organized into 4 sections. Section 2 describes how the Linear Counting and other work can estimate the cardinality for multi-set. Section 3 analyzes the accuracy along with sampling. Experimental results are presented in Section 4. The conclusion is drawn in Section 5.

2. Background

In this section, we review several estimation algorithms such as Linear Counting, CSE, MCSE, and CSES in order to understand our argument.

2.1 Linear Counting Algorithm

The number of distinct values, called n, can be counted exactly by keeping track of all unique values passing through. However, in many cases, the number of unique values is excessively large. For example, the number of IP addresses is $n = 2^{32}$. That requires impracticably large amount of memory. Whang's approach [12] reduces the storage space using a hash function $H()$ mapping the value into $s = \Theta(n)$ bit size. The algorithm is depicted in Alg. 1. Apparently, the limitation of this method is that the space is linear to n, and thus, it may cause the waste of memory when the load factor is much smaller than the memory size. This is why Linear Counting can be adopted and used for multiple set efficiently.

Algorithm 1 : The Linear Counting algorithm [12]

```
1  Initialize bits M[0] to M[s-1] to 0.
2  for all value v do
3     idx := h(v)
4     M[idx] := 1
5  end for
6  Us := number of 0 bit in M
7  return s · ln(s/Us)
```

2.2 Compact Spread Estimator algorithm (CSE)

For estimating multi-spreader, if each spreader uses its own exclusive memory, then the required space will be too much. Moreover, the space is not used efficiently when the spreader is small. The solution for this problem is that the bit array for each spreader is not stored individually and explicitly but shared over the entire memory space. The group of these bits is called a virtual bit vector V. Each bit in the vector will be mapped to the memory by a hash function. The mapping process is visualized in Fig. 1.

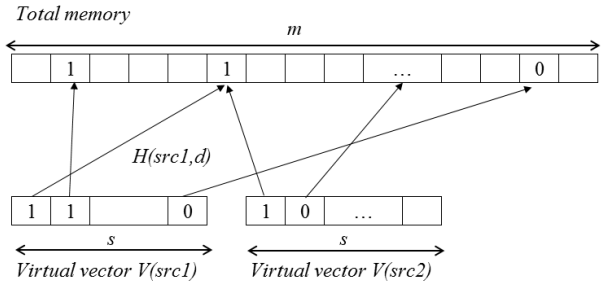


Fig. 1. Mapping of virtual bit V(src) into the memory M

How to store a contact (a source and destination IP pair) in a virtual vector and how to estimate a spread are presented in Alg. 2. In CSE, two hash functions are used, and $Hd()$ calculates the position of bits in a virtual bit vector and $H()$ calculates the position in the memory. m is the total amount of memory in bits, s is the virtual vector size, and Um and Us are the fractions of the number of zero-bits in the memory and in a virtual vector, respectively. The estimation is obtained by $\hat{k_1} = s \circ \ln(\frac{s}{U_s})$ and the noise caused by sharing memory equals to $\hat{k_2} = s \circ \ln(\frac{m}{U_m})$.

The main advantage of CSE is that it can work on

Algorithm 2 : The CSE algorithm [10]

```
1   Initialize bits M[0] to M[m-1] to 0.
2   function CseStore(src,dest)
3       for all src do
4           d := Hd(dest)
5           idx := H(src; d)
6           M[idx] := 1
7       end for
8   end function
9   function CseEstimate(src)
10      Um := number of 0-bits in M
11      Us := number of 0-bits in V (src)
12      return ( - s · ln(Um/m) + s · ln(Us/s))
13  end function
```

small memory while a previous algorithm such as OSM [9] cannot. Since the algorithm uses random bit sharing scheme, the bits of each virtual vector are used efficiently. Moreover, compared to OSM, the error is uniform, and therefore, it can be measured and removed. However, CSE also has problems. When the virtual vector size s becomes too large, the large spreaders are dominant over the total memory M. That makes more noise for other spreaders especially for small spreaders. Consequently, the constraint of s causes the limit in estimation range. For example, when s = 200, the upper bound for estimation is 200.ln(200) ≈ 1060. To overcome this limitation, two ideas are proposed in Section 2.3 and 2.4.

2.3 Multiple Compact Spread Estimator algorithm (MCSE)

To extend the estimation range, Multiple CSE is proposed. It consists of multiple separate CSE estimators with different sampling probabilities. The final result is obtained from the CSE segment with the highest maximum likelihood value.

Particularly, the memory M is divided into g segments M_1 to M_g. Each segment is selected with the sampling probability $p_i = 1/2^i$, i=1, 2, ..., g. The size of M_i, denoted as mi, is the proportional to p_i, and it is defined by $m_i = (p_i / \sum_{i=1}^{g} p_i)m$. The source and destination address pair are hashed and the segment is chosen according to the position of the rightmost 1 bit. In order to estimate the spreader, both estimation and likelihood of every segment are calculated, and the estimation is chosen from the segment that has the maximum likelihood.

Algorithm 3 : The MCSE algorithm [10]

```
1   Initialize bits for all Mi[..]
2   function McseStore(src,dest)
3       for all src do
4           b:= the largest i that the i rightmost bits in
            H(src,dest) are all 0
5           i: = b+1;
6           CseStore(i, src, dest) //Insert to segment i
7       end for
8   end function
9   function McsmEstimate(src)
10  for i=1 to g do
11      L[i] = CalculateLikelihood(i,src) //Calculate likelihood
        value of each segment
12      end for
13  j := ||i : max(L[i])|| //Choose the estimation with highest
    likelihood
14  return CseEstimate(j, src)
15  end function
```

2.4 Compact Spread Estimator with sampling (CSES)

CSES algorithm is identically the same as CSE, where each contact (source, destination) is hashed to a position in the memory and that bit position is set to one. The difference from CSE and MCSE is that CSES uses probabilistic sampling with only one virtual vector while CSE examines all packets and MCSE uses multiple virtual vectors for estimation. It is presented in Alg. 4. Note that the estimation of Efficient Spreader Classification algorithm (ESC) [11] and CSES are identical since $ln(1-p/s) - ln(1-p/m) \simeq (-p/s) - (-p/m) \simeq p/s$. (extended by its Taylor series, and s << m).

Algorithm 4 : The CSE with sampling

```
1   Initialize bits M[0] to M[m-1] to 0.
2   p is the sampling ratio, in a range (0,1)
3   function CsesStore(src, dest)
4       h := H(src,dest). //h is in a range [0,N)
5       if h <= N · p then
6           CseStore(src, dest)
7       end if
8   end function
9   function CsesEstimate(src)
10      return CseEstimate(src) / p
11  end function
```

3. Sampling and Accuracy

The first advantage of sampling in CSES is that it increases the estimation range and reduces the encoding time, but using sampling technique is considered to reduce the accuracy of estimation [13]. In this section, however, we analyze the accuracy of the spread estimation with sampling, and we show that estimation with sampling increases accuracy for high spreaders. We mainly use the results in [10] for analysis CSE estimation equation with sampling where $\hat{k}=-\hat{k_1}+\hat{k_2}=-\frac{s}{p}.\ln(\frac{m}{U_m})+\frac{s}{p}.\ln(\frac{s}{U_s})$.

Using the same technique in [10] with sampling consideration, we derive the mean and variance of $\hat{k_1}$

$$E(\hat{k_1}) \simeq \frac{s}{p}(\frac{np}{m}+\frac{e^{\frac{np}{m}}-\frac{np^2}{m}-1}{2m}) \quad (1)$$

$$Var(\hat{k_1}) \simeq \frac{s^2}{mp^2}(e^{\frac{np}{m}}-\frac{np^2}{m}-1) \quad (2)$$

And the mean and variance of $\hat{k_2}$

$$E(\hat{k_2}) \simeq \frac{s}{p}(\frac{np}{m}+\frac{kp}{s}+\frac{e^{\frac{np}{m}+\frac{kp}{s}}-\frac{kp^2}{s}-1}{2s}) \quad (3)$$

$$Var(\hat{k_2}) \simeq \frac{s}{p^2}(e^{\frac{np}{m}+\frac{kp}{s}}-\frac{kp^2}{s}-1) \quad (4)$$

To obtain $Var(\hat{k})$, we only have to derive $E(\hat{k_1}\hat{k_2})$ since we already obtained $Var(\hat{k_1}), Var(\hat{k_2}), E(\hat{k_1}), E(\hat{k_2})$. Let α = n/m; β = k/s then

$$E(\hat{k_1}\hat{k_2}) \simeq \frac{s^2}{p^2}[\alpha(\alpha+\beta)p^2+\frac{\alpha(e^{(\alpha+\beta)p}-\beta p^2-1)}{2s}+\frac{(\alpha+\beta)(e^{\alpha p}-\alpha p^2-1)}{2m}] \quad (5)$$

Therefore,

$$2[E(\hat{k_1})E(\hat{k_2})-E(\hat{k_1}\hat{k_2})] \simeq 2\frac{s^2}{p^2}[\frac{\alpha(e^{(\alpha+\beta)p}-\beta p^2-1)}{2s}.\frac{(\alpha+\beta)(e^{\alpha p}-\alpha p^2-1)}{2m}] \quad (6)$$

From (2), (4) and (6), we obtain the variance of the estimation as following :

$$Var(\hat{k}) \simeq \frac{s^2}{mp^2}(e^{\alpha p}-\alpha p^2-1)+\frac{s}{p^2}(e^{(\alpha+\beta)p}-\beta p^2-1)+2\frac{s^2}{p^2}[\frac{\alpha(e^{(\alpha+\beta)p}-\beta p^2-1)}{2s}.\frac{(\alpha+\beta)(e^{\alpha p}-\alpha p^2-1)}{2m}] \quad (7)$$

The the standard deviation of the ratio $\hat{k}/k$ is as follows :

$$StdDev(\hat{k}/k)=\frac{\sqrt{Var(\hat{k})}}{k} \quad (8)$$

The standard deviation embodies all the error caused by number of approximations. However, in [10], authors showed that the difference between analytical estimation and the result is minor.

From (8), we can examine the influence of sampling to the accuracy. We notice that α=n/m usually to be small and less than 2 to obtain the effective estimation. Thus, $\frac{s^2}{mp^2}(e^{\frac{np}{m}}-\frac{np^2}{m}-1)$ is too small to have an impact on our result. So $Var(\hat{k})$ get the minimize value when $\frac{s}{p^2}(e^{(\alpha+\beta)p}-\beta p^2-1)$ is minimize.

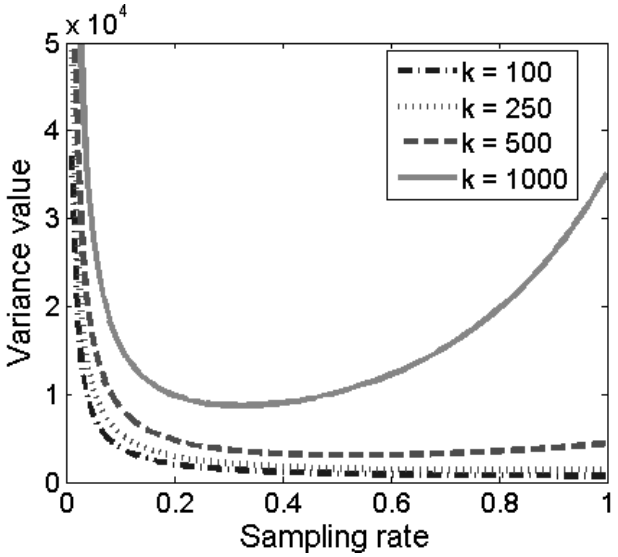


Fig. 2. The plot measures the variance of $Var(\hat{k_2})$, which mainly contribute the overall estimated variance, with different sampling rate and estimate value

Fig. 2 shows the relationship among variance, sampling rate, and spreader size, when n = 8.4 million, m = 8 million, s = 256, k has 4 different values 100; 250; 500; 1000, and the sampling rate changes from 0 to 1. It shows that when we reduce the sampling rate, the accuracy slightly decreases for low spreaders. However, for high spreaders such as with k = 1000, the variance or the standard deviation reaches the minimum when $p < 1$. This means that sampling contributes to better estimation for large range, in this case is the range more than 500. The reason is that without sampling, the information stored in the memory is too much so that the error caused by sharing memory is significantly larger than the error caused by sampling. We will see more clearly this result in Section 4.

4. Experiments

In this section, we evaluate the performance of these algorithms (CSE, MCSE, CSES) in terms of processing time and estimation accuracy. We used 1 hour network traffic trace from CAIDA dataset which hold 8.4M distinct contacts (source IP, destination IP) with nearly 6M distinct spreaders. The range of spreader size is from 0 to 25,722 and the average spread size per source is 13.3. See Fig. 3 for spreader distribution of the dataset in a log scale. The total memory allocated for the measurement is 1MB, which means 1 contact is stored in 1 bit on average.

In the first experiment, we choose the vector size s = 256, since its normal measurement range is from 0 to 500. As shown in Fig. 4a, CSE shows its limitation, that is, the estimation cannot go beyond 1000. To overcome this limitation, we setup 3 more experiments in which the range will increase 4 times. In the first one, the vector size of CSE s is increased to 1,024. Second, we use MCSE with 3 segments. CSES with sampling rate of 0.25 is the last experiment. You can find the estimation result in Fig. 4.

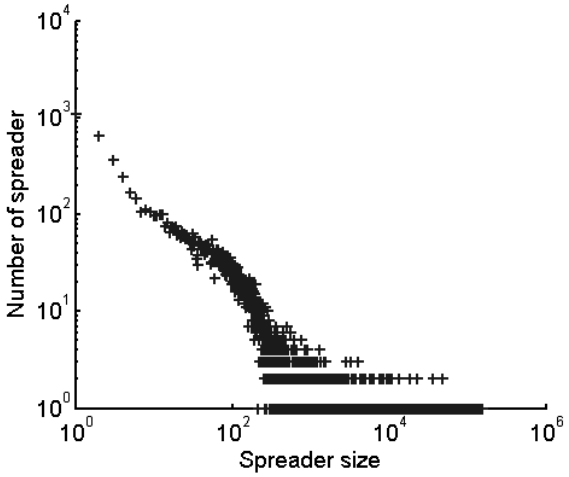
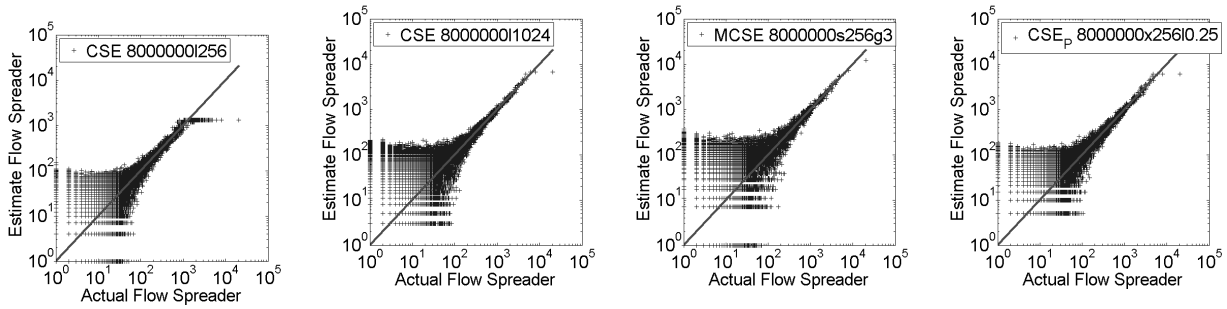


Fig. 3. Traffic distribution of CAIDA dataset. Each point represents the spreader to reach a certain size

As can be seen in Fig. 4a and Fig. 4b, when we increase s the size of virtual vector, the maximum spread estimation of CSE and the accuracy of high spreader increase. Also, high spreader uses more bits, so it causes more noise and reduces the accuracy of low spreaders. And by accessing larger vector size s then the decoding time will be increase. Fig. 4c illustrates MCSE with the number of segments g=3. Spreaders are distributed to



(a) CSE (s=256) (b) CSE (s=1024) (c)MCSE (d) CSES

Fig. 4. Estimation result with CSE (s = 256, 1024), MCSE (s=256,g=3) and CSES (s=256,p=0.25) with 1MB memory. Each point (in cross) stands for each spreader, and y = x line is the reference line. The closer the point is to y = x, the more accurate the estimation is. Note that both axes are in log scale

Table 1. Estimation result with CSE (s = 256, 1024), MCSE (s=256, g=3) and CSES (s=256, p=0.25) with 1MB memory. The fraction of error ($|\hat{k}-k|/k$) is calculated and put into a group according to size. Group i will have the spreaders which have the size in range from 2i-1 to 2i - 1

Range	(A) CSE (s=256)	(B) CSE (s=1024)	(C)MCSE	(D) CSES
1	8.9307	17.5627	16.966	14.2446
2-3	4.6239	8.8216	8.7167	7.1723
4-7	2.2008	3.977	4.1447	3.274
8-15	1.2774	2.1506	2.4086	1.8199
16-31	0.7527	1.2538	1.5486	1.0996
32-63	0.4181	0.7356	1.0148	0.6409
64-127	0.2299	0.4101	0.5842	0.3623
128-255	0.1364	0.2182	0.3094	0.2063
256-511	0.1066	0.1173	0.176	0.1324
512-1023	0.1305	0.0688	0.1174	0.0912
1024-2047	0.1947	0.0499	0.0713	0.07
2048-4095	0.5065	0.0635	0.0774	0.0966

different segments according to its size. However, because the size for each segment is smaller than that of CSE, it comes with reducing the overall accuracy. As a result, the accuracy of MCSE is less than that of CSE (*s=1024*). Moreover, MCSE is significantly slower than CSE due to using maximum likelihood calculation [10]. In other experiment in Fig. 4d, CSES uses the sampling probability p=0.25 instead of increasing the vector size s. Comparing to CSE with s=1024, in CSES, low spreaders are slightly better, while high spreaders are rather worse. However, because of the smaller vector size of CSES (s=256), the estimation time of CSES is faster than that of CSE (s=1024).

In order to look into the accuracy more clearly, we divide spreaders into groups based on its size. Group i will have the spreaders which have the size in range from 2i-1 to 2i-1. Afterwards, we do the descriptive statistics of $|\hat{k}-k|/k$ to show the accuracy of the estimation, the average values in each group are shown in Table 1. From this table, it can be seen that high spreaders (value from 512 to 2047) show better estimation in CSES than in CSE with the same vector size s = 256. Particularly, the accuracy is 1.3 times better in range 512-1023 and significantly increases to 5.24 times in range 2048-4095 when the spreader reaches the upper bound in CSE.

In summary, CSES that adopts sampling for estimation is better compared to CSE or to MCSE in terms of accuracy, and is comparable to CSE in terms of speed, which is presented in Table 2. The result was determined according to Table 1 and the total processing time of main statements in 4 algorithms.

Table 2. The algorithms discussed in section when comparing with CSE(s=256). = dthe base for comparison, - means poor, -- means worse. Similarly, + means better than =, and + + means better than +. H means hash calculation, and M means memory access

Algorithm		CSE	CSE	MCSE	CSES
Cite		[10] III.C	[10] III.C	[10] VI.D	II.2.4
Parameters		s=256	s=1024	s=256;g=3	s=256;p=0.25
Maximum Est. Range		1500	7000	12000	6000
Accuracy	Small range (1-15)	=	-	-	-
	Large range (256-4095)	=	++	+	+
Processing Time	Encoding(Store)	2H + 2M	2H + 2M	3H + 2M	3H + 3M
	Decoding(Estimate)	256*(H + M)	1024*(H + M)	>>256*3*(H + M)	256*(H + M)

5. Conclusion

In conclusion, CSE delivers better result when the spreader value is not high. With the traffic including high spreaders, however, increasing the vector size in CSE makes drawbacks in terms of estimation speed and error. MCSE tries to solve this problem by dividing into several segments and by using maximum likelihood for estimation. High spreaders are stored in all segments, and the estimation is calculated only from one segment. That makes a waste of memory, and the maximum likelihood requires a lot of time. Thus, CSES seems to be the best solution among them in terms of algorithm complexity, processing time, and accuracy when the available amount of memory is the same. It expands the range without increasing the estimation time, and improved accuracy in large range while accuracy in small range is compromised.

References

[1] Z. Bar-Yossef and T. Jayram. “Counting distinct elements in a data stream”, Randomization and Approximation Techniques in Computer Science, pp.1-10, 2002.

[2] A. Chen and J. Cao. “Distinct counting with a self-learning bitmap,” Journal of the American Statistical Association, pp. 1171-1174, Mar., 2011.

[3] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. “HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm”, DMTCS Proceedings, 2008.

[4] P. Flajolet and G. Nigel Martin. “Probabilistic counting algorithms for data base applications,” Journal of Computer and System Sciences, pp.182-209, Oct., 1985.

[5] D. Kane, J. Nelson, and D. Woodruff. “An optimal algorithm for the distinct elements problem,” Proceedings of the twenty-ninth ACM, pp.41-52, 2010.

[6] J. Cao, Y. Jin, A. Chen, T. Bu, and Z.-L. Zhang. “Identifying high cardinality internet hosts,” INFOCOM 2009, IEEE, pp. 810-818, 2009.

[7] C. Estan, G. Varghese, and M. Fisk. “Bitmap algorithms for counting active flows on high speed links”, Proceedings of the 3rd ACM SIGCOMM, pp.925-937, Oct., 2003.

[8] X. Shi, D. Chiu, and J. Lui. “An online framework for catching top spreaders and scanners”, Computer Networks, pp. 1375-1388, June, 2010.

[9] Q. Zhao, J. Xu, and A. Kumar. “Detection of Super Sources and Destinations in High-Speed Networks: Algorithms, Analysis and Evaluation”, IEEE Journal on Selected Areas in Communications, pp.1840-1852, Oct., 2006.

[10] M. Yoon, T. Li, S. Chen, and J. Peir, “Fit a spread estimator in small memory”, INFOCOM 2009, IEEE, 2009.

[11] T. Li, S. Chen, and W. Luo, “Spreader classification based on optimal dynamic bit sharing”, Networking, IEEE/ACM Transactions on, pp.817-830, 2013.

[12] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor, “A linear-time probabilistic counting algorithm for database applications”, ACM Transactions on Database Systems, pp. 208-229, June, 1990.

[13] B. Choi and S. Bhattacharyya, “Observations on cisco sampled net flow”, ACM SIGMETRICS Performance Evaluation pp.18-23, 2005.

DinhNguyen Dao

e-mail : nguyendd@seclab.inha.ac.kr

He received the B.S in Vietnam and is currently pursuing the M.S. degree in computer science and engineering at Inha University, Incheon, Korea, His research interests include cryptography and network measurement..

양 대 헌

e-mail : nyang@inha.ac.kr

He received his BS degree in electronic engineering from KAIST in 1994, the MS and the PhD degrees in computer science from YONSEI Univ. in 1996 and 2000, respectively. From 2000 to 2003, he had worked for Electronics and Telecommunications Research Institute as a senior researcher. Since 2003, he has been with INHA Univ., Korea, where he is currently assistant professor in Graduate School of Information Technology and Telecommunications. His research interests include cryptography, network security including WPAN security, ad hoc network security and wireless LAN security.

이 경 희

e-mail : khlee@suwon.ac.kr

She received B.S., M.S. and Ph.D. degrees in computer science from Yonsei University, Korea. She was a researcher at the LG Soft Company, Korea, from 1993 to 1996. She was a senior member of the engineering staff at the Electronics and Telecommunications Research Institute, Korea, from 2000 to 2005. Since 2005, she has been an assistant professor of Electrical Engineering at University of Suwon, Korea. Her research interests include information security, privacy, biometrics, image processing, artificial intelligence and pattern recognition.