http://dx.doi.org/10.3745/KTCCS.2013.2.10.421

Flow Labeling Method for Realtime Detection of Heavy Traffic Sources

KyungHee Lee† · DaeHun Nyang††

ABSTRACT

As a greater amount of traffic have been generated on the Internet, it becomes more important to know the size of each flow. Many research studies have been conducted on the traffic measurement, and mostly they have focused on how to increase the measurement accuracy with a limited amount of memory. In this paper, we propose an explicit flow labeling technique that can be used to find out the names of the top flows and to increase the counting upper bound of the existing scheme. The labeling technique is applied to CSM (Counter Sharing Method), the most recent traffic measurement algorithm, and the performance is evaluated using the CAIDA dataset.

Keywords : Traffic Measurement, Flow labeling, Counter Sharing Method

대량 트래픽 전송자의 실시간 탐지를 위한 플로우 라벨링 방법

이 경 희† · 양 대 헌††

요 약

인터넷을 통해 전송되는 트래픽의 양이 점점 더 많아지고 있고, 이에 따라 트래픽의 양을 알아내는 것이 중요해지고 있다. 트래픽을 측정하는 기법에 대한 많은 연구가 있었으며, 주로 적은 양의 메모리를 사용해서 측정의 정확도를 높이는 방향으로 연구가 진행되고 있다. 이 논문에서는 기존 기법들에서는 제공하지 않는 플로우 라벨링 기법을 제안한다. 이 기법을 통해서 관리자는 특정 값 이상의 트래픽을 생성한 플로우의 라벨을 실시간으로 알아낼 수 있으며, 기존의 기법이 가지는 카운팅의 상한선을 확장시킬 수 있다. 가장 최근에 발표된 CSM (Counter Sharing Method)에 이 기법을 적용하고, CAIDA 데이터셋을 이용해서 성능을 분석해 본다.

키워드 : 트래픽 측정, 플로우 라벨링, 카운터 공유 방법

1. 서 론

인터넷을 통해 전송되는 트래픽의 양은 1990년에 한 달에 0.001페타바이트에서 2011년에 27,483 페타바이트로 급격히 증가했다[1]. 또한, 시스코사는 2017년에는 한 달에 120,643 페타바이트가 생성될 것으로 예측했다[2]. 데이터의 양이 증가함에 따라, 트래픽을 정확히 측정하는 것이 중요해 졌지만, 동시에 매우 어려워지기도 했다. 이는 주로 초고속의 라우터에서 측정해야하는 알고리즘의 특성상 속도와 메모리의 양에 매우 민감하기 때문이다. 예를 들어, OC-192 링크에서 10 Gbps이상의 속도로 전송되는 패킷을 처리해야 하므로, 알고리즘의 속도가 매우 중요해졌다. 고속 처리를 위해서 일반적인 DRAM이 아닌 고속의 SRAM을 이용해서 트래픽을 측정하는 것이 최근의 연구 방향이다. 하지만, SRAM은 속도가 매우 빠른 대신, 가격이 매우 비싸 충분히 많은 메모리를 사용할 수 없으므로, 최소한의 메모리만을 이용해서 트래픽 측정의 정밀도를 높이는 것이 현재 이 분야에서 가장 중요한 주제이다. 하지만, 대부분의 기법들은 측정을 위한 메모리에 각 플로우의 크기만을 저장할 뿐, 실제로 그 플로우의 라벨 (송신자 IP주소, 또는 송신자/수신자 IP주소 쌍)을 저장하지는 않는다. 따라서 사후에 모든 플로우 라벨에 대해서 그 플로우의 크기를 알아낸 후, 크기가 너무 큰 플로우에 대해서 정책에 따라 처리하는 방식을 취하고 있다. 하지만, 이는 실시간으로 매우 큰 플로우에 대해 대응하지 못한다는 단점이 있다. 매우 큰 크기를 가지는 플로우는 스패머 (Spammer), 대량의 데이터를 업로드 하거나 다운로드하는 사용자, 또는 DoS 공격자일 가능성이 있고, 이들을 실시간으로 탐지하는 것이 메모리를 덜 사용하는 것보다 더 중요한 경우가 있다.

※ 이 논문은 수원대학교의 지원에 의하여 연구되었음.
† 종신회원 : 수원대학교 전기공학과 부교수
†† 정 회 원 : 인하대학교 컴퓨터정보공학과 부교수
논문접수 : 2013년 8월 16일
수 정 일 : 1차 2013년 9월 25일
심사완료 : 2013년 9월 26일
* Corresponding Author : KyungHee Lee(khlee@suwon.ac.kr)

이 논문에서는 20% 정도의 메모리를 더 사용함으로써, 크기가 큰 플로우들을 실시간으로 탐지하고, 기존의 기법이 가지는 카운팅의 상한선을 확장시키는 알고리즘을 제안한다.

2. Randomized Counter Sharing Method

이 논문에서 제시하는 기법은 Tao Li 등이 2012년 발표한 randomized counter sharing 기법 (CSM)에 새로운 데이터 구조를 추가하고, 인코딩 알고리즘을 수정함으로써 만들어진다[17].

CSM은 m개의 b비트 카운터로 이루어진 메모리에 모든 플로우의 크기를 저장한다. b비트 카운터 하나는 2^b-1개의 패킷을 셀 수 있다. 그리고 각 플로우에는 l개의 임의로 선택된 카운터가 할당 된다. 보통 플로우의 개수가 m보다 크므로 많은 경우, 하나의 카운터를 여러 개의 플로우가 공유하게 된다. 이제 패킷이 하나 도착했을 때, 송신자 IP주소에 따라 l개의 카운터 중 하나가 임의로 선택되고 그 카운터의 값을 1 증가 시킨다. 이를 인코딩이라 부르고, 이렇게 해서 해당 플로우의 패킷의 수가 세어진다. 만약, 어떤 플로우의 크기를 알아내고 싶다면, 그 플로우의 송신자 IP주소를 이용해서 l개의 카운터를 찾은 후, 그 카운터들이 가지고 있는 값들을 모두 더해서 플로우의 크기를 계산한다. 이때, 카운터가 가지고 있는 값 모두가 해당 플로우가 증가시킨 것이 아니므로, 에러를 보정해 주어야 한다. 따라서, 최종 플로우의 크기는 다음과 같이 계산한다.

$$\hat{s}=\sum_{i=0}^{l-1} C[H(f,i)]-l\times\frac{n}{m}$$

여기서 *H(f,i)*는 플로우 라벨 f와 i=1...l을 입력으로 해서 계산한 해쉬 값이고, C[]는 카운터를 의미한다. 그리고 m은 카운터의 개수, n은 전체 패킷의 개수이다. 이 과정을 디코딩이라 부른다.

3. CSM-HT: 플로우 라벨링 가능한 CSM

이 절에서는, 이 논문에서 제시하는 플로우 라벨링이 가능한 CSM-HT (CSM with Hash Table) 기법을 데이터 구조, 인코딩, 디코딩으로 나누어 설명한다.

3.1 데이터 구조

CSM과 마찬가지로 m개의 b비트 카운터로 이루어진 메모리에 플로우의 크기를 저장한다. 그리고, 해쉬 테이블이 추가된다. 해쉬 테이블은 (플로우 라벨, 플로우 크기) 쌍이 하나의 엔트리이며, 이 엔트리의 개수는 h이다. 플로우 라벨은 송신자 IP주소이므로 32비트, 그리고 플로우의 크기는 최대 플로우가 가지는 패킷의 수에 따라 결정된다. 예를 들어, 20비트로 설정하면 $(2^{20}-1)\times1000\simeq1$기가바이트까지의 플로우를 측정할 수 있다. h는 $l(2^b-1)$보다 큰 플로우의 개수의 2배로 설정한다. 이는 이 알고리즘을 설치하려는 사이트의 전형적인 트래픽 분포를 이용해서 추정할 수 있다. 2배로 설정하는 것은 해쉬 테이블의 로드 팩터 (load factor)가 0.5일 때, 오픈 어드레싱을 이용하면 키워드를 찾는데 드는 평균 시도 횟수가 2보다 작아지기 때문이다. 즉, n개의 엔트리가 m개의 버킷을 가지는 해쉬 테이블에 등록되어 있는 경우 평균 시도 횟수는 $1/(1-n/m)$보다 작다.

3.2 인코딩

CSM과 마찬가지로 l개의 b비트 카운터 중 임의로 하나를 고른 후, 1을 증가시킨다. 이때, 만약 그 카운터가 꽉 찼다면, 그 카운터의 값을 해쉬 테이블의 해당 플로우 버킷에 누적시킨다. Fig. 1에서 Inc(f, c) 함수는 해쉬 테이블의 f에 해당하는 버킷에 c를 누적한다. 또한, 만약 CSM 카운터가 이미 꽉찬 상태라면, 해쉬 테이블에 1을 증가시킨다. 이렇게 하므로써, CSM의 개별 카운터가 가지는 카운팅의 상한선인 2^b-1을 넘는 크기의 플로우를 셀 수 있게 된다.

Algorithm 1: Encoding(f)
1: $i \leftarrow$ rand()
2: $pos \leftarrow H(f,i)$
3: **if** $C[pos]=2^b-1$
4: Inc(f,1)
5: **else**
6: $C[pos] \leftarrow C[pos]+1$
7: **if** $C[pos]=2^b-1$ **then**
8: Inc(f,$2^b-1-n/m$)

Fig. 1. Encoding

3.3 디코딩

해당 플로우의 크기는 해쉬 테이블에 누적된 값과 CSM에 있는 l개의 카운터가 가지고 있는 값을 더함으로써 알 수 있다. Fig. 2에서 GetCount(f)는 해쉬 테이블의 f에 해당하는 버킷에서 카운트 값을 리턴한다. 이때, CSM 카운터가 꽉 찼다면, 이미 해쉬 테이블에서 누적이 이루어지고 있으므로, 더할 필요가 없다.

Algorithm 2: Estimate(f)
1: $est \leftarrow$ GetCount(f)
2: **for** $i=0\ldots l-1$ **do**
3: **if** $C[H(f,i)] \neq 2^b-1$ **then**
4: $est \leftarrow est+C[H(f,i)]-n/m$
5: **end for**
6: **return** *est*

Fig. 2. Decoding

3.4 Threshold 카운팅

특정 크기 이상의 플로우의 목록을 알고 싶다면 해쉬 테이블을 스캔해서 그 크기 이상인 플로우의 라벨을 출력한다. Fig. 3에서 GetC(k), GetF(k)는 해쉬 테이블에서 k번째 버킷의 카운트 값, 플로우 라벨을 각각 리턴한다. 이 threshold 카운팅은 다른 트래픽 측정 알고리즘에서는 제공되지 않는 기능이다.

Algorithm 3: List(t)

```
1:   for k = 0...h-1 do
2:     est ← GetC(k)
3:     f ← GetF(k)
4:     for i = 0...l-1 do
5:       if C[H(f,i)] ≠ 2^b - 1 then
6:         est ← est + C[H(f,i)] - n/m
7:       end for
8:       if est ≥ t then
9:         print f : est
10:      endif
11:    end for
```

Fig. 3. Threshold counting

4. 성능 분석 및 실험 결과

4.1 추가 계산량

인코딩에 추가로 소요되는 연산은 카운터가 꽉 찼을 때, 해쉬 테이블에 누적 시키는 것인데, 앞서 언급한 대로 해쉬 테이블의 load factor를 0.5 정도로 유지하면, 평균 2번 이내에 찾을 수 있으므로, 전체 평균 해쉬 연산 횟수는 1 + 0.5 * p이다. 이때, p는 CSM 카운터가 꽉 차 있을 확률이고, 실험에 사용한 데이터셋에서는 전체 패킷 16M개 중 125K개의 패킷이 카운터가 꽉찼을 때 도착했으므로, 평균 해쉬 연산 수는 1+0.5*0.008 = 1.004 이다. 디코딩에서는 두 번 미만의 해쉬 연산이 추가 되지만, l이 100~1000인 것을 고려하면 추가 되는 연산량이 크지 않다.

4.2 추가되는 메모리의 양

이 절에서는 파라미터에 따른 해쉬 테이블의 메모리 사용량과 추정의 정확도 및 상한선에 대해 알아본다. 실험을 위해 CAIDA 데이터셋을 이용했다[5]. 이 데이터셋은 라우터에서 실제 트래픽을 수집해 놓은 것으로, 프라이버시를 위한 익명처리가 되어있다. 이 실험을 위해서 사용한 데이터는 1분 동안 측정한 것으로, 전체 패킷의 수가 16M개, 플로우의 수가 690K개, 그리고 가장 큰 플로우는 120K개의 패킷을 가진다. 이 데이터셋의 트래픽 분포는 Fig. 4에서 볼 수 있다. 각 CSM 카운터의 크기는 메모리 크기별로 1MB, 2MB, 4MB이 주어졌을 때, 각각 8비트, 7비트, 6비트로 설정했고, 모든 경우에 대해 l은 100-1,000 사이에서 100씩 변화 시켜가며 실험 했다. 해쉬 테이블은 (32비트 플로우라벨, 17비트 플로우 크기)쌍으로 구성했다.

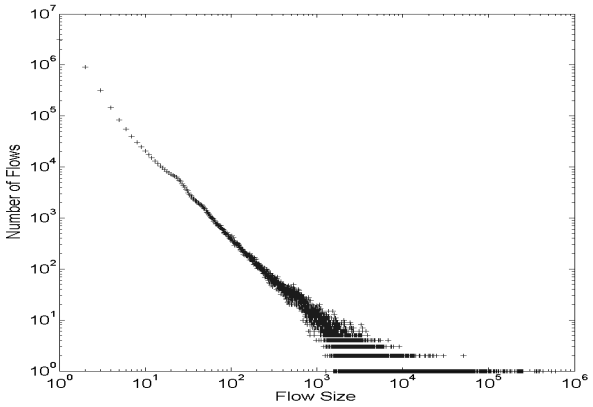

Fig. 4. Traffic Distribution

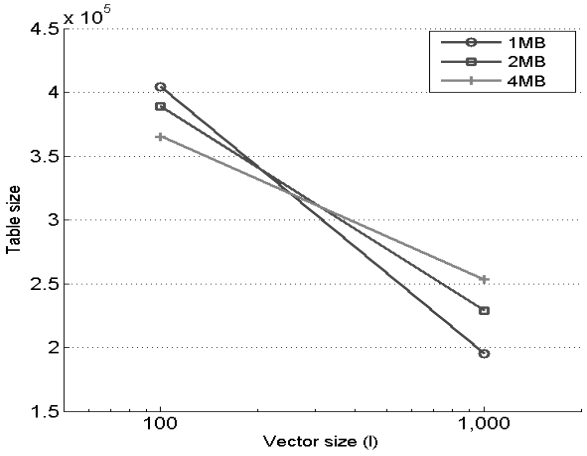

Fig. 5. Table size in bytes

Fig. 5는 l의 크기를 100에서 1000까지 100씩 변화시켜가면서 테이블의 크기 변화 (바이트)를 나타낸 것이다. 카운터의 개수 l이 클수록 테이블의 크기가 작아지는 것을 볼 수 있다. 테이블의 크기는 최소 200KB에서 최대 400KB 정도로 그리 크지 않음을 알 수 있다.

Fig. 6, 7, 8은 l이 100일 때와 1000일 때 메모리 크기 별 (1MB, 2MB, 4MB) 추정의 정확도를 나타낸 그래프이다. 로그 스케일로 그려졌으며, 가로축은 실제 플로우의 크기 (패킷의 개수), 세로축은 추정치 (패킷의 개수)를 보여준다. 이 그래프에서 하나의 플로우는 하나의 점으로 표현된다. 따라서, 추정이 정확할수록 y=x 직선에 근접해야 한다. CSM은 카운팅의 상한선이 $l\times(2^b-1)$이므로, 최소 $100*(2^6-1)$=6.4K에서 최대 $1000*(2^8-1)$=256K이다. 따라서, l=100일 때, 6.4K까지, l=1,000일 때, 256K이상을 셀 수 없지만, 그림에서 보이는 것처럼 l이 클수록 추정 에러도 커지게 된다. 반면에, CSM-HT는 그림에서 보이는 것처럼 해쉬 테이블의 누적 카운터에 할당한 비트 수 (b_h)를 이용해서 추가로 셀 수 있으므로, 최대 플로우 크기가 CSM보다 $2^{b_h}-1$만큼을 더 셀 수 있다.

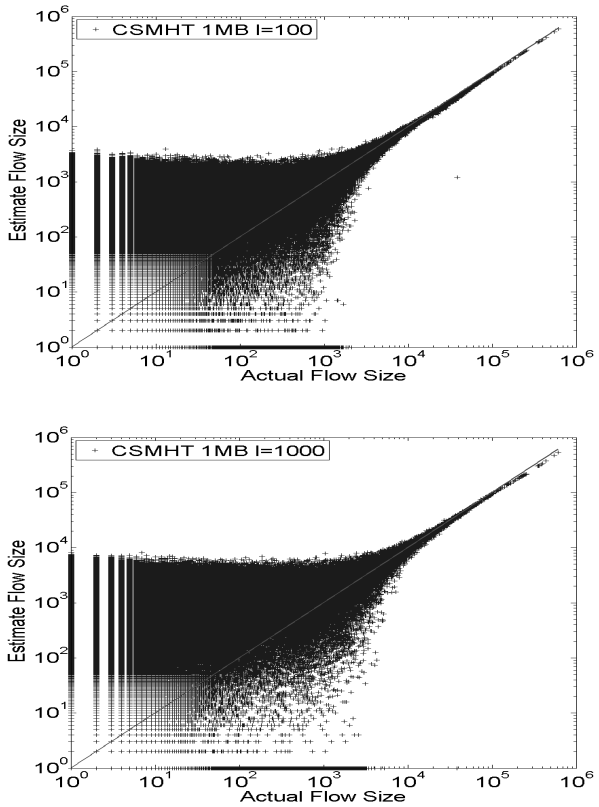

Fig. 6. CSM-HT Estimation (1MB)

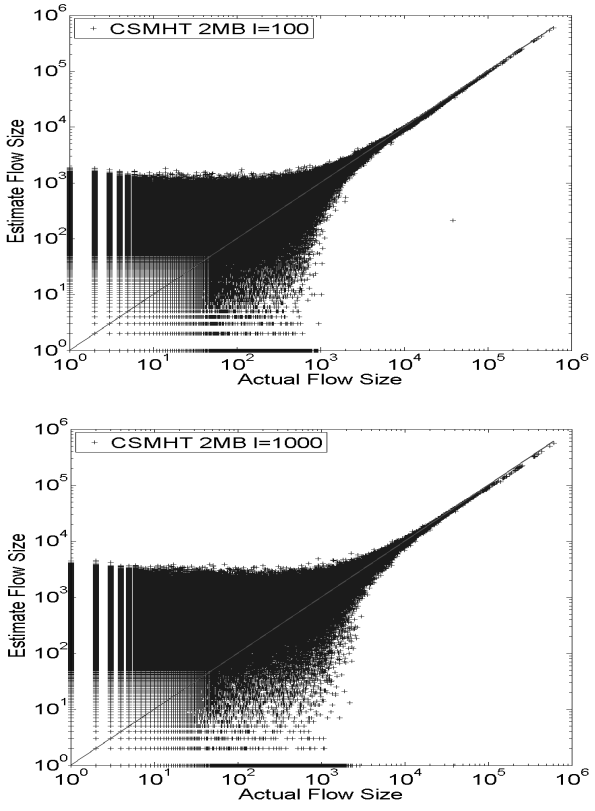

Fig. 7. CSM-HT Estimation (2MB)

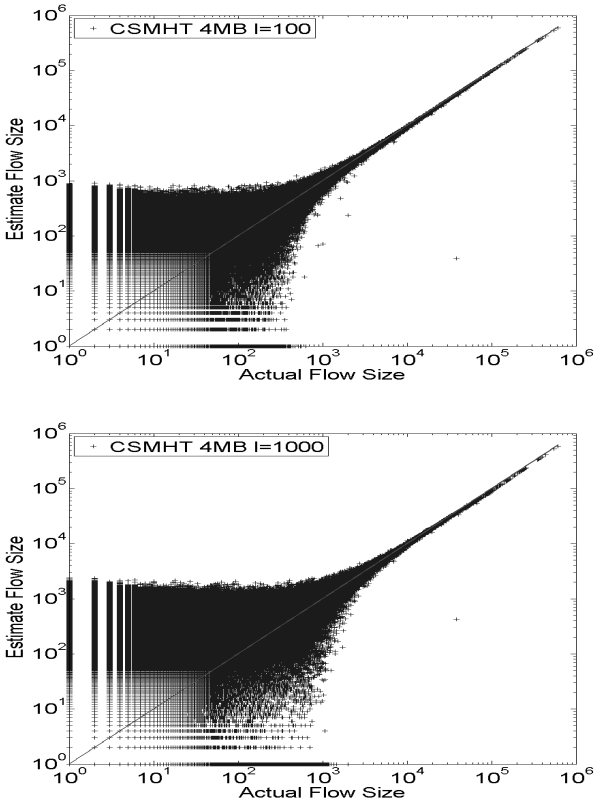

Fig. 8. CSM-HT Estimation (4MB)

5. 관련 연구

트래픽을 측정하는 방법에 대한 연구는 패킷을 샘플링해서 추정했던 Cisco사의 Netflow와 Sflow가 시작했다[3], [4]. 이후에 Estan등이 큰 플로우만을 식별하는 방법을 제안했고, 후속 연구가 뒤를 이었다[8], [10], [12], [13]. 최근에는 블룸필터를 이용해서 개별 플로우의 크기를 측정하는 방법에 대한 연구가 소개되었으며[9], [11], 이 방법들이 점차 개선되어갔다. 대표적인 것으로 Counter Braids[14], PMC[16] 등이 있다. 그리고 가장 최근에 CSM이 소개되었다[17]. 연구의 방향은 최소한의 메모리와 연산만을 사용해서, 가장 정확하게 모든 플로우의 크기를 추정하는 것으로 향하고 있다. 이 논문에서 제시한 방법은 이 연구 방향을 거스르지 않으면서, 기존의 연구에서는 지원하지 않지만 필수적인 기능의 하나인 큰 플로우의 실시간 식별 기능을 추가했다는데 의의가 있다.

6. 결 론

이 논문에서는 트래픽 측정 알고리즘의 하나인 CSM에 실시간으로 대량의 트래픽을 발생시키는 플로우의 라벨을 알아낼 수 있는 기능을 추가하는 방법을 제시했다. 이 기능은 스패머, 대량의 트래픽 유발자, DoS 공격자를 실시간으

로 탐지하는데 유용하게 사용될 수 있으며, 이를 위해 추가되는 비용이 크지 않음을 실제 데이터를 이용한 실험을 통해서 보였다. 또한, CSM의 카운팅 상한선을 늘리는 부수적인 효과도 얻게 되었다.

참 고 문 헌

[1] Wikipedia, "Internet Traffic," http://en.wikipedia.org/wiki/Internet_traffic

[2] Cisco whitepaper: "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[3] Cisco, "NetFlow Systems," http://www.cisco.com/en/US/products/ps6601/ products_ios_protocol_group_home.html

[4] InMon Corp, "sFlow Accuracy & Billing," http://www.sflow.org/sFlowOverview.pdf

[5] "The Cooperative Association for Internet Data Analysis," http://www.caida.org

[6] P. Flajolet and G. Nigel Martin, "Probabilistic Counting Algorithms for Database Applications," Journal of Computer System Science, Vol.31, pp 182-209, 1985.

[7] K.Y. Whang, B. Vander-Zanden and H. Taylor, "A linear-time probabilistic counting algorithm for database applications," ACM Transactions on Database Systems, Vol.15, 2, pp. 208-229, 1990.

[8] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," Proc. of ACM SIGCOMM, August, 2002.

[9] S. Cohen and Y. Matias, "Spectral bloom filters," Proc. ACM SIGMOD Conference on Management of Data, 2003.

[10] R. Karp, S. Shenker, and C. Papadimitriou, "A Simple Algorithm for Finding Frequent Elements in Streams and Bags," ACM Transactions on Database Systems, Vol.28, No.1, pp.51-55, 2003.

[11] A. Kumar, J. JimXu, and J. Wang, "Space-Code Bloom Filter for Efficient Per-Flow Trafc Measurement," Proc. of IEEE INFOCOM, March, 2004.

[12] N. Kamiyama and T. Mori, "Simple and Accurate Identification of High- rate Flows by Packet Sampling," Proc. of IEEE INFOCOM, April, 2006.

[13] X. Dimitropoulos, P. Hurley, and A. Kind, "Probabilistic Lossy Counting: An Efficient Algorithm for Finding Heavy Hitters," ACM SIGCOMM Computer Communication Review, Vol.38, No.1, pp.7-16, 2008.

[14] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter Braids: A Novel Counter Architecture for Per-Flow Measurement," Proc. of ACM SIGMETRICS, June, 2008.

[15] Y. Lu and B. Prabhakar, "Robust Counting Via Counter Braids: An Error- Resilient Network Measurement Architecture," Proc. of IEEE INFOCOM, April, 2009.

[16] P. Lieven and B. Scheuermann, "High-Speed Per-Flow Trafc Measurement with Probabilistic Multiplicity Counting," Proc. of IEEE INFOCOM, April, 2010.

[17] T. Li, S. Chen and Y. Ling, "Fast and Compact Per-Flow Traffic Measurement through Randomized Counter Sharing," Proc. of IEEE INFOCOM, April, 2011.

이 경 희

e-mail : khlee@suwon.ac.kr

1993년 연세대학교 컴퓨터과학과(학사)

1998년 연세대학교 컴퓨터과학과(석사)

2004년 연세대학교 컴퓨터과학과(박사)

1993년~1996년 LG소프트(주) 연구원

2000년~2005년 한국전자통신연구원 정보보호연구본부 선임연구원

2005년~현 재 수원대학교 전기공학과 부교수

관심분야 : 정보보호, 바이오인식, 컴퓨터비전, 패턴인식

양 대 헌

e-mail : nyang@inha.ac.kr

1994년 한국과학기술원 과학기술대학 전기 및 전자공학과(학사)

1996년 연세대학교 컴퓨터과학과(석사)

2000년 연세대학교 컴퓨터과학과(박사)

2000년~2003년 한국전자통신연구원 정보보호연구본부 선임연구원

2003년~현 재 인하대학교 컴퓨터정보공학과 부교수

관심분야 : 암호이론, 암호프로토콜, 인증프로토콜, 무선인터넷 보안, 네트워크 보안