한국시뮬레이션학회 논문지
Vol. 18, No. 4, pp. 207-217 (2009. 12)

# Address Prefix에 기반한 우선 순위 서비스를 이용한 DDoS 방어

김경학[1] · 이태진[2] · 남승엽[1†]

# DDoS Defense using Address Prefix-based Priority Service

**Jinghe Jin · Taijin Lee · Seung Yeob Nam**

**ABSTRACT**

We propose a two-stage Distributed Denial of Service (DDoS) defense system, which can protect a given subnet by serving existing flows and new flows with a different priority based on IP history information. Denial of Service (DoS) usually occurs when the resource of a network node or link is limited and the demand of the users for that resource exceeds the capacity. The objective of the proposed defense system is to provide continued service to existing flows even in the presence of DDoS attacks. The proposed scheme can protect existing connections effectively with a smaller memory size by reducing the monitored IP address set through sampling and per-prefix history management. We evaluate the performance of the proposed scheme through simulation.

**Key words** : DDoS, DoS, Bloom filter, Priority service

**요 약**

본 논문에서는 IP 히스토리 정보를 이용하여 기존의 연결된 flow와 새로 도착한 flow를 다른 우선순위로 서비스하여 특정 서브넷을 DDoS(Distributed Denial of Service) 공격으로부터 보호하는 방어 시스템을 제안한다. DDoS 공격은 일반적으로 네트워크 노드 혹은 링크의 리소스가 제한되어 있으며 네트워크 리소스에 대한 사용자의 수요가 실제 네트워크 노드 혹은 링크의 용량을 초과하는 경우에 발생하게 된다. 제안된 방어 시스템은 두 단계로 나뉘어서 작동하게 되는데, 첫 번째 단계에서는 접속을 시도하는 외부 IP에 대해 샘플링기법을 사용하여 모니터링 하는 IP주소의 개수를 줄이고, 두 번째 단계에서는 정상적인 IP의 목록을 관리하고 DDoS 공격이 발생하였을 때 IP 목록에 기반해 기존의 연결된 flow와 새로 도착한 flow를 구별함으로써 기존의 연결된 flow에 대해서 지속적이고 정상적인 서비스를 제공한다. 제안된 DDoS 방어 시스템의 성능은 시뮬레이션을 통해 평가한다.

**주요어** : DDoS, DoS, 블룸필터, 우선순위 서비스

## 1. 서 론

DoS(Denial of Service) 공격은 네트워크나 서버에 의해 제공되는 정상적인 서비스를 악의적으로 중단하는 것을 목적으로 하고 있다. 최근 DoS 공격은 봇넷(Botnet)[2]을 통하여 수천대 이상의 좀비(Zombie) 컴퓨터를 동원함으로써 규모 및 피해가 점점 더 커지는 양상을 보이고 있다. 지금까지 비록 많은 DDoS(Distributed Denial of Service) 방어 방법들이 제안되었지만[10], DDoS 공격에 대한 방어는 여전히 어려운 문제로 남아 있다. 특히 최근에는 공격 트래픽과 정상적인 트래픽간의 차이가 점점 줄어들고 있어 DDoS 공격 방어가 더욱 어려워지고 있다.

소수의 컴퓨터가 DoS공격에 참여하는 경우 연결된 각 flow의 상태를 관리하지 않는 [3,6]에서 제안한 방법으로 공격에 참여한 컴퓨터들의 IP 주소를 검출할 수도 있다. 반대로 많은 양의 컴퓨터가 특정 웹사이트에 DDoS 공격을 실행할 경우 대다수 공격에 참여하는 컴퓨터들의 IP 주소는 평소 웹사이트에 자주 접속하는 정상적인 사용자

* 본 연구는 지식경제부 및 한국산업기술평가관리원의 IT산업원천기술개발사업의 일환으로 수행하였음.[2009-S-038-01, 분산서비스거부공격 대응기술 개발]
2009년 11월 7일 접수, 2009년 12월 9일 채택
[1] 영남대학교 전자정보공학부
[2] 한국인터넷진흥원 정보보호본부 보호기술팀
주 저 자 : 김경학
교신저자 : 남승엽
E-mail; synam@ynu.ac.kr

의 IP와는 달리 새로운 IP라는 연구결과에 의하여 공격자의 IP 주소를 검출할 수도 있다[5]. Peng은 위의 연구결과에 기반하여 특정 웹사이트에 접속하는 IP 주소의 이력(history)을 모니터링 함으로써 DDoS 공격을 탐지하는 메커니즘을 제안하였다[8,9]. 그렇지만, [8][9]의 연구결과는 DDoS 방어 시스템의 메모리 사용량을 고려하지 않았다. 1Gbps 이상의 고속 환경에서는 고속 메모리가 필요하고, SRAM등과 같은 고속 메모리는 그 크기가 제한되는 경우가 많기 때문에 본 논문에서는 제한된 메모리 환경에서 구현 가능한 DDoS 방어 방법론에 대해서 연구한다. 특히 본 논문에서는 특정 서브넷을 보호하는데 초점을 두며 개개의 flow 정보를 모두 관리하지 않고, 많은 개수의 외부 IP들로부터 접속이 되는 특정 host만을 모니터링 하는 방법을 제안한다. DDoS 공격이 발생하는 경우 공격 대상은 많은 개수의 외부 IP들로부터 접속되므로 개개의 외부 IP들을 일일이 관측하지 않고 샘플링을 통해 작은 개수의 IP 주소를 관측하여도 공격대상을 쉽게 확인할 수 있으며, 잠재적인 공격대상에 대하여 정상적인 접속 IP 주소의 목록인 white list를 단시간에 구성하고 DoS 공격이 시작되면 white list에 있는 외부 IP주소들을 새로운 IP 주소에 비해 우선적으로 처리함으로써 기존 flow에 대해 정상적인 서비스의 연속적인 제공을 가능하게 한다. 또한 샘플링을 통하여 작은 개수의 IP만 관측함으로써 기존의 방식에 비해 작은 메모리 크기로 구현이 가능하다.

## 2. 관련 연구

Peng[8,9]이 제안한 방식 이외에도 최근에는 다양한 DDoS 방어 방법론들이 제안되고 있다. 그 중에 대표적인 방법론으로 다음과 같은 방식들이 있다. 첫 번째 TVA (Traffic Validation Architecture)[12]는 트래픽 발생 노드가 원하는 목적지로 트래픽을 보내기 이전에 목적지와 중간 라우터들로부터 먼저 허락을 받도록 하는 구조이다. 라우터들은 허락을 하는 경우 특별한 시그너처를 패킷에 집어 넣게 되고, 나중에 그러한 시그너처의 조합에 해당하는 capability라는 정보를 목적지가 다시 트래픽 발생지 쪽으로 전달해 주게 된다. 그러면, 트래픽 발생지에서는 받은 capability 정보를 해당되는 목적지로 가는 모든 패킷에 넣어서 보내게 되고, 중간 라우터는 자신이 입력한 시그너처를 확인해서 허락을 받은 패킷만 전달하고 그렇지 않은 패킷은 폐기하는 방식으로 동작하게 된다.

이와 유사하게 DoS 트래픽을 공격지 가까이에서 filtering 할 수 있는 또 다른 방식으로 [7]에서 제안된 StopIt이라는 방식이 있다. 이 방식은 DoS 공격을 받는 host가 공격 트래픽 중단 요청을 보내면 그러한 공격 중단 요청이 공격 노드 자체 또는 공격 노드에 가장 가까운 라우터에 전달되어 공격트래픽을 가능한 공격지와 가장 가까운 곳에서부터 폐기하는 방식이다. 이러한 방식들은 이론적으로는 DoS 공격이 발생하는 경우에도 트래픽 발생지에 가까운 곳에서부터 공격 패킷을 폐기할 수 있기 때문에, 상당히 효율적일 수 있는 방법이지만, 실제로는 트래픽 생성지와 목적지 뿐만 아니라 공격지 subnet 또는 중간 라우터들까지도 upgrade되어야만 제대로 동작할 수 있다. 본 논문에서는 이와 같이 네트워크 라우터의 전반적인 upgrade를 요구하지 않고서도, 즉 infrastructure 업그레이드를 최소화 하면서도 원하는 subnet을 효율적으로 방어할 수 있는 방법의 개발에 초점을 맞춘다.

## 3. IP 필터링 기반 DDoS 방어 메커니즘

본 논문에서는 [5]의 연구결과에 의거하여 정상적인 flow와 공격 flow를 구별하기 위하여 평소 정상적으로 접속하는 사용자들의 IP 리스트(LEA: List of Existing IP Addresses)를 관리한다.

제안된 DDoS 방어 메커니즘은 두 단계로 구성되며 첫 번째 단계에서는 잠재적으로 DoS 공격목표가 될 수 있는 내부 IP를 검출하고, 두 번째 단계에서는 검출된 IP에 대해 white list, 즉, LEA를 구축하고 DoS 공격이 본격적으로 시작되면 LEA에 속하는 IP와 그렇지 않은 IP를 차별적으로 서비스 한다. DoS 공격이 시작될 때 LEA가 공격 IP에 의해서 쉽게 점유되는 현상을 방지하기 위하여 IP prefix별로 일종의 신용도를 per-prefix credit table에 관리한다. 잠재적인 DoS 대상에 대한 새로운 source IP가 나타나는 경우에는 per-prefix credit table에서 계산된 확률에 따라 LEA에 등록을 하게 된다. 각 단계별 자세한 방어기술과 알고리즘은 아래와 같다.

### 3.1 첫 단계: 잠재적인 공격목표의 검출

해당 서브넷에서 잠재적인 공격목표를 찾기 위해서 첫 번째 단계에서는 특정 서브넷에 접속하는 flow를 송신측과 수신측 IP 쌍 즉(SrcIP, DstIP) 쌍으로 샘플링 한다. (SrcIP, DstIP) 쌍을 샘플링 하는 확률을 $p_s$라고 하자. $h_p$는 (SrcIP, DstIP) 쌍에 대해서 [0, 1)사이의 실수를 할당하는 uniform random 해시 함수라고 정의하면 접속 flow의 source 와 destination IP가 각각 $s$, $d$일 경우 만약 $h_p(s, d) < p_s$이면 IP쌍 ($s$, $d$)는 샘플링 되게 된다. 본 논

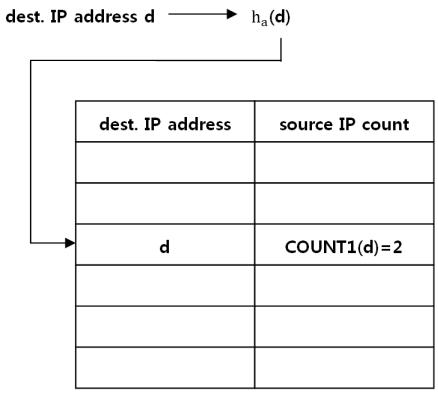


**그림 1.** 연결 상태 관리 테이블 $T_1$

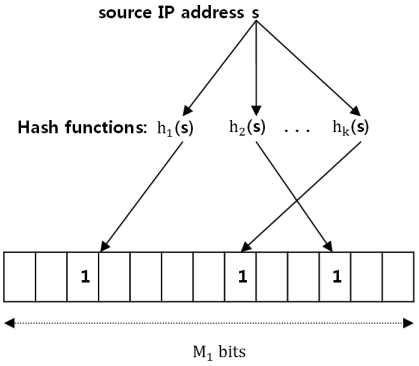


**그림 2.** source IP 추적을 위한 블룸필터 $B_1$

문에서는 샘플링 확률 $p_s$를 0.1로 설정하였다. 또한 특정 서브넷에 속한 IP 별로 flow들의 연결 상태를 관리하기 위하여 그림 1과 같이 연결 상태 관리 테이블 $T_1$을 사용하고 샘플링 된 source IP를 추적하기 위하여 그림 2와 같이 블룸필터 $B_1$을 사용한다. 해시 테이블 $T_1$은 일정한 시간 간격 $I_s$동안 서브넷에 속한 특정 IP에 접속하는 서로 다른 외부 IP의 개수를 관리한다. 즉, $I_s$의 주기로 $T_1$을 관리하게 되고 이때 $I_s$를 short time interval이라 한다. 테이블 $T_1$의 $COUNT1(d)$는 내부 IP $d$로 접속하는 샘플링 된 외부 IP의 개수를 카운트하고 중복된 source IP를 카운트하는 것을 방지하기 위하여 그림 2와 같이 길이가 $M_1$-bit이고 $k$개의 해시 함수를 사용하는 블룸필터 $B_1$을 사용한다. 메모리 공간의 효율적 사용을 위해서 블룸필터 $B_1$은 $T_1$에 등록된 서로 다른 목적지 IP주소들이 공유해서 사용한다. 블룸필터 $B_1$의 크기는 $T_1$에 속한 여러 IP가 공유하는 환경에서도 충돌 확률이 낮게 유지될 수 있도록 충분히 크게 설정되어야 한다.

만약 특정 내부 IP에 대한 $COUNT1(d)$의 값이 임계치 $N_{th}^1$을 초과하면 $d$는 잠재적인 공격목표가 된 것으로 간주하고 $d$에 대한 관리는 두 번째 단계로 넘어가며 두 번째 단계에서 $d$에 대한 LEA, 즉, 정상적인 접속 IP의 목록을 DDoS공격이 본격적으로 시작될 때까지 구축하게 된다. 여기서 주의해야 할 점은 $COUNT1(d)$는 샘플링을 거친 카운트 값이므로 실제 목적지 IP $d$에 접속한 source IP의 개수는 $COUNT1(d)=N_{th}^1$ 일 때 평균적으로 $N_{th}^1/p_s$이 된다.

블룸필터와 마찬가지로 내부 IP에 접속하는 외부 IP의 개수를 관리하는 테이블 $T_1$도 해시 테이블이기 때문에 충돌이 발생할 수 있다. IP $s$에서 $d$로 향하는 패킷이 제안한 방어 시스템이 설치된 라우터에 도착하였을 때 $d$에 대한 연결 정보는 그림 1과 같이 테이블 $T_1$의 행 $h_a(d)$($h_a$는 uniform random 해시 함수)에 저장된다. 만약 $d_1$이 해시 함수 $h_a$에 의하여 행 $r$에 매핑 되었을 때 행 $r$을 이미 다른 IP $d_2$가 차지하고 있을 경우 충돌이 발생하게 된다. 이러한 충돌은 DDoS 방어시스템 성능에 영향을 미칠 수 있기 때문에, 이러한 충돌이 발생하지 않도록 $T_1$의 등록절차를 관리할 필요가 있다.

많은 개수의 외부 IP로부터 접속되는 내부 IP는 잠재적으로 DDoS 공격목표일 가능성이 높기 때문에 이 같은 내부 IP는 해시 테이블에 남겨두고 관리를 하여야 한다. 따라서, 위와 같은 문제를 해결하기 위하여 본 논문에서는 아래와 같은 해시 테이블 엔트리 관리 정책을 사용한다. 새로운 destination IP $d_n$이 이미 destination IP $d_0$가 차지하고 있는 행 $r$로 매핑 되었을 경우 만약 $COUNT1(d_0)>1$이면 행 $r$에서는 여전히 $d_0$을 관리하고 만약 $COUNT1(d_0)=1$이면 행 $r$에서는 $d_0$을 지우고 $d_n$을 관리하게 된다.

### 3.2 IP perfix 별 traffic 속도에 기반한 신용도 관리

제안하는 방식에서는 DoS 공격이 시작되는 시점에 새롭게 나타나는 IP 주소들이 대거 white list에 등록되는 현상을 막기 위해서 IP address prefix별로 신용도를 관리하고, 신용도가 높은 prefix에 속하는 IP 주소들을 높은 확률로 두 번째 단계에서 white list, 즉, LEA에 등록하게 된다. 각 IP prefix별 신용도는 해당 라우터를 통해 평균적으로 서비스 받는 트래픽의 양이 많을수록 높은 것으로 간주한다. 각 source address prefix별 신용도는 그림 3에서 보는 것과 같이 신용도 표 $T_c$에서 관리된다.

신용도 표 $T_c$는 그림 3에서 보는 것과 같이 L개의 행과 2개의 열을 가지는 해시 테이블이기 때문에, 모두 2L개의

Source IP address s ⟶ $h_c$(s & 0xffffff00)

| IP prefix | Current traffic rate ($r_n$) (Mbps) | average traffic rate ($R_n$) (Mbps) | LEA registration probability ($p_n$) | LEA registration probability ($p_{n-1}$) | IP prefix | Current traffic rate ($r_n$) (Mbps) | average traffic rate ($R_n$) (Mbps) | LEA registration probability ($p_n$) | LEA registration probability ($p_{n-1}$) |
|---|---|---|---|---|---|---|---|---|---|
| 11.2.3 | $r_{n,1,1}$ = 7 | $R_{n,1,1}$ = 10 | $p_{n,1,1}$ = 0.2 | $P_{n-1,1,1}$ = 0.18 | 16.5.7 | $r_{n,1,2}$ = 45 | $R_{n,1,2}$ = 55 | $p_{n,1,2}$ = 1 | $P_{n-1,1,2}$ = 1 |
| 4.5.6 | $r_{n,2,1}$ = 35 | $R_{n,2,1}$ = 50 | $p_{n,2,1}$ = 1 | $P_{n-1,2,1}$ = 0.95 | 17.10.3 | $r_{n,2,2}$ = 5 | $R_{n,2,2}$ = 6 | $p_{n,2,2}$ = 0.12 | $p_{n,-12,2}$ = 0.10 |
| 7.8.9 | $r_{n,3,1}$ = 14 | $R_{n,3,1}$ = 11 | $p_{n,3,1}$ = 0.22 | $P_{n-1,3,1}$ = 0.20 | 15.3.2 | $r_{n,3,2}$ = 4 | $R_{n,3,2}$ = 7 | $p_{n,3,2}$ = 0.14 | $P_{n-1,3,2}$ = 0.16 |
| 155.6.7 | $r_{n,4,1}$ = 10 | $R_{n,4,1}$ = 8 | $p_{n,4,1}$ = 0.16 | $P_{n-1,4,1}$ = 0.15 | 9.10.11 | $r_{n,4,2}$ = 0.1 | $R_{n,4,2}$ = 0.1 | $p_{n,4,2}$ = 0.002 | $P_{n-1,4,2}$ = 0.002 |
| ⋮ | | | | | ⋮ | | | | |
| L-th prefix | $r_{n,L,1}$ | $R_{n,L,1}$ | $p_{n,L,1}$ | $P_{n-1,L,1}$ | L-th prefix | $r_{n,L,2}$ | $R_{n,L,2}$ | $p_{n,L,2}$ | $P_{n-1,L,2}$ |

**그림 3.** IP prefix별 신용도 관리표 $T_c$

IP address prefix에 대한 신용도를 관리할 수 있다. $h_c$는 $T_c$에서 사용되는 uniform random 해시 함수를 나타내고, 도착하는 패킷의 source IP 주소가 아닌 주소 prefix에 해시 함수를 적용하고 있음에 주목할 필요가 있다. 본 방식에서는 prefix를 class C, 즉, 각 IP 주소의 상위 24bit로 고정해서 관리한다. 이 경우 가장 큰 문제가 될 수 있는 점은 각 IP 주소의 상위 24bit를 prefix로 보는 경우 prefix가 모두 $2^{24}$개, 즉 1600만개 이상이 존재할 수 있기 때문에 모든 가능한 prefix에 대해서 신용도를 관리하는 경우 원래 목표인 작은 크기의 메모리를 이용한 구현이 어려워지게 된다.

신용도가 그 동안 많은 트래픽을 보내고, 서비스 받은 경우에만 높게 보는 신용도 관리 기준에 근거하여 이전에 보낸 트래픽 또는 서비스 받은 트래픽의 양이 매우 작은 IP prefix는 샘플링 확률 또한 매우 낮게 결정될 것이므로, 제안하는 방식에서는 메모리의 효율적인 사용을 위하여 많은 트래픽이 서비스 받은 IP prefix를 $T_c$에서 관리하고 신용도가 매우 낮은 prefix들은 가능한 $T_c$에서 관리하지 않거나 낮은 우선순위로 관리하는 방향으로 메모리 문제를 해결하고자 시도한다. 좀 더 구체적인 관리 방법은 먼저 $T_c$의 구조를 기술한 이후에 설명한다.

신용도 표에서는 각 IP prefix별로 트래픽 속도 및 샘플링 확률과 관련이 있는 다음 지표들을 관리한다: $r_{n,i,j}$, $R_{n,i,j}$, $p_{n,i,j}$, $p_{n-1,i,j}$. 먼저 $r_{n,i,j}$ ($1 \leqq i \leqq$ L, $1 \leqq j \leqq 2$)는 현재 $n$번째 관측구간에서 측정된 해시 테이블 $T_c$에서 $i$번째 행, $j$번째 열에 등록된 IP prefix에 대한 트래픽 서비스 속도를 나타낸다. $R_{n,i,j}$는 동일한 IP prefix에 대해서 측정된 여러 관측 구간에 걸친 평균 트래픽 서비스 속도를 의미하며, 오랜 history 관리로 인한 메모리 소모를 최소화하기 위하여 다음과 같이 $r_{n-1,i,j}$와 이전에 계산된 평균치 $R_{n-1,i,j}$에 관해서 다음과 같이 EWMA(exponentially weighted moving average) 형태로 계산한다:

$$R_{n,i,j} = \alpha r_{n-1,i,j} + (1-\alpha) R_{n-1,i,j}. \quad (1)$$

$p_{n,i,j}$는 DoS 방어의 두 번째 단계에서 white list, 즉 LEA,를 구성할 때 해당 IP prefix에 속하는 IP 주소를 white list에 등록시키는 확률을 의미한다. 이전에 언급된 바와 같이 본 논문에서는 기존에 충분히 높은 속도로 해당 subnet을 접속해 온 IP prefix는 신뢰도가 높은 것으로 보기 때문에, 그 동안 평균 트래픽 서비스 속도가 기준을 넘는 경우에 대해서 높은 등록 확률을 부여하기 위해서 $p_{n,i,j}$을 $R_{n,i,j}$에 관해서 다음과 같이 설정한다:

$$p_{n,i,j} = \min\{1, R_{n,i,j}/(\gamma C)\}. \quad (2)$$

여기에서 $C$는 제안되는 방어시스템이 설치된 노드의 출력 링크 속도를 의미하여, $\gamma C$가 해당 IP prefix로 부터의 평균 트래픽 속도가 충분히 높은지를 판단하는 기준이 된다. 현재 $\gamma$의 값과 $\alpha$의 값은 여러 번의 실험을 통해서 각각 0.05와 0.1로 설정하였으며, 최적의 값을 찾는 방법에 대해서는 추후에 연구가 될 예정이다. 현재 구간, 즉 n번째 구간의 초기에 $R_{n,i,j}$의 값이 갑작스럽게 증가하는 DoS 트래픽에 의해 오염되는 경우, DoS 트래픽에 의한 영향을 줄이기 위하여 n번째 구간에서는 이전 관측 구간이 시작하는 단계에서 미리 계산된 $p_{n-1,i,j}$을 사용한다. 따라서, 그림 3에서 보는바와 같이 이전 구간에서 계산된

LEA 등록 확률 값 $p_{n-1,i,j}$도 $T_c$에 저장하게 된다.

그림 3의 해시 테이블 $T_c$에서 각 행마다 두 개의 IP address prefix를 관리하는 공간을 둔 이유는 해시 테이블에서 발생하는 충돌을 해결하기 위해서이다. 각 행별로 두 개의 IP address prefix를 관리할 수 있는 공간을 두더라도 만약 세 개의 IP address prefix가 해시 함수에 의해서 동일한 행으로 대응이 된다면 충돌은 여전히 발생할 수 밖에 없다. 이와 같은 해시 충돌과 앞에서 언급한 제한된 메모리로 인해서 모든 IP prefix를 관리할 수 없는 문제를 해결하기 위해서 해시 테이블의 엔트리 관리 정책을 잘 설정할 필요가 있다. 본 논문에서 사용하는 $T_c$에 대한 해시 테이블 엔트리 관리 정책은 다음과 같이 요약될 수 있다. 새로운 IP prefix가 나타나는 경우 그림 3에서와 같이 prefix에 해시 함수 $h_c$를 적용함으로써 대응되는 행을 찾을 수 있다. 만약 대응되는 행에 있는 2개의 entry가운데 남는 공간이 있으면 새로 도착한 IP prefix가 바로 사용할 수 있다. 만약 대응되는 행에 남는 entry가 없는 경우에는 이미 그 행에 존재하는 IP prefix들의 평균 트래픽 서비스 속도인 $R_{n,i,1}$와 $R_{n,i,2}$ 가운데 문턱값 $r_{low}$와 비교해서 만약 $r_{low}$보다 낮은 값이 존재하는 경우에는 그 entry를 삭제하고 새로운 IP prefix를 등록 및 관리한다. 이와 같은 정책을 통하여 제한된 메모리 상황에서 신용도, 또는 트래픽 속도가 낮은 IP prefix에 비해 트래픽 속도가 높은 IP prefix를 우선적으로 관리하게 된다. $r_{low}$의 값은 일단 0.001$C$로 설정한다.

그림 3에서 해시 테이블 $T_c$의 행의 개수를 200으로 고정한다. 수식 (2)에서 $\gamma$의 값을 0.05로 두게 되면 등록 확률이 1인 IP prefix는 최대 20개까지 존재할 수 있다. 따라서, 이 경우 등록 확률이 1인 IP prefix 3개 이상이 동시에 같은 행에 대응될 확률은 2.7%보다 작기 때문에, 열의 개수를 2로 두어도 높은 등록 확률을 가지는 IP prefix들의 관리가 가능하다.

### 3.3 두 번째 단계: white list(LEA) 생성 및 우선 순위에 기반한 DoS 방어

첫 단계에서 서브넷 중의 특정 내부 IP $d$에 대한 카운트 값 $COUNT1(d)$가 첫 번째 임계치인 $N_{th}^1$을 초과하였을 경우 $d$는 잠재적으로 DDoS 공격목표가 될 가능성이 있지만 DDoS 공격을 받고 있다고는 간주하지 않는다. 그 원인은 첫 단계의 임계치 $N_{th}^1$은 잠재적인 DDoS 공격목표 검출만을 목표로 비교적 낮게 설정이 되기 때문이다. 따라서 $COUNT1(d) > N_{th}^1$ 일 경우 특정 내부 IP $d$에 대한 관리는 두 번째 단계에서부터 하게 되며 $d$에 대한 DDoS 공격이 발생할 때까지 $d$에 접속하는 외부 IP들을 저장하는 IP 리스트(List), 즉 LEA를 작성하게 된다.

첫 단계에서 $COUNT1(d) > N_{th}^1$ 인 destination IP $d$에 대해서는 접속하는 외부 IP의 개수가 작은 관계로 어떤 규제도 적용하지 않는다. 적은 양의 IP가 대부분의 대역폭을 차지하는 공격에 대해서는 추가적으로 [3][6]의 Heavy-hitter 검출 메커니즘을 이용하여 검출할 수도 있다. 그러므로 두 번째 단계에서는 첫 단계에서 $COUNT1(d) > N_{th}^1$ 인 $d$에 대해서 DDoS 공격이 발생하였을 경우 기존의 연결된 flow에 지속적으로 정상적인 서비스를 제공하기 위하여 아래와 같은 트래픽 관리 정책을 적용한다.

만약 두 번째 단계에서 관리하는 내부 IP $d$로 접속하는 외부 IP의 개수가 두 번째 임계치인 $N_{th}^2$ ( $N_{th}^2 > N_{th}^1/p_s$) 의 값을 초과할 경우 특정 IP $d$는 DDoS 공격을 받고 있다고 간주하고 기존에 연결된 flow, 즉 LEA에 저장된 IP 주소, 에 대해서는 높은 우선순위로 지속적으로 정상적인 서비스를 제공하고 LEA에 저장되지 않은 IP 주소, 즉 새로 연결된 flow, 에 대해서는 네트워크의 혼잡(Congestion) 상태에 따라 낮은 우선순위로 서비스 하거나 패킷을 폐기한다. 그러므로 제안된 방어 시스템을 작동하는 에지 라우터는 LEA의 정보에 따라 내부 IP에 접속하는 외부 IP의 개수를 제한하는 수락제어(*admission control*)를 수행하게 된다. 만약 두 번째 단계에서 관리하는 내부 IP $d$로 접속하는 외부 IP의 개수가 다시 임계치 $N_{th}^2$ 보다 낮아질 경우 새롭게 도착하는 source IP주소는 3.2절에서 설명한 IP prefix별 신용도 관리표 $T_c$에 있는 이전 관측구간에서 계산된 등록확률 $p_{n-1}$의 확률로 LEA에 등록이 되게 된다. 만약 도착하는 IP prefix 정보가 그림 3의 $T_c$에 존재하지 않으면 그 IP는 LEA에 등록이 될 수가 없으며, 이러한 경우는 DoS 공격시 새로운 IP가 나타날 때 발생할 수 있다. 만약 특정 내부 IP로 접속하는 외부 IP의 양이 다시 세 번째 임계치인 $N_{th}^3$ ($N_{th}^3 = N_{th}^1/p_s$, $N_{th}^3 < N_{th}^2$ ) 보다도 낮을 경우 특정 내부 IP $d$는 DDoS 공격을 더 이상 받지 않는 것으로 간주하여 $d$를 두 번째 단계에서 삭제하고, $d$에 대한 LEA도 더 이상 관리하지 않는다.

LEA에 대한 관리를 좀 더 자세하게 살펴본다. 두 번째 단계에서도 첫 단계와 마찬가지로 하나의 연결 상태 관리 테이블 $T_2$와 하나의 블룸필터 $B_2$를 사용하지만 그림 4 및 그림 5와 같이 첫 번째 단계의 테이블 $T_1$과 블룸필터 $B_1$과는 조금 다른 구조를 가지고 있다. 해시 테이블 $T_2$는 첫 번째 단계에서 검출된 잠재적인 DDoS 공격대상 IP $d$에

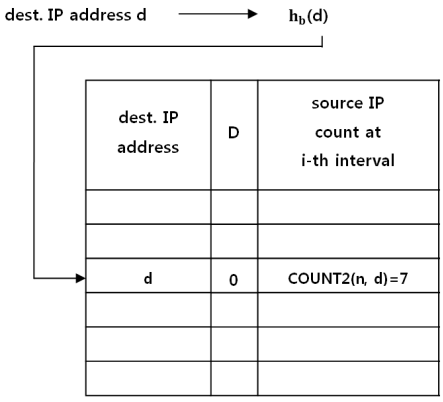


**그림 4.** 연결 상태 관리 테이블 $T_2$

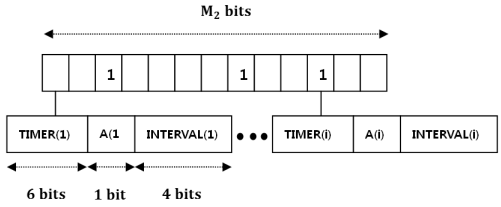


**그림 5.** source IP 추적을 위한 블룸필터 $B_2$

현재 접속하고 있는 외부 IP의 개수를 추적한다. 즉, *COUNT2*(*n*, *d*)는 현재 잠재적인 DDoS 공격목표인 내부 IP *d*로 접속하고 있는 외부 IP의 개수를 카운트하고 1-bit 필드 *D*는 0으로 초기화되어 있으며 이후 내부 IP *d*가 DDoS 공격을 받는 것으로 간주되는 순간부터 1로 설정된다.

각 flow의 연결을 시도한 시점 및 refresh(기존의 연결된 flow가 해당 구간에서 패킷을 보내는 경우)된 시기를 정확하게 추적하기 위하여 long time interval을 정의한다. $\triangle_L$은 long time interval의 길이를 나타내고, *n*번째 long time interval을 $I_{L,n}$로 표기한다. 그리고, $I_L$은 long time interval의 index 값을 나타낸다. 즉, $I_{L,n}$에서 $I_L$의 값은 *n*이 된다. 좀 더 엄밀하게는 $I_L$값을 저장하는 공간의 크기를 작게 유지하기 위해 $I_{L,n}$에서 $I_L$의 값을 $I_L = n\%R$로 계산한다. 여기에서 $R$은 flow life-time에 관련된 상수이다. 제안하는 방어시스템에서는 새로운 IP에서 패킷이 도착하면 그 시점부터 새로운 flow의 life-time이 $(R-1)\times I_S$인 것으로 인식한다. $I_S$는 $T_1$의 관리 주기, 즉, short time interval의 길이이다. 방어 시스템이 인식하는 특정 flow의 life-time은 이후 같은 flow에 속하는 패킷이 추가로 도착하면 연장이 될 수 있다.

*n*번째 long time interval $I_{L,n}$에서 $T_2$에 있는 IP *d*로 접속하는 새로운 flow의 개수와 $I_{L,n}$에서 refresh된 IP의 개수의 합을 $I_{L,n}$의 *COUNTp*(*n*, *d*) 값으로 정의한다. *COUNTc*(*n*, *d*)는 $I_{L,n}$에서 시작하여 연결된 외부 flow들 중에서 flow life-time동안 refresh된 flow의 개수를 카운트한다. *COUNTp*(*n*, *d*)와 *COUNTc*(*n*, *d*)의 역할은 *COUNT*$_2$(*n*, *d*)의 동작 방법을 설명할 때 더욱 자세하게 설명한다. 방어 시스템은 메모리 이용 효율을 높이기 위하여 두 번째 단계에서 첫 단계에 검출된 내부 IP *d*로 연결하는 외부 IP들을 블룸필터 $B_2$에 저장하여 관리하며 메모리 효율을 더욱 향상시키기 위하여 블룸필터 $B_2$는 첫 단계에서 검출된 모든 내부 IP들에 의해 공유된다. 두 번째 단계에서 사용되는 블룸필터 $B_2$는 첫 단계의 블룸필터 $B_1$과 달리 그림 5와 같이 별도의 *TIMER*와 *INTERVAL* 변수를 블룸필터의 각 bit에 할당한다. *TIMER*(*i*)와 *INTERVAL*(*i*)는 각각 블룸필터의 *i*번째 bit의 *TIMER*와 *INTERVAL*의 값을 의미한다. *A*(*i*)는 flow life-time동안 특정 서브넷에 연결된 외부 IP가 블룸필터의 *i*번째 bit을 사용하고 있음을 의미하고, 추후 *COUNT2*(*n*, *d*)를 업데이트하는데 사용된다. 그리고 제안된 방어 시스템에서는 일정한 시간, 즉 flow life-time, 동안 주어진 IP쌍 사이에 패킷의 전달이 없을 경우 두 IP 사이의 연결이 끊어진 것으로 간주한다. 특히 블룸필터 $B_2$에서 사용되는 *TIMER*는 서브넷 내부 IP와 외부 IP사이의 연결 상태를 모니터링하기 위하여 사용된다. 다른 한편으로는 새로운 flow가 생성되거나 기존의 연결된 flow가 refresh되는 경우 flow가 생성되거나 refresh된 시기를 기록 및 업데이트하기 위하여 *INTERVAL* 변수를 사용한다.

*TIMER*와 *INTERVAL*의 구체적인 작동방식은 아래와 같다. 블룸필터 $B_2$에서는 $k_2$개의 해시 함수를 사용하고, $h_i^b$는 블룸필터 $B_2$에서 사용되는 $i$번째 해시 함수를 나타낸다. 만약 새로운 Source IP $s'$이 $T_2$에서 관리하고 있는 destination IP *d*로 연결을 시도할 경우 *COUNT2*(*n*, *d*)의 값은 1이 증가하며 $k_2$ bit 위치에 해당하는 블룸필터 $B_2$의 $h_i^b(s')$ $(i=1,2,...,k_2)$의 값은 모두 1로 설정이 되고 $TIMER(h_i^b(s'))(i=1, 2, ..., k_2)$값은 정수 *R*로 설정되며 $INTERVAL(h_i^b(s'))$ $(i=1, 2, ..., k_2)$의 값은 해당 long time interval의 index($I_L$)의 값으로 설정된다. 만약 $s'$이 refresh되었을 경우 $s'$에 해당하는 *INTERVAL*값들은 다시 refresh가 발생한 long time interval의 $I_L$의 값으로 업데이트 된다. 블룸필터 $B_2$의 각 *TIMER*값은 $I_S$마다 1

씩 감소한다. 그러므로 *s'*에 해당하는 *TIMER*의 값 중에서 최소한 하나의 *TIMER*의 값이 $(R\text{-}1)\times I_S$초 후 0으로 감소되면 *s'*과 *d* 사이의 연결은 끊어진 것으로 간주된다. 만약 *TIMER*의 값이 0으로 감소하기 전에 *s'*가 refresh되었을 경우 *s'*에 해당하는 *TIMER*의 값들은 다시 정수 *R*로 재설정된다. 따라서 *R*, $I_S$와 $\triangle_L$의 구체적인 값은 정상적인 flow의 패킷 도착간격의 패턴을 고려하여 설정하여야 한다.

[1]의 연구결과에 의하면 World Wide Web (WWW) 트래픽은 자기유사성(self-similarity)을 나타내고 On-off 패턴의 웹 트래픽에서 트래픽의 침묵시간(Silent times)은 인터넷 사용자의 사고시간(Think time)의 영향으로 Heavy-tail한 경향을 나타내는 것으로 알려져 있다. 그리고, [11]의 결과에 따르면 HTTP 패킷의 도착간격은 여전히 long-tail한 분포를 가지고 있지만 패킷 도착간격이 2초를 넘는 비율은 3% 미만 이였다. 본 논문에서는 두 노드사이에 3000초 이상 패킷 교환이 없을 경우 두 노드사이의 연결은 끊어진 것으로 간주하였다. $\triangle_L$의 값은 300초로 설정하였고, 외부 IP가 블룸필터 $B_2$에 저장되어 있는 시간이 300초보다 길어야 하므로 $TIMER$의 초기 설정 값 *R*은 61로 설정하고 $I_S$의 값은 50초로 설정하였다. 따라서, 외부 IP가 블룸필터에 저장되어 있는 시간은 최소 $(R\text{-}1)\times I_S = 3000$초가 된다.

제안된 방어 시스템에서 *COUNT2*(*n, d*)의 값은 블룸필터 $B_2$의 $TIMER$의 이벤트가 만료되어도 $B_2$에 저장되어 있는 IP의 개수와 일치하도록 관리하고자 한다. 만약 *COUNT2*(*n, d*)가 $B_2$의 *TIMER*가 만료되는 이벤트를 제때 반영하지 못하면 *COUNT*2(*n, d*)의 값은 실제 $B_2$에 남아있는 IP의 개수보다 큰 값을 가지게 된다. *COUNT2*(*n, d*)의 과대평가(overestimation)로 링크 또는 내부 노드의 활용도는 최적의 경우보다 낮을 수 있지만 노드 *d*는 여전히 방어 시스템에 의해 DDoS 공격으로부터 보호를 받을 것이다. 다른 한편으로 만약 *COUNT2*(*n, d*)가 $B_2$에 있는 IP의 개수를 과소평가(underestimation)하는 경우 노드 *d*는 DDoS 공격이 발생시 *d*에 연결되는 외부 IP의 개수를 과소평가했기 때문에 DDoS 공격으로부터 보호되지 못할 수도 있다. 따라서, 제안된 방어 시스템은 *COUNT2*(*n, d*)의 값이 실제 접속 중인 외부 IP의 개수보다 작지 않도록 *COUNT2*(*n, d*)의 값을 관리하여야 한다.

그림 6은 *R* = 11인 경우 제안된 방어 시스템의 *COUNT2*(*n, d*)의 관리 방법을 보여준다. *COUNTp*(*n, d*)는 $I_{L,n}$에서 내부 IP *d*로 접속하는 새로운 flow의 개수와 같은 long time interval에서 refresh된 IP의 개수의 합을 저장한다. 그리고 *COUNTc*(*n, d*)의 관리는 아래와 같다. *n*번째 long time interval에서 시작하여 *d*와 연결중인 외부 IP들 가운데 $(R\text{-}1)\times I_S$동안 refresh된 flow의 개수가 *j*일 경우 *COUNTc*(*n, d*)의 값은 *j*가 된다. 예를 들어 그림 6에서 $I_{L,4}$에서 내부 IP *d*로 접속하는 새로운 flow의 개수는 1이고 refresh된 flow의 개수도 1이다. 따라서 4번째 long time interval에서 *COUNTp*(4, *d*)의 값은 2가 된다. 그리고 $I_{L,1}$에서 생성된 3개의 flow들 중의 *S*3에 해당하는 flow가 $I_{L,4}$에서 refresh 되었으므로 *COUNTc*(*n, d*)의 해당 필드(*n* =1)의 값 즉, *COUNTc*(1, *d*)은 1로 업데이트 되고 flow *S*3의 life-time은 연장되었다. 그러므로 제안된 알고리즘에서 *COUNTp*(*n, d*)는 $(R\text{-}1)\times I_S$ 초 후, 즉 flow life-time 후 연결이 끊길 flow의 개수를 관리하고 *COUNTc*(*n, d*)는 각 flow가 예정된 life-time동안에 refresh되어 *TIMER*값이 재설정되어 flow life-time이 길어진 flow의 개수를 관리하여 *COUNT2*(*n, d*)의 과소평가(underestimation)가 발생하지 않도록 보상한다. *COUNT2*

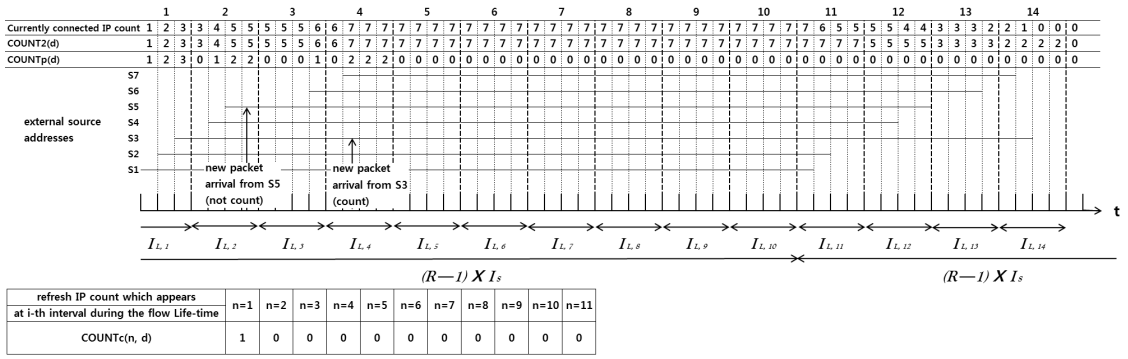


**그림 6.** 시간에 따른 *COUNT2*(*n, d*), *COUNTp*(*n, d*), *COUNTc*(*n, d*)의 관리 (*R*=11인 경우)

($n$, $d$)의 구체적인 작동 방식은 아래와 같다.

제안된 방어시스템이 가동되어 첫 flow life-time, 즉, ($R$-1)×$I_S$, 만큼의 시간 간격에서는 새로운 flow가 내부 IP $d$로 접속할 때마다 그림 3의 IP prefix 별 신용도관리표 $T_c$에 있는 등록 확률값에 따라 블룸필터 $B_2$로의 등록 여부를 결정하고, 새로운 IP가 $B_2$에 등록되는 경우 *COUNT2*($n$, $d$)의 값도 1 증가한다. 이때 블룸필터 $B_2$의 해당되는 bit에 연결된 *TIMER*의 값은 $R$로 설정되고 해당 *INTERVAL*의 값은 현재 $I_L$의 값으로 설정되고, $A$의 값은 1이 된다. 블룸필터 $B_2$에서 $A$의 값은 각 long time interval마다 0으로 재설정 되므로 $A$의 값은 이미 연결된 flow에 대해서는 도착하는 패킷이 현재 long time interval에서 처음 도착하는지 아닌지를 알려주고, 그 결과는 flow life-time 연장에 사용된다. 첫 flow life-time동안에 *COUNT2*($n$, $d$)는 새로운 flow가 내부 IP $d$로 접속할 때마다 1씩 증가하여 실제 연결된 flow의 개수보다 항상 크거나 같으므로 위에서 언급한 과소평가 상황이 발생하지 않는다.

첫 flow life-time이 지난 후부터는 *COUNT2*($n$, $d$)의 underestimation을 방지하기 위하여 아래와 같은 수식을 사용한다:

$$\begin{aligned} &COUNT2(n+1,d) = COUNT2(n,d) \\ &- COUNTp(n+1-R,d) \\ &+ COUNTc(n+1-R,d). \end{aligned} \tag{3}$$

여기서 $n$은 현재 long time interval의 index 값을 나타낸다. 접속한 외부 IP의 개수에 대한 관리는 아래와 같다. 만약 새로운 flow가 내부 IP $d$로 접속할 때 $T_c$의 확률에 따라 샘플링이 되는 경우 *COUNT2*($n$, $d$)는 여전히 1씩 증가한다. 예를 들어 $R$=11인 경우 만약 $I_{L,n}$에서 10개의 새로운 flow가 접속되었을 경우 이 10개의 flow들은 $I_{L,(n+11)}$에서 *TIMER*가 0으로 감소되므로 연결이 끊기게 되어 있다. 따라서 $I_{L,(n+11)}$에서 *COUNT2*($n$+11, $d$)의 값은 $I_{L,(n+10)}$의 *COUNT2*($n$+10, $d$)의 값에서 $I_{L,n}$의 *COUNTp*($n$, $d$)의 값을 빼주면 된다. 하지만 $I_{L,n}$에서 접속을 시작한 flow들 중에서 life-time동안 refresh되어 *TIMER*가 재설정되는 flow가 있을 수 있기 때문에 위의 $I_{L,(n+10)}$의 *COUNT2*($n$+10, $d$)의 값에서 $I_{L,n}$의 *COUNTp*($n$, $d$)의 값을 뺀 후 $I_{L,n}$의 *COUNTc*($n$, $d$)의 값을 더하면 $I_{L,(n+11)}$에서 flow의 개수와 일치하게 된다. 여기서 *COUNT2*($n$+11, $d$)의 값은 정확하게 실제 연결된 flow의 개수와 일치하게 된다. 여기서 *COUNTc*($n$, $d$)의 관리는 효율적인 메모리사용을 위하여 $R$(=11)개 필드만 사용하고 cyclic-buffer 형태로 재활용 한다.

### 3.4 알고리즘

DDoS 공격 검출 및 방어 메커니즘은 아래와 같다. *COUNT2*($n$, $d$) > $N_{th}^2$ 인 경우 LEA에 저장된 기존의 IP 주소, 즉 블룸필터 $B_2$에 저장되어 있는 IP주소들은 LEA에 저장되어 있지 않은 새로운 IP 주소에 비해 높은 우선순위로 전송이 된다. Source IP가 $s_m$인 공격자가 연결을 시도하였을 경우 블룸필터 $B_2$에서 만약 $s_m$에 해당하는 모든 bit들이 1인 경우 해시 테이블 $B_2$에서 충돌이 발생한 것이 되며, 이런 경우 공격자 $s_m$은 높은 우선순위로 서비스를 제공 받게 된다. 따라서, 블룸필터 $B_2$의 크기는 false negative 확률이 낮게 유지될 수 있도록 충분히 커야 한다.

$R_{T_2}$를 그림 4의 연결 상태 관리 해시 테이블 $T_2$의 행의 개수라고 정의할 경우 블룸필터 $B_2$는 최대로 $R_{T_2}\times N_{th}^2$ 개의 외부 IP밖에 저장할 수 없다. 그 원인은 $T_2$의 각 행에서 COUNT2 > $N_{th}^2$ 일 경우 DDoS 공격이 발생한 것으로 간주하므로 LEA추가 등록을 멈추기 때문이다. 위에서 언급한 것과 같이 $k_2$는 블룸필터 $B_2$에서 사용하는 해시 함수의 개수이다. 크기가 $M_2$인 벡터(Vector)에 $n'$개의 IP를 저장한 후 특정 bit가 여전히 0일 확률은 $(1\text{-}1/M_2)^{k_2 n'}$ 이다. 블룸필터에서의 충돌은 새로운 IP에 대해서 $k_2$ 개의 해시 함수로 지정되는 모든 bit가 1로 설정되어 있는 경우 나타나게 된다. 따라서 $(R_{T_2}N_{th}^2)$번째 IP가 저장된 후 새로운 IP가 접속시 충돌발생 확률은 다음과 같다.

$$\begin{aligned} &(1-(1-1/M_2)^{k_2 R_{T_2} N_{th}^2})^{k_2} \\ &\approx (1-e^{-k_2 R_{th}^2 N_{th}^2/M_2})^{k_2}. \end{aligned} \tag{4}$$

수식 (4)의 오른쪽은 [4]에 의해 $k_2 = \ln 2 \times M_2/(R_{T_2}N_{th}^2)$인 경우 최소화 될 수 있다. 따라서 최소인 경우 (4)는 다음과 같이 표현될 수 있다.

$$(1/2)^{k_2} = (0.6185)^{M_2/(R_{T_2}N_{th}^2)}. \tag{5}$$

수식 (5)에 의하여 $k_2$의 값이 7보다 크거나 같은 경우 해시 테이블에서의 충돌 발생 확률은 1%이하로 유지된

다. 따라서 본 논문에서는 $k_2$의 값을 7로 설정하였으며 $k_2$가 7인 경우 $M_2$는 다음과 같이 표현된다.

$$M_2 = \lceil k_2 R_{T_2} N_{th}^2 / \ln 2 \rceil \simeq \lceil 10.1 \times R_{T_2} N_{th}^2 \rceil . \quad (6)$$

$M_1$의 크기도 같은 방법으로 정해질 수 있다.

## 4. 성능 분석

제안된 DDoS 방어 시스템은 OPNET 시뮬레이션을 통하여 평가하였다. DDoS 공격시 제안된 DDoS 방어시스템이 기존의 정상적인 연결에 속하는 트래픽을 어느 정도 보호하는지 알기 위해서 DDoS 방어시스템을 설치하였을 때와 그렇지 않은 경우 throughput, 즉 송신단에서 보낸 트래픽 양에 대한 성공적으로 수신단까지 전달된 트래픽 비율, 을 측정 및 비교하였다. 또한 기존의 방식가운데 제안하는 방식과 유사한 개념으로 IP 히스토리 기반 필터링을 사용하는 Peng Tao 방식과도 성능을 비교해 보았다. 시뮬레이션 네트워크 토폴로지는 그림 7과 같다. 제안된 DDoS 방어 시스템은 에지 라우터 $R_1$에서 작동을 하며 $Z_2$개의 정상적인 사용자는 서브넷 중의 $U$개의 서버로 접속을 하며 $Z_1$개의 공격자는 일정한 시간 후 $U$개의 서버로 DDoS 공격을 실시하게 된다. 또한 DDoS 공격전 $Z_3$개의 정상적인 사용자는 동일한 서브넷에 속하지만 DDoS 공격대상이 아닌 $V$개의 사용자와 연결되어 있다. 대역폭 혼잡 공격(Bandwidth congestion attack) 상황을 고려하기 위하여 모든 외부 IP 트래픽은 $R_1$과 $R_2$사이 100 Mbps의 단일 링크를 통과하도록 하였다. 그리고 공격 트래픽과 정상적인 사용자 트래픽을 구별하지 못하는 극단적인 경우를 평가하기 위해 트래픽과 정상적인 사용자 트래픽을 모두 ON-OFF 트래픽으로 모델링하였으며 인터넷 트래픽의 특징인 Self-similarity를 반영하기 위해서 ON구간과 OFF구간의 길이는 Pareto 분포로 모델링 하였다[1]. Pareto 분포에 대한 확률 밀도 함수(Probability density function)는 $p(x) = \alpha k^{\alpha} x^{-\alpha-1}$, $x \geq k$로 표현될 수 있으며 ON과 OFF구간에 대한 각 변수는 $\alpha_{ON}$=1.2, $\alpha_{OFF}$ = 1.5, $k_{ON}$ = 0.167, $k_{OFF}$ = 10으로 설정하였다. 이때 평균 ON과 OFF 구간의 길이는 각각 1초와 30초가 된다. 트래픽은 ON 기간 동안 500 Byte의 패킷을 1.55 Mbps의 속도로 전송하므로 각 flow의 평균 전송 속도는 50 Kbps이다. 시뮬레이션 시나리오에서 $Z_1$ = 20000으로 공격 트래픽의 전체 평균속도는 1Gbps로써 $R_1$과 $R_2$사이 링크 속도 100Mbps의 10배가 된다. 공격목표 서버의 개수 $U$는 10개로 설정하고 서브넷 중의 내부 사용자 수 $V$는 100으로 설정하였다. 그리고, 공격대상 서버로 접속하는 정상적인 외부 사용자 수 $Z_2$와 같은 서브넷에 속한 내부 사용자와 연결하는 정상적인 외부 사용자 수 $Z_3$는 모두 200으로 설정하였다. 정상적인 트래픽은 0초부터 시작되고, 900초부터는 DDoS 공격이 가해지며 10000초 동안 시뮬레이션을 수행하였다. 그림 8은 제안된 방어 시스템이 에지 라우터에서 작동할 경우와 제안된 방어시스템 없이 에지 라우터의 출력 버퍼를 FIFO queue로만 운영했을 경우 기존에 연결된 flow, 즉 정상적인 사용자 $Z_2$의 트래픽이 공격받는 서버로 성공적으로 전달되는 비율과 정상적인 사용자 $Z_3$의 트래픽이 같은 서브넷에서 공격대상이 아닌 다른 사용자에게 성공적으로 전달되는 비율을 비교 평가하였다. 그림 9는 제안된 방어 시스템과 Peng Tao가 제안한 방어 시스템을 각각 에지 라우터에서 작동하였을 경우 $Z_2$의 트래픽 전달 성공률을 비교 평가하였다.

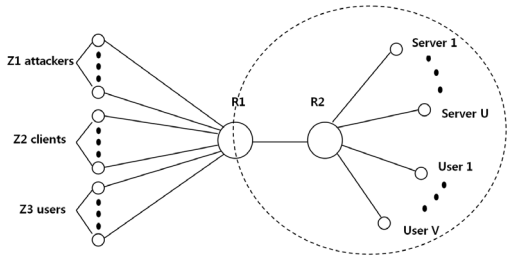


**그림 7.** 시뮬레이션 네트워크 토폴로지

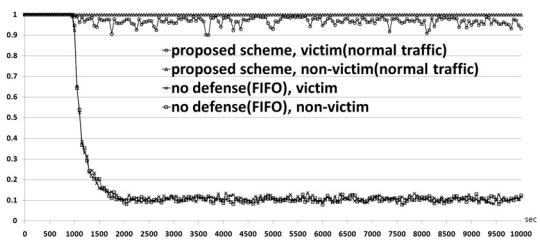


**그림 8.** 제안된 방어 시스템 작동시 정상적인 flow에 대한 Throughput의 보호

그림 7에서와 같이 제안된 방어 시스템을 작동 할 경우 그림 8은 DDoS 공격대상으로 가는 정상적인 사용자의 트래픽 가운데 90%이상이 성공적으로 전달되는 것을 보여주고, DDoS 공격대상이 아닌 곳으로는 100%에 가까운 전달 성공률이 얻어짐을 보여준다. 반면 제안된 방어 시스템을 작동하지 않고 단순 FIFO queue로만 작동하였을 경우 공격대상 서버로의 정상적인 사용자 패킷 전달

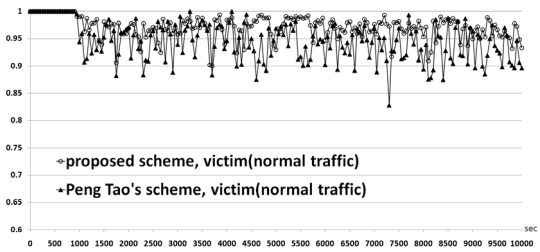


**그림 9.** 제안된 방어 시스템과 Peng Tao의 방어 시스템의 성능 비교

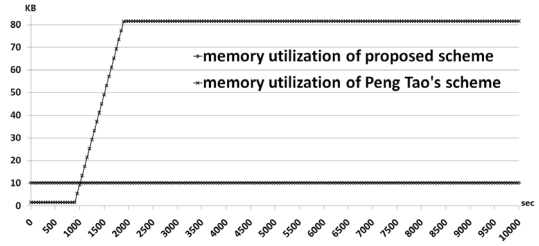


**그림 10.** 제안된 방어 시스템과 Peng Tao의 방어 시스템의 메모리 사용률 비교

성공률과 공격대상이 아닌 내부 사용자로의 패킷 전달 성공률은 모두 10% 밖에 되지 못했다. 만약 $R_{T_1}$, $R_{T_2}$와 $R_{T_c}$를 각각 테이블 $T_1$, $T_2$와 $T_c$의 행의 개수라고 정의할 경우 제안된 방어 시스템의 메모리 사용량은 $R_{T_1}$(48+10.1$N_{th}^1$)/8+$R_{T_2}$(87+10.1$N_{th}^2$)/8+$R_{T_c}$×40 Byte가 된다. 시뮬레이션에서 사용한 변수는 $R_{T_1}$=60, $R_{T_2}$=30, $R_{T_c}$=200, $N_{th}^1$=2, $N_{th}^2$=50이고, 이 경우 제안된 방어 시스템에 요구되는 메모리는 10.7KB가 된다. 그림 9는 제안된 방어 시스템을 Peng Tao의 방어 시스템과 비교할 경우 전체적으로 5%정도 나은 성능을 나타내고 있다. 그 원인은 제안한 방식은 IP prefix별 신용도 관리를 하여 각 prefix의 신용도에 따라 LEA에 저장하기 때문에 LEA가 공격 트래픽에 의해 오염되는 것을 방지하였지만 Peng Tao의 방식은 모든 IP에 대해 동등한 확률로 IP 히스토리를 관리함으로 인해 더 많은 공격 flow가 white list를 침범하였기 때문이다. 그림 10은 제안된 방식과 Peng Tao방식의 메모리 사용량을 비교하였으며, Peng Tao 방식이 본 논문에서 제안한 방어 시스템이 요구하는 메모리의 8배 가량을 소모하는 것을 관찰할 수 있다. Peng Tao 방식은 모든 flow를 관측하기 때문에 flow의 개수가 증가할수록 메모리 소모가 많아지게 되고, 특히 DDoS 공격의 경우 대량의 공격 IP 개수로 인해 메모리 소모가 더 심해지는 반면 제안하는 방식은 메모리 효율을 고려하여 설계가 되었기 때문에 10.7KB의 작은 메모리로 Peng Tao 방식에 비해 향상된 성능을 보여준다.

## 5. 결 론

본 논문에서는 작은 크기의 메모리로 정상적인 IP 주소 정보를 효율적으로 관리하여 DDoS 공격 발생시 이미 연결된 flow와 새로 도착하는 flow를 구별하여 서로 다른 우선순위로 서비스를 제공함으로써 특정 서브넷을 보호하는 두 단계로 나뉘는 DDoS 방어 시스템을 제시하였다. 첫 단계에서 샘플링기법을 이용하여 관측하는 패킷의 개수를 줄이고 잠재적인 DDoS 공격 대상만을 선택하여 관리하도록 하며, 첫 번째 단계와 두 번째 단계에서 모두 블룸필터를 사용함으로써 메모리 사용량을 감소시켰다. 시뮬레이션 결과에 따르면 제안된 방어 시스템은 방어 시스템이 설치되지 않은 경우보다 정상적인 사용자에 대해 훨씬 높은 품질의 서비스를 제공하고 있으며, 유사한 다른 방어 시스템에 비해서도 작은 메모리 크기로 우수한 성능을 보여주고 있다.

## 참 고 문 헌

1. M. E. Crovella and A. Bestavros. (1997), Self-similarity in world wide web traffic: evidence and possible causes, IEEE/ACM Trans. Networking, Vol. 5, No. 6, pp. 835-846.
2. D. Dagon, G. Gu, C. P. Lee, and W. Lee. (2007), "A Taxonomy of Botnet Structures," Proc. Annual Computer Security Applications Conference (ACSAC), pp. 325-339.
3. Estan, C., and Varghese, G. (2002), "New Directions in Traffic Measurement and Accounting," Proc. ACM SIG COMM, pp. 323-336.
4. Fan, L., Cao, P., Almeida, J., and Broder, A.Z. (1998), Summary cache: a scalable wide-area web cache sharing protocol, Technical Report 1361, Univ. of Wisconsin-Madison.
5. Jung, J., Krishnamurthy, B., and Rabinovich, M. (2002) "Flash Crowds and Denial of Service Attacks: Characterization and Implication for CDNs and Web Sites", Proc. World Wide Web (WWW) Conference, pp. 293-304.
6. Kompella R.R., Singh, S., and Varghese, G. (2004), "On Scalable Attack Detection in the Network", Proc. ACM Internet Measurement Conference (IMC), pp. 187-200.
7. X. Liu, X. Yang, and Y. Lu (2008), "To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-

node Botnets," Proc. ACM Sigcomm, pp. 195-206.
8. Peng, T., Leckie, C., and Ramamohanarao, K. (2003), "Protection from Distributed Denial of Service Attack Using History-based IP Filtering", Proc. IEEE ICC, pp. 482-486.
9. Peng, T., Leckie, C., and Ramamohanarao, K. (2004), "Proactively Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring", Proc. Networking Conference, pp. 771-782.
10. Peng, T., Leckie, C., and Ramamohanarao, K. (2007), "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems", ACM Computing Surveys, Vol. 31, No. 1, Article No. 3, pp. 1-42.
11. Z. Sun, D. He, L. Liang, and H. Cruickshank. (2004), "Internet QoS and traffic modelling", IEE Proceedings, Vol. 151, No. 5, pp. 248-255.
12. X. Yang, D. Wetherall, and T. Anderson (2008), "TVA: A DoS-Limiting Network Architecture," IEEE/ACM Trans. on Networking, Vol. 16, No. 6, pp. 1267-1280.

**Jinghe Jin** (jinjinghe@ynu.ac.kr)

2007 중국 대련민족대학교 소프트웨어공학 학사
2008～현재 영남대학교 전자정보공학부 정보통신공학전공 석사과정

관심분야 : Network Security

**이 태 진** (tjlee@kisa.or.kr)

2003 포항공대(POSTECH) 컴퓨터공학과 학사
2008 연세대학교 컴퓨터공학과 석사
2003～현재 한국인터넷진흥원(KISA) 주임 연구원

관심분야 : Network Security, VoIP

**남 승 엽** (synam@ynu.ac.kr)

1997 한국과학기술원(KAIST) 전기 및 전자공학과 학사
1999 한국과학기술원(KAIST) 전기 및 전자공학과 석사
2004 한국과학기술원(KAIST) 전자전산학과 박사
2004～2006 Post-doc, Carnegie Mellon University
2006～2007 Post-doc, 한국과학기술원(KAIST)
2007～현재 영남대학교 전자정보공학부 정보통신공학전공 교수

관심분야 : Network Measurement, Network Security, Network Architecture