병렬 구조 NIDS를 위한 효율적인 플로우 기반 부하 분산 기법에 관한 연구

김남욱*, 박민우*, 박선호*, 정태명**
*성균관대학교 전자전기컴퓨터공학과
**성균관대학교 정보통신공학부
e-mail : {nukim, mwpark, shpark}@imtl.skku.ac.kr
tmchung@ece.skku.ac.kr

A Study on Effective Flow-Based Load Balancing Scheme for Parallel-Structure NIDS

Nam-Uk Kim*, Min-Woo Park*, Seon-Ho Park*, Tai-Myoung Chung**
*Dept of Electrical and Computer Engineering, Sungkyunkwan University
**School of Information and Communication Engineering, Sungkyunkwan University

요 약

최근 네트워크를 구성하는 기반 시설의 성능이 향상됨에 따라 대량의 트래픽에 대한 네트워크 침입 탐지 시스템의 성능을 향상시키기 위한 연구가 진행되고 있다. 대규모 네트워크에서는 단일 시스템으로 네트워크 내의 모든 트래픽을 분석하는 것이 불가능하므로 병렬 구조 NIDS를 도입하여야 하는데, 이를 위해서는 병렬 구조를 이루는 각 NIDS 노드로의 부하 분산이 필요하다. 플로우 기반 부하 분산 기법은 이러한 부하 분산 기법 중 하나로, TCP 세그먼트의 재조합으로 인해 발생하는 통신 오버헤드를 줄일 수 있어 효율적이다. 본 논문에서는 네트워크 트래픽의 특성과 각 노드의 성능을 고려하여 플로우 기반 부하 분산이 효율적으로 이루어질 수 있는 방안을 제안한다.

1. 서론

오늘날 정보화가 급속도로 진전됨에 따라 정보의 저장·생산 및 가공의 대부분이 전산화되어 이루어지고 있다. 이에 국가 및 산업 주요 시설의 네트워크 의존도가 매우 높아졌으며 따라서 기밀 정보 침해와 정보처리 시스템 안정성에 대한 위협이 네트워크를 통해 이루어질 가능성이 크게 증가하였다.

이러한 위협에 대처하기 위해 Network Intrusion Detection System(NIDS)에 대한 연구가 1990년대부터 이루어져 현재는 네트워크 보호에 있어서 중요한 역할을 담당하게 되었다. NIDS는 네트워크 트래픽을 모니터링하고 분석하여 보호 대상 네트워크 외부로부터의 공격 뿐만 아니라 내부에서의 공격까지 탐지하고 이에 대응할 수 있는 시스템이다.

한편 최근에는 네트워크 기반 시설의 발달 속도가 가파르게 증가함으로 인해 차세대 대규모 네트워크에서의 침입 탐지가 효율적이면서도 정확하게 이루어지게 하기 위한 연구가 이루어지고 있다. 대규모 네트워크에서 탐지해야 할 트래픽의 양은 매우 방대하므로 단일 시스템으로 처리하기 불가능하여 다수의 침입 탐지 노드의 병렬 처리에 의한 분석이 이루어져야 하는데, 이를 위해서는 트래픽이 각 노드로 누락 없이 효율적으로 분산되어야 한다. 플로우 기반 부하 분산 기법은 이러한 부하 분산 기법 중 하나로, 같은 플로우에 속하는 트래픽이 다수의 노드로 흩어지지 않도록 하여 분석의 효율 및 정확성을 높일 수 있다. 본 논문에서는 네트워크 트래픽의 특성과 각 노드의 성능을 고려하여 플로우 기반 부하 분산이 효율적으로 이루어질 수 있는 방안을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 해시 기반 부하 분산 기법과 그 연구 사례를 소개하고, 기존 연구의 문제점을 분석한다. 또한 제안하는 기법에 활용된 네트워크 트래픽의 특성을 기술한다. 3장에서는 제안하는 시스템의 전체 구조와 주요 동작 방식에 대해 설명한다. 마지막으로 4장에서 본 논문의 결론을 맺는다.

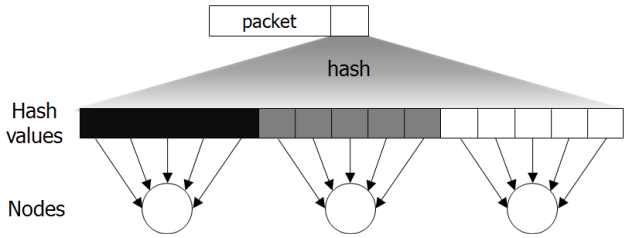
2. 관련 연구

2.1. 해시 기반 부하 분산 기법

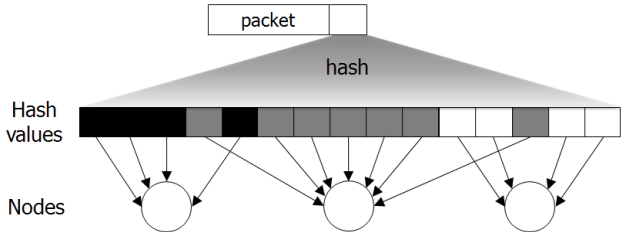
플로우 기반 부하 분산을 위한 기술로 해시 기반 부하 분산 기법이 있다. 하나의 플로우는 송신자 IP 주소와 포트 번호, 수신자 IP주소와 포트 번호, 전송 프로토콜 정보로 식별이 되는데, 이 기법에서는 이러한 식별 정보를 입력으로 하여 생성된 해시 값을 통해 해당 플로우가 탐지될 노드가 정해진다. 일반적인 해시 연산의 결과값은 직접적으로 적용하기에 그 수의 범위가 너무 크므로, mod 연산을 통해 원하는 만큼 그 범위를 줄인 해시 값을 사용하게 된다. 해시를 사용하면 해시 함수의 특성상 임의의 입력에 대해 출력 값이 정해진 범위 내에서 고르게 분포하므로 각 노드로의 부하 분산을 공평하게 할 수 있으며, 동일 플로우 식별 정보에 대한 해시 값이 항상 일정하므로

플로우 기반 부하 분산이 가능하다.

해시 기반 부하 분산 기법은 크게 정적 기법과 동적 기법으로 나뉜다. 정적 기법은 (그림 1)과 같이 각 노드에 동일한 개수의 해시 값을 할당하고 바꾸지 않는다. 이에 반해 동적 기법은 (그림 2)와 같이 경우에 따라 해시 값과 노드 간의 매핑을 변화시킬 수 있는데 이를 재배치라 한다. 따라서 필요시 각 노드로의 분산을 조정할 수 있다[1].



(그림 1) 정적 해시 기반 부하 분산 기법



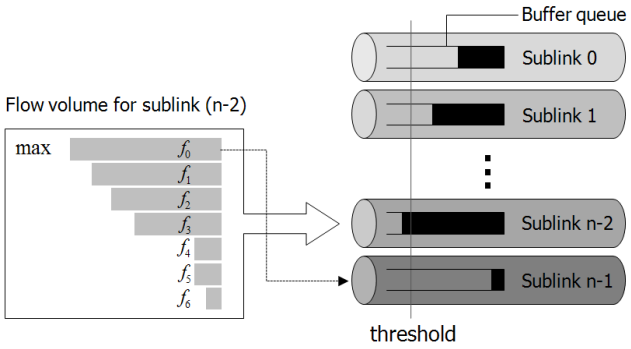
(그림 2) 동적 해시 기반 부하 분산 기법

2.2. 동적 해시 기반 부하 분산 기법의 연구 사례

병렬 구조 NIDS에서 동적 해시 기반 부하 분산 기법을 사용한 예로 SPANIDS가 있다. SPANIDS에서 부하 분산을 담당하는 부하 분산기(Load balancer)는 각 해시 값과 노드 간의 매핑 정보를 저장하기 위한 해시 테이블을 가지며 이 테이블내의 엔트리를 조작함으로써 동적으로 부하 분산을 조정할 수 있다. 이 NIDS를 이루고 있는 각 노드들은 자신의 입력 버퍼를 감지하고 있다가 버퍼에 존재하는 트래픽의 양이 임계치를 벗어나면 플로우 컨트롤 메시지를 보낸다. 부하 분산기는 이 메시지를 받았을 경우 해당 노드에 매핑 된 해시 값 중 임의의 하나를 다른 노드로 재배치 시킨다. 이로 인해 플로우 컨트롤 메시지를 보낸 노드로의 트래픽 양이 줄어들어 패킷 손실을 막을 수 있다 [1].

Dynamic hashing with flow volume (DHFV)는 인터넷 환경에서의 동적 해시 기반 부하 분산 기법 중 하나로 플로우 크기(Flow volume)를 이용하여 기존의 동적 해시 기반 분산 기법인 ECMP, OSPF-OMP 등에 비해 패킷 손실률을 현저히 낮추었다. 여기서 플로우 크기은 한 플로우의 단위 시간당 트래픽 양을 뜻하는 것으로 IP 헤더내의 패킷 크기 정보를 이용해서 구한다 [2].

그림은 DHFV가 동작하는 방식을 나타낸다. DHFV에서는 해시 값의 범위를 충분히 크게 하여 각 플로우와 해시 값이 거의 일대일로 매핑 되도록 하고, 일정 주기마다 각 노드와 연결된 링크로의 버퍼를 감지하여 트래픽의 양이 임계치를 벗어난 링크를 찾는다. 그러한 링크를 찾았다면 그 링크에 할당된 각 플로우의 플로우 크기를 계산하여 해당 주기의 최대 플로우 크기를 가지는 플로우를 찾아내어 다른 링크로 재배치 시킨다.



(그림 3) DHFV의 동작 방식

선행 연구된 기술들은 위에서 설명한 것과 같이 단순히 특정 노드로의 트래픽 집중을 발견한 후에 대응하는 방식으로 동작하므로 네트워크 트래픽의 특성과 노드의 성능을 충분히 반영한다고 보기 힘들다. 또한 상황이 급변하는 네트워크 환경에서의 부하 분산에 있어서 패킷 손실을 온전히 막기 어렵다. 본 논문에서는 선행된 계산을 통해 트래픽이 각 노드의 성능에 맞추어 고르게 분산될 수 있는 방안을 제시한다.

2.3. 네트워크 트래픽의 특성

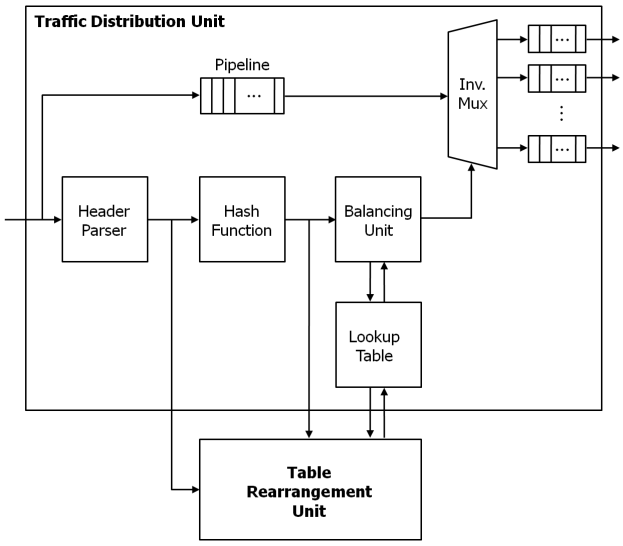
Carey Williamson은 2001년 인터넷 상의 트래픽을 측정하여 분석한 결과로 네트워크 트래픽의 주요 특성을 정리하였다[3]. 다소 오래 전 연구이지만 네트워크의 기반을 이루는 프로토콜이 거의 변하지 않았고 이러한 프로토콜에 의해 트래픽의 특성이 대부분 결정된다고 볼 수 있기에 충분히 고려할 만하다. 플로우 기반 부하 분산을 위해 고려할 수 있는 네트워크 트래픽의 특성을 요약하면 다음과 같다.

- 네트워크 상에서 패킷의 크기는 양극화된 분포를 지닌다. 네트워크 상에 존재하는 패킷의 약 50%가 MTU에 의한 최대 패킷 크기를 가지며 약 40%가 헤더만 있어 최소 패킷 크기를 가진다.
- 플로우를 기반하여 보았을 때 트래픽은 매우 비균일한 분포를 지니는데, 이는 소수의 특정 호스트들이 대부분의 트래픽을 발생시키기 때문이다.
- 특정 플로우의 크기는 일시적 국지성을 가진다. 네트워크 트래픽을 분석한 다른 연구에서 50ms를 한 주기로 하여 각 플로우의 크기를 측정한 후 비교한 결과 N번째 주기의 플로우 크기가 클수록 N+1번째 플로우 크기가 비슷해 연관성이 매우 높은 것으로 나타났다. 이는 특정 주기에 플로우의 크기가 클수록 다

음 주기에 그 플로우가 유지될 확률이 높다는 것을 의미한다[2].

3. 제안 방법

본장에서는 앞서 소개한 네트워크 트래픽의 특성을 활용하고 각 노드의 성능을 고려하여 병렬 구조 NIDS에서 플로우 기반 부하 분산을 효율적이고 정확성 있게 수행하는 방안을 제시한다.



(그림 4) 부하 분산기의 구조

(그림 4)는 제안하는 부하 분산기의 구조를 나타낸다. 이는 Gero Dittmann이 소개한 부하 분산기를 응용한 것으로, 테이블 재배치 유닛(Table rearrangement unit)을 추가하여 각 주기마다 테이블 재배치에 필요한 계산이 이루어지게 하였다 [4].

트래픽 분산 유닛(Traffic distribution unit)은 유입된 패킷들의 플로우 정보로 생성된 해시 값에 매칭되는 노드를 검색 테이블(Lookup Table)로부터 찾아내어 해당 노드로 트래픽을 포워딩한다.

테이블 재배치 유닛은 트래픽 분산을 수행하는 트래픽 분산 유닛의 성능에 영향을 미치지 않도록 독립적인 프로세서로서 동작하게 되며, 각 패킷의 크기 정보와 각 플로우의 해시 값을 입력받아 효율적인 분산에 필요한 계산을 한 후 주기적으로 검색 테이블을 업데이트한다.

<표 1>은 테이블 재배치 유닛의 동작과정을 설명하기 위해 필요한 변수 및 용어를 나타낸 표이다. 테이블 재배치 유닛에서는 패킷 크기 정보를 이용해 각 플로우 f_i의 크기를 주기적으로 계산하는데, 크게 두 분류로 나누어 구한다. 첫째는 플로우의 일시적 국지성을 활용하기 위해 비교적 짧은 시간 간격 P_s동안 크기를 측정한 값 l_i이며 둘째는 일정 시간동안 각 노드로 포워딩된 플로우 크기를 계산하기 위해 시간 간격 P_l ($= t \bullet P_s$) 동안 측정한 값 s_i이다.

테이블 업데이트는 시간 간격 P_l마다 일어나는데, 이때 다음과 같은 계산이 이루어진다. 먼저 각 노드에 속하는 플로우들의 l_i들을 모두 더하여 $l_{node(j)}$를 구한다. 또한 이렇게 구해진 l_{node0} ~ $l_{node(N-1)}$과 이들의 총합 l_{all}을 이용하여 해당 주기 동안 각 노드에 포워딩 된 트래픽의 비율 $T_{node(j)}$을 구한다. 이후 각 노드의 $T_{node(j)}$와 $W_{node(j)}$를 비교하여 $T_{node(j)}$가 $W_{node(j)}$보다 크면 해당 노드의 플로우들 중 적절한 수를 취하여 $T_{node(j)}$가 $W_{node(j)}$보다 작은 노드로 재배치한다. 이때 재배치할 플로우를 정하기 위해 각 플로우의 s_i가 사용된다.

<표 1> 용어 및 변수 정리

용어/변수	내용
N	노드의 개수
m	해시 값의 범위 (해시 값은 0 ~ $m-1$의 범위를 가짐)
f_i	플로우i (플로우 정보에 대한 해시 연산의 결과를 mod m 연산한 값이 i임을 의미)
$node_i$	노드 i
P_s	단주기 플로우 크기 (Short period flow volume)를 구하기 위한 시간 간격
P_l	장주기 플로우 크기 (Long period flow volume)를 구하기 위한 시간 간격 (=$t \bullet P_s$) 및 테이블 업데이트 주기
t	P_l은 P_s의 배수이며 정수 t를 곱한 값임
l_i	f_i의 장주기 플로우 크기
$l_{node(j)}$	노드 j에 속하는 각 플로우의 l_i를 모두 더한 값
l_{all}	l_{node0} ~ $l_{node(N-1)}$을 모두 더한 값
s_i	f_i의 단주기 플로우 크기
$W_{node(i)}$	노드 i의 성능을 나타내는 가중치 값
$T_{node(i)}$	노드 i에 포워딩된 트래픽의 비율을 나타내는 가중치 값
W	W_{node0} ~ $W_{node(N-1)}$을 모두 더한 값, 또는 T_{node0} ~ $T_{node(N-1)}$을 모두 더한 값

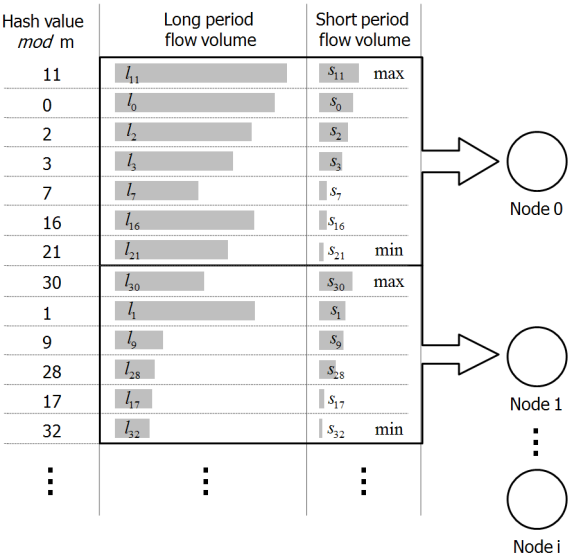
네트워크 트래픽의 특성상 짧은 시간 간격동안 현재 주기의 플로우 크기가 클수록 다음 주기에도 그 플로우가 유지될 가능성이 높다. 그러므로 플로우 기반 분산을 위해서는 이전 주기의 플로우가 그 크기가 클수록 다른 노드로 재배치 시키지 않는 것이 좋다. 따라서 재배치할 플로우를 정하기 위해 각 노드 별로 플로우를 s_i크기 순으로 정렬하여 다음 주기에 플로우가 유지될 가능성이 가장 적은 순, 즉 s_i가 작은 순으로 재배치가 일어나도록 하였다.

또한 재배치될 플로우들의 l_i와 재배치 대상 노드들의 $W_{node(j)}$, $T_{node(j)}$ 간의 차를 이용한 계산을 통해 각 플로우와 노드간의 매핑이 최종적으로 정해지며 이를 기반으로 테이블이 업데이트된다. 이에 대한 구체적인 계산법 및 앞서 설명한 내용들에 대한 상세 알고리즘은 다음에 나오는 의사 코드에 명시되어 있다.

다음은 테이블 재배치 유닛이 주기 P_l마다 수행하는 동작 과정을 나타낸 알고리즘이다.

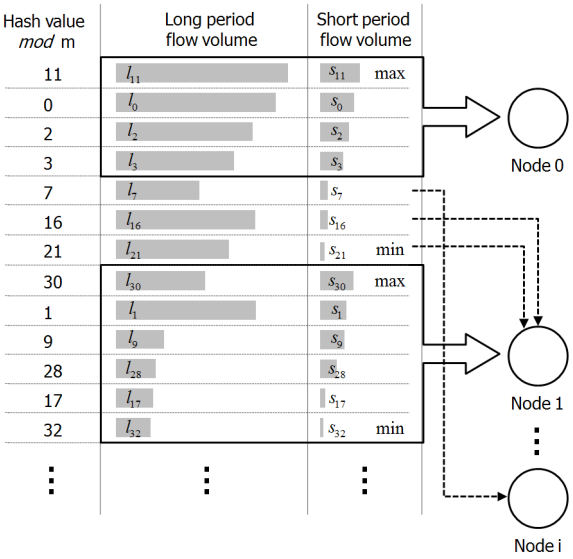
```
for i = 0 to m−1
    l_all += l_i
unit_w = ⌊ l_all/w ⌋
for j = 0 to N−1
    for all f_i s in node_j
        l_node(j) += l_i
    T_node(j) = ⌊ l_node(j)/unit_w ⌋
    if T_node(j) − W_node(j) > 1
        E_node(j) = 0, l_sum = 0, i = 0
        sort f_i s in node_j with s_i by ascending order and
        insert them into fsorted[]
        while l_sum < (l_node(j) − W_node(j) • unit_w)
            l_sum += l_fsorted[i]
            insert fsorted[i] into queue_f
            i++
    else if T_node(j) − W_node(j) < −1
        E_node(j) = W_node(j) • unit_w − l_node(j)
    else E_node(j) = 0
while queue_f is empty
    dequeue f_i from queue_f
    for j = 0 to N−1
        if E_node(j) > l_i
            rearrange f_i to node_j
            E_node(j) −= l_i
```

(그림 5)와 (그림 6)은 테이블 재배치 유닛의 동작과정의 예를 보여주기 위한 그림이다. 본 예에서 NIDS를 구성하는 노드의 수 N은 4, 전체 성능 가중치 W는 20이며 $node_0$와 $node_1$의 성능 가중치 W_{node0}, W_{node1}은 5로 같다고 가정한다.



(그림 5) 특정 주기에서 재배치가 일어나기 전

(그림 5)에서 특정 주기 동안 계산된 각 플로우들의 l_i와 s_i를 볼 수 있다. 여기서 l_{node0}은 550, l_{node1}은 200이며 l_{all}은 1600이라고 했을 때 이 주기 동안 $node_0$으로 포워딩된 트래픽의 비율이 해당 노드의 성능 가중치에 비해 상대적으로 많으며, $node_1$으로 포워딩된 트래픽은 상대적으로 적다는 것을 알 수 있다. 따라서 $node_0$에 속한 플로우들을 s_i가 작은 순으로 정렬을 하여 $node_1$와 트래픽 비율이 상대적으로 적었던 그 외 노드들에게 순차적으로 재배치시킨다. (그림 6)은 재배치가 이루어지고 난 후 플로우와 노드간의 매핑 상태를 나타낸 것이다.



(그림 6) 특정 주기에서 재배치가 일어난 후

4. 결론

본 논문에서는 단순히 특정 노드로의 트래픽 집중을 발견한 후에 대응하는 방식으로 동작하는 기존의 해시 기반 부하 분산 기법을 발전시켜 네트워크 트래픽의 특성과 각 노드들의 성능을 반영하여 병렬 구조 NIDS를 위한 부하 분산이 이루어질 수 있는 방안을 제시하였다. 하지만 효과적인 부하 분산을 위한 P_s과 P_l등의 크기, 각 노드의 성능 가중치, 해시 값의 범위 등에 대해서는 구체적으로 언급하지 않았다. 향후 연구를 통해 이러한 사항을 보완하여 시스템을 구현하고 기존 기법들과의 비교 분석을 통해 성능을 입증할 계획이다.

참고문헌

[1] Lambert Schaelicke, et. al., "SPANIDS: A Scalable Network Intrusion Detecion Loadbalancer", Conference On Computing Frontiers, New York, USA, 2005

[2] Ju-Yeon Jo, et. al., "Internet Traffic Load Balancing using Dynamic Hashing with Flow Volume", In Proc. SPIE ITCom, 2002

[3] C. Williamson, "Internet Traffic Measurement", IEEE Internet Computing, 2001

[4] G. Dittman, et. a.l, "Network Processor Load Balancing for High-Speed Links", in International Symposium on Performance Evaluation of Computer and Telecommunication System, 2002