연결설정 지연 단축을 위한 바이러스 쓰로틀링의 가변 비율 제한기

심 재 홍†

요 약

연결요청(connection request) 패킷의 전송비율을 일정 비율 이하로 제한함으로써 웜 발생을 탐지하는 바이러스 쓰로틀링(virus throttling)은 대표적인 웜 조기 탐지 기술 중의 하나이다. 기존 바이러스 쓰로틀링은 비율 제한기의 주기를 고정시키고 지연 큐 길이를 감시하여 웜 발생 여부를 판단한다. 본 논문에서는 가중치 평균 지연 큐 길이를 적용하여 비율 제한기의 주기를 자율적으로 조절하는 알고리즘을 제안하고, 가중치 평균 지연 큐 길이에 따른 다양한 주기결정 기법을 제시한다. 실험결과 제안 알고리즘은 웜 탐지시간에는 크게 영향을 미치지 않으면서도 연결설정 지연시간을 단축하여 사용자가 느끼는 불편함을 줄여 줄 수 있음을 확인하였다.

키워드 : 바이러스 쓰로틀링, 웜 조기 탐지, 인터넷 웜, 연결설정 지연

Variable Rate Limiter in Virus Throttling for Reducing Connection Delay

Jaehong Shim†

ABSTRACT

Virus throttling technique, one of many early worm detection techniques, detects the Internet worm propagation by limiting the connect requests within a certain ratio. The typical virus throttling detects worm occurrence by monitoring the length of delay queue with the fixed period of rate limiter. In this paper, we propose an algorithm that controls the period of rate limiter autonomically by utilizing the weighted average delay queue length and suggest various period determination policies that use the weighted average delay queue length as an input parameter. Through deep experiments, it is verified that the proposed technique is able to lessen inconvenience of users by reducing the connection delay time with having just little effect on worm detection time.

Key Words : Virus Throttling, Worm Early Detection, Internet Worm, Connection Delay

1. 서 론

인터넷을 기반으로 하는 많은 웜[1-5]들은 스스로 전파되는 속성을 가지고 있으며, 이는 주로 인터넷상에 존재하는 호스트의 응용 프로그램들의 취약성을 이용한다. 따라서 웜에 감염된 호스트는 취약성이 있는 또 다른 호스트를 찾기 위해서 대상 호스트의 IP 주소를 무작위로 생성하여 스캐닝을 하고, 스캐닝한 정보를 이용하여 다른 호스트에 웜 코드를 전파한다.

일반적으로 웜은 가능한 많은 다른 시스템에 자기 자신을 복제하고 전파시키는 행위를 반복한다. 예를 들어, 취약성을 가진 시스템을 탐색하기 위한 목적으로 Nimda[1]는 초당 300~400개, SQLSlammer[2]는 초당 850개, Code Red[3]는 초당 약 200개의 연결요청을 위한 패킷을 전송한다. 이들은 대부분 일정한 주기로 연결요청 패킷을 전송하지만 Code Red II[4]처럼 초당 패킷 발생 비율이 동일하더라도, 한번에 많은 패킷을 발생시킨 후 일정시간 동안 패킷을 발생시키지 않다가 다시 많은 패킷을 발생시킬 수도 있다. 또한 I Love You[5]와 같은 많은 전자우편 바이러스들은 감염된 시스템에서 찾을 수 있는 모든 전자우편 주소로 메일을 전송한다.

이처럼 스스로 전파되는 웜이 유해한 데이터를 전달하지 않을지라도 웜의 전파과정에서 발생하는 엄청난 양의 트래픽은 네트워크의 성능을 저하시킬 뿐 아니라, DOS 공격 등도 가능하게 한다[6]. 또한 웜의 전파 속도가 워낙 빠르기 때문에 사용자들이 일일이 그것에 대항하여 반응하기에는 역부족이다[7]. 따라서 새로운 웜이 발생하였을 때 웜의 발생 여부를 빠르게 탐지하여 더 이상의 웜 전파를 차단하는 것이 중요하다.

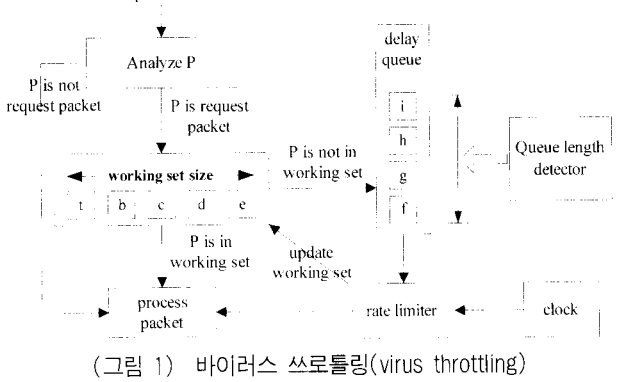
이에 비해 정상적인 트래픽은 상당히 낮은 비율(초당 약

† 정 회 원 : 조선대학교 컴퓨터공학부 조교수
논문접수 : 2006년 6월 2일, 심사완료 : 2006년 9월 5일

1개)로 연결요청이 이루어지며, 또한 연결 대상 시스템이 웜처럼 불특정 다수가 아닌 소수의 특정 시스템들로 국한되어 있다. 따라서 웜과 정상 트랙픽과의 이 같은 속성 차이를 이용하면 쉽게 웜 발생을 탐지할 수 있다.

바이러스 쓰로틀링[8,9]은 새로운 세션의 연결(connection) 비율을 제한하여 웜의 전파 속도를 늦추고 차단하는 기법이다. 웜에 감염되지 않은 정상적인 호스트에서 이루어지는 새로운 연결들은 보통 낮은 비율로 이루어지나, 웜은 상대적으로 높은 비율로 연결요청을 시도한다. 바이러스 쓰로틀링은 정상 트래픽과 웜 트래픽의 이러한 차이점을 이용하여 세션의 연결요청 패킷들을 지연시키면서 단위 시간당 전송되는 연결요청 개수를 인위적으로 제한함으로써, 웜의 전파 속도를 지연시키는 것은 물론 웜 탐지 시 더 이상의 전파를 차단한다.

(그림 1)은 바이러스 쓰로틀링 기법에 의해 제어되는 전송 패킷들의 흐름을 나타낸 것이다. 그림에서 워킹 셋(working set)은 기존에 전송된 패킷들의 수신 IP 주소들을 일정 개수(일반적으로 5개)만큼 FIFO방식으로 보관하고 있는 자료 구조체이다. 새로운 연결요청 패킷(P)의 전송요청이 상위 계층에서 내려오면 워킹 셋에서 P와 동일한 수신 IP 주소가 존재하는지 확인하고, 만약 존재한다면 P를 정상 트래픽으로 간주하여 지연 없이 즉시 전송한다. 그렇지 않은 경우 P를 지연 큐(delay queue)에 저장한다. 즉, 상대적으로 최근에 접속이 이루어졌던 호스트에 대해서는 지연 없이 바로 연결요청 패킷을 전송하고 그렇지 않은 호스트에 대해서는 패킷을 지연 큐에 보관한 후 적당한 시기에 비율 제한기(rate limiter)에 의해 전송되게 한다. 비율 제한기는 일정 시간 간격을 두고 주기적으로 지연 큐에서 제일 오래된 패킷을 꺼내어 이를 전송한다. 이때 이 패킷과 동일한 수신 IP 주소를 가지는 지연 큐 내의 다른 패킷들도 동시에 전송한다. 비율 제한기는 매 패킷을 처리할 때마다 해당 패킷의 수신 IP 주소를 워킹 셋에 추가한다. 마지막으로 지연 큐 길이 감시자(queue length detector)는 패킷이 지연 큐에 저장될 때마다 지연 큐의 길이를 검사하여 사전에 정의된 경계값(threshold: 일반적으로 큐 크기임)을 초과하면, 웜이 발생하였다고 판단한다. 일단 웜이 탐지 되면 시스템은 모든 패킷을 차단하여 더 이상의 웜 전파를 차단한다.



(그림 1) 바이러스 쓰로틀링(virus throttling)

기존의 바이러스 쓰로틀링에서는 비율 제한기의 주기를 1초로 고정시켰다. 따라서 외부 시스템으로 전송해야 할 새로운 세션을 위한 연결요청 패킷을 1초에 1개씩 전송하므로 (워킹 셋과 동일한 IP 주소는 즉시 전송), 이 경우 일시적으로 서로 다른 IP 주소를 가진 세션에 대한 연결 요청이 폭주할 경우 지연 큐에 늦게 삽입된 연결요청 패킷들은 상당한 지연시간을 가지게 되며, 사용자들은 갑자기 네트워크 부하가 심해져 속도가 떨어진 것과 같은 불편함을 겪게 된다.

따라서 본 논문에서는 비율 제한기의 주기를 고정시키고 지연 큐 길이만으로 웜 발생 여부를 판단하는 기존 바이러스 쓰로틀링의 방법을 탈피하고, 비율 제한기의 주기를 능동적으로 조절하는 방법을 제안하여 웜 탐지시간에는 크게 영향을 미치지 않으면서도 사용자가 느끼는 세션의 연결설정 지연시간을 단축하고자 한다. 또한 가변 비율 제한기의 다양한 주기 결정기법들을 제시하고 이들의 성능을 비교 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 웜 발생 탐지를 위한 기존 연구를 분석하고, 3장에서 비율 제한기의 주기를 고정시켰던 기존 바이러스 쓰로틀링 기법 대신 주기를 능동적으로 변경할 수 있는 가변 비율 제한기을 제안하고, 다양한 주기결정 전략을 제시한다. 4장에서는 제안된 가변 비율 제한기의 효율성을 검증하기 위한 다양한 실험결과를 분석한다. 마지막 5장에서는 본 논문의 결론과 향후 연구방향을 제시한다.

2. 관련연구

웜 조기 탐지(worm early detection) 기술은 바이러스 쓰로틀링(virus throttling)[8,9], 수신-송신 상호관계(DSC: destination-source correlation)를 이용한 시스템[10], 순차적 가설 시험 (sequential hypothesis testing)을 이용한 알고리즘[11,12] 등이 있다. Xinzhou Qin[10]은 패킷의 수신 포트와 송신 주소를 슬라이딩 윈도우를 이용하여 추적하며, 동일한 송신 주소에서 동일한 수신 포트로 패킷을 보내는 횟수를 계산하여 웜을 탐지하는 DSC 방법을 제안하였다. Jaeyeon Jung은 순차적 가설 시험을 이용한 온라인 웜 탐지 알고리즘[11]을 제안하였고, 또한 이 알고리즘과 연결 비율 제한(connection rate limiting)을 동시에 사용하는 하이브리드 접근 방법[12]을 제안하였다.

그러나 위에서 소개한 방법들은 경계값-기반(threshold-based) 기술로 탐지의 오판(false alarm) 가능성이 높다는 단점을 가지고 있다. 보통 탐지 오판은 경계값(threshold)을 어떻게 주는가에 따라 성능상의 차이가 발생하는데, 오판율을 낮추기 위해서는 경계값을 좀더 높게 설정하여야 한다. 그러나 이는 결국 웜 탐지 성능을 저하시키기 때문에 동일한 경계값을 가지고도 오판율을 낮추는 것은 아주 중요한 문제이다. C. Zou[13]는 경계값-기반 기술의 경우 오판 가능성이 높다는 단점을 지적하고, 동적 검역 방어 (dynamic quarantine defense)라는 기법을 사용하여 오판율을 낮추는

방법을 제안하였다. 또한 경계값-기반 변이 탐지(threshold-based anomaly detection) 시스템에 트래픽 추세(trend)라는 개념을 칼만-필터(Kalman-filter)를 통해 추가하여 웜 탐지 오판율을 줄이는 방법에 대해 발표하였다[14].

본 연구와 직접적으로 관련된 연구로는 기존 바이러스 쓰로틀링의 지연 큐 관리에 있어 정상 트래픽의 연결요청 패킷들에 대해 지역성[15]을 반영하지 못한다는 단점을 지적하고, 이를 해결하기 위해 지연 큐의 구조를 일차원에서 이차원으로 변경하여 동일한 수신 IP 주소를 지연 큐 길이 산정 시 중복하여 계산하지 않음으로써 웜 탐지 오판을 줄이는 알고리즘을 제안하였다[16]. 또한 동일한 크기의 지연 큐를 가지고도 웜 탐지시간을 줄이고 전송된 웜 패킷 수를 줄일 수 있는 새로운 웜 탐지 알고리즘을 제안했으며[17], 이는 지연 큐 길이 산정 시 현재 큐 길이뿐 아니라 과거 큐 길이와 향후 트래픽 경향을 반영함으로써, 웜의 발생 가능성을 사전에 예측하여 보다 빠른 시간 내에 웜을 탐지할 수 있는 방안을 제시하였다. 그러나 기존 바이러스 쓰로틀링의 연구는 웜 탐지시간 단축 또는 웜 탐지 오판율 감소에 초점을 두어 연구가 진행되었고, 바이러스 쓰로틀링으로 인한 연결설정 지연과 관련한 연구는 진행된 바가 없다. 따라서 본 연구에서는 웜 탐지시간을 단축하는 것이 아니라, 평상시의 정상 패킷에 대해 바이러스 쓰로틀링으로 인한 연결설정 지연시간을 단축하고자 하는 것이며, 반면에 웜 탐지시간은 기존 연구에서의 웜 탐지시간과 거의 비슷하게 하는 것이 목적이다.

3. 가변 비율 제한기와 주기 결정 전략

바이러스 쓰로틀링은 새로운 연결요청 패킷일 경우 이를 바로 전송하는 것이 아니라 지연 큐에 보관했다가 비율 제한기가 주기적으로 지연 큐에 저장된 패킷을 하나씩 빼내어 전송한다. 따라서 연결요청 패킷은 비율 제한기로 인해 최소한의 연결설정 지연을 가지며, 비율 제한기의 주기를 어떤 값을 설정하는가에 따라 웜 탐지시간 및 연결설정 지연시간이 달라 질 수 있다.

주기가 짧을 경우 빠른 속도로 연결요청 패킷이 외부로 전송되므로 연결 지연시간이 짧아지고 사용자는 연결설정 요구시 속도 저하를 느끼지 못한다. 그러나 지연 큐의 패킷이 빠른 속도로 빠져 나가기 때문에 웜 발생시 큐 길이가 경계값에 도달하기 위해서는 더 많은 시간이 소요되므로 웜 탐지시간이 길어지는 문제점이 있다.

반면, 주기가 길어질 경우 패킷들이 지연 큐에 존재하는 시간이 길어지면서 상대적으로 빠르게 큐 길이가 경계값에 도달하므로 웜 탐지시간은 빨라진다. 그러나 사용자들이 서로 다른 시스템들과 새로운 세션에 대한 연결요청을 동시에 많이 할 경우 연결설정 지연시간이 길어져 불편함을 겪게 된다. 즉, 네트워크 속도가 갑자기 저하된 듯한 느낌을 갖게 된다. 사용자에게는 바이러스 쓰로틀링의 설치로 인한 접속 지연이나 불편함을 가능한 느끼지 않게 해야 한다. 또한 지연 큐가 빠르게 증가하기 때문에 동일한 경계값에 대해 정상적인 트래픽임에도 불구하고 웜이 발생한 것으로 오판할 확률도 상대적으로 증가한다.

비율 제한기의 주기를 적절한 값으로 고정하는 것도 중요하지만 현재의 트래픽의 상태를 정확히 파악하여 주기를 적절히 조절할 경우 웜 탐지시간에는 큰 영향을 주지 않으면서 연결설정 지연을 단축해 줄 수 있다. 일반적으로 정상 트래픽의 경우 지연 큐 길이가 일시적으로 증가하였다가 서서히 감소하는 과정을 반복하지만, 웜 트래픽은 큐 길이의 증가 또는 감소 없이 지속적으로 증가하거나 또는 큐 길이가 증가 및 감소를 반복하면서 지속적으로 증가하기도 한다. 따라서 이러한 두 트래픽의 차이를 활용하면 효율적으로 비율 제한기의 주기를 조절할 수 있다.

예를 들어, 지연 큐 길이가 증가하고 있을 경우 웜인지 정상 트래픽인지 판단할 수 없으므로 기존 바이러스 쓰로틀링과 같이 비율 제한기의 주기를 고정시킨다. 만약 큐 길이가 꾸준히 계속 증가하면 웜일 가능성이 크고, 증가하다가 감소하기 시작하면 일시적인 트래픽 증가일 가능성이 높다. 증가하던 지연 큐 길이가 갑자기 감소하기 시작하면 이는 일시적인 트래픽 증가로 판단할 수 있으며, 이 경우 비율 제한기의 주기를 서서히 감소시켜 지연 큐 내에 대기 중이던 연결요청 패킷을 점점 빠르게 외부로 전송한다. 그래도 계속해서 큐 길이가 감소하면 비율 제한기의 주기를 더 감소시켜 보다 빠르게 연결요청 패킷을 전송한다. 만약 큐 길이가 다시 증가하기 시작하면 웜 탐지시간에 영향을 미치지 않게 비율 제한기의 주기를 원래 주기로 바로 복원한다.

이러한 아이디어를 구현하기 위해서는 현재의 지연 큐 길이가 지속적으로 증가하고 있는지 아니면 감소하고 있는지를 판단할 수 있어야 한다. 일반적으로 지연 큐 길이는 현재의 트래픽 상황에 따라 빠르게 증가 또는 감소를 반복하기 때문에 전체적으로 큐 길이가 증가하거나 감소하는지를 판단하기 어렵다. 따라서 본 논문에서는 널리 잘 알려진 지수 평균(exponential average) 수식을 활용하여 가중치 평균 큐 길이(weighted average queue length: WAQL)를 정의하고 이를 활용한다. 수식은 다음과 같다.

$$WAQL_n = a * avgQL_n + (1 - a) * WAQL_{n-1}$$

수식에서 $WAQL_{n-1}$은 직전의 주기에서 구해진 가중치 평균 큐 길이이며, $avgQL_n$은 현 샘플링 주기 동안의 평균 큐 길이이고, a는 새로운 $WAQL_n$을 계산할 때 현 주기의 $avgQL_n$과 과거의 $WAQL_{n-1}$을 반영하는 비율을 결정하는 가중치 상수이다. 가중치 평균 큐 길이는 최근의 트래픽 상황과 이전의 트래픽 상황을 적절히 반영시킬 수 있으므로 지속적으로 큐가 증가하는지 아니면 감소하는지를 쉽게 판단할 수 있는 근거를 제공한다.

(그림 2)는 본 논문에서 제안하는 가변 비율 제한기의 주기결정 알고리즘이다. 제안 알고리즘은 샘플링(sampling time: 일반적으로 1초) 때마다 한번씩 수행된다. 알고리즘에서 downWAQL는 가중치 평균 큐 길이가 증가하다가 감소

```
WAQL = ALPHA * avgQL + (1 - ALPHA) * WAQL;
if (WAQL >= prevWAQL)  /* 큐 길이가 계속 증가 중임 */
     RLP = 1;          /* 1 sec:원래 주기 값으로 복원 */
else {                 /* 큐 길이가 감소 중임  */
     if (RLP == 1)     /* 큐 길이가 감소하기 시작했음 */
          downWAQL = prevWAQL;
     RLP = GetRLP() * 0.5 + 0.5;
     prevWAQL = WAQL;
}
```

(그림 2) 가변 비율 제한기의 주기결정 알고리즘

하기 시작한 시점(감소시점)의 WAQL 값을 가지며, 이는 향후 실제 주기결정 함수인 GetRLP() 함수에서 사용된다. RLP는 GetRLP()에 의해 결정된 비율 제한기의 새로운 주기 값이다.

제안 알고리즘은 샘플링 때마다 가중치 평균 큐 길이를 계산한 다음, 큐 길이가 증가하고 있는 추세면 비율 제한기의 주기를 원래의 주기(기존 바이러스 쓰로틀링의 주기)인 1초로 복원한다. 그러나 큐 길이가 감소하고 있는 경우엔 주기결정 함수인 GetRLP()를 호출하여 비율 제한기의 주기를 새로이 결정한다. 새로 결정된 주기 RLP는 이후 비율 제한기에 의해 새로운 주기로 설정된다. GetRLP()는 현재의 WAQL를 참조해서 0~1까지의 상대적인 주기를 계산해서 반환하고, RLP는 0.5~1초 사이의 값을 가진다.

주기결정 함수인 GetRLP()는 다양한 방법으로 구현될 수 있다. 그러나 여기서 고려해야 할 사항은 WAQL가 지속적으로 감소할 경우 비율 제한기의 주기도 계속해서 감소할 것인가 아니면 처음에는 감소시키다가 일정 시점 이후에는 다시 증가할 것인가를 고려해야 한다. 왜냐하면 정상 트래픽은 지연 큐가 일시적으로 증가했다가 천천히 감소하여 큐 길이가 0이 되는 패턴을 반복하지만, 웜의 경우 지연 큐 길이가 지속적으로 증가하는 경우도 있지만 어떤 경우는 일시적 증가 후 조금 감소하고, 다시 증가 후 감소를 반복하면서 꾸준히 증가하는 추세를 보이는 경우도 있기 때문이다.

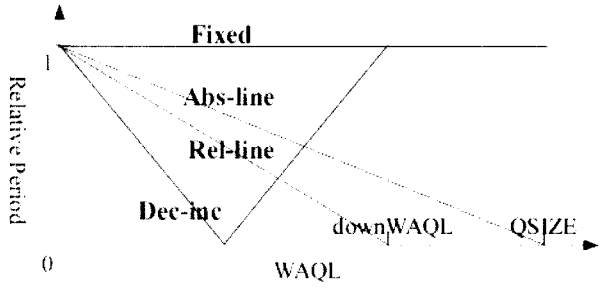
정상 트래픽이 일시적으로 연결 패킷이 증가한 이후 지속적으로 지연 큐 길이가 감소할 때 비율 제한기의 주기도 계속 감소할 경우, 큐에서 대기 중이던 연결요청 패킷이 빠르게 외부로 전송되어 연결설정 지연시간은 상당히 줄어들 수 있다. 그러나 지연 큐 길이가 증가 및 감소를 반복하면서 지속적으로 증가하는 웜일 경우, 큐 길이가 감소한다고 해서 비율 제한기의 주기를 지속적으로 줄여 주면 지연 큐 길이 증가율이 감소하여 웜 탐지시간이 반대로 증가할 수 있다.

따라서 주기결정 함수의 주기결정 전략은 웜 탐지시간과 연결설정 지연시간에 상당한 영향을 미친다. 본 논문에서는 직관적이면서 단순한 주기결정 전략들을 정의하고, 가변적으로 비율 제한기의 주기를 조절하고자 하는 아이디어가 타당성이 있는지를 실험을 통해 확인하고 또한 어떤 주기결정 전략이 더 효율적인지를 찾고자 한다.

<표 1>은 본 논문에서 정의된 주기결정 전략들을 보여주며, (그림 3)은 이를 그래프로 표현한 것이다. 각 전략에서 입력 매개변수는 WAQL이며, 나머지 값들은 상수이거나 이미 값이 결정된 변수이다. QSIZE는 지연 큐의 크기이며, 이는 바이러스 쓰로틀링의 지연 큐 길이 감시자가 웜 탐지를 위해 사용하는 경계값과 동일한 값이다. 각 전략은 QSIZE 또는 감소시점 downWAQL에 대한 WAQL의 상대적인 값을 계산해서 0~1 사이의 상대적 주기를 반환한다. Fixed 전략은 기존의 바이러스 쓰로틀링이 사용하는 전략으로서 지연 큐 길이와 상관없이 항상 고정된 값을 반환한다.

〈표 1〉 가변 비율 제한기의 주기 결정 전략

전략 이름	결정된 상대적 주기 값
Fixed	1
Abs-line	(QSIZE-WAQL) / QSIZE
Rel-line	(downWAQL-WAQL) / downWAQL
Dec-inc	abs(2*WAQL-downWAQL) / downWAQL



(그림 3) 주기결정 전략별 함수 그래프

Abs-line 전략은 지연 큐 크기에 대한 현재의 WAQL의 상대적인 크기 값과 반비례 한다. 즉, WAQL가 QSIZE와 비슷한 값일수록 0에 가깝다가 지속적으로 줄어들수록 1에 가까운 값을 반환한다. 이 전략의 주기결정 전략은 WAQL가 증가하다가 감소하기 시작한 감소시점 downWAQL과는 상관없이 WAQL가 절대적으로 크면 비율 제한기의 주기를 작고, 반대로 WAQL가 작으면 주기는 크게 설정된다. 따라서 WAQL가 지속적으로 줄어들면 이에 반비례하여 주기는 서서히 증가되어 원래의 주기로 복구되는 전략이다.

Rel-line 전략은 감소시점 downWAQL에 대한 WAQL의 상대적인 크기 값과 반비례 한다. 즉, WAQL가 감소하기 시작한 초기에는 0이었다가 이후 WAQL가 지속적으로 줄어들수록 1에 가까운 값을 반환한다. 이 전략은 감소시점인 downWAQL의 크기에는 상관없이 초기에는 최소 주기로 설정한 후 이후 WAQL가 지속적으로 줄어들면 주기를 증가시켜 원래 주기로 복구한다. 따라서 downWAQL이 작으면 빠르게 주기가 원래의 주기로 복귀하고, downWAQL이 크면 서서히 복귀하게 된다.

Dec-inc 전략은 WAQL가 초기 감소시점인 downWAQL에서 1이었다가 downWAQL의 중간 지점(반환점)까지 0으로 감소했다가 이후 WAQL가 지속적으로 줄어들면 다시 1로 증가한다. 즉, 주기를 처음부터 갑자기 감소시키는 것이 아니라, WAQL가 지속적으로 감소함에 따라 원래 주기에서 최소 주기로 감소한 후 다시 원래 주기로 복귀시키는 전략

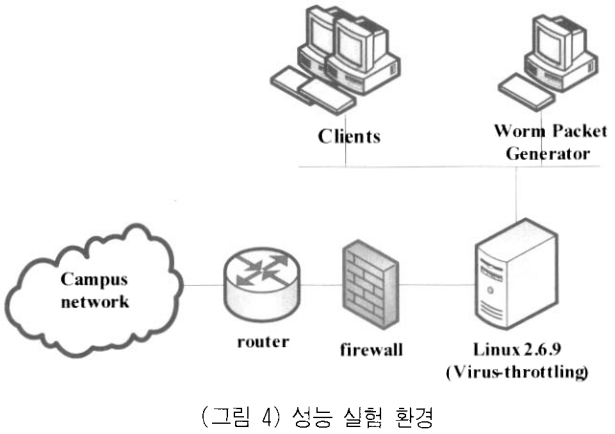
이다. 정상 트래픽은 지연 큐가 일시적으로 증가했다가 천천히 감소하여 큐 길이가 0이 되는 패턴을 반복한다. 그러나 웜의 경우 지연 큐 길이가 지속적으로 증가하는 경우도 있지만 어떤 경우는 일시적 증가 후 조금 감소하고, 다시 증가 후 감소를 반복하면서 꾸준히 증가하는 추세를 보이는 경우도 있다. 따라서 Dec-inc는 이런 웜에 대해 효과적으로 대처할 수 있다.

4. 성능 평가

비율 제한기의 주기를 변경하여 새로운 세션에 대한 연결 지연시간을 단축하고자 하는 제안 알고리즘의 성능을 알아보기 위해서 Linux 시스템에 바이러스 쓰로틀링과 제안된 알고리즘들을 구현하였다. Linux 커널 버전 2.6.9에 기존 알고리즘[9]을 참고하여 구현하였으며, (그림 4)와 같은 성능 실험 환경을 꾸몄다.

리눅스 시스템은 크게 2개의 모듈로 구성된다. 첫째가 커널에 구현된 바이러스 쓰로틀링 모듈로서, 리눅스에서 외부로 전달되는 모든 패킷들을 바이러스 쓰로틀링을 통하여 새로운 연결 비율을 조절하는 역할을 하며, 실험결과를 측정하기 위한 여러 통계 데이터도 수집하게 된다. 둘째는 웜 패킷 생성기(Worm Packet Generator)라는 것으로, 다양한 시간적 특성을 가지는 웜 패킷을 생성하거나 또는 정상 트래픽용 패킷을 생성하기 위해서 사용된다. 그리고 라우터에는 방화벽이 설치되어 있으며, 실험을 위해 생성되는 유해한 패킷들이 외부 인터넷으로 전달되지 않도록 설정하였다.

실험에서 가변 비율 제한기 주기결정 알고리즘은 1초에 한번씩 실행되고, 가중치 평균 큐 길이를 구하는 수식의 a = 0.2로 설정하였고, 참고문헌 [8]에서처럼 웜 탐지를 위한 경계값은 100, 그리고 워킹셋 크기는 5로 설정하였다. 가중치 평균 큐 길이 수식의 a 값 선정은 다양한 실험을 통해 바이러스 쓰로틀링에 가장 적합하게 평균 지연 큐 길이의 증가 또는 감소를 표현해 주는 값으로 선택하였다. 각 실험에서 연결설정 패킷들은 잘 알려진 Pareto on/off 트래픽 생성기법을 사용하여 생성하였으며, Pareto 분포의 세이프(shape) 파라미터는 1.5로 설정하였다.



(그림 4) 성능 실험 환경

〈표 2〉 웜들의 시간적 특성

(단위: 초)

웜 분류	웜 이름	초당 연결 패킷 수	Off 기간
연속적 웜	Nimda	300	0
	Code Red	200	0
	CW 1	100	0
	CW 2	50	0
	CW 3	10	0
산발적 웜	SW 1	100	5
	SW 2	100	10
	SW 3	50	5
	SW 4	100	15
	SW 5	50	10
	SW 6	50	15

<표 2>는 가변 비율 제한기로 인한 웜 탐지시간의 변화를 알아보기 위해 생성된 다양한 시험용 웜들(CW1~CW3, SW1~SW6)과 실제 웜인 Nimda[1], Code Red[3]의 시간적 특성을 보여 준다. CW1~CW3, Nimda, Code Red 등과 같은 연속적 웜들은 Off 기간 없이 매초 지속적으로 연결 패킷을 생성하는데, 대부분의 웜들이 이 범주에 속한다. 그러나 Code Red II[4]와 같은 일부 산발적 웜들은 초당 패킷 발생 비율은 동일하더라도, On 기간 동안 한번에 많은 패킷을 발생시킨 후 일정시간(Off 기간) 동안 패킷을 발생시키지 않다가 다시 많은 패킷을 발생시키는 과정을 반복하는 웜들도 있다. 이런 산발적 웜들은 일반적인 트래픽과 유사한 반복 패턴을 가지지만 패킷을 발생하지 않는 Off 기간이 짧고, On 기간 동안 생성되는 연결 패킷의 수가 일반 트래픽 보다는 매우 많다는 점에서 차이를 보인다. SW1~SW6는 산발적 웜을 실험하기 위해 생성되었으며, 이들은 On 기간을 1초로 동일하게 설정하고 On 기간 동안에 생성되는 패킷 수와 Off 기간의 길이를 서로 다르게 설정하였다.

<표 3>은 연속적 웜에 대해 비율 제한기 주기결정 전략들이 웜 탐지시간에 어떠한 영향을 미치는지 알아보기 위한 실험 결과이다. 연속적 웜들에 대해서는 웜 탐지시간이 주기결정 방법에 상관없이 모두 동일하다. 이는 이들 웜들의 경우 매초 연속적으로 웜 패킷을 발생하기 때문에 지연 큐나 또는 WAQL가 지속적으로 증가만 하여 비율 제한기의 주기가 전혀 변동되지 않기 때문이다. 즉, 가변 비율 제한기의 주기는 WAQL이 감소하기 시작할 때부터 조절되는데, 이런 웜들의 경우 1초 이내에 지연 큐가 바로 경계값에 도달하거나 이 보다 길어질 경우 WAQL가 증가만하고 감소를 하지 않기 때문에 주기가 전혀 변동되지 않는다.

<표 4>는 산발적 웜에 대한 주기결정 전략별 웜 탐지시간을 보여준다. 괄호 안은 기존 Fixed 전략에 비해 웜 탐지시간의 증가율이다. 연속적 웜과는 달리 산발적 웜들은 1초간 웜 패킷이 생성되고 Off 기간 동안 전혀 웜 패킷이 생성되지 않기 때문에 WAQL가 감소하기 시작한다. 이때부터 주기결정 전략에 의해 비율 제한기의 주기가 짧아지게 되어 지연 큐의 연결요청 패킷이 기존 Fixed 방식보다 약간씩 더 빠르게 큐에서 제거되어 외부로 전송된다. 그러나 다시 On 기간 동안 새로운 패킷이 지연 큐에 유입되면 비율 제한기

〈표 3〉 연속적 웜들의 탐지시간

(단위: 초)

웜 이름	Fixed	Abs-line	Rel-line	Dec-inc
Nimda	0.28	0. 28	0.28	0. 28
Code Red	0.44	0. 44	0.44	0. 44
CW 1	0.94	0.94	0.94	0.94
CW 2	1.96	1. 96	1.96	1. 96
CW 3	10.88	10.88	10.88	10.88

〈표 4〉 산발적 웜들의 탐지시간

(단위: 초)

웜 이름	Fixed	Abs-line	Rel-line	Dec inc
SW 1	6.71	6.72 (0.14%)	6.74 (0.44%)	6.72 (0.06%)
SW 2	13.37	13.43 (0.48%)	13.52 (1.13%)	13.42 (0.39%)
SW 3	15.09	15.15 (0.39%)	15.22 (0.85%)	15.13 (0.22%)
SW 4	20.64	21.13 (2.40%)	21.44 (3.87%)	20.76 (0.58%)
SW 5	31.38	32.01 (2.01%)	32.80 (4.53%)	31.82 (1.41%)
SW 6	52.31	54.29 (3.78%)	56.64 (8.28%)	53.24 (1.77%)

의 주기는 원래대로 복구되었다가 Off 기간에 다시 감소하기를 반복한다. Off 기간 동안 주기가 감소했기 때문에 기존 방식보다 지연 큐 길이가 조금 더 작게 되며, 이로 인해 기존의 고정된 주기를 사용하는 방법에 비해 웜 탐지시간이 약간씩 증가하게 된다.

Fixed를 제외한 세 전략 중에 Dec-inc 전략의 웜 탐지 성능이 기존의 Fixed 전략과 가장 비슷한 성능을 보이고, Rel-line이 가장 떨어진다. 이는 Rel-line의 경우 WAQL가 감소하기 시작한 시점인 downWAQL의 크기에는 상관없이 감소시점부터 최소 주기로 설정한 후 이후 WAQL가 지속적으로 줄어들면 주기를 증가시켜 원래 주기로 복구하기 때문에, Abs-line과 Dec-inc 전략보다도 비율 제한기의 주기를 더 작게 하여 상대적으로 빠르게 지연 큐 내의 패킷을 외부로 전송하기 때문이다. Abs-line과 Dec-inc 전략은 SW1~SW3와 같은 초당 패킷 생성율이 높고 Off 기간이 짧은 웜들에 대해서는 웜 탐지시간이 거의 똑 같다가 SW 4~SW6과 같이 패킷 생성율이 낮고 Off 기간이 길어질수록 약간의 성능 차이를 보이기 시작한다. Dec-line과 기존의 Fixed와 비교해 볼 때 큰 성능차이를 보이지 않으며, 가장 느리게 전파되는 SW6에 대해 약 1.8%정도 웜 탐지시간이 증가했다. 이 정도의 웜 탐지시간 차이는 1초 이내의 차이이기 때문에 실제 전송되는 웜 패킷의 수도 기껏해야 1~2 패킷 정도 내로 차이가 발생하므로 거의 유사한 성능이라 할 수 있다.

〈표 5〉 정상 트래픽에 대한 연결설정 평균 지연시간

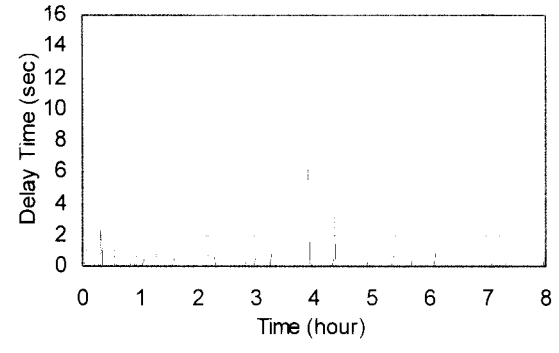
(단위: 초)

On 기간	패킷 생성율	Fixed	Abs-line	Rel-line	Dec-inc
1	1	0.59	0.59 (0.0%)	0.59 (0.0%)	0.59 (0.0%)
1	2	1.09	1.09 (0.0%)	1.07 (1.6%)	1.09 (0.2%)
2	2	1.94	1.93 (0.3%)	1.84 (5.3%)	1.91 (1.3%)
1	4	2.57	2.55 (0.9%)	2.38 (7.5%)	2.51 (2.3%)
2	3	3.53	3.47 (1.7%)	3.16 (1.4%)	3.39 (4.0%)
1	6	4.32	4.22 (2.3%)	3.76 (12.9%)	4.10 (5.2%)
2	4	5.24	5.06 (3.4%)	4.45 (15.0%)	4.89 (6.6%)
1	8	6.13	5.85 (4.4%)	5.08 (17.1%)	5.66 (7.6%)
1	10	8.21	7.62 (7.2%)	6.47 (21.2%)	7.41 (9.8%)

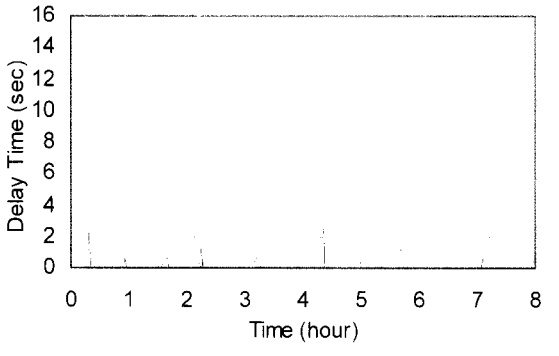
다음은 가변 비율 제한기가 얼마나 효율적인지 알아보기 위해 웜이 아닌 정상 트래픽에 대해 주기결정 전략별 평균 연결설정 지연시간을 측정하였다. 여기서 연결설정 지연시간이란 연결요청 패킷이 지연 큐에서 삽입된 이후부터 외부로 전송되기 직전까지 지연 큐에서 대기한 시간만을 의미한다. 실험에서 Off 기간은 40초로 고정시키고, On기간과 On 기간 동안의 연결설정 패킷 생성율을 다양하게 변경시켰다. <표 5>는 실험을 통해 측정된 각 전략별 연결설정 평균 지연시간이다. 괄호 안은 기존 Fixed 전략에 비해 평균 연결설정 지연시간의 감소율이다.

전체적으로 각 전략들이 기존의 Fixed 방식보다 연결설정 지연시간이 단축되었음을 확인할 수 있다. 또한 주기결정 전략에 상관없이 단위시간당 패킷 생성율이 증가할수록 연결설정 지연시간 단축율이 더 증가한다는 것이다. 이는 지연 큐의 길이가 길어질수록 지연시간이 더 많이 단축된다는 것을 의미한다. 연결설정 지연시간은 Abs-line, Dec-line, Rel-line 전략 순으로 더 많이 단축되었다. 그러나 Rel-line의 경우 지연시간이 가장 많이 단축된 반면 앞서 <표 4>의 실험 결과에서 알 수 있듯이 웜 탐지시간이 상대적으로 가장 많이 길어졌다. 따라서 이 방식은 다른 두 방식에 비해 웜 탐지시간과 연결설정 지연시간 측면에서 너무 극단적인 성능을 보이므로 적절하다고 할 수 없다. 반면, Dec-inc 전략의 경우 비록 지연시간은 Rel-line에 못 미치지만 웜 탐지시간은 기존의 Fixed와 가장 유사했다. 따라서 두 실험결과에서 Dec-inc 함수가 웜 탐지시간과 연결설정 지연시간 측면에서 가장 적합하다는 것을 알 수 있다.

(그림 5)는 지연 큐에서 제거된 패킷들이 큐에서 대기한 평균 지연시간을 매초 별로 표시한 것이다. 동일한 트래픽에 대해 그림 (a)는 기존의 Fixed 전략을 사용한 경우이고,



(a) Fixed 전략



(b) Dec-inc 전략

(그림 5) 시간대별 연결설정 지연시간

그림 (b)는 Dec-inc 전략을 사용한 가변 비율 제한기를 설치했을 때의 시간대별 평균 지연시간을 보여 준다. 실험에서 Off 기간은 40초, On기간은 1초, On 기간 동안의 패킷 생성율은 2로 설정하였다.

그림 (a)와 (b)를 비교했을 때, 트래픽이 적을 경우의 지연시간 차이는 근소한 차이 밖에 나지 않지만, 트래픽이 많은 경우 Fixed 보다는 Dec-inc 전략이 눈에 띄게 지연시간이 단축되었음을 확인할 수 있다. 이는 지연 큐 길이가 길어질수록 비율 제한기의 주기가 서서히 감소되었다가 천천히 원래의 주기로 복귀하는 과정에서 지연 큐 내의 패킷들이 Fixed 보다 빠르게 제거되기 때문이다. 반면, 큐 길이가 짧을 경우에는 가변 비율 제한기의 변경된 주기가 적용되는 기회가 그만큼 줄어들기 때문에 지연시간 단축에는 큰 영향을 미치지 못하기 때문이다.

(그림 5)의 실험결과는 사용자들에게 상당히 긍정적인 이점을 제공할 수 있다. 지연시간이 1초 이내인 경우 사용자는 연결설정 지연을 피부로 느끼지 못하지만, 일시적으로 연결설정 요청이 증가하여 지연시간이 더 증가할 경우 사용자는 갑자기 네트워크 부하가 늘어 데이터 전송 속도가 떨어졌다고 생각할 수 있다. 그러나 이는 네트워크 문제가 아니라 바이러스 쓰로틀링의 설치로 인해 발생하는 문제이다. 따라서 지연 큐의 길이가 갑자기 증가했을 때 발생할 수 있는 과도한 지연시간을 제안된 가변 비율 제한기를 적용하여 기존의 Fixed 전략보다 더 많이 지연시간을 줄여 줄 수 있다. 이는 사용자에게 연결설정 지연으로 인한 불편함을 최소화 할 수 있음을 의미한다.

이상의 실험결과를 통해 비율 제한기의 주기를 고정하기보다는 능동적으로 변경시킬 경우 웜 탐지시간에는 커다란 영향을 미치지 않으면서 연결설정 지연시간을 상당히 감소시킬 수 있음을 알 수 있다. 이는 곧 사용자에게 바이러스 쓰로틀링의 설치로 인한 불편함을 상대적으로 줄여 줄 수 있음을 의미한다. 또한 주기결정 전략에 있어 WAQL가 감소하기 시작한 시점부터 주기를 서서히 감소시켰다가 다시 원래의 주기로 복귀시키는 Dec-inc 전략이 가장 효율적이라는 것을 확인할 수 있었다.

5. 결 론

본 논문에서는 비율 제한기의 주기를 고정시키고 지연 큐 길이만으로 웜 발생 여부를 판단하는 기존 바이러스 쓰로틀링의 방법을 탈피하고, 가중치 평균 지연 큐 길이를 적용하여 비율 제한기의 주기를 가변적으로 조절하는 방법과 다양한 주기결정 전략을 제안했다.

실험결과 제안 알고리즘은 웜 탐지시간에는 큰 영향을 미치지 않으면서 연결설정 패킷의 지연시간을 단축시켜 사용자에게 바이러스 쓰로틀링의 설치로 인한 불편함을 줄이는 장점이 있다는 것을 확인할 수 있었다. 또한 다양한 주기결정 방법을 소개하고 이들의 성능을 비교 분석함으로써 가중치 평균 큐 길이가 감소하기 시작한 시점부터 비율 제한기의 주기를 서서히 감소시켰다가 다시 원래의 주기로 복귀시키는 방식으로 비율 제한기의 주기를 변경하는 것이 가장 효과적이라는 것을 실험을 통해 보였다.

그러나 연결설정 지연시간이 단축되었다 해도 산발적 웜에 대해서는 웜 탐지시간이 기존의 전략들에 비해 미세하게 증가됨을 볼 수 있는데, 이러한 연결설정 지연시간과 웜 탐지시간의 서로 상반되는 특성에 대해서는 사용자가 시스템의 목적에 맞게 적절히 선택 가능하다고 판단한다.

제안된 가변 비율 제한기의 최소주기는 0.5초로 고정하였고, 또한 주기를 감소시켰다가 다시 증가시키기 시작하는 지점을 감소시점(downWAQL)의 중간지점(반환점)으로 고정하였다. 향후 최소주기와 반환점을 동적으로 변경하여 웜 탐지시간과 연결설정 지연을 동시에 줄일 수 있는 가장 적절한 최소주기와 반환점을 자동으로 찾는 연구를 진행할 계획이다.

참 고 문 헌

[1] CERT, "CERT Advisory CA-2001-26 Nimda Worm, Sept. 2001. http://www.cert.org/advisories/CA-2001-26.html.

[2] CERT Advisory CA-2003-04: "MS-SQL Server Worm," Jan., 2003. http://www.cert.org/advisories/CA-

2003-04.html.

[3] CERT, "CERT Advisory CA-2001-08 Code Red Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," July, 2001. http://www.cert.org/incident_notes/IN-2001-08.html.

[4] CERT Advisory CA-2001-09: "Code Red II: Another Worm Exploiting Buffer Overflow," *IIS Indexing* Service DLL, Aug. 2001. http://www.cert.org/incident_notes/IN-2001-09.html.

[5] CERT, "CERT Advisory CA-2000-04 Love Letter Worm, May 2002. http://www.cert.org/advisories/CA-2000-04.html.

[6] S. Sidiroglou and A. D. Keromytis, "A Network Worm Vaccine Architecture," *Proc. of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Workshop on Enterprise Security, pp.220-225, June, 2003.

[7] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver, "Inside the Slammer worm," *IEEE Security and Privacy*, Vol.1, pp.33-39, July, 2003.

[8] Matthew M. Williamson, "Throttling Viruses: Restricting propagation to defeat malicious mobile code," *Proc. of the 18th Annual Computer Security Applications Conference*, Dec., 2002.

[9] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," *Proc. of the 12th USENIX Security Symposium*, pp.285-294, Aug., 2003.

[10] X. Qin, D. Dagon, G. Gu, and W. Lee, "Worm detection using local networks," Technical report, College of Computing, Georgia Tech., Feb., 2004.

[11] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast port-scan detection using sequential hypothesis testing," *Proc. of the IEEE Symposium on Security and Privacy*, May, 2004.

[12] J. Jung, S. E. Schechter, and A. W. Berger, "Fast Detection of Scanning Worm Infections," *Proc. of 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Sophia Antipolis, French Riviera, France. Sept., 2004.

[13] C. C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense," *ACM CCS Workshop on Rapid Malcode (WORM'03)*, Washington DC, Oct., 2003.

[14] C. Zou, L. Gao, W. Gong, D. Towsley, "Monitoring and early warning for Internet worms," *ACM Conference on Computer and Communications Security*, Washington, DC, Oct., 2003.

[15] N. Gulati, C. Williamson and R. Bunt, "LAN traffic locality: Characterization and application," *Proc. of the First International Conference of Local Area Network Interconnection*, pp.233-250. Plenum, Oct., 1993.

[16] 심재홍, 김장복, 최경희, 정기현, "Virus Throttling의 윔 탐지오판 감소 및 탐지시간 단축," 정보처리학회논문지 C, 제12-C권, 제6호, pp. 847-854, 2005. 10.

[17] J. Kim, J. Shim, G. Jung, and K. Choi, "Reducing Worm Detection Time and False Alarm in Virus Throttling," LNAI, Vol. 3802, pp.297~302, Dec., 2005.

심 재 홍

e-mail : jhshim@chosun.ac.kr

1987년 서울대학교 전산과학과(학사)

1989년 아주대학교 컴퓨터공학과(석사)

2001년 아주대학교 컴퓨터공학과(박사)

1989년~1994년 서울시스템(주) 공학연구소

1999년~2000년 University of Arizona 객원연구원

2001년~2001년 9월 아주대학교 정보통신전문대학원 BK21 전임연구원

2001년 10월~현재 조선대학교 컴퓨터공학부 조교수

관심분야 : 임베디드시스템, 운영체제, 분산시스템, 실시간 및 멀티미디어시스템