Virus Throttling의 웜 탐지오판 감소 및 탐지시간 단축

심 재 홍[†] · 김 장 복[††] · 최 경 희[†††] · 정 기 현[††††]

요 약

인터넷 웜(worm)의 전파속도는 매우 빠르기 때문에, 발생초기에 웜의 전파를 탐지하여 막지 못하면 큰 피해를 초래 할 수 있다. 새로운 세션에 대한 연결요청을 일정 비율이하로 제한함으로써 웜의 발생여부를 탐지하는 바이러스 쓰로틀링(virus throttling)[6, 7]은 대표적인 웜 조기 탐지 기술 중의 하나이다. 대부분의 기존 기술은 지연 큐 관리에 있어서 동일한 수신 IP 주소들을 개별적으로 처리함으로써 웜 탐지의 오판 가능성을 증가시켰고, 지연 큐가 가득 찼을 때에만 웜이 발생했다고 판단하는 단순 판단 기법을 사용했다. 본 논문은 지연 큐에서 동일 수신 IP 주소들을 하나의 연결 리스트로 묶어 별도로 관리함으로써 동일 수신 IP들을 중복하여 지연 큐에 저장하지 않는 이차원 지연 큐 관리방안을 제안한다. 개선된 바이러스 쓰로틀링은 지연 큐 길이 산정 시 동일 수신 IP 주소들을 중복하여 계산하지 않기 때문에 웜 탐지 오류를 줄일 수 있다. 그리고 동일한 크기의 지연 큐를 가지고도 웜 탐지시간을 줄이고 웜 패킷 전송 수를 줄일 수 있는 가중치 평균 큐 길이 기반의 새로운 웜 탐지 알고리즘을 제안한다. 지연 큐 길이 산정 시 현재의 큐 길이 뿐 아니라 과거의 큐 길이를 반영하는 방법이 웜의 발생 가능성을 사전에 예측하여 기존 기법보다 빠르게 웜을 탐지할 수 있음을 실험을 통해 확인하였다.

키워드 : 바이러스 쓰로틀링, 웜 조기 탐지, 인터넷 웜

Reducing False Alarm and Shortening Worm Detection Time in Virus Throttling

Jae-Hong Shim[†] · Jang-bok Kim[††] · Kyung-Hee Choi[†††] · Gi-Hyun Jung[††††]

ABSTRACT

Since the propagation speed of the Internet worms is quite fast, worm detection in early propagation stage is very important for reducing the damage. Virus throttling technique, one of many early worm detection techniques, detects the Internet worm propagation by limiting the connection requests within a certain ratio.[6, 7] The typical throttling technique increases the possibility of false detection by treating destination IP addresses independently in their delay queue managements. In addition, it uses a simple decision strategy that determines a worm intrusion if the delay queue is overflown. This paper proposes a two dimensional delay queue management technique in which the sessions with the same destination IP are linked and thus a IP is not stored more than once. The virus throttling technique with the proposed delay queue management can reduce the possibility of false worm detection, compared with the typical throttling since the proposed technique never counts the number of a IP more than once when it checks the length of delay queue. Moreover, this paper proposes a worm detection algorithm based on weighted average queue length for reducing worm detection time and the number of worm packets, without increasing the length of delay queue. Through deep experiments, it is verified that the proposed technique taking account of the length of past delay queue as well as current delay queue forecasts the worm propagation earlier than the typical virus throttling techniques do.

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

1. 서 론

인터넷을 기반으로 하는 많은 웜[1, 2, 13-15]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. 이처럼 스스로 전파되는 웜이 유해한 데이터를 전달하지 않을지라도 웜의 전파과정에서 발생하는 엄청난 양의 트래픽은 네트워크의 성능을 저하시킬 뿐 아니라, DOS 공격 등도 가능하게 한다[3]. 또한 웜의 전파 속도가 워낙 빠르기 때문에 사용자들이 일일이 그것에 대항하여 반응하기에는 역부족이다[4]. 따라서 새로운 웜이 발생하였을 때 웜의 발생 여부를 빠르게 탐지하여 더 이상의 웜 전파를 차단해야 한다.

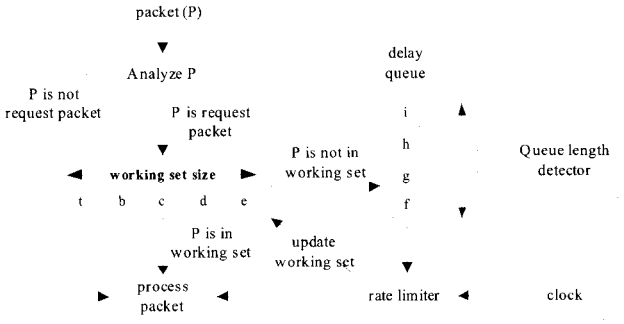
† 정 회 원 : 조선대학교 인터넷소프트웨어공학부 교수
†† 준 회 원 : 아주대학교 정보통신전문대학원 박사과정
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
†††† 정 회 원 : 아주대학교 전자공학부 교수
논문접수 : 2005년 7월 22일, 심사완료 : 2005년 9월 12일

웜 조기 탐지(worm early detection) 기술은 바이러스 쓰로틀링(virus throttling)[6, 7], 수신-송신 상호관계(DSC : destination-source correlation)[10]을 이용한 시스템, 순차적 가설 시험 (sequential hypothesis testing)을 이용한 알고리즘[8, 9] 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[9]을 제안하였고, 또한 이 알고리즘과 연결 비율 제한(connection rate limiting)을 동시에 사용하는 하이브리드 접근 방법[8]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[11]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[5].

바이러스 쓰로틀링[6, 7]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 단위시간당 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. 새로운 연결요청 패킷(P)의 전송요청이 들어오면 워킹 셋(working set)에서 P와 동일한



(그림 1) 바이러스 쓰로틀링(virus throttling)

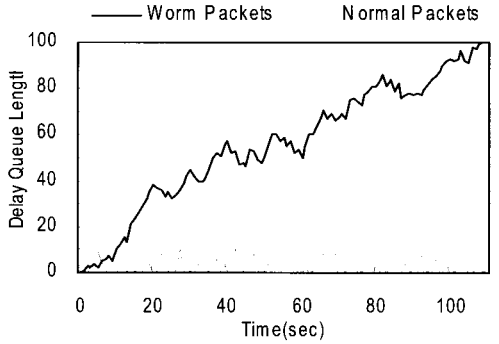
수신 IP 주소가 존재하는지 확인하고, 만약 존재한다면 P를 정상 트래픽으로 간주하여 지연 없이 즉시 전송한다. 그렇지 않은 경우 P를 지연 큐(delay queue)에 저장한다. 즉, 상대적으로 최근에 접속이 이루어졌던 호스트에 대해서는 지연 없이 바로 연결요청 패킷을 전송하고 그렇지 않은 호스트에 대해서는 패킷을 지연 큐에 보관한 후 적당한 시기에 비율 제한기(rate limiter)에 의해 전송되게 한다. 비율 제한기는 일정 시간 간격을 두고 주기적으로 지연 큐에서 제일 오래된 패킷을 꺼내어 이를 전송한다. 이때 이 패킷과 동일한 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.

그러나 기존 바이러스 쓰로틀링은 지연 큐 관리에 있어서 동일한 수신 IP 주소들을 다른 수신 IP 주소들과 구분 없이 중복하여 저장한다. 이는 동일한 호스트에 새로운 연결요청이 일시적으로 폭주할 경우 정상적인 트래픽임에도 불구하고 잘못된 웜 탐지 판정을 내리게 할 수 있다. 따라서 본 논문에서는 지연 큐에서 동일 수신 IP 주소들을 하나의 연결 리스트로 묶어 별도로 관리함으로써 동일 수신 IP 주소들을 중복하여 지연 큐에 저장하지 않는 이차원 지연 큐 관리방안을 제안한다. 또 다른 개선점으로 기존 바이러스 쓰로틀링의 지연 큐 길이 감시자는 현재 지연 큐의 길이를 경계값과 비교하여 이를 초과했을 때만 웜이 발생했다고 판단하는 단순 판단 기법을 사용한다. 본 논문에서는 지연 큐 길이 산정 시 현재 큐 길이 뿐 아니라 과거 큐 길이도 반영함으로써 웜의 발생 가능성을 사전에 예측하여 보다 빠르게 웜 발생을 탐지할 수 있는 새로운 웜 탐지 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 바이러스 쓰로틀링 기법에서 개선해야 할 사항들에 대해 논의하고, 이를 해결할 수 있는 방안을 3장에서 제시한다. 4장에서는 제안된 해결방안의 효율성을 검증하기 위한 실험결과를 제시하고 이를 분석한다. 마지막 5장에서는 본 논문의 결론과 향후 연구방향을 제시한다.

2. 연구 배경

일반적으로 인터넷 웜은 가능한 많은 다른 시스템에 자기 자신을 복제하고 전파시키는 행위를 반복한다. 예를 들어, 취약성을 가진 시스템을 탐색하기 위한 목적으로 Nimda[13]는 초당 300~400개, SQLSlammer[1]는 초당 850개, Code Red [14]는 초당 약 200개의 연결요청을 위한 패킷을 전송한다. 이들은 대부분 일정한 주기로 연결요청 패킷을 전송하지만 Code Red II[2]처럼 초당 패킷 발생 비율이 동일하더라도, 한 번에 많은 패킷을 발생시킨 후 일정 시간동안 패킷을 발생



(그림 2) 정상 패킷과 웜 패킷의 지연 큐 길이 변화

시키지 않다가 다시 많은 패킷을 발생시키기도 한다. 또한 I Love You[15]와 같은 많은 전자우편 바이러스들은 감염된 시스템에서 찾을 수 있는 모든 전자우편 주소로 메일을 전송한다. 그러나 정상적인 트래픽은 상당히 낮은 비율(초당 약 1개)로 연결요청이 이루어지며, 또한 연결 대상 시스템이 웜처럼 불특정 다수가 아닌 소수의 특정 시스템들로 국한되어 있다.

바이러스 쓰로틀링에서 지연 큐 길이 감시자는 패킷이 지연 큐에 저장될 때마다 큐 길이를 검사하여 사전에 정의된 경계값을 넘어서게 되면 웜이 발생하였다고 판단한다. 지연 큐 길이와 관련하여 먼저 웜 발생 상황과 정상 상황에서의 지연 큐 길이 특성을 비교 분석해 볼 필요가 있다. 이를 위해 TCP/IP 패킷에 대해서 바이러스 쓰로틀링을 실제로 구현하였다. 비율 제한기의 주기를 0.25로 설정하고, 평균 초당 5개의 웜 패킷을 임의로 생성하여 시뮬레이션을 하였으며, 실험결과를 (그림 2)에 보였다.

웜 패킷의 경우 지연 큐의 길이가 경계값 100에 도착할 때까지 큐 길이가 늘어났다 줄어들었다 하지만 전체적으로 상승하는 추세를 보인다. 즉, 웜 패킷이 평균 초당 5개가 발생하기 때문에 시간이 흐를수록 큐에 쌓이는 패킷 수는 증가한다. 물론 모든 웜의 큐 길이 패턴이 그림과 같은 동일한 패턴을 가지는 것은 아니다. 이에 반해 정상적인 패킷 흐름에서는 웜과 같은 패턴이 발생하지 않으며, 갑자기 패킷의 수가 증가하였다가 감소하는 패턴을 가지며, 큐 길이가 작고 (그림에선 10 이하) 지속적으로 큐의 길이가 증가하지는 않는다.

다른 경계값-기반 웜 탐지 기법들과 마찬가지로 바이러스 쓰로틀링도 지연 큐의 경계값을 낮추게 되면 웜을 탐지하는 시간이 감소하게 된다. 그러나 이것은 정상적인 트래픽이 일시적이지만 갑작스럽게 증가할 경우 웜이 발생한 것으로 잘못 판단할 가능성이 높아지게 된다. 반대로 지연 큐의 경계값을 높이게 되면 큐에 보관해야 할 패킷의 개수가 많아지면서 필요한 버퍼 용량도 증가하게 된다. 또한 웜 발생 오판을 줄일 수는 있겠지만 실제 웜이 발생할 경우 웜 탐지 시간이 증가하게 되며 그 만큼 많은 양의 웜을 외부로 누출하게 된다.

따라서 본 논문에서는 정상 트래픽의 특징 중 하나인 지역성(locality)[12]과 웜 트래픽의 특징을 이용하여 기존 바이러스 쓰로틀링 기법과 동일한 경계값을 가지고도 웜 탐지 오판을 줄일 수 있는 간단한 이차원 지연 큐 관리방안을 제안하고자 한다.

〈표 1〉 비율 제한기의 주기에 따른 웜 탐지시간 및 처리된 패킷 수

주기 (초)	웜 탐지시간 (초)	처리된 패킷 수
1.00	20.49	20
0.75	24.51	32
0.50	27.94	55
0.25	108.15	432

바이러스 쓰로틀링의 또 다른 고려사항은 비율 제한기의 주기 값의 결정이다. 비율 제한기가 지연 큐의 패킷을 전송하는 주기는 사용자가 느끼게 되는 접속 지연시간과 웜 탐지시간, 전송된 웜 패킷 수에 민감한 영향을 주는 중요한 요소이다. 따라서 본 논문에서는 비율 제한기의 주기에 따라 웜 탐지시간과 웜 발생 이후 탐지까지 처리된 웜 패킷 수가 어떻게 달라지는 지를 측정하였다. <표 1>은 시뮬레이션 프로그램을 이용하여 웜 패킷을 생성하고, 생성된 패킷을 구현된 바이러스 쓰로틀링에 적용한 실험결과를 보여준다. 실험에서 웜 패킷 생성율을 초당 평균 5개로 하고, 지연 큐의 최대 큐 크기(경계값)를 100으로 설정하였으며, 동일한 웜 패킷들에 대해 비율 제한기의 주기를 다르게 하여 실험하였다.

예상한 바와 같이 비율 제한기의 주기가 빨라질수록 웜을 탐지하는데 걸리는 속도가 느리고, 웜 발생 이후 웜이 탐지되기까지 처리된 패킷의 개수도 많아짐을 알 수 있다. 웜 탐지시간은 비율 제한기의 주기 1초와 0.25초를 비교하였을 때, 거의 5배에 달하는 차이를 보였으며, 처리된 패킷의 개수는 거의 21배의 차이를 보였다.

만약 비율 제한기의 주기가 길어지면 웜을 탐지하는 시간이 줄어들게 될 뿐 아니라, 초당 생성되는 웜 패킷의 수가 작아도 웜을 정상적으로 탐지할 수 있다. 하지만 사용자가 느끼게 되는 접속 지연 시간은 상대적으로 길어지게 된다. 사용자에게는 바이러스 쓰로틀링으로 인한 접속지연이나 불편함을 가능한 느끼지 않게 해야 한다. 반면, 비율 제한기의 주기가 짧아질 경우, 연결 요청 패킷을 상대적으로 빠르게 전송함으로써 사용자가 느끼는 접속 지연 시간은 상대적으로 줄어드는 이점이 있다. 그러나 웜 탐지 시간은 반대로 길어지게 되고 그 만큼 많은 양의 웜 패킷을 전송하게 된다. 따라서 비율 제한기의 주기 값을 여러 환경에 따라 적절하게 결정하는 것은 아주 중요하면서도 쉽지 않은 문제이다.

위의 간단한 실험결과를 통해 접속 패킷의 지연시간을 가능한 짧게 하면서도 웜 탐지시간은 빠르게 유지할 수 있는 방법이 필요하다는 것을 알 수 있다. 기존 바이러스 쓰로틀링의 지연 큐 길이 감시자는 단순히 현재 큐 길이가 경계값을 초과하는지만 비교하여 웜 발생 여부를 판단한다. 그러나 (그림 2)에서 알 수 있듯이 웜 발생 시 나타나는 지연 큐의 증가 또는 감소 패턴을 웜 탐지 조건에 추가하면, 경계값이나 비율 제한기의 주기를 조정하지 않더라도 웜 탐지 속도를 향상 시킬 수 있을 것으로 예측된다. 따라서 본 논문에서는 지연 큐 길이 산정 시 현재의 큐 길이 뿐 아니라 과거의 큐 길이 변화를 반영하는 새로운 웜 탐지 알고리즘을 제안하고자 한다.

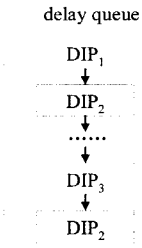
3. 개선된 바이러스 쓰로틀링

3.1 지연 큐 관리기법

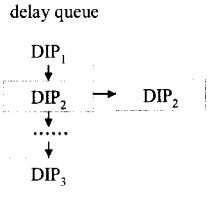
기존 바이러스 쓰로틀링[6, 7]은 단순한 일차원 지연 큐를 관리한다. 따라서 동일한 수신 IP 주소들이 지연 큐에 삽입되어도 지연 큐 길이를 산정할 때 동일한 각각의 IP 주소들이 따로 계산(count)되어, 큐 길이 산정에 중복으로 계산된다. 일반적으로 정상적인 연결요청 패킷들의 경우에는 지역성(locality)[12]을 가지기 때문에 동일한 수신 IP 주소들을 가질 확률이 많다. 반면 웜에 의해 발생되는 연결요청 패킷들은 지역성을 가지지 않으므로 동일한 수신 IP 주소들을 가질 확률은 거의 없다. 지연 큐 길이는 웜이 발생하였음을 나타내는 중요한 매개변수이기 때문에, 지연 큐의 길이 산정에 있어 동일한 수신 IP 주소들에 대해 중복으로 계산하지 않을 경우 정상적인 연결요청에 의한 웜 탐지 오판을 줄일 수 있다.

(그림 3)은 기존 기법에서 사용하던 큐의 구조를 나타낸 그림이며, (그림 4)는 웜 탐지 오판을 줄이기 위해 본 논문에서 설계된 이차원 지연 큐를 나타낸 그림이다.

(그림 3)에서는 동일한 두 개의 DIP_2 주소를 다른 주소들과 구분 없이 지연 큐에 포함시켜서 처리한다. 그러나 (그림 4)에서는 동일한 DIP_2에 대해서는 별도의 연결 리스트를 이용하여 보관함으로써 다른 주소들과는 다르게 관리한다. 지연 큐의 구조가 (그림 3)에서 (그림 4)로 변경되면, 큐에 연결요청 패킷을 삽입할 때와 제거할 때 알고리즘상 차이점이 발생한다. 새로운 연결요청 패킷이 도착하여 패킷을 큐에 삽입(enqueue)하고자 할 때, (그림 3)에서는 큐의 맨 끝에 패킷을 추가하기만 하나, (그림 4)에서는 패킷을 삽입할 때마다 그 패킷과 동일한 수신 IP 주소를 가지는 패킷이 큐에 있는지 검색하여야 한다. 그러나 패킷을 빼내(dequeue)고자 할 경우, (그림 4)에서는 동일한 수신 IP 주소를 연결 리스트로 연결해 놓았기 때문에 검색이 필요하지 않다. 그러나 (그림 3)에서는 제일 오래된 패킷과 동일한 수신 IP 주소를 가지는 패킷들을



(그림 3) 기존 일차원 지연 큐



(그림 4) 제안된 이차원 지연 큐

```
FUNCTION enqueue(packet P)
  Insert a new packet P into the end of delay queue
END FUNCTION

FUNCTION process_rate_limiter()
  Remove the first(oldest) packet P from the delay queue
  Send P to its destination
  FOR each packet S in the delay queue
    IF (P's destination IP address ==
        S's destination IP address) THEN
      Remove S from the delay queue
      Send S to its destination
    END IF
  END FOR
  Update the working set
END FUNCTION
```

(그림 5) 기존 일차원 큐 관리 알고리즘

```
FUNCTION enqueue(packet P)
  FOR each chained list L in the delay queue
   IF (P's destination IP address ==
       L's destination IP address THEN
     Add P to the list L
   END IF
  END FOR
END FUNCTION

FUNCTION process_rate_limiter()
  Remove the first list L from the delay queue
  Send all packets in L to the same destination
  Update the working set
END FUNCTION
```

(그림 6) 제안된 이차원 큐 관리 알고리즘

동시에 처리하기 위해서 지연 큐를 순차적으로 검색하여야 한다.

(그림 5)는 기존 일차원 지연 큐를 이용한 큐 관리 알고리즘이며, (그림 6)는 본 논문에서 제안하는 이차원 지연 큐 관리 알고리즘이다. 그림에서 함수 enqueue()는 연결요청 패킷이 도착할 경우 이를 지연 큐에 삽입하기 위해 호출되는 함수이며, process_rate_limiter()는 큐에서 가장 오래된 패킷을 꺼내기 위해 비율 제한기에 의해 호출되는 함수이다.

기존 알고리즘은 매 패킷 제거 시에, 제안 알고리즘은 매 패킷 삽입 시에 동일한 수신 IP 주소를 가진 패킷들을 찾기 위해 지연 큐를 모두 검색해야 한다. 따라서 두 알고리즘의 시간 복잡도는 $O(n^2)$으로 동일하다. 그러나 제안 알고리즘은 지연 큐 길이 산정 시 동일한 수신 IP 주소들을 중복하여 계산하지 않기 때문에 정상 트래픽의 일시적 폭주로 인한 웜 탐지 오판을 줄일 수 있는 이점을 제공한다.

3.2 웜 탐지 알고리즘

기존 바이러스 쓰로틀링의 큐 길이 감시자(queue length detector)는 패킷이 지연 큐에 쌓일 때마다 현재 지연 큐 길이를 검사하여 큐 길이가 경계값(threshold : 일반적으로 지연 큐의 최대 크기)을 넘어서게 되면, 웜이 발생한 것으로 간주하는 단순 판단기법을 사용한다. 본 절에서는 큐 길이 감시자가 웜 발생 여부를 결정하기 위해 현재 지연 큐 길이 뿐 아니라, 과거 큐 길이의 변화 추이도 함께 반영하는 방법을 제안하고자 한다.

```
IF ((CurrentQueueLength + WAQL) > Threshold)  THEN
   Worm is detected.
ELSE
   Worm is not detected.
END IF
```

(그림 7) 제안된 웜 탐지 알고리즘

과거 큐 길이를 반영하기 위해 본 논문에서는 가중치 평균 큐 길이(weighted average queue length : WAQL)를 사용한다. WAQL은 다음의 널리 잘 알려진 지수 이동 평균(exponential moving average) 수식을 이용하여 정해진 주기마다 구한다.

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

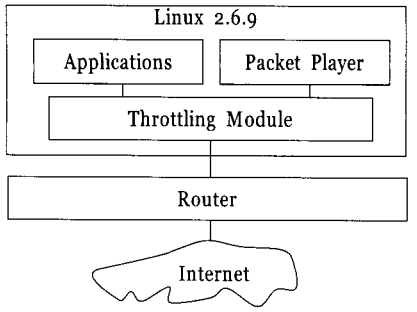
수식에서 $WAQL_{n-1}$은 직전의 주기에서 구해진 가중치 평균 큐 길이이며, $avgQL_n$는 현 주기 동안의 평균 큐 길이이다. 또한 a는 새로운 $WAQL_n$를 계산할 때 현 주기의 $avgQL_n$와 과거의 $WAQL_{n-1}$를 반영하는 비율을 결정하는 가중치 상수이다. 상수 a를 조절함으로써 최근의 트래픽 상황과 그 이전의 트래픽 상황을 적절히 반영시킬 수 있다.

(그림 7)은 큐 길이 감시자가 웜의 발생 여부를 판단할 때 사용하는 웜 탐지 알고리즘이다. 그림에서 CurrentQueueLength는 현재 지연 큐의 길이를 나타내는 변수이고, Threshold는 경계값을 표현하는 상수이며 일반적으로 지연 큐의 최대 크기이다.

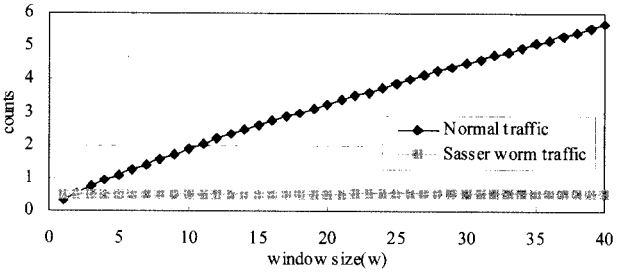
기존의 바이러스 쓰로틀링 기법에서는 단순히 CurrentQueue Length와 Threshold만을 비교하였으나, 제안 알고리즘에선 CurrentQueueLength와 WAQL를 더한 값을 Threshold와 비교한다. 이 경우 직관적으로 CurrentQueueLength가 Threshold의 반만 되어도 WAQL와 더한 값이 Threshold를 초과하여 웜 탐지 오판을 하지 않을까 염려할 수 있다. 그러나 정상 환경에서 트래픽이 일시적으로 증폭하여 CurrentQueueLength가 Threshold의 반 이상을 초과해도, 이는 일시적인 증폭이므로 WAQL는 매우 작은 값을 가진다. 본 연구에서는 이러한 점을 고려하여 정상 환경에서의 CurrentQueueLength와 WAQL를 오랜 기간동안 조사한 결과 두 값이 동시에 Threshold의 반 이상으로 증가하는 경우는 없었다. 이는 곧 새로운 연결의 일시적 증가로 인해 CurrentQueueLength가 커지면 그 순간 WAQL는 작고, 일시적 증가로 인해 일정기간 WAQL가 점점 증가하면 반대로 CurrentQueueLength는 감소하기 때문이다.

4. 성능 평가

제안된 이차원 지연 큐와 웜 탐지 알고리즘의 성능을 알아보기 위해서 실제로 바이러스 쓰로틀링을 Linux 시스템에 구현하였다. Linux 커널 버전 2.6.9에 기존 알고리즘[7]을 참고하여 구현하였으며, (그림 8)과 같은 성능 실험 환경을 꾸몄다.



(그림 8) 성능 실험 환경



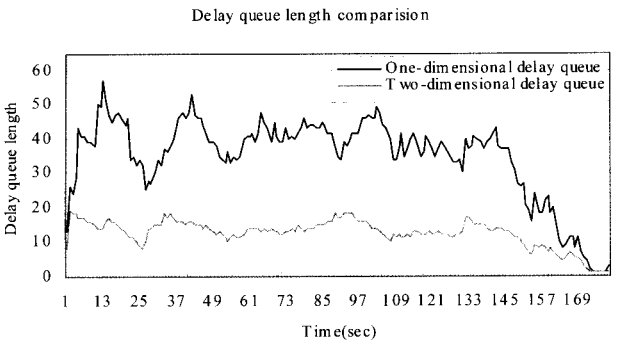
(그림 9) 슬라이딩 윈도우 크기에 따른 패킷 중복 회수

리눅스 시스템은 크게 2개의 모듈로 구성된다. 첫째가 커널에 구현된 바이러스 쓰로틀링 모듈로서, 리눅스 머신에서 외부로 전달되는 모든 패킷들을 바이러스 쓰로틀링을 통하여 새로운 연결 비율을 조절하는 역할을 하며, 실험결과를 측정하기 위한 여러 통계 데이터도 수집하게 된다. 둘째는 패킷 재생기(Packet Player)라는 것으로, 패킷을 생성시키거나, 캡쳐된 패킷을 재생하기 위해서 사용된다. 그리고 라우터에는 방화벽이 설치되어 있으며, 실험을 위해 생성되는 유해한 패킷들이 외부 인터넷으로 전달되지 않도록 설정되었다.

먼저, 일차원 지연 큐와 이차원 지연 큐의 차이점을 확인하기 위해 먼저 웹 브라우저를 이용하여 웹 서핑을 한 후 그 때 생성되는 패킷을 캡쳐하였다. 두 지연 큐 관리 방법의 차이점을 쉽게 확인 할 수 있도록 비교적 많은 양의 정상적인 연결요청 패킷을 발생시켰다. 이를 위해 하나의 PC에서 총 16개의 웹 브라우저를 이용하여 서로 다른 웹 사이트를 동시에 접속하였을 때 발생하는 트래픽을 캡쳐하여 실험에 사용하였다. 캡쳐된 패킷은 패킷 재생기를 통해 재생하였다.

캡쳐된 패킷이 웜 트래픽에 비해 지역성을 가지는지 알아보기 위해 캡쳐된 패킷과 Sasser 웜에 의해 발생된 패킷을 분석하였다. Sasser 웜 트래픽은 Windows PC에 Sasser 웜을 감염시킨 후 해당 PC에서 발생되는 패킷을 캡쳐하여 얻었다.

(그림 9)는 외부로 나가는 패킷들 중 연결요청 패킷에 대해 한 개의 패킷을 기준으로 동일한 수신 IP 주소가 슬라이딩 윈도우 크기 내에서 몇 개가 중복되어 나타나는지를 정상 트래픽과 Sasser 웜 트래픽 별로 평균값을 나타낸 그래프이다. 정상 트래픽의 경우에는 윈도우 크기가 커지면 커질수록 수신 IP 주소가 중복되어 나타나는 패킷의 수가 증가함을 알 수 있다. 이에 비해 Sasser 웜 트래픽은 거의 고정된 값을 가진다. 웜 트래픽의 경우에 그래프에서 값이 평균 0이 아니라



(그림 10) 두 가지 큐 구조에서의 큐 길이 변화

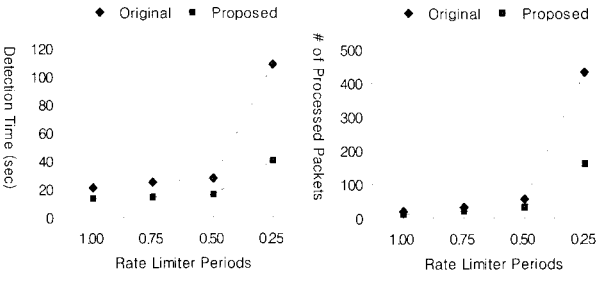
0.5 부근인 이유는 Sasser 웜이 스캐닝(scanning)할 때 동일한 수신 IP 주소에 대해서 거의 동시에 두개의 TCP SYN 패킷을 전송하기 때문이다. 따라서 Sasser 웜 트래픽에 대해서는 지역성을 가진다고 말할 수 없으나, 정상 트래픽은 수신 IP 주소에 대해서 충분한 지역성을 가진다고 말할 수 있다.

정상 트래픽의 지역적인 특성을 활용하여 제안된 이차원 큐 관리 방법이 기존 일차원 큐 관리 방법에 비해 어느 정도 웜 탐지 오판을 줄일 수 있는지 알아 볼 필요가 있다. 이를 위해 앞서 웹 서핑을 통해 캡처된 정상적인 트래픽에 대해 각각의 큐 관리 방법에 의해 관리되는 대기 큐의 길이 변화를 측정하였다. 이 실험에서 워킹 셋 크기는 5, 비율 제한기의 주기는 1초로 고정하였다.

(그림 10)은 이에 대한 실험결과를 그림으로 나타낸 것이다. 일차원 지연 큐 길이와 이차원 지연 큐 길이는 평균 22, 최대 43으로 큰 차이를 보이는 것을 알 수 있다. 이는 곧 정상적인 연결요청 패킷들의 상당수가 동일한 수신 IP 주소를 가지고 있음을 의미한다.

만약 지연 큐의 경계값을 50으로 설정하였을 경우, 일차원 지연 큐를 사용하는 바이러스 쓰로틀링은 웜이 발생한 것으로 잘못 판단하는 웜 탐지 오판을 일으키게 된다. 그러나 이차원 지연 큐 구조에서는 웜 탐지 오판은 일어나지 않는다. 즉, 일차원 지연 큐에서는 위의 실험 환경에서 경계값을 60으로 잡아야 하나, 이차원 지연 큐에서는 경계값을 20으로 잡아도 웜 탐지 오판이 일어나지 않는다. 이는 이차원 지연 큐를 사용할 경우 트래픽의 지역성이라는 특성을 활용하므로 기존 일차원 지연 큐보다 동일한 수신 IP 주소들의 일시적인 폭주로 인해 발생할 수 있는 웜 탐지 오판을 줄일 수 있다는 것을 의미한다.

다음은 기존 웜 탐지 알고리즘과 본 연구에서 제안된 웜 탐지 알고리즘의 성능을 비교하고자 한다. 앞서 2장에서 언급했듯이 대부분의 웜들의 초당 발생되는 패킷의 수는 다양하다. 따라서 다양한 웜을 시뮬레이션하기 위해 패킷 재생기에 실험용 웜(test worm)을 생성할 수 있는 기능을 삽입하고, 초당 생성하는 연결 패킷 수를 지정할 수 있게 했다. 제안된 웜 탐지 알고리즘의 가중치 평균 큐 길이(WAQL)는 1초마다 갱신하였으며, 상수 a 값은 0.2로, 경계값(지연 큐의 최대 크기)은 100으로 설정하였다. 상수 a는 여러 번의 실험결과 가장 우수한 성능을 보이는 값으로 선택하였다.



(a) 웜 탐지 시간 (b) 처리된 패킷 수

(그림 11) 비율 제한기의 주기에 따른 성능 비교

(그림 11)은 비율 제한기의 주기를 변경하면서 실시한 실험결과를 보여준다. 이 실험에서 패킷 재생기에 의해 생성되는 실험용 웜은 초당 5개의 연결을 시도한다. 비율 제한기의 주기 값을 1, 0.75, 0.5, 0.25 초로 변경하면서 웜을 탐지하는데 걸린 시간과 웜을 탐지할 때까지 처리한 패킷 수를 비교하였다.

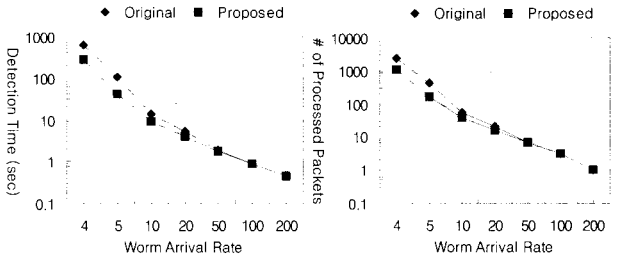
전체적으로 두 시스템 모두 비율 제한기의 주기가 짧을수록 웜 탐지시간과 처리된 패킷 수가 줄어든다. 이는 웜 패킷 생성율은 고정된 반면 지연 큐에 저장된 연결요청 패킷의 단위시간당 실제 전송 패킷 수가 증가하기 때문이며, 이는 곧 지연 큐가 상대적으로 느리게 가득 차게 되는 요인이 된다. 따라서 웜 탐지시간은 길어지게 되고 그 시간동안 전송된 패킷의 수도 증가하게 된다.

그림에서 비율 제한기의 모든 주기에 대해 개선된 시스템이 기존 시스템보다 더 좋은 성능을 보이고 있다. 이는 제안 알고리즘이 웜 탐지 시 현재의 큐 길이 뿐 아니라 큐 길이 변화 패턴도 함께 고려하기 때문이다. 또한 주기가 1에서 0.25초로 감소될 때 기존의 시스템의 경우 웜 탐지시간은 5배, 처리된 패킷 수는 22배 정도 증가되었지만, 개선된 시스템의 경우 각각 3배, 12배 정도로 증가되는 폭이 상대적으로 감소하는 것을 볼 수 있다. 이는 웜 탐지 시 비록 현재 큐 길이는 동일할지라도 웜 발생 지속시간이 길어질수록 제안 알고리즘의 가중치 평균 큐 길이(WAQL)는 상대적으로 더 커지기 때문이다. 즉, 주기가 짧아질수록 웜 탐지시간은 증가하게 되는데, 이럴수록 제안 알고리즘의 가중치 평균 큐 길이는 점점 더 커지기 때문에 개선된 시스템이 기존 시스템보다 웜 탐지 시간이 상대적으로 짧아지게 된다.

이러한 실험결과는 비율 제한기로 인해 사용자는 느끼는 연결요청 지연시간을 더 줄여주기 위해 비율 제한기의 주기를 상대적으로 짧게 했을 때, 기존 시스템보다 더 빠른 시간 내에 웜을 탐지할 수 있는 장점을 제공한다.

다음은 다양한 종류의 웜들에 대한 제안 알고리즘의 성능을 시뮬레이션해 보고자 한다. 실험을 위해 7가지 종류의 실험용 웜들을 생성하였으며, 이들의 초당 연결횟수는 각각 4, 5, 10, 20, 50, 100, 200이다. (그림 12)는 초당 연결횟수가 다른 실험용 웜들에 대한 웜 탐지 시간과 패킷 처리 수의 변화를 비교한 그림이다. 실험에서 비율 제한기의 주기는 0.25초로 설정하였다.

두 시스템 모두 웜의 초당 연결횟수가 증가할수록 웜 탐지 시간과 처리 패킷 수가 감소함을 볼 수 있다. 이는 단위시간



(a) 웜 탐지 시간 (b) 처리된 패킷 수

(그림 12) 초당 연결횟수가 다른 실험용 웜들에 대한 성능 비교

에 생성되는 웜 패킷의 수가 많을수록 지연 큐에 저장되는 연결요청 패킷 수가 지연 큐의 최대 크기에 빠르게 도달하기 때문이다.

웜의 초당 연결횟수에 상관없이 개선된 시스템이 기존 시스템보다 더 우수한 성능을 보임을 알 수 있다. 웜 패킷 생성율이 낮을수록 기존 시스템과 개선된 시스템의 웜 탐지시간에 대한 격차가 늘어남을 알 수 있는데, 이는 앞서 언급한 바와 같이 웜 발생 후 지속시간이 길어질수록 제안 알고리즘의 가중치 평균 큐 길이가 점점 더 커지기 때문이다. 즉, 두 시스템 모두 초당 웜 패킷 발생율이 적을수록 웜 탐지시간이 증가하게 되는데, 이럴수록 현재 큐 길이는 동일할지라도 제안 알고리즘의 가중치 평균 큐 길이가 점점 더 커지기 때문에 개선된 시스템이 더 빨리 웜을 탐지하게 된다. 따라서 개선된 시스템은 초당 연결횟수가 낮은 웜에 더더욱 유용하다는 것을 알 수 있다.

5. 결 론

본 논문에서는 기존 바이러스 쓰로틀링의 지연 큐 관리에 있어 정상 트래픽의 연결요청 패킷들에 대해 지역성을 반영하지 못한다는 단점을 지적하고, 이를 해결하기 위해 지연 큐의 구조를 일차원에서 이차원으로 변경하였다. 제안된 이차원 지연 큐는 일차원 지연 큐와 비교하여 패킷 처리 성능에는 영향을 주지 않지만, 동일한 수신 IP 주소를 지연 큐 길이 산정 시 중복하여 계산하지 않음으로써 웜 탐지 오판을 줄일 수 있었다.

또한 동일한 크기의 지연 큐를 가지고도 웜 탐지시간을 줄이고 전송된 웜 패킷 수를 줄일 수 있는 새로운 웜 탐지 알고리즘을 제안했다. 제안 알고리즘은 지연 큐 길이 산정 시 현재 큐 길이 뿐 아니라 과거 큐 길이도 반영함으로써, 웜의 발생 가능성을 사전에 예측하여 보다 빠른 시간 내에 웜을 탐지할 수 있었다. 제안 알고리즘은 비율 제한기의 주기가 짧은 경우 기존 시스템보다 더 좋은 성능을 보이므로 사용자에게 보다 짧은 연결요청 지연시간을 제공할 수 있으며, 또한 초당 연결횟수가 낮은 웜에 대해 기존 시스템보다 더 좋은 성능을 보인다는 것을 확인했다.

그러나 본 논문에서 제안한 개선된 바이러스 쓰로틀링을 실제 네트워크 또는 시스템에 설치하여 가동했을 때 사용자가 느끼는 새로운 세션에 대한 연결 지연이 얼마정도인지를 충분히 조사하지 못했다. 이는 비율 제한기의 주기와 밀접한 연관성이 있다. 사용자에게 가능한 짧은 연결 지연을 느끼게 하기 위해선 비율 제한기의 주기를 가능한 작게 설정해야 하며, 이 경우 웜 탐지시간은 상대적으로 증가하게 된다. 따라서 향후 이들의 상관관계를 면밀히 분석하여 비율 제한기의 주기를 작게 하면서도 빠르게 웜을 탐지할 수 있는 방안에 대해 연구할 예정이다.

참 고 문 헌

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

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

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

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

[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] C. C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine

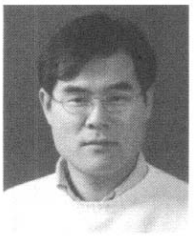
Defense," ACM CCS Workshop on Rapid Malcode (WORM'03), Washington DC, Oct., 2003.

[12] 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, Oct., 1993.

[13] 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

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

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



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 인터넷소프트웨어공학부 조교수

관심분야 : 임베디드시스템, 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템

김 장 복

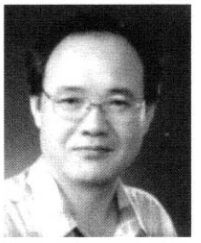
e-mail : bbok@ajou.ac.kr

2003년 아주대학교 정보및컴퓨터공학부 (학사)

2005년 아주대학교 정보통신전문대학원 정보통신공학과(석사)

2005년~현재 아주대학교 정보통신전문대학원 컴퓨터공학과 박사과정

관심분야 : 네트워크 보안, 임베디드시스템, 운영체제, 임베디드 소프트웨어 테스팅



최 경 희

e-mail : khchoi@madang.ajou.ac.kr

1976년 서울대학교 수학교육과(학사)

1979년 프랑스 그랑데꼴 Enseeiht대학 (석사)

1982년 프랑스 Paul Sabatier대학 정보공학부(박사)

1982년~현재 아주대학교 정보통신전문대학원 교수

관심분야 : 운영체제, 분산시스템, 실시간 및 멀티미디어시스템

정 기 현

e-mail : khchung@madang.ajou.ac.kr

1984년 서강대학교 전자공학과(학사)

1988년 미국 Ilinois주립대 EECS(석사)

1990년 미국 Purdue대학 전기전자공학부 (박사)

1991년~1992년 현대반도체 연구소

1993년~현재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템