# 바이러스 감속기의 가변 비율 제한기를 위한 자율적 주기 결정

정회원 심 재 홍*, 손 장 완**

# Autonomic Period Determination for Variable Rate Limiter of Virus Throttling

Jae-Hong Shim*, Jang-Wan Sohn** *Regular Members*

요 약

바이러스 감속기(virus throttling)은 연결요청 패킷의 전송비율을 일정 비율 이하로 제한함으로써 웜을 탐지하는 대표적인 조기 웜 탐지 기술 중의 하나이다. 기존 바이러스 감속기 연구에서는 웜 탐지시간에는 크게 영향을 미치지 않으면서도 연결설정 지연시간을 단축시키기 위해 가중치 평균 지연 큐 길이를 적용하여 비율 제한기의 주기를 자율적으로 조절하였다. 기존 연구에서는 비율 제한기의 최소주기를 고정하고 또한 주기 값을 감소시키다가 다시 증가시키기 시작하는 반환점도 미리 고정하였다. 그러나 이러한 두 성능결정 요소는 웜 탐지시간과 연결설정 지연에 서로 다른 영향을 미친다. 따라서 본 논문에서는 가변 비율 제한기의 최소주기와 반환점이 어떤 영향을 미치는지 실험을 통해 분석하고, 상황에 따라 이들 성능결정 요소들의 값을 결정할 수 있는 방안을 제시하고자 한다. 제안된 방법은 성능결정 요소를 고정시킨 기존 방법보다는 웜 탐지시간이나 연결설정 지연시간 단축에 더 효율적이라는 사실을 실험을 통해 확인하였다.

**Key Words** : 바이러스 감속기, 웜 조기 탐지, 인터넷 웜, 연결설정 지연

ABSTRACT

Virus throttling technique, one of many early worm detection techniques, detects Internet worm propagation by limiting connect requests within a certain ratio. The typical virus throttling controls the period of rate limiter autonomically by utilizing weighted average delay queue length to reduce connection delay time without hanving a large effect on worm detection time. In the existing virus throttling research, a minimum period of variable rate limiter is fixed and a turning point which is a point that the period of rate limiter has been being decreased and starts to be increased is also fixed. However, these two performance factors have different effects on worm detection time and connection delay. In this paper, we analyze the effect of minimum period and turning point of variable rate limiter, and then propose an algorithm which determines values of performance factors by referencing current traffic pattern. Through deep experiments, it is verified that the proposed technique is more efficient in respect of reducing worm detection time and connection delay than the existing virus throttling which fixed the performance factors.

※ 본 연구는 문화관광부 및 한국문화콘텐츠진흥원의 문화콘텐츠기술연구소(CT)육성사업의 연구결과로 수행되었습니다.
* 조선대학교 컴퓨터공학부 및 BK21 사업팀, 교수 (jhshim@chosun.ac.kr),
** 조선대학교 디자인학부, 교수 (jwson@chosun.ac.kr)
논문번호 : KICS2006-06-260, 접수일자 : 2006년 6월 4일, 최종논문접수일자 : 2007년 1월 15일

## Ⅰ. 서 론

인터넷을 기반으로 하는 많은 웜[1-5]들은 스스로 전파되는 속성을 가지고 있으며, 이는 주로 인터넷 상에 존재하는 호스트의 응용 프로그램들의 취약성을 이용한다. 따라서 웜에 감염된 호스트는 취약성을 가진 또 다른 호스트를 찾기 위해서 대상 호스트의 IP 주소를 무작위로 생성하여 조사하고, 수집된 정보를 이용하여 다른 호스트에 웜 코드를 전파한다. 웜들은 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.

웜 조기 탐지(worm early detection) 기술은 바이러스 감속기(virus throttling)[6,7], 수신-송신 상호관계(DSC: destination-source correlation)를 이용한 시스템[8], 순차적 가설 시험 (sequential hypothesis testing)을 이용한 알고리즘[9,10] 등이 있다. 바이러스 감속기[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 감속기는 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. Xinzhou Qin[8]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 혼합형(hybrid) 접근 방법[10]을 제안하였다.

그러나 위에서 소개한 방법들은 경계값-기반(threshold-based) 기술로 웜 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[12].

바이러스 감속기와 관련한 연구로는 바이러스 감속기의 지연 큐 관리에 있어 정상 트래픽의 연결요청 패킷들에 대해 지역성을 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 알고리즘이 제안되었다[13]. 또한 동일한 크기의 지연 큐를 가지고도 웜 탐지시간을 줄이고 전송된 웜 패킷 수를 줄일 수 있는 새로운 웜 탐지 알고리즘이 제안되었으며[14], 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 있게 하였다. 참고문헌 [14]는 실제 웜이 발생했을 때 보다 빠르게 웜을 탐지할 수 있는 새로운 웜 탐지 알고리즘을 제안한 것이며 이는 웜 탐지시간 단축이 주 목적이다. 이에 비해 참고문헌 [13]은 실제 웜이 발생하지도 않았는데도 웜이 발생한 것으로 잘못 판정하는 웜 발생 오판율을 줄이는 것이 주 목적이다.

그러나 본 연구에서는 웜 탐지시간의 단축 또는 웜 오판율을 감소하는 것이 아니라, 바이러스 감속기의 설치로 인해 평상시의 정상 연결 패킷에 대한 연결설정 지연시간을 단축하고자 하는 것이 목적이다. 반면에 웜 탐지시간은 가능하면 기존의 웜 탐지 시간과 거의 같아야 한다.

바이러스 감속기로 인한 연결 지연 단축을 위한 연구로는 비율 제한기의 주기를 고정시키는 기존 방법[6]을 탈피하고, 비율 제한기의 주기를 자율적으로 조절하는 가변 비율 제한기가 제안되었다[15]. 가

변 비율 제한기는 웜 탐지시간에는 큰 영향을 미치지 않으면서 연결설정 패킷의 지연시간을 단축시켜 사용자에게 바이러스 감속기의 설치로 인한 불편함을 줄여 주었지만, 최소주기와 반환점 등을 고정시켰다. 따라서 본 연구에서는 가변 비율 제한기의 두 성능결정 요소인 반환점과 최소주기가 웜 탐지시간과 연결설정 지연에 어떤 영향을 미치는지 분석해 보고, 상황에 따라 적절한 값을 선택할 수 있는 방안을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 바이러스 감속기와 이와 관련한 연구를 분석하고, 3장에서 가변 비율 제한기와 주기결정 알고리즘을 제안한다. 4장에서는 제안된 가변 비율 제한기의 성능에 영향을 미치는 반환점과 최소주기의 영향을 실험을 통해 분석하고, 5장에서는 반환점과 최소주기를 자동으로 결정할 수 있는 새로운 알고리즘을 제시하며 수정된 알고리즘의 성능을 분석한다. 마지막 6장에서는 본 논문의 결론을 기술한다.

## Ⅱ. 바이러스 감속기

일반적으로 정상적인 트래픽은 새로운 세션을 위한 연결요청이 상당히 낮은 비율(초당 약 1개)로 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 감속기[6,7]는 정상 트래픽과 웜 트래픽의 이러한 차이점을 이용하여 세션의 연결 설정 비율을 인위적으로 제한하여 웜의 전파 속도를 늦추고 차단하는 기법이다. (그림 1)은 바이러스 감속기에 의해 제어되는 전송 패킷들의 흐름을 나타낸 것이다. 새로운 연결요청 패킷(p)의 전송요청이 들어오면 워킹 셋(working set)에서 p와 동일한 수신 IP 주소가 존재하는지 확인하고, 만약 존재한다면 p를 정상 트래픽으로 간주하여 지연 없이 즉시 전송한다. 그렇지 않은 경우, 일단 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 가장 오래된 패킷을 꺼내어 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 패킷 전송을 중단하여 더 이상의 웜 전파를 차단한다.

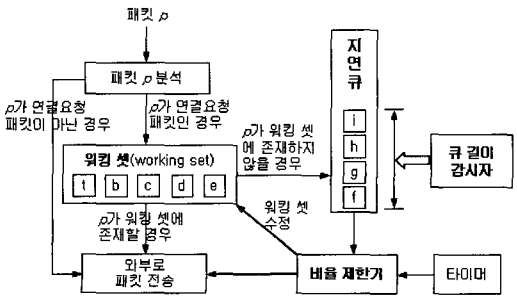



그림 1. 바이러스 감속기(virus throttling)

본 연구팀은 비율 제한기의 주기를 고정시키고 지연 큐 길이만으로 웜 발생 여부를 판단하는 기존 바이러스 감속기[6]의 방법을 탈피하고, 가중치 평균 지연 큐 길이를 적용하여 비율 제한기의 주기를 자율적으로 조절하는 가변 비율 제한기를 제안하였다[15]. 기존의 제안된 가변 비율 제한기의 기본 아이디어는 다음과 같다.

평균 지연 큐 길이가 증가하기 시작할 경우, 웜 때문인지 아니면 일반 패킷의 일시적 증가인지 구분하기 힘들기 때문에 기존 바이러스 감속기와 같이 비율 제한기의 주기를 고정시켜 운영한다. 이때 큐 길이가 꾸준히 계속 증가하면 웜일 가능성이 크고, 증가하다가 감소하기 시작하면 일시적인 트래픽 증가일 가능성이 높다. 평균 큐 길이가 감소하기 시작하여 일시적인 트래픽 증가로 판단될 경우 비율 제한기의 주기를 서서히 감소시켜 점점 빠르게 지연 큐 내에 대기 중이던 연결요청 패킷을 외부로 전송한다. 그래도 계속해서 평균 큐 길이가 감소하여 큐 길이가 짧아지면 이번에는 반대로 주기를 다시 증가시켜 서서히 원래의 주기로 되돌아 오게 하는 것이다. 만약 평균 큐 길이가 다시 증가하기 시작하면 웜 탐지시간에 영향을 미치지 않게 비율 제한기의 주기를 원래 주기로 바로 복원시킨다. 이러한 전략은 기존의 웜 탐지시간에는 큰 영향을 미치

지 않으면서 연결설정 지연시간을 단축시켜 사용자에게 연결설정 지연으로 인한 네트워크 속도 저하 등과 같은 불편함을 줄여줄 수 있는 장점이 있다.

그러나 기존 연구[15]에서는 주기 값을 감소할 때 최소주기 값을 0.5초로 설정하였다. 또한 주기 값을 감소시키다가 다시 증가시키기 시작하는 시점(반환점)도 미리 고정하였다. 그러나 이러한 두 매개변수는 설정된 값이 무엇인가에 따라 웜 탐지시간과 연결설정 지연에 서로 다른 영향을 미친다. 따라서 본 연구에서는 실험을 통해 두 매개변수가 미치는 영향을 분석해 보고, 상황에 따라 적절한 매개변수 값을 동적으로 변경할 수 있는 방안을 제시하고자 한다.

## Ⅲ. 가변 비율 제한기와 주기결정 함수

가변 비율 제한기를 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. 그러나 널리 잘 알려진 지수 평균 수식을 활용하여 가중치 평균 큐 길이(weighted average queue length)를 정의하고 이를 활용하면 이 문제는 쉽게 해결할 수 있다. 수식은 다음과 같다.

$$WAQL_n = \alpha * avgQL_n + (1 - \alpha) * WAQL_{n-1} \quad (1)$$

수식 (1)에서 $WAQL_{n-1}$은 직전의 주기에서 구해진 가중치 평균 큐 길이이며, $avgQL_n$은 현 샘플링 기간 동안의 평균 큐 길이이고, α는 새로운 $WAQL_n$을 계산할 때 현 샘플링 기간의 avgQLn과 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 가중치 평균 큐 길이를 활용하는 가변 비율 제한기의 주기결정 알고리즘이다. 제안 알고리즘은 매 샘플링(sampling time) 때(일반적으로 1초임)마다 한번씩 수행된다. 주기결정 알고리즘은 가중치 평균 큐 길이를 계산한 다음, 큐 길이가 증가하고 있는 추세면 비율 제한기의 주기를 기존 바이러스 감속기의 주기인 1초로 복원한다[6]. 그러나 큐 길이가 감소하고 있는 경우엔 주기결정 함수인

```
WAQL = ALPHA * avgQL + (1 - ALPHA) * WAQL;
if (WAQL >= prevWAQL) /* 큐 길이가 증가 추세 */
  RLP = 1;            /* 1 sec:원래 주기 값 */
else {                /* 큐 길이가 감소 추세 */
  if (RLP == 1)       /* 큐 길이가 감소 시작 */
    downWAQL = prevWAQL;
    RLP = GetRLP(WAQL);
    prevWAQL = WAQL;
}
```

그림 2. 가변 비율 제한기의 주기결정 알고리즘

GetRLP()를 호출하여 비율 제한기의 주기를 새로이 결정한다. 새로 결정된 주기 RLP는 이후 비율 제한기에 의해 새로운 주기로 설정된다.

알고리즘에서 WAQL은 가중치 평균 큐 길이이며, downWAQL는 WAQL가 증가하다가 감소하기 시작한 시점(이후 이를 감소지점이라 함)의 WAQL 값을 가지며, 이는 향후 실제 주기결정 함수인 GetRLP() 함수에서 사용된다. RLP는 GetRLP()에 의해 결정된 비율 제한기의 새로운 주기 값이다. prevWAQL는 직전의 샘플링 시점에서 구해진 WAQL이다.

주기결정 함수인 GetRLP()는 가중치 평균 큐 길이인 WAQL를 기반으로 다양한 방법으로 구현될 수 있다. 참고문헌 [15]에서 다양한 특성을 가진 웜에 대비해 가중치 평균 큐 길이가 감소하기 시작하면서 주기를 감소시켰다가 큐가 지속적으로 감소하여 큐 길이가 줄어들면 다시 원래의 주기로 서서히 증가시키는 전략이 가장 적절하다는 것을 실제 실험을 통해 확인하였다. 그러나 기존 주기 결정함수[15]는 감소시킬 수 있는 비율 제한기의 최소주기를 0.5로 고정시켰으며, 또한 주기를 감소시키다가 최소주기에 도달한 후 다시 증가시키는 지점인 반환점(turning point)을 WAQL가 계속 감소하여 downWAQL의 중간 지점에 도착할 때를 기점으로 하였다.

그러나 본 논문에서는 기존의 주기결정 함수를 확장시켜 보다 유연하게 이러한 값을 조절할 수 있는 새로운 주기결정 함수를 제안하고자 한다. 제안

```
double GetRLP (double WAQL)
{
if (WAQL < TURN_POINT)
  tmpRLP = (TURN_POINT - WAQL) / TURN_POINT;
else
  tmpRLP = (WAQL - TURN_POINT) /(downWAQL
  - TURN_POINT);
return tmpRLP * (1 - MIN_RLP) + MIN_RLP;
}
```

그림 3. 주기결정 함수

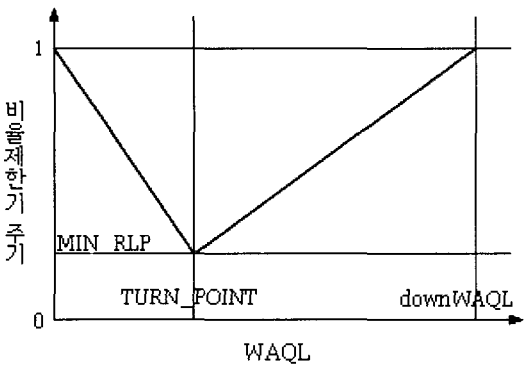



그림 4. 주기결정 함수 그래프

된 주기결정 함수는 (그림 3)과 같으며, (그림 4)는 주기결정 함수를 그림으로 표현한 그래프이다.

주기결정 함수에 의해 결정되는 가변 비율 제어기의 주기는 WAQL가 증가하다가 감소하기 시작한 시점, 즉 감소지점 downWAQL에서 원래 주기인 1이었다가 WAQL가 감소하면서 서서히 최소주기인 MIN_RLP까지 감소한다. WAQL가 반환점인 계속 감소하여 반환점인 TURN_POINT보다 작아지는 시점부터 주기는 다시 증가하기 시작하여 궁극적으로 원래 주기 1로 복귀한다.

주기결정 함수에 의해 결정되는 주기의 증감 비율은 WAQL을 매개변수로 하는 일차함수의 특성을 가지만, 최소주기인 MIN_RLP와 반환점인 TURN_POINT에 따라 증감 비율은 달라진다. MIN_RLP는 0~1 사이의 값을 가지지만, TURN_POINT는 절대값을 가질 수는 없으며 감소지점인 downWAQL에 대한 상대적인 값을 가진다. 예를 들어, TURN_POINT는 WAQL가 증가하다가 감소하기 시작한 감소지점인 downWAQL의 1/4, 1/2, 또는 3/4에 해당하는 값 등으로 설정될 수 있다.

## Ⅳ. 최소주기와 반환점의 영향분석

가변 비율 제한기의 주기결정 함수의 주요 성능 결정 요인인 최소주기와 반환점이 웜 탐지와 연결 지연에 어떠한 영향을 미치는지 알아보기 위해서 Linux 시스템에 바이러스 감속기와 제안된 알고리즘들을 구현하였다. Linux 커널 버전 2.6.9에 기존 알고리즘[7]을 참고하여 구현하였으며, (그림 5)와 같은 성능 실험 환경을 꾸몄다.

리눅스 시스템은 크게 2개의 모듈로 구성된다. 첫째가 커널에 구현된 바이러스 감속기 모듈로서,

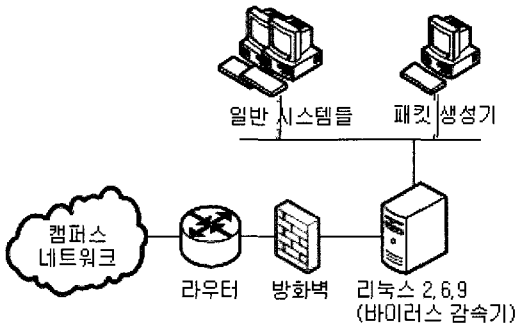



그림 5. 성능 실험 환경

리눅스에서 외부로 전달되는 모든 패킷들을 바이러스 감속기를 통하여 새로운 연결 비율을 조절하는 역할을 하며, 실험결과를 측정하기 위한 여러 통계 데이터도 수집하게 된다. 제안된 알고리즘도 이 모듈에 추가되었다. 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.

각 실험에서 비율 제한기 주기결정 알고리즘은 1초에 한번씩 실행되고, 가중치 평균 큐 길이를 위한 수식 (1)의 $\alpha$는 0.2로 설정하였고, 참고문헌 [6]에서처럼 웜 탐지를 위한 경계값은 100, 그리고 워킹셋 크기는 5로 설정하였다. 각 실험에서 연결설정 패킷들은 잘 알려진 파레토 온-오프(Pareto on/off) 트래픽 생성기법을 사용하여 생성하였다. 파레토 분포의 형상(shape) 매개변수는 1.5로 설정하였다.

최소주기와 반환점이 사용자에게 미치는 영향을 알아보기 위해 평균 연결설정 지연시간을 측정하였다. 여기서 연결설정 지연시간이란 연결요청 패킷이 지연 큐에 삽입된 이후부터 외부로 전송되기 직전까지 지연 큐에서 대기한 시간만을 의미한다. 실험에서 평균 온(on), 오프(off) 기간은 각각 1, 40초로 고정시키고, 온 기간 동안의 초당 평균 패킷 생성율은 5로 설정하였다.

<표 1>은 비율 제한기의 최소주기를 0.75, 0.5, 0.25등으로 변경하고, 또한 반환점을 감소지점 downWAQL의 1/4, 1/2, 3/4 지점 등으로 설정하였을 때 각각의 평균 연결설정 지연시간이다. 참고로 비율 제한기의 주기를 고정시킨 기존 바이러스 감속기 방식은 동일한 조건에서 3.41초의 평균 지연

표 1. 성능결정 요소별 평균 연결설정 지연시간 (단위: 초)

| 최소주기 \ 반환점 | downWAQL* 0.25 | downWAQL* 0.50 | downWAQL* 0.75 |
|---|---|---|---|
| 0.75 | 3.36 (1.4%) | 3.34 (1.9%) | 3.31 (2.8%) |
| 0.50 | 3.31 (2.8%) | 3.28 (3.7%) | 3.22 (5.4%) |
| 0.25 | 3.26 (4.2%) | 3.22 (5.4%) | 3.14 (7.9%) |

표 2. 성능결정 요소별 웜 탐지시간 (단위: 초)

| 최소주기 \ 반환점 | downWAQL* 0.25 | downWAQL* 0.50 | downWAQL* 0.75 |
|---|---|---|---|
| 0.75 | 6.71 (0.01%) | 6.71 (0.02%) | 6.72 (0.06%) |
| 0.50 | 3.72 (0.04%) | 6.72 (0.06%) | 6.72 (0.08%) |
| 0.25 | 3.72 (0.07%) | 6.73 (0.25%) | 6.81 (1.42%) |

시간을 보였다. 표에서 괄호안은 평균 연결 지연시간이 기존 방식에 비해 줄어든 단축율(%)이다. 전체적으로 최소주기와 반환점에 상관없이 기존 방식보다는 지연시간이 1~8% 줄어들었다. 이는 가변 비율 제한기를 사용할 경우 주기와 반환점에 따라 정도의 차이는 있지만 확실히 연결설정 지연시간이 줄어든다는 것을 의미한다.

표에서 반환점에 상관없이 가변 비율 제한기의 최소주기가 짧을수록 지연시간이 더 줄어든다. 이는 최소주기가 작을수록 그만큼 지연 큐의 패킷이 빠른 속도로 빠져나가기 때문이며, 쉽게 예측이 가능하다. 특이한 것은 반환점 역시 최소주기에 상관없이 가능한 감소지점인 downWAQL에 가까울수록 지연시간이 더 줄어든다는 사실이다. 이는 곧 가중치 평균 큐 길이가 감소하기 시작할 때부터 가능한 빠르게 최소주기로 감소시키고 반환점 이후 천천히 증가시키는 것이 보다 효율적이라는 것을 의미한다. 이 실험에서 연결설정 지연시간을 단축하기 위해서는 비율 제한기의 최소주기는 가능한 작아야 하고, 반환점은 가능한 감소지점에 가까워야 한다는 사실을 알 수 있다.

이번에는 최소주기와 반환점이 웜 탐지에 미치는 영향을 알아보기 위해 웜 탐지시간을 측정하였다. 실험에서 온 기간 동안 웜 패킷을 한꺼번에 많이 발생한 후 오프 기간 동안 전혀 패킷을 생성하지 않다가, 다시 온 기간 동안 웜 패킷 생성을 반복하는 탐지하기 까다로운 웜을 대상으로 하였다. 실험에서 웜의 평균 온, 오프 기간은 각각 1, 5초로 설정하였고, 온 기간 동안의 평균 웜 패킷의 생성율은 100으로 설정하였다. <표 2>는 다양한 최소주기와 반환점에 대한 웜 탐지시간이다. 참고로 비율 제한기의 주기를 고정시킨 기존 방식은 동일한 조건에서 6.71초의 평균 웜 탐지시간을 보였다. 괄호안은 기존 방식에 비해 웜 탐지시간이 증가한 증가율(%)이다.

전체적으로 최소주기와 반환점에 상관없이 기존 방식보다는 웜 탐지시간이 0.01~1.42% 증가하였다. 표에서 반환점에 상관없이 가변 비율 제한기의 최소주기가 길수록 웜 탐지시간이 기존 고정방식과 비슷해진다. 반환점의 경우 최소주기에 상관없이 가능한 감소지점인 downWAQL에서 멀어질수록 웜 탐지시간이 기존 방식과 유사해진다. 실험에서 확인할 수 있는 것은 웜 탐지시간을 가능한 기존 고정방식과 유사하게 하기 위해서는 비율 제한기의 최소주기는 가능한 커야 하고, 반환점은 가능한 감소지점에서 멀어져야 한다는 사실이다.

문제는 웜 탐지시간을 가능한 기존 방식과 동일하게 하고 연결설정 지연시간을 단축시키고자 하는 두 가지 목적을 동시에 만족하기 위해서는 <표 1>과 <표 2>에서 확인할 수 있듯이 최소주기와 반환점이 서로 반대가 되어야 한다는 것이다. 즉, 웜 탐지시간을 단축하기 위한 최소주기와 반환점의 요구조건이 연결설정 지연 단축을 위한 요구조건과는 서로 상반된다는 것이다.

그러나 만약 일반 트래픽과 웜 트래픽을 구분할 수 있다면 이 두 가지 목적을 동시에 만족할 수 있다. 예를 들어, 일반 트래픽일 경우 연결설정 지연을 단축하기 위해 <표 1>에서 확인했듯이 비율 제한기의 최소주기를 가능한 작게 하고 반환점을 감소지점에 가깝게 설정한다. 반면에 웜 트래픽일 경우 웜 탐지시간 단축을 위해 <표 2>에서 확인한 바와 같이 최소주기와 반환점을 일반 트래픽과는 반대로 설정하는 것이다.

## V. 최소주기와 반환점의 가변운영

본 논문에서는 일반 트래픽과 웜 트래픽을 구분하기 위해 기존 지연 큐의 특성을 면밀히 분석하였다. 그 결과 일반 트래픽일 경우 가중치 평균 큐 길이가 일시적으로 증가하다가 감소하는 시점인 downWAQL(감소지점)이 대체적으로 작은 값을 보이는 반면에, 웜이 발생한 경우는 감소지점이 주로 큰 값을 가진다는 사실을 발견하였다. 따라서 감소지점이 작을 때는 일반 트래픽인 것으로 간주하여, 연결설정 지연을 단축하기 위해 비율 제한기의 최소주기를 가능한 작게 하고 반환점을 감소지점에 가깝게 설정한다. 반면, 감소지점이 큰 값일 경우에

```
...
if (RLP == 1) {        /* 큐 길이가 감소 시작 */
  downWAQL = prevWAQL;
  ratio = 1.3 * downWAQL / QSIZE;
  MIN_RLP = ratio;
  TURN_POINT = (1 - ratio) * downWAQL;
}
...
```

그림 6. 최소주기와 반환점 결정

는 웜일 가능성이 있으므로 최소주기를 크게 하고 반환점을 가능한 감소지점으로부터 먼 곳으로 설정하며 가능한 지연 큐에서 패킷이 천천히 빠져 나가게 한다.

이러한 고찰을 기반으로 가중치 평균 큐 길이가 일시적으로 증가하다가 감소하는 시점에서 감소지점의 상대적인 크기에 따라 최소주기와 반환점을 자동으로 결정하도록 (그림 2)의 주기결정 알고리즘을 (그림 6)과 같이 수정하였다. 알고리즘에서 QSIZE는 지연 큐 길이 감시자가 웜 탐지를 위해 사용하는 경계값과 동일한 값인 지연 큐 크기이며, ratio는 QSIZE에 대한 downWAQL의 상대적인 크기이다. 따라서 감소지점인 downWAQL이 증가함에 따라 최소주기인 MIN_RLP 역시 증가하게 되고, 반면에 반환점인 TURN_POINT는 반대로 감소하게 된다. ratio를 계산할 때 상수 1.3을 곱한 것은 다양한 실험결과 가장 좋은 성능을 보였기 때문이며, 따라서 최소주기인 MIN_RLP가 기존의 고정방식의 비율 제한기 주기인 1보다 큰 값을 가질 수 있으므로 주기가 기존 방식보다 길어질 수도 있다.

주기결정 함수의 주요 성능결정 요소인 최소주기와 반환점을 자동으로 계산하는 수정된 주기결정 알고리즘이 얼마나 효율적인지 확인하기 위해 평균 연결 지연시간을 측정하였다. 실험에서 오프 기간은 평균 40초로 고정시키고, 각 트래픽 별로 온 기간과 온 기간 동안의 평균 연결설정 패킷 생성율을 다양하게 변경시켰다.

표 3. 평균 연결설정 지연시간 (단위: 초)

| 트래픽 번호 | 온 기간 | 패킷 생성율 | Fixed | Line_half | Line_auto |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0.59 | 0.59 | 0.59 |
| 2 | 1 | 2 | 1.09 | 1.09 | 1.06 |
| 3 | 2 | 2 | 1.94 | 1.91 | 1.83 |
| 4 | 1 | 4 | 2.57 | 2.51 | 2.37 |
| 5 | 2 | 3 | 3.53 | 3.39 | 3.18 |
| 6 | 1 | 6 | 4.32 | 4.10 | 3.82 |
| 7 | 2 | 4 | 5.24 | 4.89 | 4.58 |
| 8 | 1 | 8 | 6.13 | 5.66 | 5.31 |
| 9 | 1 | 10 | 8.21 | 7.41 | 7.09 |

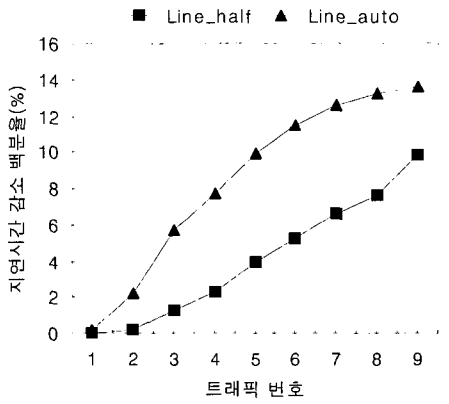



그림 7. 연결설정 지연시간의 감소율 (백분율)

<표 3>은 실험을 통해 측정된 평균 연결설정 지연시간이다. 표에서 Fixed는 비율 제한기의 주기를 고정하였던 기존 방식이고[6], Line_half는 최소주기와 반환점을 각각 0.5초, downWAQL*0.5로 고정시킨 채 비율 제한기의 주기를 변경시키는 방법[15]이며, Line_auto는 최소주기와 반환점을 자동으로 계산하여 비율 제한기의 주기를 변경시키는 본 논문에서 제안된 주기결정 알고리즘을 사용하는 방식이다. Line_half나 Line_auto 방식 모두 기존의 Fixed 방식보다 연결설정 지연시간이 단축되었음을 확인할 수 있다. 또한 Line_half에 비해 Line_auto가 지연시간이 더 단축되었음을 알 수 있다. (그림 7)은 Fixed와 비교했을 때, Line_half나 Line_auto의 연결설정 지연시간 감소율을 백분율로 표시한 것이다. 대체적으로 Line_auto 전략이 Line_half보다 감소 백분율이 2~3배 정도 더 향상되었다는 것을 알 수 있다. 이는 최소주기와 반환점을 고정시키는 것 보다는 감소지점의 상대적 크기에 따라 변경시키는 것이 더 우수한 성능을 보인다는 것을 의미한다.

표 4. 웜들의 시간적 특성

| 웜 분류 | 웜 이름 | 초당 연결 패킷 수 | 오프 기간(초) |
|---|---|---|---|
| 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 |

<표 4>는 가변 비율 제한기로 인한 웜 탐지시간의 변화를 알아보기 위해 생성된 다양한 시험용 웜들(CW1~CW3, SW1~SW5)과 실제 웜인 Nimda[1], Code Red[3]의 시간적 특성을 보여 준다. CW1~CW3, Nimda, Code Red 등과 같은 연속적 웜들은 오프 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, 온 기간 동안 한번에 많은 패킷을 발생시킨 후 일정시간(오프 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 오프 기간이 짧고, 온 기간 동안 생성되는 연결 패킷의 수가 일반 트래픽보다는 매우 많다는 점에서 차이를 보인다. SW1~SW5는 산발적 웜을 실험하기 위해 생성되었으며, 이들은 온 기간을 1초로 동일하게 설정하고 온 기간 동안에 생성되는 패킷 수와 오프 기간의 길이를 서로 다르게 설정하였다.

<표 5>는 연속적 웜에 대해 수정된 주기결정 알고리즘이 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 5. 연속적 웜들에 대한 웜 탐지시간 (단위: 초)

| 웜 이름 | Fixed | Line_half | Line_auto |
|---|---|---|---|
| Nimda | 0.28 | 0. 28 | 0. 28 |
| Code Red | 0.44 | 0. 44 | 0. 44 |
| CW 1 | 0.94 | 0.94 | 0.94 |
| CW 2 | 1.96 | 1. 96 | 1. 96 |
| CW 3 | 10.88 | 10.88 | 10.88 |

표 6. 산발적 웜들에 대한 웜 탐지시간 (단위: 초)

| 웜 이름 | Fixed | Line_half | Line_auto |
|---|---|---|---|
| SW 1 | 6.71 | 6.72 | 6.71 |
| SW 2 | 13.37 | 13.42 | 13.41 |
| SW 3 | 15.09 | 15.13 | 15.12 |
| SW 4 | 20.64 | 20.76 | 20.72 |
| SW 5 | 31.38 | 31.82 | 31.73 |

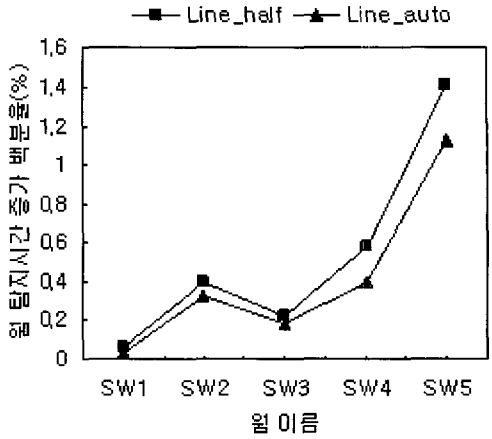



그림 8. 산발적 웜들에 대한 웜 탐지시간 감소율 (백분율)

<표 6>은 산발적 웜에 대한 수정된 주기결정 알고리즘(Line_auto)의 웜 탐지시간을 보여준다. 연속적 웜과는 달리 산발적 웜들은 1초간 웜 패킷이 생성되고 오프 기간 동안 전혀 웜 패킷이 생성되지 않기 때문에 WAQL가 감소하기 시작한다. 이때부터 주기결정 함수에 의해 비율 제한기의 주기가 짧아지게 되고, 따라서 지연 큐의 연결요청 패킷이 기존 Fixed 방식보다 약간씩 더 빠르게 큐에서 제거되어 외부로 전송된다. 그러나 다시 온 기간 동안 새로운 패킷이 지연 큐에 유입되면 가중치 평균 큐 길이가 증가하기 시작하므로 비율 제한기의 주기는 원래대로 복구된다. 이후 오프 기간에 다시 감소하기를 반복한다. 오프 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. 따라서 Line_half나 Line_auto 방식 모두 기존의 Fixed 방식보다 산발적 웜에 대한 웜 탐지시간은 모두 미세한 증가를 보인다. 비록 웜 탐지시간은 미세하게 증가했지만 (그림 7)에서 확인했듯이 연결설정 지연시간은 상대적으로 더 큰 폭으로 감소하므로 Fixed 방식에 비해 상대적으로 이점이 더 있다고 할 수 있다. (그림 8)은 Fixed와 비교했을 때, Line_half나 Line_auto의 웜 탐지시간 증가율을 백분율로 표시한 것이다. 대체적으로 Line_half나 Line_auto 모두 Fixed 전략에 비해 1.4%이내에서

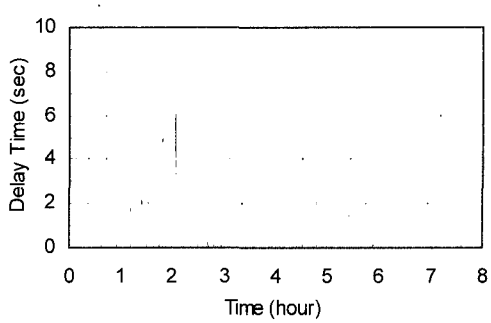
미세하게 증가했지만, Line_half에 비해 Line_auto가 웜 탐지시간 증가율은 상대적으로 더 작다는 것을 알 수 있다.

<표 5>와 <표 6>의 두 실험에서 최소주기와 반환점을 고정시키는 Line_half 전략 보다는 감소지점인 downWAQL의 상대적 크기에 따라 이들을 동적으로 변경시키는 Line_auto 전략이 웜 탐지시간이나 연결설정 지연시간에서 보다 더 효율적이라는 사실을 확인할 수 있다. 또한 <표 1>과 <표 2>에서 확인한 바와 같이 웜 탐지시간을 단축하기 위한 최소주기와 반환점의 요구조건이 연결설정 지연 단축을 위한 요구조건과는 서로 상반되는 특성을 가졌음에도 불구하고, <표 5>와 <표 6>처럼 최소주기와 반환점을 가변적으로 조절하여 줌으로써 연결설정 지연시간은 더 줄이되 웜 탐지시간은 오히려 늘지 않는 효과를 거둘 수 있음을 확인했다.

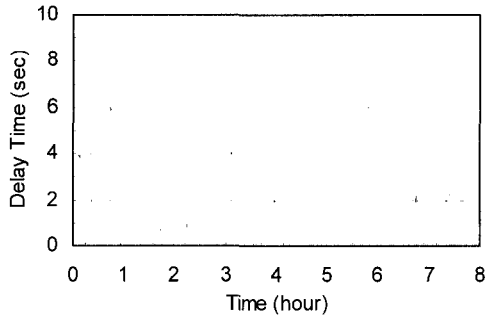
(그림 9)는 지연 큐에서 제거된 패킷들이 큐에서 대기한 평균 지연시간을 매초 별로 표시한 것이다. 동일한 트래픽에 대해 그림 (a)는 기존의 Fixed 전략을 사용한 경우이고, 그림 (b)는 Dec-inc 전략을 사용한 가변 비율 제한기를 설치했을 때의 시간대별 평균 지연시간을 보여 준다. 실험에서 오프 기간은 40초, 온 기간은 1초, 온 기간 동안의 패킷 생성율은 2로 설정하였다.

(그림 9)의 (a)와 (b)를 비교했을 때, 트래픽이 적을 경우의 지연시간 차이는 근소한 차이 밖에 나지 않지만, 트래픽이 많은 경우 Fixed 보다는 Line_auto 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, 트래픽이 작아 큐 길이가 짧을 경우에는 가변 비율 제한기의 변경된 주기가 적용되는 기회가 그만큼 줄어들기 때문에 지연시간 단축에는 큰 영향을 미치지 못하기 때문이다.

(그림 9)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 감속기의 설치로 인해 발생하는 문제이다. 따라서 지연 큐의 길이가 갑자기 증가했을 때




(a) Fixed 전략




(b) Line_auto 전략

그림 9. 시간대별 연결설정 지연시간

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 고정하기 보다는 능동적으로 변경시킬 경우 웜 탐지시간에는 커다란 영향을 미치지 않으면서 연결설정 지연시간을 상당히 감소시킬 수 있음을 알 수 있다. 이는 곧 사용자에게 바이러스 감속기의 설치로 인한 불편함을 상대적으로 줄여 줄 수 있음을 의미한다. 또한 주기결정 전략에 있어 최소주기와 반환점을 고정시키는 것보다는 감소지점의 상대적 크기에 따라 이들을 동적으로 변경시키는 것이 웜 탐지시간이나 연결설정 지연시간에서 보다 더 효율적이라는 사실을 확인할 수 있었다.

## Ⅵ. 결 론

본 논문에서는 가변 비율 제한기의 두 성능결정 요소인 최소주기와 반환점이 웜 탐지시간과 연결설정 지연에 미치는 영향을 실험을 통해 분석하였다. 그 결과 연결설정 지연시간을 단축하기 위해서는

최소주기는 가능한 작아야 하고, 반환점은 가능한 감소지점에 가까워야 하지만, 반대로 웜 탐지시간에 영향을 미치지 않기 위해서는 최소주기는 커야 하고 반환점은 작아야 하는 서로 상반되는 요구사항이 존재한다는 사실을 발견했다.

이 문제를 해결하기 위해 가중치 평균 큐 길이가 일시적으로 증가하다가 감소하는 시점에서 지연 큐 크기에 대한 감소지점의 상대적인 크기에 따라 최소주기와 반환점을 자동으로 선택하여 비율 제한기의 주기를 능동적으로 변경하는 알고리즘을 제안했다.

다양한 실험을 통해 제안된 알고리즘은 비율 제한기의 주기를 고정시킨 기존 전략[6]보다 정상 트래픽에 대한 연결설정 지연시간을 최고 13%까지 감소시킬 수 있었으며, 이는 최소주기와 반환점을 고정시킨 채 비율 제한기의 주기를 변경하는 기존 전략[15]보다 연결설정 지연시간의 감소율을 2~3배 정도 향상시킨 결과이다. 이러한 효과에도 불구하고 실제 웜이 발생했을 때의 웜 탐지시간은 비율 제한기의 주기를 고정시킨 기존 전략과 동일하거나 약 1% 정도의 미세한 증가를 보였지만, 최소주기와 반환점을 고정시킨 기존 전략에 비해서는 웜 탐지시간이 오히려 감소된 고무적인 결과를 보였다.

## 참 고 문 헌


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

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

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

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

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

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

[9] 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.

[10] 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.

[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] 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.

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

[14] 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.

[15] 심재홍, "바이러스 쓰로틀링의 연결설정 지연 단축," 한국멀티미디어학회 2006 춘계학술발표대회 논문집, pp. 405-408, May 2006.

**심 재 홍** (Jae-Hong Shim) 정회원

1987년 서울대학교 전산과학과 졸업(학사)
1989년 아주대학교 컴퓨터공학과 졸업(석사)
2001년 아주대학교 컴퓨터공학과 졸업(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

**손 장 완** (Jang-Wan Sohn) 정회원

1990년 조선대학교 공과대학 전기공학과 졸업
2002년 조선대학교 대학원 시각디자인전공 (미술학 석사)졸업
1995년~1998년 (주)A.G.M Com, puter Graphic사업부문 이사
1998년~2002년 조선대학교 산업디자인 특성화사업단 연구원
2005년 12월~현재 다온미디어 대표
2006년 4월~현재 조선대학교 문화콘텐츠기술연구소 선임연구원
2004년 3월~현재 조선대학교 미술대학 디자인학부 겸임교수
<관심분야> 3D, 영상