네트워크 시스템의 세션 관리 부하를 감쇄하기 위한 사건 기반 타임아웃 정책

임 강 빈[†] · 최 창 석[††] · 문 종 욱[†††] · 정 기 현[††††] · 최 경 희[†††††]

요 약

방화벽이나 침입탐지시스템과 같은 세션 관리를 요하는 시스템은 관리하는 세션 테이블의 크기가 증가함에 따라 각 세션에 대한 타임아웃 처리 시 발생하는 오버헤드가 커지게 된다. 본 논문은 기존의 타이머를 이용한 시간 기반 타임아웃 관리 방법에 비하여 시스템의 부하를 현저히 감쇄하여 네트워크 시스템의 패킷 처리량을 증가시킬 수 있는 사건 기반 타임아웃 관리 방법을 제안한다. 또한, 실제 구현한 시스템을 이용한 실험을 통하여 제안한 방법이 기존의 방법에 비하여 보다 많은 패킷을 처리할 수 있음을 확인한다.

An event-based timeout policy to decrease the overhead of session managements in network systems

Kangbin Yim[†] · Changseok Choi[††] · Jongwook Moon[†††]
Gihyun Jung[††††] · Kyunghee Choi[†††††]

ABSTRACT

The session management overhead on the network systems like firewalls or intrusion detection systems is getting grown as the session table is glowing. In this paper, we propose the event-based timeout management policy to increase packet processing throughput on network systems by decreasing the system's timeout management overhead that is comparable to the existing time-based timeout management policies. Through some empirical studies using a session management system implemented in this paper we probed that the proposed policy provides better packet processing throughput than the existing policies.

키워드 : 네트워크 시스템(Network Systems), 세션 관리(Session Management), 타임아웃(Timeout)

1. 서 론

요즘 널리 사용되는 TCP/IP 프로토콜 스택은 상당히 많은 테이블들을 동적으로 처리하고 있다. 이처럼 TCP/IP 프로토콜 스택에 구현된 ARP cache, DNS cache, session state table과 같이 동적으로 증가하는 테이블들은 필히 타임아웃 관리를 필요로 한다. 이들 테이블들이 증가함에 따라 검색 시간이 증가하여 시스템의 성능이 떨어지고 메모리 사용량이 증가하여서 메모리 이용율이 떨어지게 된다. 이를 방지하기 위하여 여러 가지 타임아웃 관리 기법을 통해서 테이블의 크기를 줄여 주어야 한다.

타임아웃 관리는 여러 응용분야에 사용되는 기법으로 각 용도에 따라 적절한 방법이 요구되고 또한 그 타임아웃 관리 방법이 얼마나 효율적인가에 따라 시스템의 성능에 큰 영향을 미친다[1].

현재 보안 분야의 대표적인 장비인 방화벽(firewall)[11]과 침입탐지시스템(IDS)[12]에서는 과도한 트래픽에 따른 많은 양의 세션 관리가 큰 이슈가 되고 있다. 이러한 세션 관리에 있어서 타임아웃 관리 기법이 중요한 역할을 하고 있다. 보안 시스템에서 세션 타임아웃 관리는 기존에 주로 연구 되어 오던 타임아웃 관리와는 성격 면에서 차이가 난다. 기존의 연구들은 얼마나 세밀한 단위로 성능의 저하를 최소화 하면서 타임아웃 처리를 해낼 수 있느냐에 대한 것이다[1]. 하지만 보안 시스템에서의 세션 타임아웃 관리는 타임아웃 이벤트가 발생하더라도 경우에 따라서는 즉시 동작을 요하지는 않는다. 이러한 성격을 이용해서 본 논문에서는 기존의 시간 기반 방식을 일부 이용한 사건 기반 방식의 타임아웃 관리 기법을 제안한다.

본 논문의 실험에 따르면 세션의 수가 많아지는 경우 사건 기반 방식을 사용하게 되면 성능 면에서 예측 성이 뛰어나고 통신 지연이 시간 기반 방식이 경우 보다 좋은 결

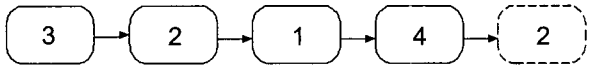
※ 본 논문은 과기부 국가지정연구실사업 및 정통부 IT분야 해외교수초빙 국제공동연구사업의 지원으로 연구되었음.
† 정 회 원 : 순천향대학교 정보보호학과 교수
†† 준 회 원 : 삼성전자 디지털 미디어 총괄 연구원
††† 준 회 원 : 시큐아이닷컴 연구원
†††† 정 회 원 : 아주대학교 전자공학부 교수
††††† 정 회 원 : 아주대학교 정보 및 컴퓨터공학부 교수
논문접수 : 2004년 1월 13일, 심사완료 : 2004년 2월 23일

과를 보인다.

앞으로 이 논문에서는 2장에서 본 논문에서 사용할 용어를 정의하고 기존에 연구되어온 타이머를 이용한 타임아웃 관리 기법에 대해 조사한다. 이어서 3장에서는 세션 관리 시스템에 적합한 사건 기반 타임아웃 관리 방법을 제시 하게 된다. 4장, 5장에서는 실제 세션 관리 시스템을 사용하여 시간 기반 방식을 사용했을 경우와 사건 기반 방식을 사용했을 때의 시스템의 성능의 차이를 실험을 통해 알아볼 것이다. 마지막 6장에서 실험결과를 통한 결론을 도출한다.

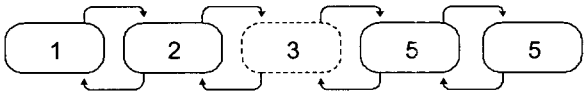
2. 타임아웃 메커니즘

일반적으로 운영체제에서의 태스크 상태나 네트워크 프로토콜에서의 연결 상태 또는 시스템의 가용성(availability) 증진을 위한 기동 상태의 천이를 위하여 타임아웃 기법이 자주 사용된다. 이러한 타임아웃 기법들은 다양한 형태로 구현되어 사용되고 있으나 그 구조를 기반으로 분류하면 Straightforward 방식, Ordered list 방식, Hierarchical Timing Wheels 방식 등 크게 세 가지로 나누어 볼 수 있다.



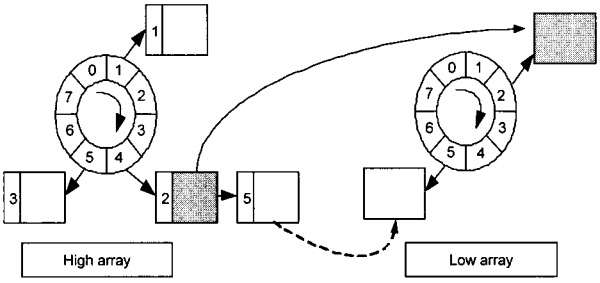
(그림 1) Straightforward 방식

Straightforward 방식은 새로운 타이머 설정 과정에서 (그림 1)에서와 같이 해당 타임아웃 값을 타이머 리스트의 끝에 추가 등록하고 주기적으로 모든 타이머의 만료를 차례로 검사한다[2]. 이 방식은 관리대상이 적을 경우와 타임아웃 값이 작은 경우 좋은 성능을 보이며 구현하기가 간단한 방법이다. 그러나 관리대상이 많아지면 타이머의 만료 검사를 위한 오버헤드가 커지게 된다.



(그림 2) Ordered list 방식

Ordered List 방식은 Straightforward 방식에 비하여 타이머 만료 검사에 대한 오버헤드를 줄일 수 있는 방법이다[1, 2]. 이 방법은 (그림 2)에서와 같이 새로운 타이머를 등록하는 과정에서 만료시각이 빠른 순서대로 정렬하여 저장한다. 만료의 검사는 주기마다 리스트를 앞에서부터 현재 시각보다 큰 항목을 만날 때까지 이루어진다. 검사하는 과정에서 현재 시각보다 작은 시각을 가진 항목을 만나면 그 항목을 지우고 만료 처리 루틴을 호출한다. 이 방법은 새로운 타이머의 등록 과정에서 해당 타이머가 위치할 곳을 탐색하기 위한 오버헤드가 발생한다. 다만, 모든 타이머가 동일한 타임아웃 값을 갖는 응용에서는 새로운 타이머의 등록이 리스트의 끝에서만 이루어질 수 있으므로 상기의 오버헤드가 발생하지 않아 좋은 성능을 보이게 된다. 이 방법은 단순한 연결 리스트 방식뿐만 아니라 트리 기반의 자료구조를 사용하는 unbalanced binary trees, heaps, post-order and end-order trees, leftist-trees 등의 방법을 사용하면 보다 성능을 높일 수 있다[3, 4].



(그림 3) Hashed and hierarchical Timing Wheels 방식

Hierarchical Timing Wheels은 시간을 계층적으로 나누어 각각에 여러 개의 슬롯을 가지는 timing wheel을 두는 방법이다. 타이머의 등록은 만료시각에 따라 각 Wheel의 특정 슬롯으로 나누어 등록된다. 타이머의 만료 검사는 현재 시각에 해당하는 특정 Wheel들을 따라가다가 등록된 타이머 리스트가 존재하면 해당 리스트를 만료시킨다. 이 방법은 타이머의 등록 시에 계층의 수가 많아질수록 오버헤드가 증가하고 구현이 복잡하며 많은 메모리를 사용하게 되는 단점이 있다[1].

타임아웃을 관리하는 방법은 어플리케이션의 성격에 따라 적절하게 선택되어야 한다. 이를 위하여 다음과 같은 세 가지 항목을 고려할 수 있다. 첫째는 타임아웃 검사 주기의 정밀도(granularity)이다. 이는 타임아웃 값의 만료를 검사하는 간격을 말하며 이는 타이머의 정밀도에 밀접한 관련이 있는 요소이다. 세밀한 정밀도의 검사 주기는 높은 정확성을 보장하지만 시스템의 성능에 오버헤드로 작용한다. 운영체제 커널에서의 프로세스 스케줄링, TCP Reno 등의 응용이 비교적 높은 정밀도를 요구하는 대표적인 어플리케이션들이며 TCP Vegas 및 방화벽과 침입탐지시스템에서 세션 타임아웃 관리 때 사용되는 응용들이 낮은 정밀도를 요하는 대표적인 어플리케이션이다[1, 5]. 둘째, 타임아웃 관리 대상의 양과 종류이다. 타임아웃 관리 대상의 양은 타임아웃 만료를 검사하는 시점에서 타이머 모듈이 관리해야 하는 대상의 양을 말한다. 이 양은 어플리케이션에 따라 불과 수십 개에서 수 만개에 이르기까지 다양하다. 이처럼 관리대상의 양이 많을수록 일반적으로는 타임아웃 검사 정밀도를 세밀하게 적용하기가 힘들어진다. 관리대상의 종류는 각 대상에 적용되는 타임아웃 값의 종류를 말하며 종류가 많아지면 타임아웃 처리의 효율성을 위한 여러 가지 정열 방법이 사용된다[1]. 셋째, 반응 특성이다. 이는 타임아웃 만료 시의 반응의 성향을 말한다. 이 반응은 크게 수동적 반응(passive reaction)과 능동적 반응(active reaction)으로 나뉠 수가 있다. 수동적 반응은 응용의 성격에 따라 어느 정도의 지연을

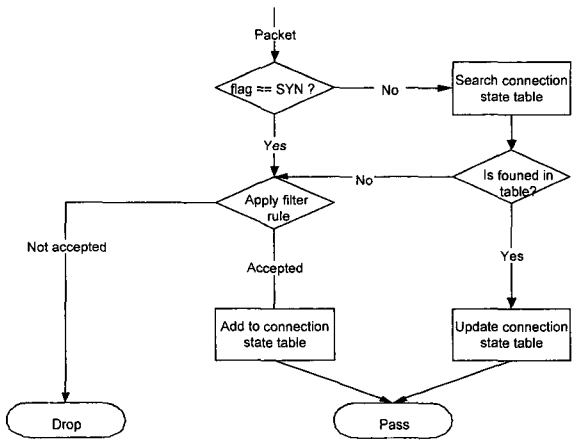
허용하는 soft real-time 성격을 가지는데 반하여 능동적 반응은 지연을 거의 허용하지 않는 hard real-time 성격을 가진다. 예를 들어 패킷의 재전송의 경우, 그 동작이 늦어지면 전체적인 성능에 큰 영향을 미치기 때문에 이를 위한 타임아웃 만료는 능동적 반응을 요구하는 응용이다.

타임아웃 관리 방법을 선택함에 있어 추가적으로 고려해야 할 요소는 그 방법이 가지는 오버헤드이다. 여기서의 오버헤드는 공간과 시간에 따라 크게 두 종류로 나누어 볼 수 있는데, 공간적 오버헤드란 관리 방법이 사용하는 자료구조의 메모리 요구량이고 시간적 오버헤드란 타임아웃 값의 설정과 갱신, 만료, 만료검사 등의 동작에 소요되는 지연시간이다[1]. 그러나, 오버헤드란 일반적으로는 시간적 오버헤드를 말한다.

3. 세션 타임아웃 관리

방화벽이나 침입탐지시스템은 상태검사(stateful inspection)[6]와 같은 기능을 사용하여 망에 흐르는 패킷의 연결상태(connection state)를 기록하고 그에 따른 적절한 동작을 수행한다. 이러한 네트워크 상의 각 연결에 대한 상태를 기록하고 관리하는 기능을 세션 관리라고 한다.

상태검사는 네트워크 계층에서 동작하는 패킷 여과(packet filtering) 기술로서 강력한 보안 기능을 제공한다. 기존의 패킷 필터링 방식이 헤더의 정보만을 이용했다면, 상태검사 방식은 이보다 발전된 방식으로 헤더 정보와 함께 내용까지 해석하여 패킷을 여과하는 방식이다. 상태검사는 네트워크 계층에서 접속을 가로채어 검사엔진에 전달하고, 응용계층으로부터 정책 결정에 필요한 상태 관련 정보를 동적 상태테이블(Dynamic State Table)에 보관하게 되는데, 이후에 들어오는 접속에 대해서는 이것과 비교해서 통과 여부를 결정하게 된다. 일례로 상태검사 방화벽의 동작 과정은 (그림 4)와 같다[9].



(그림 4) Stateful inspection 동작 flow(TCP)

방화벽을 거치는 모든 TCP 패킷은 (그림 4)의 과정을 거쳐서 세션 테이블에 기록되거나 버려진다. 이러한 세션 관리에서 사용하는 연결 상태 테이블의 주요 항목을 다음 (그림 5)에 보였다.

(그림 5)에 따르면 연결을 구분 하기 위한 정보로 목적지와 출발지 IP 주소와 포트를 사용하였고 TIMEOUT을 통하여 해당 연결이 유지되는 만료시간을 적용하여 세션 테이블의 무한 증식을 막는다. 해당 세션의 패킷이 들어올 때마다 STATE 항목과 TIMEOUT 항목이 갱신된다.

```
{
unsigned long SOURCE_IP ;
unsigned long DESTINATION_IP ;
unsigned short SOURCE_Port ;
unsigned short DESTINATION_Port ;
unsigned long TIMEOUT ;
unsigned short STATE ;
}
```

(그림 5) 세션 table 주요 항목

리눅스(LINUX)는 패킷 여과 도구인 iptables에서 connection tracking이라는 기능으로 상태검사 기능을 구현하고 있다[10]. iptables의 경우 각 프로토콜 별로 적용하는 타임아웃을 <표 1>에 보였다. Stateless 프로토콜인 UDP의 경우는 각각의 UDP 세션에 상태 정보를 유지하여 요구에 대한 실제의 응답만 전달하고 다른 모든 패킷은 버린다. 만약 정해진 시간 내에 응답이 오지 않으면, 연결은 타임아웃 된다.

<표 1> LINUX의 주요 Protocol 별 타임아웃

PROTOCOL	STATE	TIMEOUT
TCP	ESTABLISHED	5 DAYS
	SYN_SENT	2 MINS
	SYN_RECV	60 SECS
	FIN_WAIT	2 MINS
	TIME_WAIT	2 MINS
	CLOSE	10 SECS
	CLOSE_WAIT	60 SECS
	LAST_ACK	30 SECS
	LISTEN	2 MINS
UDP		30 SECS

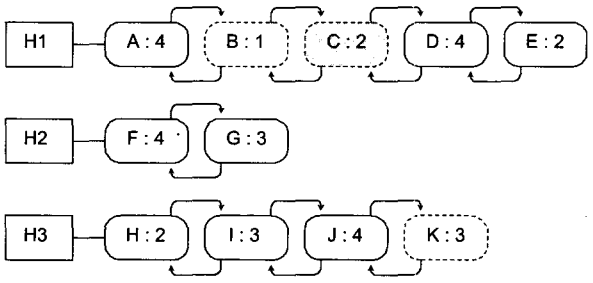
이러한 세션 관리 하에서 타임아웃 처리의 특징을 살펴보면 타임아웃 처리 대상의 양이 많다는 것, 타임아웃 값의 갱신이 빈번하다는 것, 그리고 타임아웃 만료시 즉각적인 반응 동작이 없어도 된다는 것의 세 가지를 발견할 수 있다. 약 50,000~100,000개 세션의 타임아웃을 처리해야 하는 세션 관리에서 기존의 타이머를 사용한 방법을 사용시 많은 관리 대상으로 인한 PERTICKBOOKKEEPING(주기적인 타임아웃 만료 검사)의 오버헤드가 발생한다. 또한 100M 네트워크인 경우 최대 초당 약 150,000개의 패킷이 지나가게 된다. 이런 상황에서 세션 테이블에 존재하는 패킷마다

타임아웃 값을 갱신하게 되는데 이는 STARTTIMER(타임아웃 값의 등록과 갱신)의 오버헤드가 매우 큼을 알 수 있다. 본 논문에서는 이러한 세션 관리에서의 타임아웃 처리가 가지는 오버헤드를 효율적으로 줄임으로써 시스템의 성능 향상을 위한 방안을 제안한다.

4. 사건 기반 세션 타임아웃 관리

본 논문에서 제안하는 세션 타임아웃 방법은 기존의 타이머를 사용한 시간 기반 타임아웃 관리 방식이 아닌 사건 기반 타임아웃 관리 방법이다. 세션 관리 시스템은 매 패킷마다 연결 상태를 검사하여 세션 테이블에 STATE와 TIMEOUT 값을 갱신하게 된다. 즉 매 패킷마다 세션 테이블을 검색하는 루틴을 거치게 된다. 이처럼 세션 관리 시스템이 필수적으로 행하게 되는 세션 테이블 검색 루틴을 이용하여 타임아웃 관리의 오버헤드를 줄일 수 있다. 즉 패킷이 도착(사건 발생)하여 시스템 동작의 필요에 의해 행하는 세션 테이블을 검색하는 과정 동안 거치게 되는 세션들의 타임아웃 만료를 검사하는 것이다. 세션 관리 시스템의 특성 중 타임아웃 만료시 그 반응 동작이 즉각적일 필요가 없음에 의하여 타임아웃이 발생하여도 즉시 해당 항목을 제거할 필요가 없다. 차후에 다시 참조되는 시점에 만료를 확인한 후 제거해도 되는 것이다. (그림 6)에 제안하는 방법을 나타낸다.

제안하는 방법에서 사건에 대한 해시는 각 세션의 출발지, 도착지 IP address와 port를 사용하여 분산 시켰고, 타임아웃 값은 절대 시간을 기록하는 방식을 사용 하였다. 또한 세션 관리 시스템은 내부 네트워크와 외부 네트워크 사이의 세션 만을 관리하며 관리 시스템이 내부나 외부와 세션을 맺지는 않는다고 가정한다.



(그림 6) Event based timeout management 방식

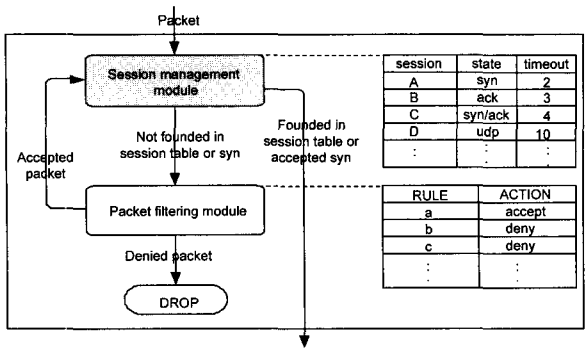
(그림 6)에서 H1, H2, H3는 패킷 도착 시의 패킷의 세션 정보를 가지고 해시를 사용하여 분산되는 루틴을 말한다. A~K까지의 알파벳은 각 패킷의 세션 정보를 나타내고 뒤의 숫자는 타임아웃 만료시의 절대시간을 나타낸다. 타임아웃 처리 시의 동작을 보면 다음과 같다. 즉, 세션 테이블을 검색하는 동안에 거치게 되는 세션에 대해서만 타임아웃 만료를 검사하며 검색 루틴 중에 만료된 세션은 제거한다. 또한, 검색 결과 새로 들어온 세션은 해당 해시 리스트의 종단에 위치시키며, 메모리의 효율성을 위해 세션의 개수가 기준치(S)에 다다르면 전체 리스트를 체크하여 만료된 세션을 제거한다. 예를 들어 현재 시간이 3인 시점에서 D 세션의 패킷이 도착하면 D까지 검색하는 루틴 중에서 시간 3보다 적은 값을 가지는 항목들(B, C)은 제거한다. 또한 H3 루틴으로 분류된 K 라는 새로 들어온 세션 정보는 리스트의 제일 뒤에 삽입된다.

이 방법은 기존의 타이머를 이용한 방법 중 Straightforward 방법과 Ordered list 방법 그리고 Hashed and hierarchical Timing Wheels 방법의 장점을 취하였다. 다시 말하면 Straightforward 방법의 STARTTIMER 루틴의 장점을 사용하였고 Ordered list 방법의 PERTICKBOOKKEEPING 루틴의 장점과 세 번째 방법의 해시를 사용하여 오버헤드를 줄였다. 차이점은 시간을 기준으로 한 것이 아니라 사건을 기준으로 검색과 hash를 한 것이다. Time based 방식은 별도의 타이머를 사용함으로써 해당 tick 마다 주기적으로 인터럽트를 발생시키게 된다. 이는 시스템의 전체적인 성능면에서 많은 리소스를 잠식하는 결과를 초래한다. 이에 대하여 사건 기반 방식은 시스템의 필요에 의하여 수행되는 필수 루틴의 과정에서 타임아웃을 관리하므로 과중한 인터럽트 오버헤드를 크게 줄일 수 있게 된다.

5. 실험 및 결과 분석

본 장에서는 실험을 위한 네트워크 환경과 방법 그리고 실험을 위해 사용된 성능 측정 도구들에 대해서 기술하고 이를 통하여 도출된 결과를 분석한다. 우선 실험을 위하여 사용한 소프트웨어 도구들을 살펴 보면 analyzer[7]와 mgen/drec[8]이 있다. analyzer는 WinPcap library를 이용하여 패킷 캡쳐 및 트래픽 모니터링을 할 수 도구이다. 또한 프로토콜 별 트래픽 모니터링 기능과 사용자가 런타임에 변경 가능한 프로토콜 양식대로 네트워크 패킷을 분석할 수 있는 기능을 가진다. 본 실험에서는 analyzer를 사용하여 세션 관리 시스템이 처리하는 패킷 처리량을 측정하였다. mgen은 UDP 트래픽을 발생하여 IP 네트워크의 성능을 테스트하기 위해 만든 도구이며, 패킷의 크기와 전송률 등을 사용자가 임의로 변경할 수 있는 일종의 패킷 생성기라고 볼 수 있다. drec은 패킷 수신기라고 볼 수 있는데 mgen에서 패킷을 생성할 때 순서 번호를 붙여서 보내기 때문에 중간에 생긴 패킷 손실을 drec을 사용하여 측정할 수 있다. 본 실험에서는 랜덤한 세션을 생성할 수 있도록 mgen의 소스코드를 수정하여 사용하였다.

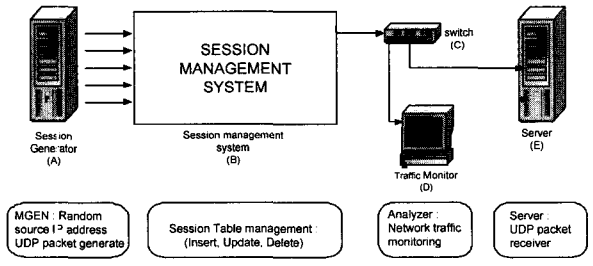
실험은 통신 프로세서인 MPC8250을 사용한 세션 관리 시스템을 사용하였다. 세션 관리 시스템의 주요 동작 모습을 다음 (그림 7)에 나타내었다. 실험에서는 (그림 7)의 Session management module에서 타임아웃을 관리하는 방법에 따른 시스템의 패킷 처리량을 알아본다.



(그림 7) 세션 관리 시스템의 동작도

실험 방법은 세션 생성 서버에서 세션이 랜덤하게 바뀌는 64byte 크기의 UDP 패킷을 내보내고 Traffic Monitor를 통해 세션 관리 시스템이 처리하는 양을 관찰하였다. 세션이 랜덤하게 바뀌도록 UDP 패킷 생성 프로그램인 mgen의 코드를 수정하여 사용하였다. 전체 세션의 양을 4,000~130,000개로 조정해가면서 실험하였다. 패킷은 64byte 크기의 UDP 패킷을 초당 50,000개씩 전송하였다.

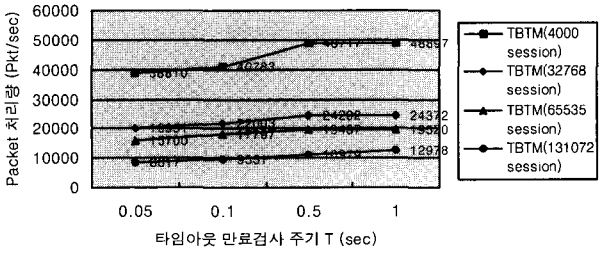
비교할 두 방식을 각각 Event Based Timeout Management(EBTM)과 Time Based Timeout Management(TBTM)이라고 하고 다음과 같이 측정하였다. EBTM 방식은 타임아웃 만료의 검사 및 타임아웃의 설정과 갱신 방식을 그림 6의 방식을 사용하였다. 메모리 사용량을 제한하기 위해서 세션 테이블 마진 S를 두어 S개의 세션이 테이블에 등록되면 테이블 전체의 타임아웃 만료를 검사하였다. 실험 결과는 S를 변경하면서 세션 관리 시스템이 처리하는 패킷의 양을 측정하여 나타냈다. TBTM 방식은 타임아웃 만료의 검사는 주기적으로 전체의 세션 테이블을 검사하였고 타임아웃 설정과 갱신은 (그림 6)의 방식을 그대로 사용하였다. 실험 결과는 타임아웃 만료를 검사하는 간격을 1초, 0.5초, 0.1초, 0.05초로 바꿔가며 세션 관리 시스템이 처리하는 패킷의 양을 측정하여 나타냈다. 테스트 환경은 (그림 8)과 같다.



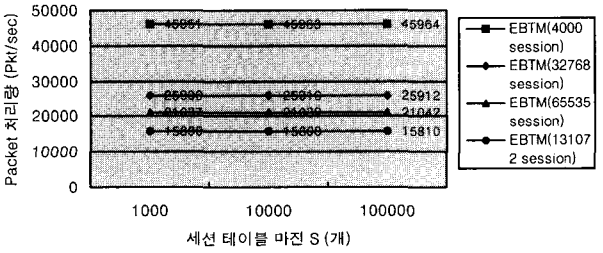
(그림 8) 실험 네트워크 환경

(그림 8)의 실험 환경에서 세션 생성기(A)는 UDP 패킷 생성 툴인 mgen을 사용하여 64byte 크기의 UDP 패킷을 생성하였고 랜덤한 세션을 생성하기 위하여 mgen의 코드를 수정하여 UDP 패킷의 source IP 부분을 랜덤하게 하여 보내도록 하였다. 세션 생성기(A)에서 생성된 64byte 패킷은 세션 관리기(B)를 거쳐 서버(E)로 전송된다. 세션 관리기와 서버 사이의 스위치(C)는 UDP 트래픽을 트래픽 모니터(D)로 미러링(mirroring)해 준다. 트래픽 모니터는 analyzer를 사용하여 세션 관리기와 서버간의 트래픽 양을 보여준다. 실험에 사용한 세션 관리기는 단지 세션 테이블 관리만을 하며 양단간에 패킷을 전달하는 역할만을 한다.

(그림 8)의 실험 환경에서 세션 생성기(A) 장비에서 세션 관리기를 통하여 서버(E) 장비로 보내는 UDP 패킷의 양을 트래픽 모니터(D) 장비로 관찰하였다. 모든 실험은 UDP 트래픽을 가한 후 20초 이상 경과 후에 트래픽 모니터(D) 화면의 패킷 처리량의 변화량이 안정화된 시점에서 초당 패킷 처리량을 기록하였다. 총 10회의 실험을 한 후 순간적인 통신망 부하의 이상변화에 의한 영향을 줄이기 위해 최대치와 최저치를 제외한 나머지의 8개의 실험치의 평균값으로 결과를 표시하였다. TBTM 방식에서는 타임아웃 만료검사 주기 T의 값을 바꾸어 실험한 결과를 (그림 9)에 보였다. (그림 9)에서 T가 세밀해 질수록 패킷 처리이 떨어지는 결과를 볼 수 있다. 또한 세션의 양이 많아질수록 패킷 처리량이 떨어지는 결과를 볼 수 있다.



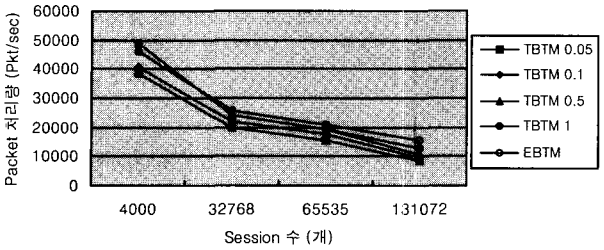
(그림 9) TBTM



(그림 10) EBTM

EBTM 방식으로 세션 테이블 마진 S를 바꾸어서 실험한 결과를 (그림 10)에 보였다. (그림 10)의 결과를 보면 세션 테이블 마진 S의 변화에는 거의 무관한 패킷 처리량을 보이고 있다. 또한 EBTM 방식도 세션의 수가 많아질수록 패킷 처리량이 적어지는 것을 볼 수 있다.

위 두 개의 그래프를 통합하여 EBTM 방식과 TBTM 방식을 비교한 그래프를 (그림 11)에 나타내었다. 실험 결과 EBTM의 경우 세션 테이블 마진 S의 변화와는 거의 무관한 결과를 보여서 실험 평균치를 사용하였다. (그림 11)을 보면 세션의 수가 30,000개 이상이면 EBTM의 경우가 더 좋은 성능을 보이고 있다. 또한 타임아웃 만료검사 주기 T가 세밀한 경우에는 세션의 수가 작은 경우에도 EBTM 방식이 좋은 결과를 보이고 있다.



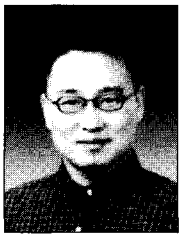
(그림 11) EBTM과 TBTM 비교

6. 결 론

본 논문에서는 네트워크 시스템의 세션 관리에서 기존의 시간 기반 타임아웃 관리 방법이 가지는 부하를 줄이기 위하여 새로운 사건 기반 타임아웃 관리 방법을 제안하였다. 제안한 방안은 실제의 세션 관리가 요구되는 네트워크 시스템을 이용하여 실험을 통하여 그 성능을 입증하였다. 실험 결과, 관리되는 세션의 수가 많아질수록 제안한 사건 기반 타임아웃 관리를 사용한 것이 기존의 시간 기반 방식보다 더 좋은 성능을 보임을 확인하였다. 이는 관리 대상의 양이 많아질 수록 주기적으로 타이머 인터럽트를 발생시켜 세션 테이블을 검색해야만 하는 시간 기반 방식의 오버헤드가 커짐에서 기인한다. 세션이 적은 경우에도 타이머의 해상도가 세밀해지면 사건 기반 방식이 더 좋은 성능을 보이는 것을 알 수 있었다.

참 고 문 헌

[1] G. Varghese, A. Lauck, "Hashed and Hierarchical Timing Wheels : Efficient Data Structures for Implementing a Timer Facility," IEEE/ACM Transactions, Vol.5, pp.824-834, 1997.
[2] A. S. Tanenbaum, "Computer Networks," 3rd ed., Prentice-Hall.
[3] T. Cormen, C. Leiserson, R. Rivest, "Introduction to Algorithms," McGraw-Hill.
[4] J. G. Vaucher, P. Duval, "A comparison of simulation event list algorithms," Commun.ACM, Vol.18, pp.223-230, 1975.
[5] L. Brakmo, S. O. Malley and L. Peterson, "TCP Vegas : New techniques for congestion detection and avoidance," Proc. ACM SIGCOMM, pp.24-35, 1994.
[6] Stateful inspection, http://www.checkpoint.com/products/downloads/Stateful_Inspection.pdf.
[7] Analyzer, http://analyzer.polito.it/, 2002.
[8] MGEN, http://manimac.itd.nrl.navy.mil/MGEN/.
[9] N. A. Noureldien, I. M. Osman, "A stateful inspection module architecture," TENCO 2000, Vol.2, pp.259-265, 2000.
[10] Net filter/iptables connection tracking, http://www.netfilter.org/, 2003.
[11] NAT, EnteraSys white paper, http://www.enterasys.com/products/whitepapers/ssr/network-trans/index.pdf, 2001.



임 강 빈

e-mail : yim@sch.ac.kr
1992년 아주대학교 전자공학과(학사)
1994년 아주대학교 전자공학과(석사)
2001년 아주대학교 전자공학과(박사)
1999년~2000년 (미)아리조나 주립대 객원연구원
2001년~2002년 아주대학교 정보통신대학 대우조교수
2003년~현재 순천향대학교 정보보호학과 전임강사
관심분야 : 네트워크 보안, 실시간 운영체제, 임베디드 시스템, 멀티미디어 시스템 등

최 창 석

e-mail : dadaz@naver.com
2002년 아주대학교 전자공학과(학사)
2004년 아주대학교 전자공학과(석사)
2004년~현재 삼성전자 디지털 미디어 총괄 연구원
관심분야 : 실시간 운영체제, 임베디드 시스템 등

문 종 욱

e-mail : lache@ajou.ac.kr
2000년 아주대학교 전자공학과(학사)
2002년 아주대학교 전자공학과(석사)
2004년 아주대학교 전자공학과 박사수료
2004년~현재 시큐아이닷컴 연구원
관심분야 : 네트워크 보안, 실시간 운영체제, 임베디드 시스템 등

정 기 현

e-mail : khchung@ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 Univ. of Illinois, EECS(석사)
1990년 Univ. of Purdue, 전기전자공학부(박사)
1991년~1992년 현대반도체 연구소
1993년~현재 아주대학교 전자공학부 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

최 경 희

e-mail : khchoi@ajou.ac.kr
1976년 서울대학교 사범대학 수학교육과(학사)
1979년 프랑스 그랑데꼴 ENSEIHT, 정보공학 및 응용수학(석사)
1982년 프랑스 Univ. of Paul Sabatier(박사)
1991년~1991년 프랑스 렌느 IRISA 연구소 교환 교수
1982년~현재 아주대학교 정보 및 컴퓨터 공학부 교수
관심분야 : 운영체제, 분산처리, 실시간 시스템 등