네트워크 대역폭 고갈 공격에 대한 정책 기반 재구성 가능 대역폭제어기

박 상 길[†] · 오 진 태[††] · 김 기 영[†††]

요 약

초고속 인터넷 망등의 국내 인터넷의 저변확대로 인해 전자상거래, 인터넷뱅킹, 전자정부, 이메일등 의 많은 서비스와 다양한 정보의 보고로서 인터넷이 사용되고 있다. 근래에는 가상생활환경의 제공과 멀티미디어 서비스를 제공하고자 새로운 미래형 네트워크인 NGN(Next Generation Network)로서 발전하고 있다. 인터넷은 원격지에서도 원하는 정보를 취득할 수 있는 장점이 있는데, 반대 급부로서 상대방의 정보를 허가없이 몰래 추출,변조하거나 서비스를 제공하는 경쟁사의 서버를 다운시키는 등의 공격이 증대되고 있다. 2000년부터 님다(Nimda) 바이러스, 코드레드(Code Red) 바이러스, 분산서비스 거부 공격(DDoS : Distributed Denial of Service)이 인터넷 전반에 걸쳐 수행되어 네트워크의 사용을 불편하게 하며, 내부 네트워크 트래픽의 비정상적인 증가를 수반했다. 이러한 대역폭 고갈 침해공격에 대하여 네트워크의 유입점에 위치하는 게이트웨이 시스템에 기가비트 이더넷 인터페이스를 갖는 보안네트워크 카드에 재구성 가능한 하드웨어 기능을 제공 가능한 FPGA (Field Programmable Gate Arrary)상에 대역폭 제어기능인 폴리싱(Policing)을 구현한다.

Policy-based Reconfigurable Bandwidth-Controller for Network Bandwidth Saturation Attacks

Sang-kil Park[†] · Jin-tae Oh[††] · Ki-young Kim[†††]

ABSTRACT

Nowadays NGN is developed for supporting the e-Commerce, Internet trading, e-Government, e-mail, virtual-life and multimedia. Internet gives us the benefit of remote access to the information but causes the attacks that can break server and modify information. Since 2000 Nimda, Code Red Virus and DDoS attacks are spreaded in Internet. This attack programs make tremendous traffic packets on the Internet. In this paper, we designed and developed the Bandwidth Controller in the gateway systems against the bandwidth saturation attacks. This Bandwidth controller is implemented in hardware chipset(FPGA) Virtex II Pro which is produced by Xilinx and acts as a policing function. We reference the TBF(Token Bucket Filter) in Linux Kernel 2.4 and implemented this function in HDL(Hardware Description Language) Verilog. This HDL code is synthesized in hardware chipset and performs the gigabit traffic in real time. This policing function can throttle the traffic at the rate of bandwidth controlling policy in bps speed.

키워드 : 토큰버켓(Token Bucket), 폴리싱(Policing), DDoS, 공격대응(Attack Response), 트래픽제어(Traffic Control)

1. 서 론

인터넷을 이용한 유용한 정보의 제공과 멀티미디어의 활용으로 인해 인터넷에 대한 관심이 증대되는 반면, 이를 악용하여 허가없이 타인의 정보를 추출, 변조하거나 경쟁상대의 서비스제공을 방해하는 등의 공격이 증대되고 있다. 또한 불특정 다수를 대상으로 하는 블라인드 공격 또한 증대되고 있는 실정이다. Yahoo, eBay 사 등 웹 사이트에 대한 분산 서비스 거부(DDoS : Distributed Denial of Service, 이하 DDoS) 공격과 그에 대한 피해사례의 보고는 인터넷에 개방되어 있는 시스템들이 DDoS 공격에 대해 매우 취약하다는 것을 보여준다. DDoS 공격은 아주 짧은 시간 안에 다량의 패킷을 공격의 목표로 삼는 타겟 시스템 또는 타겟 네트워크에 전송함으로써, 서비스를 일시적으로 중단시키거나 원활한 서비스를 유지할 수 없도록 유도한다. 전세계의 인터넷을 운영하는 핵심체인 ICANN(Internet Corporation for Assigned Names and Numbers)측의 발표에 의하면 2002년 10월 22일 전세계의 인터넷 트래픽을 관리하는 중앙 DNS 서버에 대한 다량의 분산서비스 거부 공격으로 인해 13대 중 7대의 서버가 다운되었다. 관리자의 적극적인 빠른 대응으로 인해 DNS 서버가 복구되어 인터넷의 이용에 극심한

† 정 회 원 : 한국전자통신연구원 연구원
†† 정 회 원 : 한국전자통신연구원 선임연구원
††† 정 회 원 : 한국전자통신연구원 책임연구원
논문접수 : 2004년 9월 21일, 심사완료 : 2004년 11월 8일

불편을 초래하지는 않았지만, 인터넷에 대한 공격이 얼마나 심각한 결과를 초래하는지 보여주는 사건이었다. 또한 2003년 1월 25일에는 Slammer 웜을 통한 불특정 다수로의 서비스 거부공격과 동일한 결과를 갖는 웜이 활동하였다[2, 4, 8, 11].

본 논문의 2장에서는 분산서비스 거부 공격의 동작원리와 공격의 형태, 그리고 이러한 공격을 최소화하기 위한 기존 네트워크 및 보안장비의 한계점을 살펴보며, 3장에서는 논문에서 제시하는 대역폭제어기가 구현되는 네트워크 오용타지 및 대응카드의 구조와 재구성 가능한 폴리싱모듈과 그 구현 기법에 대하여 제시한다. 이러한 구조를 통하여 하드웨어 칩셋인 Xilinx 사의 Virtex II Pro에 구현된 네트워크 대역폭제어기를 통하여 실제 네트워크 오용 공격이 발생하였을 경우 이를 탐지하고, 이러한 공격에 대하여 대역폭을 제어하여 합법적인 사용자의 트래픽을 보호하고, 유해한 트래픽만을 제어하는 실험과 결과를 4장에서는 제시한다.

2. 분산 서비스 거부 공격

서비스 거부 공격(DoS : Denial of Service) 공격은 1990년대 후반부터 공격으로서 분류되기 시작하였으며, 하나의 근원지 주소에서 하나의 타겟 시스템에 대한 공격으로 시작되었다. 그러나, 최근 들어서는 N 대 N 관계를 갖는, 즉 분산 서비스 거부 공격으로 진화되었다.

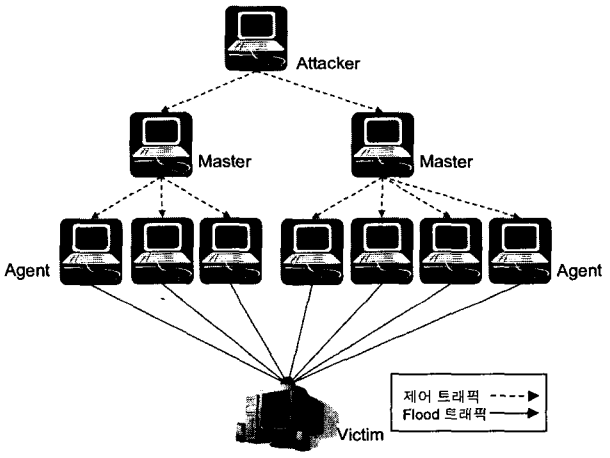
DDoS는 인터넷의 구조적인 취약성을 이용하는 공격으로서 심각한 경우 몇 시간 또는 몇 일 동안 정상서비스의 마비상황을 야기시키는 일련의 네트워크 공격을 말한다. 이러한 분산서비스 거부 공격을 당한 기관이나 회사에게는 엄청난 피해사태를 일으켜 기관이나 기업의 신뢰도를 심각하게 저하시키고, 직·간접적인 피해보상을 해야 하는 최악의 결과를 가져올 수도 있다.

기존의 일반적인 해킹 방법은 해커가 기업의 네트워크에 불법적인 방법으로 침투해서 중요한 시스템의 슈퍼 사용자 권한을 획득하고, 그 내부에 들어있는 기밀 정보를 탈취하는 것이 대표적이었다. 하지만 DDoS 공격은 기업이 고객 서비스를 위해 운영하는 서버들(웹 서버, DNS 서버 등)과 네트워크 장비(라우터, 방화벽 등)에 임의로 조작된 엄청난 양의 공격성 트래픽을 전송해서 시스템 자체를 지연 혹은 마비시켜 버림으로써 고객들이 기업의 서비스를 이용할 수 없는 서비스거부(Denial of Service) 상황을 만드는 목적을 가진다는 점에서 그 성격이 다르다[9].

최근에는 기업의 서비스용 웹 사이트나 서버들에 대한 공격을 통해 서비스를 하지 못하도록 네트워크를 마비시키는 공격 형태에서 점차 ISP(Internet Service Provider)의 코어(Core) 라우터, DNS 서버들을 직접 공격해서 인터넷 인프라 자체를 완전히 마비시키는 형태로 발전해 가고 있다. 인터넷의 IP등을 관리하는 기관인 ICANN에 대한 공격으로 2002년 10월에 발생했던 13개의 최상위 루트 DNS(Domain Name Service) 서버들에 대한 대규모의 무차별적 공격으로 적지 않은 혼란을 주었던 사례가 있었다. 또한, 2003년 1월에 발생했던 1.25 인터넷 대란은 슬래머 웜에 의한 공격이 발단이 되었지만 결국 그 형태는 DDoS 공격이었다[8, 9, 11].

2.1 DDoS 공격의 동작 원리

인터넷의 표준인 TCP/IP 프로토콜은 태생적으로 구조적인 보안의 취약점을 가지고 있다. 모든 인터넷 사용자는 TCP/IP 프로토콜을 이용하여 임의의 데이터 패킷을 발송자의 IP 주소(Source IP)를 가지고 목적지의 IP 주소(Destination IP)로 발송할 수 있다. DDoS 공격은 수백 혹은 수천 개의 Zombi들을 이용해서 Victim System을 공격하는 형태를 가진다. 이러한 많은 Zombi들은 공격 명령이 떨어지면 그 순간 피해자에서 가해자로 돌변하여 일제히 타깃 시스템을 공격하게 된다. 이렇게 수많은 Zombi들이 각자 생성하는 무차별적인 공격의 트래픽 볼륨을 생성하여 서비스 불능상태를 만들어 낸다. (그림 1)은 DDoS 공격을 시작하려는 준비단계와 실제 DDoS공격 트래픽인 Flood 트래픽을 전송하는 모습을 나타낸다. Attacker는 텔넷 또는 암호화된 전용 클라이언트 프로그램을 이용해서 Master에 접속하고 공격명령을 전달한다. 그러면, 공격명령을 받은 Master는 자신에게 속한 Agent에게 다시 공격명령을 하달하고, Agent들은 동시에 Victim을 향해서 대역폭 고갈 공격을 시도한다. 이러한 DDoS 공격을 발생시키는 공격으로 Code Red, Code Red II, Nimda 등의 자동화된 인터넷 웜(Worm)을 통해서 이러한 DDoS 공격도구의 배포가 가능하다.



(그림 1) 분산서비스 거부 공격의 구조

2.2 DDoS 공격의 형태

DDoS 공격은 기본적으로 다음 두 가지 형태를 가진다.

- 대역폭 공격(Bandwidth Attacks) : 이 형태의 공격은 엄청난 양의 패킷을 전송해서 네트워크의 대역폭이나 장

비 자체의 리소스를 모두 사용하는 형태이다. 라우터, 서버, 방화벽 같은 주요 장비들은 제한적인 처리용량을 가지고 있기 때문에 그 용량을 초과하는 이런 형태의 공격을 받게 되면 정상적인 서비스를 처리 못하게 되거나 장비 자체가 작동 불능상태가 되어서 네트워크 전체가 서비스 불능 사태를 초래할 수 있다. 이 형태의 대표적인 공격은 패킷 오버플로 공격인데, 이것은 정상적으로 보이는 엄청난 양의 TCP, UDP, ICMP 패킷들을 특정한 목적지로 보내는 것이다. 또한, 발송지의 주소를 스푸핑해서 발송하기 때문에 공격의 탐지가 쉽지 않다.

- 애플리케이션 공격(Application Attacks) : 이 형태의 공격은 TCP와 HTTP 같은 프로토콜을 이용해서 특정한 반응이 일어나는 요청 패킷을 발송하여 해당 시스템의 연산처리 리소스를 소진시켜서 정상적인 서비스 요청과 처리가 불가능한 상태로 만드는 것이다.

2.3 기존 네트워크 및 보안 장비의 한계점

DDoS 공격이 발생되는 빈도에 대한 통계자료를 보면 결코 운 나쁜 어떤 기업이 어쩌다가 불행한 사고를 당했다고 생각하고 넘어갈 성격이 아니라는 것을 알 수 있다. 저명한 인터넷 트래픽 측정 연구기관인 CAIDA(Cooperative Association for Internet Data Analysis)가 집계한 통계자료에 의하면 2001년 한해 동안 주당 4000건의 DDoS 공격이 발생하였고, 시간당 100건 이상의 공격이 동시에 일어나고 있다.

DDoS 공격에 대응하기 어려운 가장 큰 이유는 해커가 발송하는 비정상적인 공격성 패킷과 고객이 발송하는 정상적인 서비스 요청 패킷을 정확히 구분하는 것이 무척 어렵기 때문에 DDoS 공격을 탐지하는 것 자체가 힘들다는 점이다. 또한 정확히 탐지를 한다고 해도 라우터나 방화벽을 통한 필터링을 하게 되면 그 장비 자체가 죽어버리는 경우가 많기 때문이다

이러한 DDoS 공격에 대한 대응책으로 다음과 같은 기술들이 적용되고 있으나, 완벽한 해결책으로는 미비한 실정이다.

- 블랙홀링(BlackHoling) : 블랙홀링 기술은 라우터에서 특정 목적지(Victim)로 전송되는 모든 트래픽에 대하여 FIB(Forwarding Information Base) 테이블에 등록하여 Null Interface로 불리우는 트래픽 폐기장소로 보내서 소멸시키는 방법이다. 하지만, 해당 목적지로 전송되는 악성공격 패킷들뿐만 아니라 정상적인 패킷들도 포함한 모든 트래픽이 소멸되기 때문에 이 방법은 해결책이 될 수가 없다.
- 싱크홀링(SinkHoling) : 싱크홀링 기술은 라우터에서 특정 목적지(Victim)으로 전송되는 모든 트래픽에 대하여 FIB(Forwarding Information Base)를 조작하여 트래픽 분석이 가능한 특정 머신으로 패킷데이터를 전송하는 기술이다. 이를 통하여 대량의 이상 공격트래픽에 대하여 유해성 여부를 탐지할 수 있다. 하지만, 이러한 공격은 실시간적인 대응을 제공할 수 없는 한계점이 있다.
- 라우터(Router) : 라우터는 ACL(Access Control List)을 이용한 필터링 기능을 제공하는데, 이 기능만으로는 현재의 고도화된 DDoS 공격을 방어하기 어렵다. 첫째, 라우터는 핑(Ping) 공격같은 인터넷 통신에 꼭 필요하지 않은 몇 가지 간단한 DDoS 공격에 대해서는 필터링 메커니즘을 통해 방어할 수 있다. 하지만 최근의 DDoS 공격은 인터넷을 사용하기 위해서 가장 기본적이고 필수적인 프로토콜을 사용하기 때문에 특정 프로토콜 자체를 모두 필터링하는 방법은 사용할 수가 없는 것이다. 둘째, 스푸핑된 공격에 대해서는 uRPF를 이용한 방어법이 권고되고 있으나 실제로 구현하는 것은 한계가 있다. 셋째, 라우터의 ACL기능은 소스가 스푸핑되었는가 여부에 상관없이 HTTP 에러와 HTTP half-open 커넥션 공격 같은 애플리케이션 레이어의 공격에 대해선 그 효과를 발휘하기 어렵다는 한계점을 가진다.
- 방화벽(Firewall) : 방화벽은 몇가지 한계점을 가지고 있다. 첫째, 방화벽은 네트워크 트래픽이 흐르는 경로의 내부(In-line)에 위치하기 때문에 방화벽 자체가 공격의 대상이 되어 대용량의 공격 트래픽에 의해 다운되게 되면 방화벽을 통과하여 전달되는 모든 트래픽이 단절되어, 전체 네트워크가 마비되는 문제가 발생할 수 있다. 둘째, 방화벽이 비정상적인 행위를 정확히 탐지할 수 있다고 할지라도 개별 패킷들에 대해서 정상적인 것인지 아닌지의 여부를 구분할 수 있는 기능이 없기 때문에 스푸핑된 소스에서의 공격은 방어하기 어렵다.
- 침입탐지시스템(IDS) : 첫째, 기존의 시그니처 방식에만 의존하는 IDS는 정상적인 패킷으로 가장한 변형된 형태의 공격에 대해서는 해당 공격 시그니처가 없이는 방어가 어렵다. 둘째, DDoS 공격을 방어하는 수단으로서 IDS의 가장 큰 한계점은 단지 공격을 탐지만 할 수 있고, 방어할 수 있는 어떠한 수단도 제공하지 못한다는 점이다. IDS는 효과적인 방어 솔루션과 통합/연계되어 상호 보완적인 기능을 수행하는 형태로 활용되어야 할 것이다.
- 매뉴얼 반응(Manual Response) : 사람이 직접 수작업을 통해 방어를 하는 경우는 그 대응이 너무 부분적이며 즉각적인 반응이 어려울 것이다. DDoS 공격을 당했을 때 전형적인 첫번째 반응은 해당 ISP에 연락해서 소스를 밝혀달라고 요청하는 일이다. 스푸핑된 주소일 경우에는 다수의 ISP들이 협력해서 검증해야 하는 복잡하고 오랜 작업이 필요하다. 또한, 소스가 밝혀진다고 할지라도 그것을 차단한다는 것은 모든 트래픽을 차단한다는 의미가 되므로 장단점이 있다고 할 수 있다.

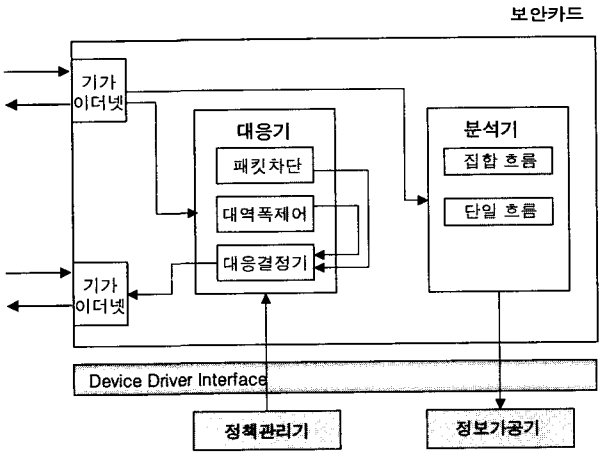
- 로드 밸런싱(Load Balancing) : 마지막으로 로드 밸런싱 혹은 이중화, 삼중화 등을 통해서 더욱 용량이 큰 트래픽에 대해서도 처리할 수 있도록 네트워크의 대역폭 및 성능을 강화시키는 방법이다. 이러한 방법은 대량의 트래픽을 어떻게 해서든 서비스 하려는 목적을 갖지만, 종국적으로는 그 이상의 트래픽에 의해 서비스 불가하는 상황을 초래할 수 있다.

3. 네트워크 고갈 공격에 대응하기 위한 재구성 가능한 폴리싱

본 논문에서는 분산서비스거부 공격과 같은 네트워크 고갈공격에 대응하기 위하여 네트워크의 접속점에 In-line형태로 위치하는 게이트웨이 시스템 내에 탐재되는 보안카드 를 통하여 네트워크 유해 트래픽을 탐지하고 대응을 수행한다.

3.1 네트워크 오용탐지 및 대응카드의 구조

네트워크 오용탐지 및 대응을 위한 기가비트 보안카드는 기가비트 이더넷 인터페이스를 통하여 패킷이 유입된다. 이렇게 유입된 패킷은 Xilinx 사의 Virtex-II Pro FPGA 칩셋 내에 HDL 코드를 기반으로 합성된 네트워크 오용탐지 및 대응기능에 의해 패킷의 오용여부와 그에 적절한 대응이 지연없이 실시간상에서 집행된다. 대응기능에 의해 패킷의 패기 또는 전송여부가 결정되는데, 이런 결정사항은 기가비트 이더넷 인터페이스를 제어하는 MAC칩을 통하여 패킷을 패기 또는 전송하는 제어가 이루어진다[13].



(그림 2) 네트워크 침해대응카드의 구성

(그림 2)와 같은 구조를 갖는 보안카드를 장착한 시스템을 게이트웨이 위치에 설치하여 운영하는 경우, 유입되는 모든 트래픽은 Xilinx FPGA내에 합성되어 있는 분석기에 의하여 서브도메인 단위의 군집화된 집합흐름이 형성된다. 집합흐름에 의해 실시간으로 트래픽마다 네트워크의 대역폭 사용량(BPS : Bit Per Second)와 PPS(Packet Per Second)가 산정된다. 이렇게 산정된 정보를 기반으로 소프트웨어 블럭인 정보가공기에 전달되어 근원지 주소에 대한 분석, 목적지 주소에 의한 분석, 목적지 주소와 목적지 포트에 대한 분석을 통하여 3가지 형태의 분석결과를 보안정책 생성블럭인 정책관리기에 전달한다. 정책관리기는 이러한 분석결과를 토대로 대역폭 제어를 시행할 5-tuple 정보를 정책으로 생성하여 보안카드의 대응기내에 있는 대역폭 제어 모듈과 패킷 차단 모듈에 전달한다. 이렇게 전달된 정책은 <표 1>과 같이 144-bit 형태로서 TCAM(Ternary Contents Addressable Memory)에 저장된다. 이렇게 TCAM에 해당하는 정책이 기록된 후부터, 실시간으로 유입되는 모든 트래픽은 TCAM에 존재하는 규칙에 해당하는지 조회하고, TCAM에 존재하는 트래픽의 경우 해당 블록(대역폭 제어 모듈 또는 패킷 차단 모듈)에 의해 처리된다.

〈표 1〉 TCAM에 기록되는 패킷차단 및 대역폭 제어를 위한 보안 정책

필드명	비트열	설 명
reserved	[143 : 129]	Reserved(15bits)
Block	[128]	대역폭제어(0)/패킷차단(1)
Valid	[127]	Rule Valid
Tcam_Port	[126]	Direction[0/1]
ICMP_CODE	[125 : 118]	ICMP Code(8bits)
ICMP_TYPE	[117 : 110]	ICMP Type(8bits)
TCP_flags	[109 : 104]	TCP Flag 정보(6bits)
TCP_Dport	[103 : 88]	TCP/UDP Destination Port(16bits)
TCP_Sport	[87 : 72]	TCP/UDP Source Port(16 bits)
Protocol	[71 : 64]	IP Header(8 bits)
DST IP	[63 : 32]	Destination IP Address(32 bits)
SRC IP	[31 : 0]	Source IP Address(32 bits)

상기와 같은 보안카드에 구현되는 분석기와 대응기는 하드웨어 기술 언어(HDL : Hardware Description Language) 중 하나인 verilog를 통하여 구현되었고, 합성툴을 통하여 칩셋에 합성되어 운영된다. 소프트웨어 모듈인 정보가공기 와 정책관리기는 임베디드 소프트웨어에 설치되어 Main CPU에 설치된 각 블록의 서버블록과 통신을 통하여 데이터 및 정책을 상호 전달한다.

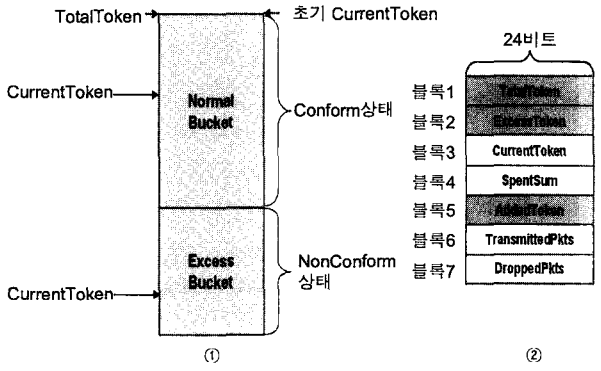
3.2 재구성 가능한 폴리싱 모듈과 기법

본 논문에서 지원하기 위한 대역폭 제어기능(Bandwidth Controller)는 Shaping과 Policing으로 구현 가능하다. Shaping과 Policing을 제공하기 위하여 Queue를 이용하는 Leaky Bucket 모델과, Queue를 이용하는 Token Bucket 모델이 존재한다. Leaky Bucket 모델은 동일한 패킷 사이즈를 갖는 ATM에서 Shaping 기법으로 사용되고 있고, 가변 적인 패킷 사이즈를 갖는 IP 기반 트래픽에 대하여서는 Token Bucket 모델을 이용한다. Diffserv망의 지원을 위하여 Linux Kernel

에서는 TC(Traffic Controller)내부에 TBF(Token Bucket Filter)를 적용 가능하도록 지원하고 있다[1, 3, 5-7, 10, 12, 14].

3.2.1 더블 토큰 버켓 메커니즘의 적용과 데이터 구조

보안카드내에 존재하는 대역폭 제어기능은 더블 토큰 버켓 메커니즘을 적용하여 구현하였다. 더블 토큰 버켓 메커니즘을 일반적인 스택의 데이터 구조로서 변경하면 (그림 3)①과 같은 구조를 갖는다. 관리자에 의해 대역폭을 제어하고자 하는 BPS 값을 입력받는다. 입력받는 값을 bit 값으로 환산하기 위하여 다음과 같은 절차를 통하여 <표 2>와 같이 (그림 3)②에 해당하는 데이터 구조에 맞는 변수 값을 계산하게 된다. 토큰 버켓과 관련된 데이터는 기가비트 트래픽을 지원하기 위하여 최대 1GigaBPS까지 값을 기록할 수 있도록 고려되었다. 1Gigabit 스피드를 지원하기 위해서는 23Bit의 필드가 필요하므로, 24bit의 데이터 크기를 부여하였다.



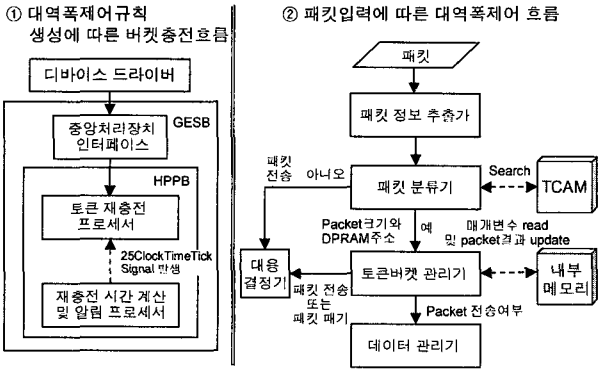
(그림 3) 더블 토큰 버켓을 이용한 대역폭 제어개념과 관련 변수와의 관계

대역폭 제어 규칙에 일치하는 패킷이 기가비트 이더넷 인터페이스를 통하여 입력되면, 패킷의 크기를 고려하여 CurrentToken의 값이 (그림 3)①에서 Conform 상태에 해당하는지, NonConform 상태에 해당하는지 판정하게 된다. 이러한 판정결과에 의해 패킷을 전송할 것인지 패기할 것인지 결정하게 된다.

〈표 2〉 토큰 버켓 메커니즘에 적용하기 위한 변수값의 계산

- Normal Bucket = 입력받은 bps 값/8
- Excess Bucket = Normal Bucket * 2
- ExcessToken = Normal Bucket
- TotalToken = Normal Bucket + Excess Bucket
- CurrentToken의 초기값 = TotalToken
- AddedToken = Normal Bucket * (1/4000)
- SpentToken = ExcessToken - CurrentToken (음수일 경우 값을 산정하지 않는다.)
- SpentSum = SpentSum + SpentToken
- SpentToken 초기값 = 0
- SpentSum 초기값 = 0

본 논문에서 적용한 더블 토큰 버켓을 적용하는 경우, 2개의 프로세스로 구분할 수 있다. (그림 4)①은 같이 버켓에 일정주기마다 일정량의 토큰을 충전하는 프로세스이고, (그림 4)②는 입력된 패킷의 크기만큼 버켓에서 제거하는 프로세스이다. 이러한 프로세스의 서로 종속적이지 않는 데이터 흐름을 통하여 관리자가 설정한 대역폭내에서 패킷을 전달하고, 대역폭을 초과하는 입력 트래픽에 대하여 패킷단위의 폐기가 집행된다.



(그림 4) 토큰버켓 충전 프로세스와 대역폭 제어 프로세스

(그림 4)의 두 프로세스는 동일한 데이터를 갖는 내부메모리에 대하여 접근하여 그 변수의 값을 접근시마다 변경한다. 이 변수의 정확한 값을 유지하기 위하여 소프트웨어의 쓰래드 개념처럼 하나의 프로세스에서 메모리 변수에 접근해서 사용하고 있는 동안, 타 프로세스가 메모리에 접근하지 못하도록 하는 메모리 접근을 제한하는 메모리 접근 인터페이스를 통하여 특정 시각에 하나의 프로세스만 접근하도록 제어한다.

대역폭 제어 모듈은 125㎒의 Clock을 입력으로 동작하게 된다. 1clock은 8ns에 해당한다. 250㎲마다 동일한 대역폭 제어 엔트리에 대해 충전을 수행하기 위해 1ms를 계산하는 서브 프로세스와 이 서브프로세스의 250번 수행을 알리는 재충전시간 알림 프로세스가 (그림 4)①에 포함되어 있다.

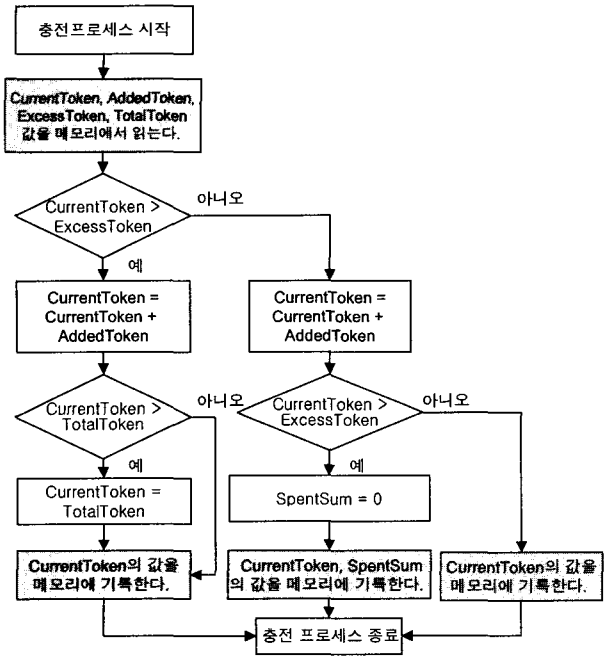
3.2.2 토큰 버켓 충전 프로세스

토큰 버켓 충전 프로세스는 동일한 엔트리에 대하여 250㎲마다 충전 작업이 적용되어야 한다.

(그림 5)는 (그림 4)①내에 존재하는 토큰 버켓 충전 프로세스의 흐름도이다. 각각의 흐름은 1Clock마다 진행된다. 세부적인 흐름은 다음과 같이 진행된다.

(그림 3)②와 같은 데이터 구조에서 CurrentToken, AddedToken, ExcessToken, TotalToken의 값을 메모리에서 읽어와 해당 레지스터에 기록한다. 이 기록된 값 중 CurrentToken과 ExcessToken과의 비교를 통하여 현재 Token이 Bucket의 Confirm 상태인지, NonConfirm 상태인지 파악한다. Confirm 상태의 경우에는 CurrentToken에 충전시기마다 더해지는 토큰수를 나타내는 AddedToken을 더한 결과가 TotalToken보다 크지 않도록 설정하는 프로세스이다. NonConfirm 상태의 경우에는 충전후 CurrentToken이 ExcessToken보다 크게 되는 경우 토큰 버켓 프로세스에 의해 생성되었던 SpentSum의 값을 0으로 초기화하여, 오랜 시간

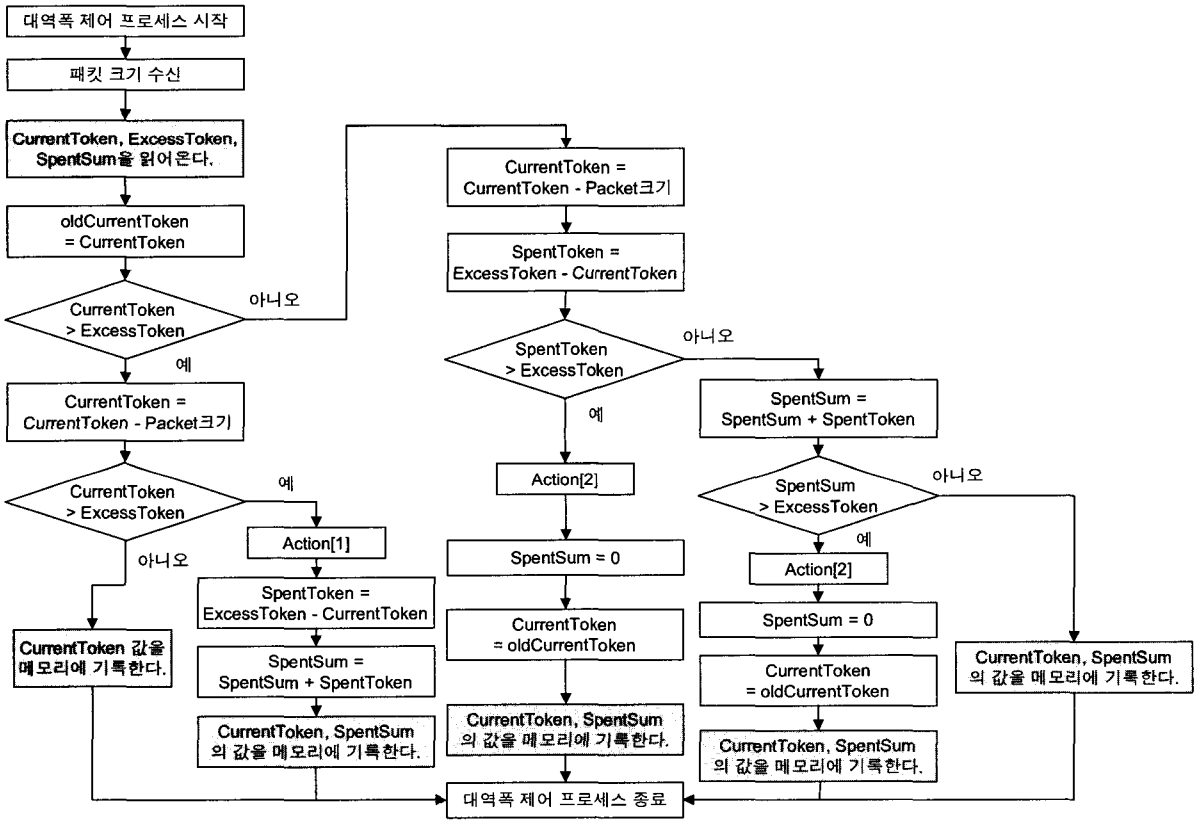
동안 지연으로 인한 패킷의 패기 현상을 방지한다. 이상의 절차를 통하여 각 엔트리에 존재하는 메모리 필드의 값을 업데이트 한다.



(그림 5) 토큰 버켓 충전 프로세스의 흐름도

3.2.3 패킷 입력에 따른 대역폭 제어 프로세스

패킷이 보안카드을 통하여 입력되면, 패킷 내의 헤더필드를 조회하여 해당 패킷의 내용이 TCAM 내에 존재하는 필드와 일치하는지 조회하다. 이렇게 조회한 결과 대역폭 제어규칙에 해당하는 패킷이 유입되었을 경우 (그림 6)과 같은 토큰 버켓 메커니즘을 적용한 대역폭 제어 프로세스가 구동된다. 패킷 입력에 따른 대역폭 제어 프로세스는 (그림 6)과 같이 운영된다. 패킷이 보안카드B의 기가비트 이더넷 인터페이스를 통하여 유입되면, 해당 Packet의 5-tuple(근원지 주소, 근원지 포트, 목적지 주소, 목적지 포트, 프로토콜) + α의 정보를 추출하여 TCAM의 모든 엔트리와 비교작업을 한다. 이 작업은 패킷 분류작업으로서 이를 통하여 대역폭 제어규칙에 해당하는지 알 수 있다. 만약, 유입된 패킷이 대역폭 제어규칙에 해당하는 경우, 토큰의 정보를 가지고 있는 내부 메모리의 주소 를 수신 받게 되고 이를 토대로 토큰 버켓 프로세서 를 운영한다. 토큰 버켓 프로세서는 일반적인 토큰 버켓 메커니즘과 동일하다. 즉, 토큰 버켓 프로세서는 유입되는 패킷의 패킷 크기를 구하여, 그 바이트(byte) 값 만큼 버켓에서 토큰을 제거하고 패킷을 전달한다. 만약, 현재 버켓에 존재하는 토큰 수보다 더 큰 패킷이 유입될 경우에는 현 패킷을 폐기하고, 버켓 내의 토큰 값은 패킷이 유입되기 전 값을 유지한다.



(그림 6) 토큰버켓 메커니즘을 통한 대역폭 제어 흐름도

이러한 패킷의 히스토리를 기록하기 위해서 레지스터인 SpentToken과 메모리 값인 SpentSum을 이용한다. SpentToken은 두 번째 버켓의 토큰을 사용하는 시기에 매 패킷마다 계산되고, 이 값은 SpentSum에 더해져서 패킷의 지속적인 대역폭 초과에 대한 히스토리로서 사용된다. 특히, 버켓에 남아있는 토큰 수보다 훨씬 큰 트래픽이 유입되게되어 SpentToken의 값이 ExcessToken의 값, 즉 2번째 버켓 값보다 크게 되면 패킷은 패기된다. 이외에 지속적으로 두 번째 버켓의 토큰을 소비하게 되는 경우 SpentSum이 두 번째 버켓 값인 ExcessToken보다 크게 되면, 패킷은 패기되고 버켓 내의 토큰 값은 이전값을 유지한다. 이전값을 유지하는 것은 패킷을 패기 하였으므로 대역폭의 계산치에서 제거하기 위함이다.

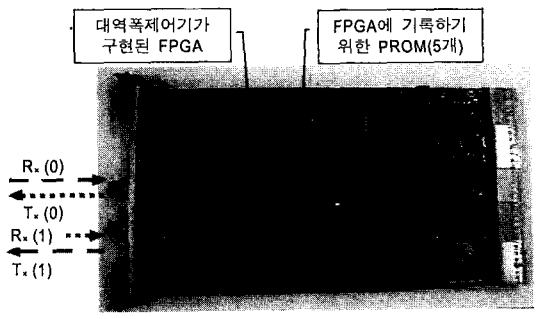
일반적으로 2개의 토큰 버켓을 이용하는 메커니즘은 분리된 2개의 버켓을 이용하여서 첫 번째 버켓에 남아있는 토큰이 없을 경우, 두 번째 버켓에서 토큰을 빌려와서 제거하는 방식을 취한다.

본 논문에서는 두개의 버켓을 별도로 구성하지 아니하고, 두개의 한개치(Threshold)를 갖도록 스택을 구성하였다. 이렇게 구성된 스택형 메모리는 충전 메커니즘에 의하여 정기적으로 토큰이 더해지고, 패킷크기만큼 패킷 유입시마다 토큰을 제거한다. 이러한 절차에 의하여 첫 번째 한 개치에 달하게 되면, 관리자에게 경고메시지등을 전송하고, 두 번째 한 개치에 달하게 되는 경우 패킷을 패기한다.

하드웨어로 토큰버켓 메커니즘을 구현하기 위하여 듀얼 포트 메모리를 이용하였다. 듀얼 포트 메모리를 이용하는 경우 서로 다른 프로세스가 동시에 메모리에 대하여 접근이 가능하다. 하지만 메모리에 같은 clock에 접근하는 경우, 충전 프로세스에 우선순위를 부여하고, 토큰버켓 메커니즘은 한 clock후에 접근하도록 메모리 인터페이스를 작성하였다.

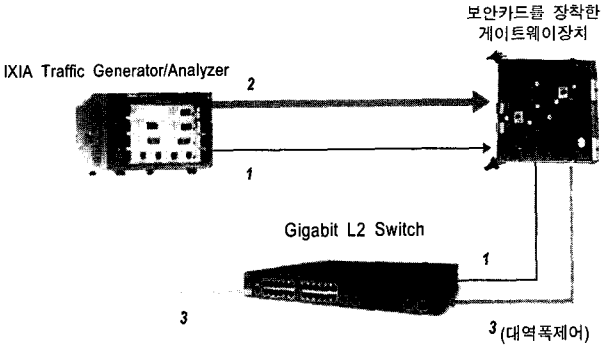
4. IXIA 장비를 이용한 기가비트 트래픽에 대한 실험

본 논문에서 구현한 보안카드는 (그림 7)과 같이 양방향을 지원하는 2개의 기가비트 이더넷 인터페이스을 지원한다. 따라서 보안카드에 대하여 첫 번째 기가비트 이더넷 인터페이스(Port A)의 입력 트래픽은 두 번째 기가비트 이더넷 인터페이스(Port B)의 출력 트래픽으로 전송되고, 두 번째 기가비트 이더넷 인터페이스(Port B)의 입력 트래픽은 첫 번째 기가비트 이더넷 인터페이스(Port A)의 출력 트래픽으로 전송된다.



(그림 7) 양방향 기가비트 속도를 지원하는 대역폭제어보안카드

트래픽 생성 및 분석 장치인 IXIA 장비를 이용하여 양방향 트래픽에 대하여 기가비트 이더넷을 이용하여 구현된 보안카드를 장착한 게이트웨이 시스템에 (그림 8)과 같이 테스트 하였다.



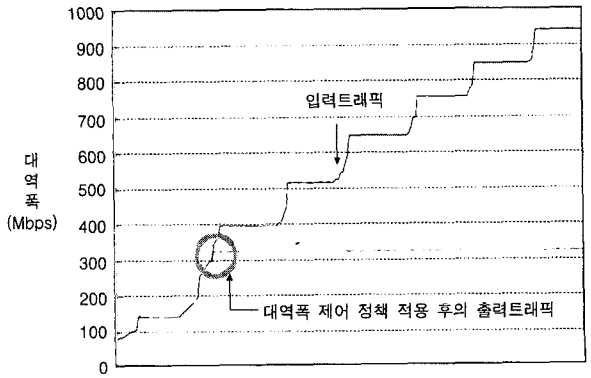
(그림 8) IXIA를 이용한 보안카드를 장착한 게이트웨이장치의 대역폭 제어 테스트환경

- A Port 송신 트래픽으로 40 Mbps의 정상트래픽 생성(1)
- B Port 송신 트래픽으로 8의 플로우를 이용하여 40 Mbps에서 1 Gbps까지 점진적으로 증가하는 비정상 과다트래픽 생성(2)
- 이상트래픽에 대한 대역폭 제어 규칙 적용 및 제어확인(3)
  - 대역폭 제어 정책을 생성하는 정책관리기에 의해서 보안카드의 A Port 수신 트래픽에 대하여 플로우별 40 Mbps의 규칙생성 및 적용
- 이상트래픽 제어를 통한 정상트래픽의 손실방지

상기와 같은 테스트에 대하여 <표 3>과 같은 결과를 나타내었다.

〈표 3〉 정상트래픽의 보호와 비정상트래픽의 대역폭제어

패킷크기	A Port 송신 트래픽	B Port 송신 트래픽	A Port 수신 트래픽	B Port 수신 트래픽
64Bytes	40Mbps	40Mbps ~ 1Gbps	20Mbps ~ 320Mbps	40Mbps
512Bytes	40Mbps	40Mbps ~ 1Gbps	20Mbps ~ 320Mbps	40Mbps
1024Bytes	40Mbps	40Mbps ~ 1Gbps	20Mbps ~ 320Mbps	40Mbps
1518bytes	40Mbps	40Mbps ~ 1Gbps	20Mbps ~ 320Mbps	40Mbps



(그림 9) 대역폭 제어에 의한 과다트래픽 제어(320Mbps)

상기와 같은 실험을 통하여 보안카드내의 FPGA에 구현된 대역폭 제어기능은 1Gbps의 속도를 갖는 양방향 트래픽에 대하여 패킷의 유실없이 송신/수신을 제공한다. 대역폭 제어를 하려는 패킷에 대하여 각 플로우별로 40Mbps로 대역폭을 제어하여 총 8개의 플로우에 대하여 (그림 8)과 같이 320Mbps로 제어하는 결과를 보여준다. 보호되어야 할 정상 트래픽에 대하여서도 40Mbps의 입력 트래픽을 패킷의 유실없이 전달함을 볼 수 있다.

5. 결 론

DDoS 공격을 위한 보다 강력한 공격 도구들이 개발되어 인터넷에서 광범위하게 유통되고 있고, 여전히 인터넷은 구조적인 보안의 문제점들을 가지고 있고, DDoS 공격은 그 규모나 심각성 측면에서 점차 문제로 부각되고 있는 추세다. 또한, 이를 방어하기 위해 존재하는 현재의 기술들은 최신의 DDoS 공격에 대한 완벽한 방어에 부족한 점이 많은 것이 사실이다. 기존의 정적인 차단솔루션에 접목된 수동적인 탐지기술만으로는 최신의 고도화된 공격을 효과적이며 즉각적으로 방어하기 어렵다. 본 논문에서 제안한 2개의 토큰버켓을 이용한 재구성 가능한 대역폭 제어 메커니즘은 현재 2Gbps Ethernet에 맞게 설계되고 구현되었고, 현재 PoS망에 적용 가능하도록 설계 및 구현이 진행중이다. 이를 ISP의 Core망인 Backbone에 위치시키기 위해 각 기가비트 보안카드 또는 Pos 보안카드를 실장한 10Gbps, 또는 20Gbps의 속도를 지원하는 게이트웨이 시스템의 구현과 메커니즘의 튜닝, 세부적인 기법의 연구가 추가 연구 및 개발되어야 할 것이다.

참 고 문 헌

[1] Paul Ferguson, Geoff Huston, *Quality of Service - Delivering QoS on the Internet and in Corporate Networks*, Wiley Computer Publishing, 1998.

[2] Pars Mutaf, "Defending against a Denial-of-Service Attack on TCP," In Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID'99), 1999.

[3] Werner Almesberger, "Linux Network Traffic Control - Implementation Overview," http://lcawww.epfl.ch/Publications/Almesberger/TR98_037.ps, EPFL ICA, 1999.

[4] I. Garber, "Denial-of-Service Attacks Rip the Internet," IEEE Computer, pp.12-17, April, 2000.

[5] David McDysan, *QoS & Traffic Management in IP & ATM Networks*, MacGraw-Hill, 2000.

[6] Geoff Huston, *Internet Performance Survival Guide - QoS Strategies for Multiservice Networks*, Wiley Computer Publishing, 2000.

[7] Werner Almesberger, "Linux Network Traffic Control - Implementation Overview 2001," EPFL ICA, 2001.

[8] David Moore, Geoffrey M, Voelker and Stefan Savage, "Interfering Internet Denail-of-Service Activity," In Proceedings of the 10th USENIX Security Symposium, pp.9-22, August, 2001.

[9] NIPC(National Infrastructure Protection Center), "find ddos," http://www.nipc.gov/warnings/advisories/2001/01-005.htm, 2001.

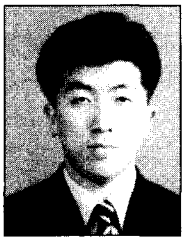
[10] H. Jonathan Chao, Xiaolei Guo, *Quality of Service Control in High-Speed Networks*, Wiley-Interscience Publicaiton, 2002.

[11] Packet Strom, "DDoS Attack Tools," http://packetstrom.widexs.nl/distributed/ indexdate.shtml, 2002.

[12] Netherlabs, Gregory Maxwell, Remco van Mook, Martijn van Oosterhout, Paul B Schroeder, Jasper Spaans, "Linun 2.4 Advanced Routing HOWTO," TLDP.

[13] Xilinx company, http://www.xilinx.com.

[14] 이대붕, 송황준, "토큰 버킷 통신망 모델의 매개변수 재협상 과정을 이용한 효과적인 동영상 트래픽 전송기법에 관한 연구", 영상통신, 한국통신학회 2002 하계종합학술대회.



박 상 길

e-mail : wideideal@etri.re.kr
1995년 조선대학교 컴퓨터 공학과 공학사
2000년 전남대학교 전산학과 이학석사
2000년~현재 한국전자통신연구원 연구원
보안게이트웨이연구팀
관심분야 : 네트워크 보안, 침해방지 시스템, 라우터 및 스위치 보안

오 진 태

e-mail : showme@etri.re.kr
1990년 경북대학교 전자공학과 공학사
1992년 경북대학교 전자공학과 석사
1992년~1998년 한국전자통신연구원 선임연구원
1998년~1999년 MinMax Tech. 연구원
1999년~2001년 Engedi Networks Inc. Director
2001년~2003년 Winnow Networks Inc. CTO 부사장
2003년~현재 한국전자통신연구원 선임연구원, 보안게이트웨이 연구팀 과책
관심분야 : 네트워크 보안, 비정상행위탐지기술, 고성능탐지엔진 기술

김 기 영

e-mail : kykim@etri.re.kr
1988년 전남대학교 전산통계학과 이학사
1993년 전남대학교 전산통계학과 석사
2001년 충북대학교 대학원 전자계산학과 이학박사
1988년~현재 한국전자통신연구원 책임연구원, 정보보호연구단 보안게이트웨이연구팀 팀장
관심분야 : 네트워크 보안, 고성능 네트워크 침입탐지 및 대응 기술 등