IPv4 방화벽에 호환성을 갖는 IPv6 터널링

이 정 남[†] · 장 주 욱[††]

요 약

본 논문은 방화벽에 독립적인 IPv6 터널링 기법의 연구에 관한 것이다. IPv4망의 인프라를 유지하면서 점진적으로 IPv6망을 확대해 나가고 있는 현재, IPv6망간의 연동을 위해서 터널링을 널리 사용하고 있으나 방화벽에 의해 IPv4로 캡슐화된 IPv6 패킷이 방화벽을 통과하지 못하는 문제점이 확인되었다. 즉, 방화벽 내부의 사용자들은 IPv6망의 접속에 제한을 받게 되며 방화벽 없이 IPv6망을 구축해야 한다. 본 논문에서는 방화벽에 의해 캡슐화된 패킷이 차단되는 것을 해결하기 위한 방법으로 Double-encapsulation 방식과 HTTP 터널링 기법을 응용한 방식을 제안하였으며 실험결과 패킷 차단없이 IPv6망간의 연동이 이루어짐을 확인하였다.

IPv6 over IPv4 tunneling compatible with IPv4 Firewalls

Jung-nam Lee[†] · Ju-wook Jang[††]

ABSTRACT

During the period of co-existence of IPv4 and IPv6, IPv6 over IPv4 tunneling technique is intended as a start-up transition mechanism. However, most of IPv4 firewalls do not support the IPv6 over IPv4 tunneling packet filtering. Finally, it is impossible that a user inside IPv4 firewall connects with an IPv6 host across IPv4 network. Without any additional hardware or changing the policy of IPv4 firewall, we solve this problem using proposed Double-encapsulation and applied-HTTP tunneling technique that are end-to-end solutions. This enables cheaper IPv6 migration solutions.

키워드 : 차세대 인터넷 프로토콜(IPv6), 터널링(Tunneling), 캡슐화(Encapsulation), 방화벽(Firewall)

1. 서 론

1996년 IETF가 IPv6를 차세대 인터넷 프로토콜 표준으로 제정한 이후 세계 각국은 차세대 인터넷으로의 진화를 준비하고 있으며 현재 연구개발 단계를 거쳐 실제 응용에 적용되는 시점에 이르렀다.

지금까지의 IPv4 인프라를 유지하면서 점진적으로 IPv6망을 확장해 나가고 있는 현재, IPv6망 사이의 연동을 위해서 터널링 기법이 사용되고 있다[1, 2]. 그러나 IPv4 터널링에 의해 캡슐화된 IPv6 패킷이 기존의 IPv4 방화벽을 통과하지 못하는 문제점이 나타났다. 캡슐화된 패킷은 IPv4 헤더의 프로토콜 필드 값이 41이 되는데 대부분의 방화벽이 이것을 차단하는 정책을 채택하고 있다. 물론 방화벽에서 이와 같이 캡슐화된 패킷을 통과시키는 방법이 있으나[3] IPv4망에 의해 격리된 전세계의 수많은 방화벽의 정책을 바꾸도록 하는 일은 비용과 시간 측면에서 어려운 문제이다.

최근 IPv6 터널링과 관련하여 제안된 ISATAP의 경우[4] 방화벽으로 인한 터널링 패킷 차단문제를 6to4 라우터를 이용하여 해결하고 있어 실현이 어려운 해결책이라고 생각된다.

† 준 회 원 : 서강대학교 대학원 전자공학과
†† 정 회 원 : 서강대학교 전자공학과 교수
논문접수 : 2002년 11월 30일, 심사완료 : 2003년 5월 6일

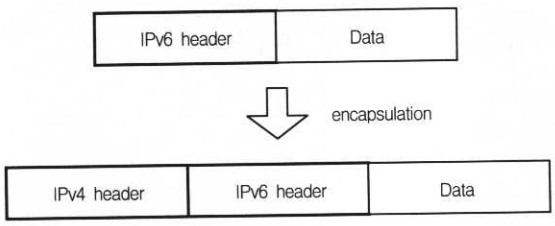
본 논문에서는 이러한 문제에 대한 해법으로 Double-encapsulation방식과 HTTP 터널링을 응용한 방식을 제안하였고 이를 직접 KAME[5]의 IPv6 FreeBSD커널에 구현하여 그 효용성을 입증하였다. 본 논문의 의의는 IPv4 방화벽의 설정을 변경할 필요없이 IPv6 패킷이 IPv4 방화벽을 통과하게 하는 진정한 의미의 End-to-End 솔루션이어서 효용성이 높을 것으로 예상된다는 점이다.

2장에서는 KAME의 IPv6 터널링 패킷이 IPv4 방화벽에 의해 차단되는 문제점과 최근 IETF에 제안된 ISATAP를 이용한 터널링의 문제점을 보이고, 3장에서는 이러한 문제점을 해결할 수 있는 새로운 터널링 기법을 제안하고, 4장에서는 제안된 터널링 기법의 구현에 따른 문제점 및 해결방안을 서술하였다. 5장에서는 실험을 통해 기존의 터널링과 제안된 터널링 방식을 비교하였으며, 6장에서는 결론 및 향후 연구 과제에 대해 기술하였다.

2. 기존 IPv6 터널링과 방화벽의 비호환성

2.1 기존 IPv6 터널링

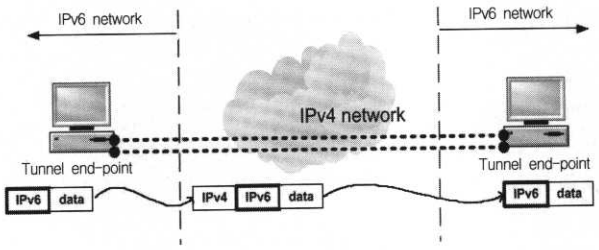
IPv4망을 통해 IPv6망의 연동을 위한 기존의 터널링 방식은 (그림 1)과 같이 IPv6 패킷에 IPv4 헤더를 덧붙이는 형태를 갖는다[2].



(그림 1) IPv4 패킷으로 캡슐화

이와 같이 캡슐화된 패킷을 통해 IPv6망 사이의 통신이 가능하며 패킷이 캡슐화되는 지점이 터널의 종단이 된다.

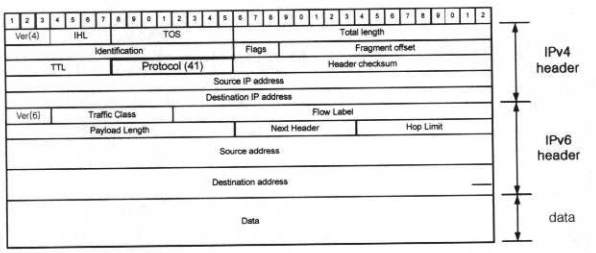
(그림 2)는 터널의 양 종단과 캡슐화된 패킷의 흐름을 나타낸 것이다.



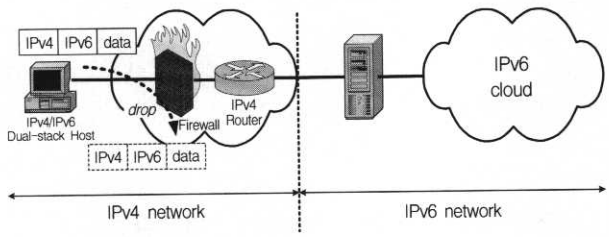
(그림 2) 기존의 터널링

2.2 IPv4 방화벽과의 비호환성 문제

방화벽에서는 기본적으로 IP 헤더의 필드 값들을 필터링하여 패킷의 차단여부를 가린다. 터널링 패킷의 경우 (그림 3)의 IPv4 헤더의 프로토콜 필드의 값을 통해 다음에 나오는 헤더가 무엇인지 판별하게 된다[6].



(그림 3) 터널링 패킷 IP 헤더



(그림 4) 방화벽에 의한 터널링 패킷 차단

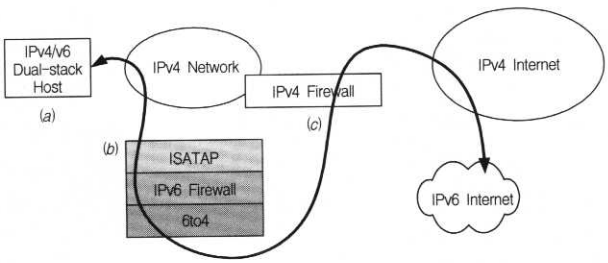
그러나 전세계의 수많은 방화벽들은 아직 IPv6가 지원되지 않는 상황에서 IPv4 다음에 나오는 헤더가 IPv6인 패킷인 경우 제대로 인식하지 못하고 패킷을 차단시키는 것이다[4]. 즉, (그림 4) 및 (그림 15)와 같이 방화벽 내부에 있는 호스트의 IPv6-over-IPv4 터널링 패킷은 방화벽에 의해 차단되어 IPv6망으로의 접속이 불가능하다(FreeBSD 기반의 KAME snap을 이용한 IPv6 터널링 패킷이 IPv4 방화벽에서 100% 유실됨을 (그림 15)에서 확인할 수 있다. 자세한 내용은 5.2절 참조).

2.3 ISATAP와 방화벽

ISATAP는 IPv4망의 사용자가 터널링을 통해 IPv6망에 접속을 원하는 경우 자동적으로 주소할당을 해주는 프로토콜이다[4]. 여기서는 방화벽의 문제점을 고려하여 IPv6 방화벽과 6to4 라우터를 도입하였다.

(그림 5)에서와 같이 Dual-stack Host(*a*)는 IPv6망에 접속하기 위해 ISATAP 라우터(*b*)를 통해 IPv6 주소를 자동으로 할당받게 된다[4]. 이후 IPv6망으로 전송되는 패킷은 IPv6 방화벽을 거쳐 6to4 라우터를 통해 캡슐화 된 후 다시 IPv4 방화벽(*c*)을 지나게 된다. 이때 IPv4 방화벽은 프로토콜 필드 41인 패킷을 통과시킬 수 있는 정책을 수용해야 하는 것이다.

결국 ISATAP를 이용한 방법 또한 캡슐화된 패킷을 위해 IPv4 방화벽에서 프로토콜 필드 값이 41인 패킷을 통과시켜주도록 세팅을 변경해야 한다. 이는 IPv6망으로의 접속을 위해 라우터에 ISATAP 기능을 추가적으로 삽입해야 하며 6to4 라우터의 도입이 필요하다는 것을 의미한다.



(그림 5) ISATAP를 이용한 IPv6망으로의 접속

그러나 본 논문에서는 ISATAP 및 6to4 라우터의 도입없이 기존 프로토콜의 응용을 통해 IPv6망과의 연동을 가능하게 하는 터널링 기법들을 제안하였다.

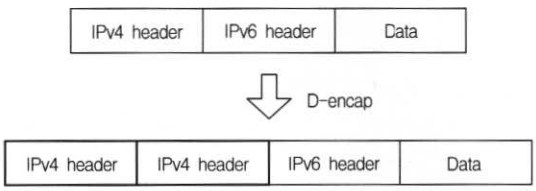
3. 새로운 IPv6 터널링 기법의 제안

방화벽에 의해 터널링 패킷이 차단되는 문제의 근본적인 해결방안은 터널링 패킷을 방화벽이 인식할 수 있도록 방화벽을 업그레이드 하거나 IPv4와 IPv6 모두 지원되는 방화벽을 새로 도입하는 것이다. 그러나 이와 같은 장비교체는 비용과 시간의 문제가 발생한다. 물론 점진적으로 방화벽을 교체한다고 하더라도 IPv6망의 확산 속도를 따라가지 못하여 결국 IPv6망의 사용자가 IPv6망에 접속하는데는 제한을 받게 된다.

이에 본 논문에서는 방화벽에 의해 IPv6 터널링 패킷이 차단되는 문제를 해결하기 Double-encapsulation(이하 D-encap)방식 및 HTTP 터널링을 응용한 방식을 제안하였다.

3.1 제안된 D-encap 방식

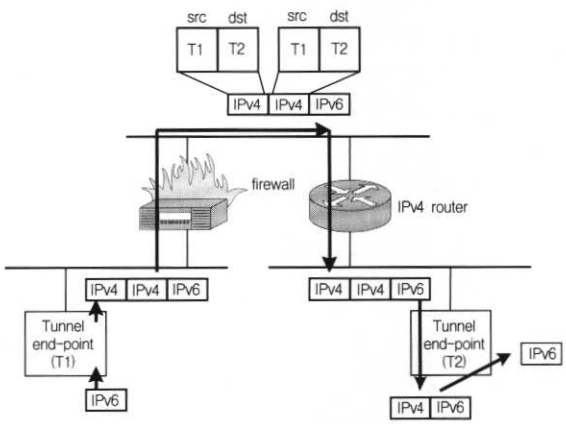
본 논문에서 새롭게 제안하는 D-encap 방식은 IP 헤더의 프로토콜 필드 값을 통해 패킷 필터링을 하는 방화벽에 따른 대처방안이다. IPv4로 캡슐화된 패킷을 한번 더 IPv4로 캡슐화하여 프로토콜 필드 값이 4(IPinIP)가 되도록 하여 방화벽을 통과하도록 한다. 즉, D-encap 방식에 의해 캡슐화된 패킷은 (그림 6)과 같은 헤더를 갖는다.



(그림 6) D-encap된 패킷

위와 같은 패킷 생성시 외부 IPv4 헤더의 송신측 주소는 터널의 생성지이며 수신측 주소는 터널의 종단이 된다. D-encap 되는 내부 IPv4 헤더의 또한 송신측 주소는 터널의 생성지가 되며 수신측 주소는 터널의 종단이 되어 패킷이 터널 종단에서 완전히 벗겨지도록 한다.

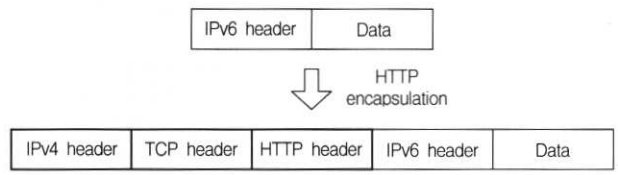
(그림 7)은 D-encap 방식으로 캡슐화된 패킷의 생성부터 소멸까지를 보여준다. T1과 T2 이하는 IPv6망이며 T1에서 T2까지는 IPv4망에 해당한다. IPv6망에서 생성된 IPv6 패킷은 T1에서 D-encap과정을 거쳐 IPv4망을 지나갈 수 있는 패킷이 된다. 이와같은 패킷은 기존의 방화벽 및 IPv4망을 지나 T2에서 IPv4 헤더가 벗겨져 T2 이하의 IPv6망에 도달하게 된다.



(그림 7) D-encap의 패킷의 생성과 소멸

3.2 HTTP 터널링 방식의 응용

HTTP 터널링 방식은 IPv6 패킷을 HTTP 패킷의 형태로 캡슐화하는 것이다[7]. 즉, 패킷을 캡슐화하는 터널의 종단에서 (그림 8)과 같이 IPv6 헤더 앞에 HTTP 형태의 헤더를 붙여주고 캡슐을 벗겨내는 종단에서는 HTTP 형태의 헤더를 떼어내는 방식이다. 결국, HTTP 터널링 방식의 패킷의 생성과 소멸은 D-encap 방식의 패킷과 같은 과정을 통해 이루어지게 되나 캡슐의 헤더만 다른 것이다.



(그림 8) HTTP로 캡슐화된 패킷

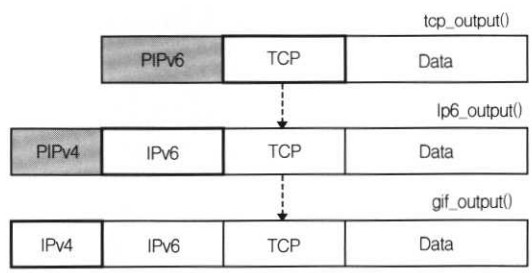
이와 같은 방법은 대부분의 방화벽이 웹 접속을 위한 80 포트를 차단하지 않는다는 점을 이용한 것으로 IP 헤더와 전송계층의 포트 및 그 이하 프로토콜까지 필터링하는 형태의 방화벽을 완벽하게 통과할 수 있도록 지원한다.

4. 구현상의 문제점 및 해결방안

본 논문에서는 방화벽이 존재하는 IPv4망과 외부의 IPv6망의 연동을 위해 IPv6 KAME[5]에서 구현한 기존의 터널링 방식을 바탕으로 D-encap 방식과 HTTP 터널링을 응용한 방식을 구현하였다.

4.1 기존 방식

IPv6 KAME에서 구현된 FreeBSD 커널에서는 패킷 생성 과정에서 gif_output()를 통해 IPv4 헤더를 덧씌워서 IPv6 패킷을 캡슐화 한다[5]. 이때 덧씌워지는 IPv4 헤더를 위한 버퍼 할당은 이미 전송계층의 헤더가 생성되는 과정에서 이루어진다. 즉, (그림 9)와 같이 Pseudo IP 헤더 영역은 실제로 IP 헤더가 생성되기 이전에 메모리를 할당받게 되며 이때부터 IP 헤더의 각 필드 값이 채워지기 시작한다.



(그림 9) 기존의 터널링

4.2 구현상의 문제점

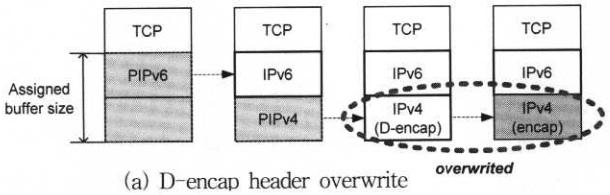
본 논문에서 제안하는 D-encap 방식과 HTTP 터널링을 응용한 방식 KAME에서 구현된 FreeBSD 커널을 기반으로 구현되었다. (그림 10)은 커널에서 터널링 패킷 생성을 위한

과정의 일부를 나타낸 것으로 IP 헤더를 위한 버퍼는 전송계층에서 미리 할당되는 것을 알 수 있다. 즉, 캡슐화를 위한 IPv4 헤더 뿐만이 아니라 제안한 두 가지 방식의 헤더를 위한 버퍼는 각 헤더 값을 확정하는 과정 전에 할당해야 한다.

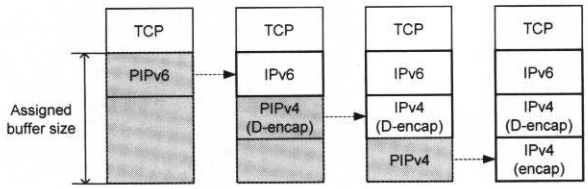
```
int
tcp_output(tp)
        register struct tcpcd *tp ;
{
...
        struct ip6_hdr *ip6 = NULL ;
        register struct mbuf *m ;
        int isipv6
...
        if (isipv6)
                hdrlen = sizeof (struct ip6_hdr) + sizeof (struct tcphdr) ;
        else
...
        hdrlen = sizeof (struct tcpiphdr) ;
...
        if (isipv6) {
                ip6 = mtod(m, struct ip6_hdr *) ;
                th = (struct tcphdr *)(ip6 + 1) ;
                bcopy((caddr_t)tp -> t_template -> tt_ipgen, (caddr_t)ip6,
                        sizeof(struct ip6_hdr)) ;
                bcopy((caddr_t)&tp -> t_template -> tt_t, (caddr_t)th,
                        sizeof(struct tcphdr)) ;
        }
...
}
```


(그림 10) Pseudo IP 헤더 할당

D-encap 방식은 기존의 방식에 비해 IPv4 헤더를 하나 더 덧붙여주는 방식인데 이를 구현하기 Pseudo IP 영역을 고려하지 않고 간단히 헤더를 덧씌우려 할 경우 (그림 11)(a)와 같이 D-encap 헤더에 기존 IPv4 헤더에 겹쳐쓰게 되므로 (그림11)(b)와 같이 IPv6 헤더 값을 채워 넣는 과정에서 미리 D-encap을 위한 메모리를 확보해야 한다.



(a) D-encap header overwrite



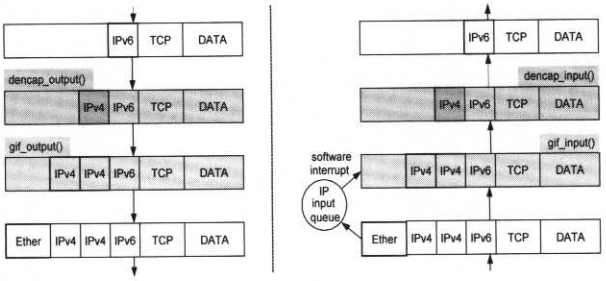
(b) Buffer assignment for D-encap

(그림 11) Pseudo IP 헤더용 버퍼 할당

마찬가지로 HTTP 터널링 방식 또한 추가되는 HTTP 헤더를 위한 버퍼할당을 IPv6 헤더 값을 채워 넣는 과정에서 메모리를 미리 확보해야 한다.

4.3 D-encap 방식의 구현

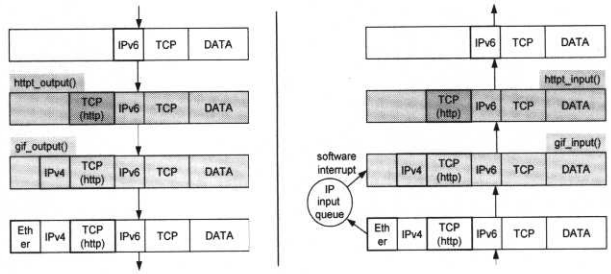
위에서 언급한 대로 Pseudo IP를 위한 버퍼 할당 문제를 고려하여 D-encap 방식을 통한 IPv6 패킷의 캡슐화 과정은 (그림 12)와 같다. 이와 같이 생성된 패킷은 IPinIP 패킷으로 실험에서 사용한 (그림 14)의 네트워크에서 방화벽을 통과하여 IPv6망과 연동이 된다. 이 경우 전체 패킷의 크기는 기존의 터널링 방식에 비해 20Bytes가 더 증가한다.



(그림 12) D-encap 과정 삽입

4.4 HTTP를 이용한 IPv6 터널링의 구현

방화벽의 정책이 좀 더 복잡하여 IP 헤더 내부에 포함된 내용까지 필터링하게 되는 경우 IPv4 헤더 내부의 TCP 헤더까지 HTTP 형태로 생성하여 방화벽을 통과하여 IPv6망과의 연동을 가능하게 한다. (그림 13)은 D-encap 과정 대신 HTTP 터널링 과정을 삽입하여 구현한 HTTP 터널링 패킷의 캡슐화 과정이다.

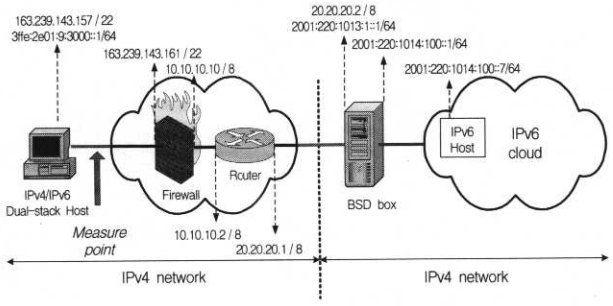


(그림 13) HTTP 터널링 과정 삽입

5. 실험 환경 및 결과

5.1 실험 환경

IPv4망 내부의 호스트가 IPv6망에 접속하기 위한 방법으로 터널링을 사용하였으며 방화벽에 의해 패킷이 차단되는 것을 D-encap 방식과 HTTP 터널링 방식을 사용하여 기존의 터널링 방식과 비교 실험하였다. (그림 14)는 실험에 사용된 네트워크 환경을 나타낸다. IPv4/IPv6 Dual 스택의 터널 양 종단은 FreeBSD4.3과 KAME를 이용하여 <표 1>과 같이 구성하였다.



(그림 14) 실험에 사용된 네트워크

〈표 1〉 실험 환경

구 분	사 양	운영체제	프로토콜스택
Dual-stack Host	Pentium-Ⅲ 800Mhz	FreeBSD 4.3 kame-freebsd43-snap	IPv4/IPv6 dual stack
Firewall	Pentium-Ⅱ 233Mhz	Linux 2.4.2 (ipchains 사용)	IPv4 only
Router	CISCO 2600	c2600-is-mz.122-2.T.bin	IPv4 only
BSD box	Pentium-Ⅱ 233Mhz	FreeBSD4.3 kame-freebsd43-snap	IPv4/IPv6 dual stack
IPv6 Server	Pentium-Ⅲ 800Mhz	FreeBSD4.3 kame-freebsd43-snap	IPv6 only

방화벽의 정책은 <표 2>와 같이 기본적인 패킷 필터링 방식을 통해 각 실험별로 Policy1~4를 적용하였다. Policy1은 방화벽이 적용되지 않는 상태, 즉 모든 패킷을 통과시키는 정책이며 Policy2는 프로토콜 41 필터링으로 IPv4 헤더로 캡슐화된 IPv6 패킷을 차단시키는 정책이다. Policy3의 스크립트는 방화벽을 초기화 한 후 프로토콜 41 필터링을 적용하는 정책이며 Policy4는 프로토콜 41 필터링과 전송계층 필터링을 통해 HTTP 포트만을 통한 패킷만 통과시키는 정책이다.

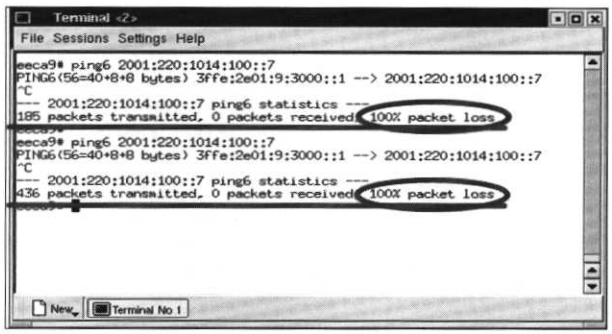
〈표 2〉 실험에 사용된 방화벽 정책

방화벽 정책	스크립트 내용
Policy1	ipchains -F ipchains -P input ACCEPT ipchains -P output ACCEPT ipchains -P forward ACCEPT
Policy2	ipchains -A input -i eth0 -p 41 -j DENY ipchains -A output -i eth0 -p 41 -j DENY ipchains -A forward -i eth0 -p 41 -j DENY
Policy3	ipchains -F ipchains -P input ACCEPT ipchains -P output ACCEPT ipchains -P forward ACCEPT ipchains -A input -i eth0 -p 41 -j DENY ipchains -A output -i eth0 -p 41 -j DENY ipchains -A forward -i eth0 -p 41 -j DENY
Policy4	ipchains -P input DENY ipchains -P output DENY ipchains -P forward DENY ipchains -A input -i eth0 -p \ tcp --destination-port 80 -j ACCEPT

IPv6망의 접속 여부는 캡슐화된 IPv6 패킷이 IPv4 방화벽과 IPv4망을 통해 IPv6 서버에 송신 후 이에 대한 응답(reply) 패킷의 도착여부로 IPv4 방화벽과의 호환성을 확인하였다.

5.2 기존의 IPv6 터널링 방식

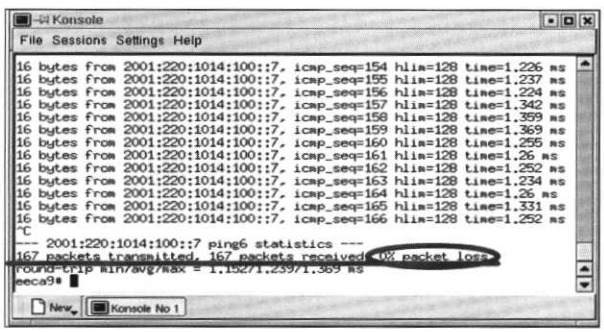
(그림 14)의 실험 환경에서 각 IPv6 호스트는 FreeBSD와 KAME를 통해 구현하였으며 방화벽 정책은 <표 2>의 Policy2을 적용하여 패킷의 IP 헤더만을 필터링하도록 설정하였다. 이와 같은 환경에서 (그림 15)과 같이 ping을 통해 IPv4로 캡슐화된 IPv6 패킷의 전송 여부를 확인한 결과 전송되는 모든 패킷은 방화벽에 차단되었으며 결국 아무런 응답이 없음을 확인할 수 있다.



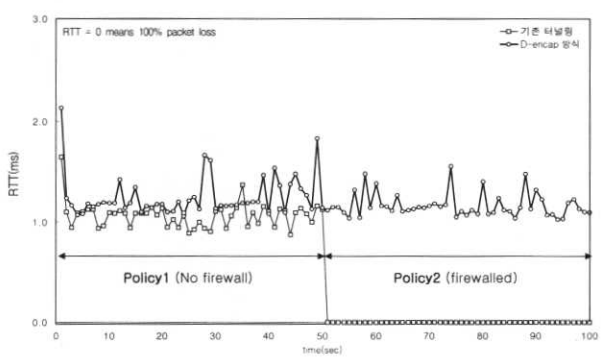
(그림 15) 기존의 터널링 방식의 실험결과

5.3 제안된 D-encap 방식의 실험 결과

<표 2>의 방화벽 정책중 Policy2를 적용한 위의 실험과 같은 환경에서 터널 종단인 Dual-stack 호스트와 BSD-box에 D-encap 과정을 삽입한 후 같은 실험을 하였다. 그 결과 (그림 16)과 같이 ping을 통해 IPv6 패킷의 송수신을 확인할 수 있다. 즉, Dual-stack 호스트에서 생성한 모든 패킷이 방화벽 및 IPv4망을 통해 IPv6망의 서버와 통신되는 것을 알 수 있다.



(그림 16) D-encap 방식의 실험 결과

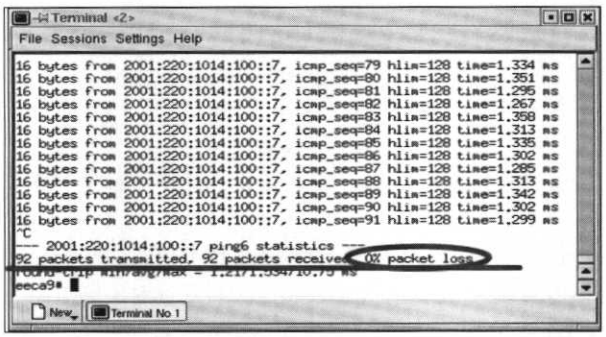


(그림 17) 기존 터널링와 D-encap 방식 비교

(그림 17)은 기존의 터널링과 D-encap 방식의 패킷 송수신 여부를 비교한 것이다. 방화벽이 없는 상태(Policy1 적용)에서는 두 방식 모두 패킷이 정상적으로 패킷이 송·수신되는 것을 확인할 수 있다. 그러나 방화벽을 동작시키는 순간(Policy 2 적용) 기존 터널링 방식은 방화벽에 차단되나 D-encap 방식은 방화벽에 영향 없이 계속해서 IPv6망에 연결되어 있는 것을 확인할 수 있다.

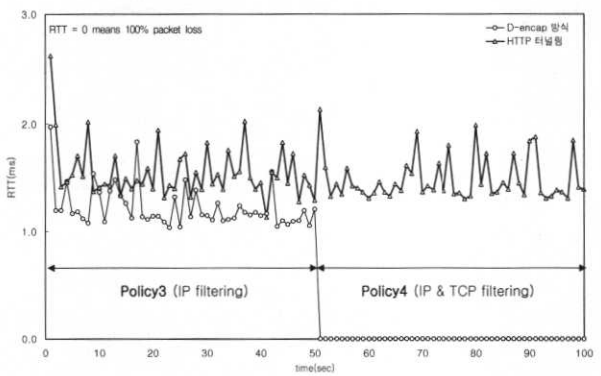
5.4 제안된 HTTP를 이용한 IPv6 터널링의 실험결과

위 실험과 같은 환경에서 방화벽은 패킷의 IP 헤더와 TCP 헤더를 필터링 하도록 Policy 3과 Policy 4를 적용하였으며 D-encap 과정을 제거하고 HTTP 터널링 과정을 삽입하였다. 이와 같은 환경에서 (그림 18)과 같이 ping을 통해 패킷의 송수신 여부를 확인한 결과 모든 패킷이 방화벽을 통과함을 확인 할 수 있었다.



(그림 18) HTTP 터널링 방식의 실험 결과

(그림 19)는 본 논문에서 제안한 D-encap 방식과 HTTP 터널링 방식을 비교한 것이다. D-encap 방식과 HTTP 방식 모두 IP 필터링을 하는 방화벽(Policy3 적용)을 통해 패킷 송수신이 정상적으로 이루어지고 있으나 IP와 포트를 필터링하는 방화벽(Policy4 적용)에서는 HTTP 터널링 방식만이 IPv6망에 연결되는 것을 확인할 수 있다.



(그림 19) D-encap 방식과 HTTP 터널링 방식 비교

6. 결론 및 향후 연구 과제

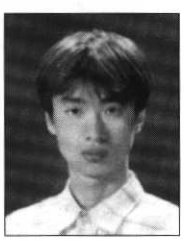
본 논문에서는 IPv6망에 접속하기 위한 기술로 현재 널리 사용되고 있는 터널링 방식이 방화벽내의 사용자들에게는 제한되는 문제점을 제시하고 이를 해결하기 위한 방안으로 D-encap 방식과 HTTP 터널링을 응용한 방식을 제안하였다.

D-encap 방식을 통해 방화벽 환경에서도 IPv6망과의 연동이 가능함을 확인하였을 뿐만 아니라 HTTP 터널링을 통해 IP 및 패킷의 내부까지 필터링하는 방화벽 환경에서도 IPv6망에 접속할 수 있는 방법을 제시하였다. 즉, D-encap 방식과 HTTP 터널링 방식은 현재 사용되는 방화벽의 패킷 필터링 정책 변경이나 추가적인 업그레이드 없이 IPv6망으로의 접속을 가능하게 한다. 또한, IPv6망으로의 전환에 필요한 추가적인 장비 도입없이 터널 종단의 캡슐화 과정만 삽입함으로써 방화벽 환경의 사용자들도 IPv6망에 접속할 수 있게 한다.

그러나, 본 논문에서 제안된 터널링 기법은 추가적인 헤더의 삽입으로 인해 기존의 터널링 기법에 비해 헤더의 크기가 커지게 된다. 그러므로 기존의 터널링 기법과 비슷한 크기의 헤더를 갖으면서도 IPv4 방화벽에 호환성을 갖는 터널링 기법의 연구와 헤더가 더 커짐으로 인해 발생하는 응답지연의 분석에 대한 연구가 필요하겠다.

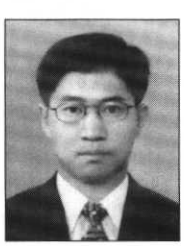
참 고 문 헌

[1] A. Durand, P. Fasano, I. Guardini, D. Lento, "IPv6 Tunnel Broker," RFC3053, January, 2001.
[2] B. Carpenter, K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC3056, February, 2001.
[3] http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/ipv6_sol/6bmctnl.pdf.
[4] F. Templin, T. Gleeson, M. Talwar, D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," draft-ietf-ngtrans-isatap, April, 2002.
[5] KAME project, "www.kame.net."
[6] Robert L. Ziegler, "Linux Firewalls," New Riders, 1999.
[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol-HTTP/1.1," RFC2068, January, 1997.
[8] W. Simpson, "IP in IP Tunneling," RFC1853, October, 1995.
[9] R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," RFC2893, August, 2000.
[10] A. Conta, S. Deering, "Generic Packet Tunneling in IPv6 Specification," draft-ietf-ipv6-tunnel, July, 2002.



이 정 남

e-mail : jungnamy@sogang.ac.kr
2001년 서강대학교 전자공학과 공학사
2001년~현재 서강대학교 전자공학과 석사과정
관심분야 : 인터넷 프로토콜, IPv6



장 주 욱

e-mail : jjang@mail.sogang.ac.kr
1983년 서울대학교 전자공학과 학사
1985년 한국과학기술원 전기 및 전자공학과 (공학석사)
1993년 University of Southern California 컴퓨터공학과(공학박사)
1995년~현재 서강대학교 전자공학과 부교수
관심분야 : 인터넷 프로토콜, 병렬처리