방화벽에 호환성을 갖는 IPv6 터널링 기법 및 구현

이정남, 장주욱
서강대학교 전자공학과
jungnamy@eeca1.sogang.ac.kr, jjang@mail.sogang.ac.kr

IPv6 tunneling compatible with Firewall

Jung-nam Lee, Ju-wook Jang
Dept. of Electronic Engineering, Sogang University

요 약

본 논문은 방화벽에 독립적인 IPv6 터널링 기법의 연구에 관한 것이다. 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 encapsulation된 패킷이 차단되는 것을 해결하기 위한 방법으로 Double-encapsulation 기법과 HTTP 터널링 기법을 제안하였으며 실험결과 패킷 차단없이 IPv6망간의 연동이 이루어짐을 확인하였다.

1. 서 론

인터넷이 급속히 보급됨에 따라 32비트의 IPv4 주소가 곧 고갈될 것으로 예상되어 IPv6라는 새로운 128비트 주소체계가 필요한 시점이 되었다[5]. 1996년 IETF가 IPv6를 차세대인터넷 프로토콜 표준으로 제정한 이후 세계 각국은 차세대인터넷으로의 진화를 준비하고 있으며 현재 연구개발 단계를 거쳐 실제 응용에 적용되는 시점에 이르렀다.

지금까지의 IPv4 인프라를 유지하면서 점진적으로 IPv6망을 확대해 나가고 있는 현재, IPv6망 사이의 연동을 위해서 6to4 터널링 기법이 사용되고 있다[3][4]. 그러나 IPv4 터널링에 의해 캡슐화된 IPv6 패킷이 기존의 IPv4 방화벽을 통과하지 못하는 문제점이 나타났다. 캡슐화된 패킷은 IPv4 헤더의 프로토콜 필드 값이 0x29가 되는데 대부분의 방화벽이 이 값을 인식하지 못하여 차단시키는 것이다. 물론 방화벽에서 이 값을 인식하고 encapsulation된 패킷을 통과하도록 하는 것이 근본적인 대안이나 전세계의 수많은 방화벽의 세팅을 바꾸어 주는 일은 비용과 시간 측면에서 생각할 때 쉽지 않은 문제다. 최근 이와 관련하여 ISATAP와 같은 프로토콜이 제안되었으나 이는 방화벽 외부에서 동작하는 것으로 방화벽에 의한 터널링 패킷 차단을 해결할 수 있는 대안이 되지 못하고 있다.[13].

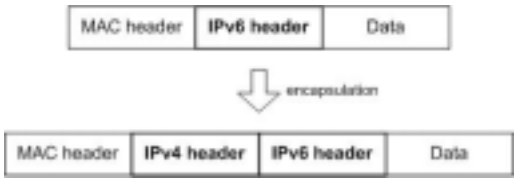
결국 IPv6를 연구하기 위해서는 방화벽 외부에 IPv6과 연동될 수 있도록 망을 구성해야하며 방화벽 내부의 호스트는 IPv6 망과의 연동에 제한을 받게된다. 즉, IPv6용 방화벽이 완전하게 구현되기 전까지 수많은 IPv6 사용자들은 안전하지 않은 방화벽 외부에서 망을 구축해야하며 방화벽 내부에서는 외부의 IPv6망과의 연동이 불가능하게 된다.

본 논문에서는 이러한 문제에 대한 해법으로 Double-encapsulation 방식과 HTTP 터널링을 응용한 방식을 제안하였으며 실험을 통해 기존의 6to4 방식의 터널링 기술과 비교하였다.

2. 기존 IPv6 터널링과 방화벽의 비호환성

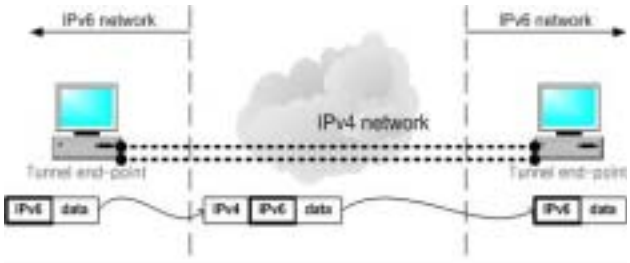
2.1 기존 IPv6 터널링

IPv4망을 통해 IPv6망의 연동을 위한 기존의 터널링 방식은 <그림1>과 같이 IPv6 패킷에 IPv4 헤더만을 덧붙이는 형태를 갖는다[4].



<그림1> IPv4 패킷으로 캡슐화

이와 같이 캡슐화된 패킷을 통해 IPv6망 사이의 통신이 가능하며 패킷이 캡슐화되는 지점이 터널의 종단이된다. <그림2>는 터널의 양 종단과 캡슐화된 패킷의 흐름을 나타낸 것이다.



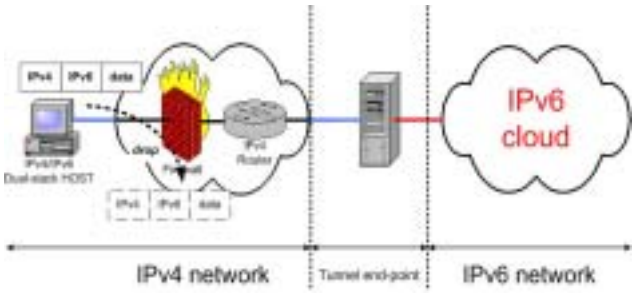
<그림2> 6to4 터널링

2.2 방화벽에 따른 문제점

현재 전세계 대부분의 ISP 및 각 기관에서는 외부로부터의 불법적인 침입의 차단 및 망의 관리를 위해서 방화벽을 도입하였다. 방화벽의 도입으로 불필요한 접속의 차단으로 망의 안정성 확보 및 대역폭 낭비를 줄이는 혁신적인 성과를 이루어냈다[11]. 그러나 방화벽으로 인해 IPv4망과 IPv6망의 연동 및 차세대 프로토콜의 실험이 차단되는 현상이 발생한 것이다. 이는 IPv6 패킷이 IPv4로 캡슐화될 경우 IPv4 헤더의 프로토콜 필드 값 41을 방화벽이 인식하지 못하고 패킷을 차단시켜 생겨난 현상이다. <그림3>는 방화벽에 의해 차단되는 패킷의 헤더를 보여준다. 본 논문에서는 IPv6의 대표적인 사례인 KAME 프로젝트의 결과를 이용하여 방화벽과의 호환성 실험결과 이와 같이 IPv6 패킷이 차단되는 것을 5장의 실험결과에서 확인하였다.

<그림3> 캡슐화된 패킷의 IP 헤더

방화벽에서는 IPv4 헤더의 다음에 나오는 헤더가 무엇인지 판별하기 위해 패킷 필터링을 통해 프로토콜 필드의 값을 뽑아내어 패킷의 차단 여부를 가린다[11]. 그러나 대부분의 방화벽에서는 IPv4 다음에 나오는 헤더가 IPv6인 패킷인 경우 이를 인식못하고 차단한다. 즉, <그림4>와 같이 방화벽내부에 있는 호스트의 IPv6-over-IPv4 터널링 패킷은 방화벽에 의해 차단되어 IPv6망으로의 접속이 불가능하다.



<그림4> 방화벽에 의한 터널링 패킷 차단

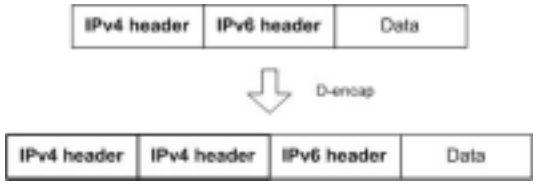
3. 제안된 IPv6 터널링

방화벽에 의해 터널링 패킷이 차단되는 문제의 근본적인 해결방안은 IPv4-over-IPv6 패킷을 방화벽이 인식할 수 있도록 방화벽의 세팅을 변경하는 것이다. 그러나 이미 전세계에 널리 퍼져있는 방화벽의 세팅을 바꿔 방화벽 내부의 사용자가 IPv6망에 접속할 수 있도록 하는데 걸리는 시간과 비용의 문제가 발생한다. 물론 점진적으로 방화벽의 세팅을 바꾼다 하더라도 IPv6망의 확산 속도를 따라가지 못하여 결국 IPv4망의 사용자가 IPv6망에 접속하는데 제한을 받게 된다.

본 논문에서는 이와 같은 문제를 Double- encapsulation(이하 D-encap)방식 및 HTTP 터널링을 응용한 방식으로 해결하고자 하였다. D-encap 방식은 기본적으로 터널링과 같이 IPinIP 방식의 캡슐화이다. 캡슐화된 패킷을 한번 더 캡슐화하여 프로토콜 필드 값이 바뀌도록하여 방화벽을 통과하도록 하는 방법이다. 이에 반해 HTTP 터널링은 보다 복잡한 방법으로 IPv6 패킷을 HTTP 패킷으로 캡슐화하여 방화벽을 통과하도록 하는 방법이다.

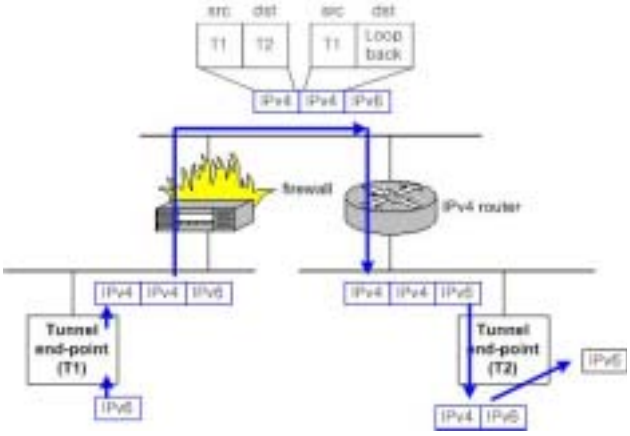
3.1 D-encap 방식

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 0x04(IPinIP) 되도록 하여 방화벽을 통과하도록 한다. 즉, D-encap 방식에 의해 캡슐화된 패킷은 <그림5>과 같은 헤더를 갖는다.



<그림5> D-encap된 패킷

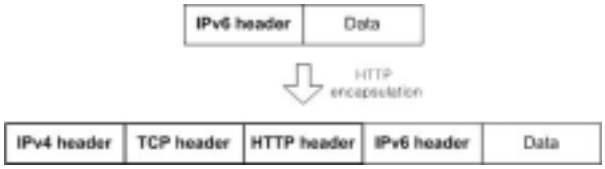
위와 같은 패킷 생성시 외부 IPv4 헤더의 송신측 주소는 터널의 생성지이며 수신측 주소는 터널의 종단이 된다. 내부 IPv4 헤더의 송신측 주소는 터널의 생성지가 되며 수신측 주소는 수신단의 Loopback 주소가 되어 패킷이 터널 종단에서 완전히 벗겨지도록 한다. <그림6>은 D-encap 방식으로 캡슐화된 패킷의 생성부터 소멸까지를 보여준다. T1과 T2이하는 IPv6망이며 T1에서 T2까지는 IPv4망에 해당한다. IPv6망에서 생성된 IPv6 패킷은 T1에서 D-encap 과정을 거쳐 IPv4망을 지나갈 수 있는 패킷이 된다. 이와같은 패킷은 방화벽 및 IPv4망을 지나 T2에서 IPv4 헤더가 벗겨져 T2이하의 IPv6망에 도달하게 된다.



<그림6> D-encap의 패킷의 생성과 소멸

3.2 HTTP 터널링 방식

HTTP 터널링 방식은 IPv6 패킷을 HTTP 패킷의 형태로 캡슐화하는 것이다[6]. <그림7>은 HTTP 터널링 방식으로 캡슐화된 패킷을 나타낸 것이다.



<그림7> HTTP로 캡슐화된 패킷

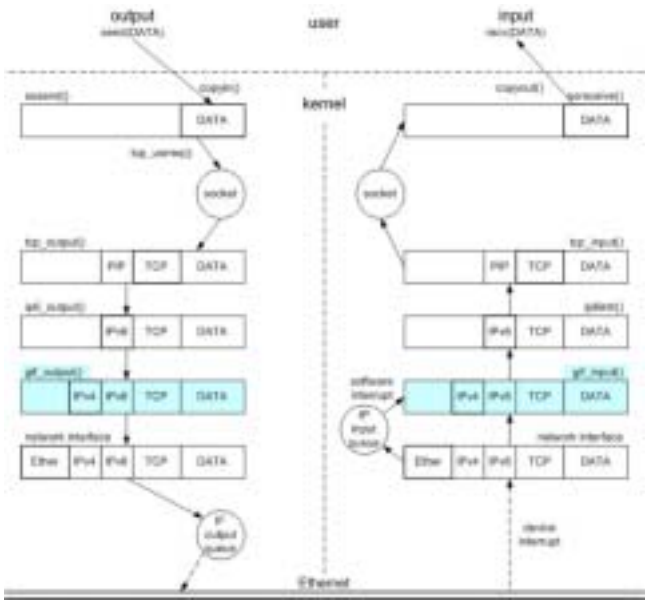
이와 같은 대부분의 방화벽이 웹 접속을 위한 80포트를 차단하지 않으므로 HTTP로 캡슐화된 패킷은 IP 헤더와 포트 및 그 이하 프로토콜까지 스캔하여 필터링하는 방화벽을 완벽하게 통과할 수 있도록 지원한다.

4. 구현

본 논문에서는 방화벽이 존재하는 IPv4망과 외부의 IPv6망의 연동을 위해 KAME에서 구현한 기존의 터널링 방식(6to4)을 바탕으로 D-encap 방식과 HTTP 터널링을 응용한 방식을 구현하였다.

4.1 기존의 터널링

FreeBSD와 KAME를 사용하여 호스트를 구성할 경우 gif(generic tunnel interface)를 통해 6to4 터널링을 지원한다[12]. gif를 통해 터널링을 할 경우 <그림8>와 같이 커널에서 IPv4 패킷을 한번 덧씌워 주는 과정을 통해 encapsulation 된 패킷을 생성한다.

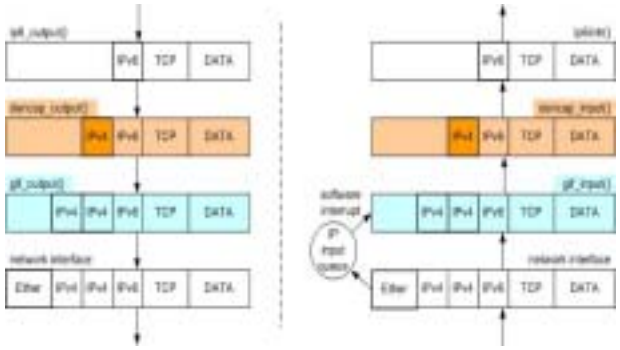


<그림8> IPv4-over-IPv6 패킷 생성 과정

이와 같이 생성된 패킷의 IP헤더 크기는 모두 60Bytes이며 <그림11>와 같은 환경의 네트워크에서 방화벽에 의해 모두 차단된다.

4.2 D-encap 방식

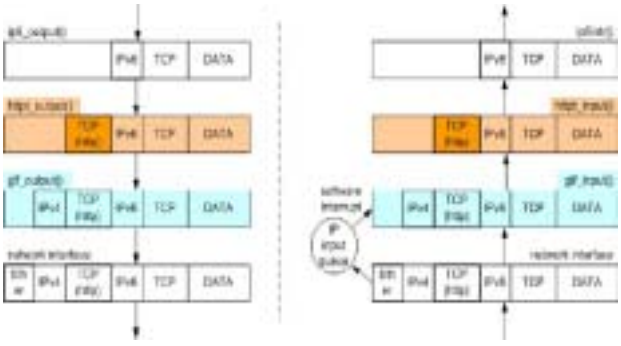
위의 IP패킷 생성 과정에서 gif를 통해 IPv4 헤더가 씌워지기 전에 D-encap 과정을 <그림9>과 같이 삽입하여 Double encapsulation된 패킷을 생성할 수 있다. 이와 같이 생성된 패킷은 IPinIP 패킷으로 <그림11>의 네트워크에서 방화벽을 통과 하여 IPv6망과 연동이 된다. 이 경우 패킷헤더는 기존의 터널링 방식에 비해 20Bytes가 더 증가한다.



<그림9> D-encap 과정 삽입

4.3 HTTP 터널링

D-encap 방식은 간단하게 IPv4 패킷을 한번 더 사용함으로써 IPv4 방화벽에 의해 패킷이 차단되는 것을 방지하고 IPv6망과의 연동을 가능하게 한다. 그러나 방화벽의 정책이 좀더 복잡하여 IP헤더 내부에 포함된 내용까지 필터링하게 되는 경우 IPv6망과의 연동이 불가능하다. 이에 HTTP 터널링 통해 IPv4 헤더 내부의 TCP 헤더까지 HTTP 형태로 생성하여 방화벽을 통과하여 IPv6망과의 연동을 가능하게 한다. 즉, <그림10>과 같이 D-encap 과정 대신 HTTP 터널링 과정을 삽입하여 구현하였다.



<그림10> HTTP 터널링 과정 삽입

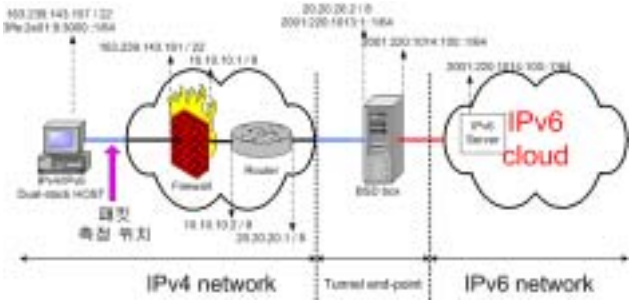
위와 같은 HTTP 터널링 방식의 경우 D-encap 방식에 비해 헤더크기가 8Bytes 증가하게 되나 보다 안정적으로 IPv6망과의 연동이 가능하다.

5. 실험 결과 및 분석

5.1 실험 환경

IPv4망 내부의 호스트가 IPv6망에 접속하기 위한 방법으로 터널링을 사용하였으며 방화벽에 의해 패킷이 차단되는 것을 D-encap방식과 HTTP 터널링 방식을 사용하여 기존의 터널링 방식과 비교 실험하였다. <그림11>는 실험에 사용된 네트워크 환경을 나타낸다. IPv4/IPv6 Dual 스택의 터널 양 종단은 FreeBSD4.3과 KAME를 이용하여 <표1>과 같이 구성하였다.

방화벽의 정책은 기본적인 패킷 필터링 방식을 적용하되 D-encap 방식의 실험에서는 IP헤더까지만 필터링 할 수 있도록 설정하였으며 HTTP 터널링 방식의 실험에서는 전송계층(Transport layer)까지 필터링 할 수 있도록 설정하였다.



<그림11> 실험에 사용된 네트워크

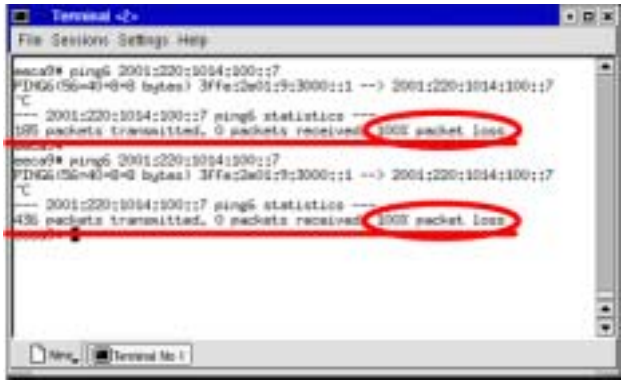
구 분	사양	운영체제	프로토콜 스택
Dual-stack Host	Pentium-III 800Mhz	FreeBSD 4.3 kame-freebsd43-snap	IPv4/IPv6 dual stack
Firewall	Pentium-II 233Mhz	Linux 2.4.2 (ipchains 사용)	IPv4 only
Router	CISCO 2600	c2600-is-mz.122-2.T.bin	IPv4 only
BSD box	Pentium-II 233Mhz	FreeBSD4.3 kame-freebsd43-snap	IPv4/IPv6 dual stack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

<표1> 실험 환경

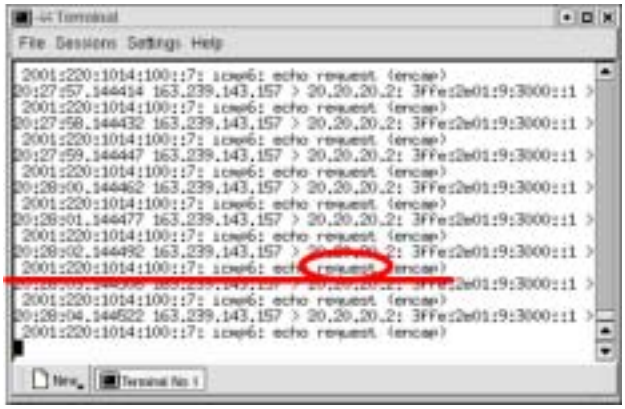
기존의 터널링 방식, D-encap, 그리고 HTTP 터널링 방식으로 패킷의 송수신 실험은 IPv6 패킷이 방화벽과 IPv4망을 통해 IPv6 서버에 송신후 이에 대한 응답(reply) 패킷의 도착여부로 방화벽을 통과하는지 확인하였으며 그 결과는 다음과 같다.

5.2 기존의 터널링 방식

<그림11>와 같은 환경에서 방화벽은 패킷의 IP 헤더만을 필터링하도록 설정하였으며 기본적으로 FreeBSD와 KAME를 통해 호스트를 구현하였다. 이와 같은 환경에서 <그림12>과 같이 ping을 통해 패킷의 전송 여부를 확인한 결과 전송되는 모든패킷은 방화벽에 차단되었으며 결국 아무런 응답이 없음을 확인할 수 있다.



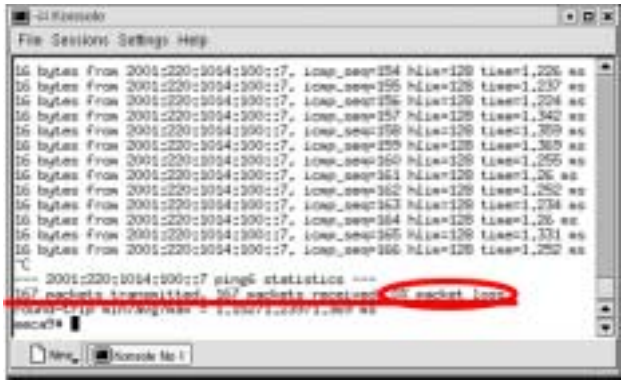
<그림12> 기존의 터널링 방식의 실험결과



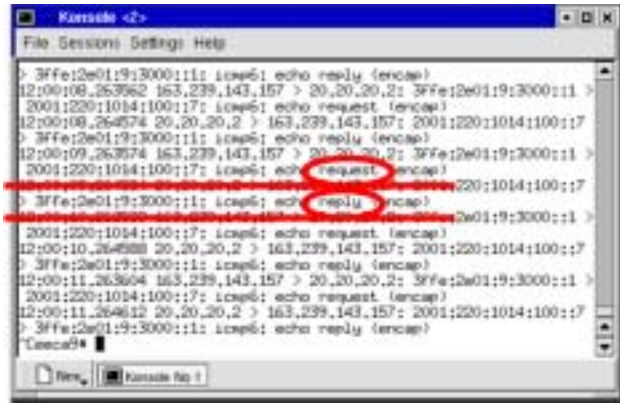
<그림13> 기존의 터널링 방식의 dump 결과

5.3 D-encap 방식

위 실험과 똑같은 환경에서 터널 종단인 Dual-stack 호스트와 BSD-box에 D-encap 과정을 삽입한후 같은 실험을 하였다. 그 결과 <그림14>와 같이 ping을 통해 패킷의 송수신을 확인할 수 있다. 즉, Dual-stack 호스트에서 생성한 모든 패킷이 방화벽 및 IPv4망을 통해 IPv6망의 서버와 통신되는 것을 알 수 있다.



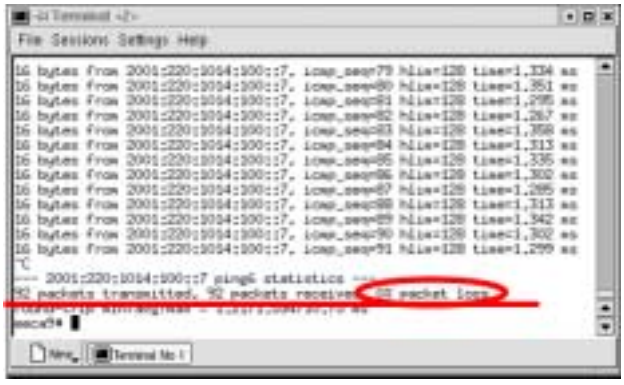
<그림14> D-encap 방식의 실험 결과



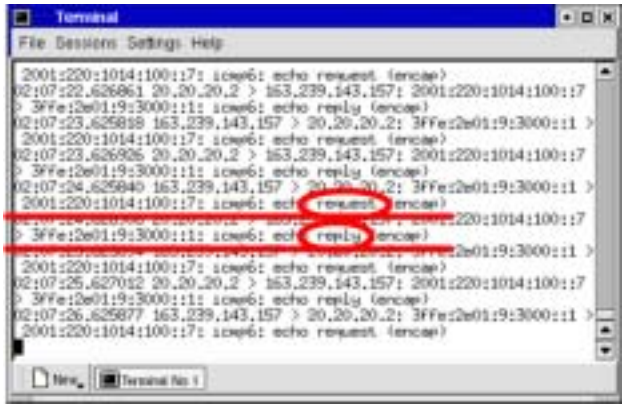
<그림15> D-encap 방식의 dump 결과

5.4 HTTP 터널링

위 실험과 같은 환경에서 방화벽은 패킷의 IP 헤더와 TCP 헤더를 필터링하도록 설정하였으며 D-encap 과정을 제거하고 HTTP 터널링 과정을 삽입하였다. 이와 같은 환경에서 <그림16>와 같이 ping을 통해 패킷의 송수신 여부를 확인한 결과 모든 패킷이 방화벽을 통과함을 확인 할 수 있었다.



<그림16> HTTP 터널링 방식의 실험 결과



<그림17> HTTP 터널링 방식의 dump 결과

5.5 실험 결과

실험결과 방화벽 환경에서는 기존의 터널링 방식으로 IPv6 망과의 연동이 불가능하지만 본 논문에서 제안한 D-encap 방식과 HTTP 방식을 통해서 IPv6망과의 연동이 가능함을 확인 할 수 있다.

구분	방화벽 허용여부	IP헤더 필터링	IP&TCP 헤더 필터링	헤더크기 (IPv6헤더 포함)
기존의 터널링 (KAME-IPv6)	O	X	X	60Bytes
D-encap 방식	O	O	X	80Bytes
HTTP 터널링	O	O	O	88Bytes

<표2> 터널링 패킷 송수신 결과

6. 결론

본 논문에서는 IPv6망에 접속하기 위한 기술로 현재 널리 사용되고 있는 터널링 방법이 방화벽내의 사용자들에게는 제한되는 문제점을 제시하고 이를 해결하기 위한 방안으로 D-encap 방식과 HTTP 터널링을 응용한 방식을 제안하였다. D-encap방식 보다는 HTTP 터널링 방식이 보다 나은 해결책이나 패킷 헤더의 크기가 커지는 문제점을 안고 있다. 이와 같은 터널링 방법이 근본적으로 방화벽에 의한 문제를 해결하기는 부족하나 이와 같은 연구가 지속되어 최소한 IPv6망의 활성화 및 IPv6망으로의 진화를 앞당길 수 있는 초석이 되었으면 한다.

참 고 문 헌

[1] W. Simpson, "IP in IP Tunneling", RFC1853, October 1995.
[2] R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC2893, August 2000.
[3] A. Durand, P. Fasano, I. Guardini, D. Lento, "IPv6 Tunnel Broker", RFC3053, January 2001.
[4] B. Carpenter, K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC3056, February 2001.
[5] Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture", RFC2373, July 1998.
[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol--HTTP/1.1", RFC2068, January 1997.
[7] R. Hinden, M. O'Dell, S. Deering, "An IPv6 Aggregatable Global Unicast Address Format", RFC2374, July 1998.
[8] Hinden, R., R. Fink, and J. Postel, "IPv6 Testing Address Allocation", RFC2471, December 1998.
[9] W. Richard Stevens, "TCP/IP Illustrated, Volume 1: The Protocols.", Addison-Wesley, Reading, Massachusetts, 1994.
[10] Keith Bostic, Michael J. Karels, "The Design and Implementation of the 4.4BSD Operating System", Addison-Wesley, 1996.
[11] Robert L. Ziegler, "Linux Firewalls", New Riders, 1999.
[12] KAME project, "www.kame.net"
[13] F. Templin, T.Gleeson, M.Talwar, D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", draft-ietf-ngtrans-isatap, April 2002.