리눅스 기반의 6to4 연동 메커니즘 설계 및 구현

이상도*, 신명기*, 김형준*
*한국전자통신연구원 표준연구센터
e-mail:{sdlee,mkshin,hjkim}@etri.re.kr

Implementaion of 6to4 Mechanism based on Linux

SangDo Lee*, Myung-Ki Shin*, Hyoung-Jun Kim*
*Electronics and Telecommunications Research Institue
Protocol Engineering Center

요 약

6to4 는 IPv6 을 지원하지 않는 IPv4 네트워크에 연결되어 있는 고립된 IPv6 네트워크의 상의 호스트들이 자동 터널링 방식을 통해 순수한 IPv6 네트워크나 6to4 로 구성된 네트워크상의 노드들과 통신할 수 있는 기술이다. 이 기술은 국제 민간 표준화 기구인 IETF 의 NGTrans 워킹 그룹에 의하여 표준화된 IPv6 전환 기술 중에 하나이며 독립적으로 구성된 IPv6 망 간의 통신을 하는 경우에 널리 사용될 것으로 예상이 된다. 본 논문은 IPv6 전환 기술 중 하나인 6to4 를 리눅스 환경에서 구현한 소프트웨어 모듈에 관한 설계 및 운용에 관한 것이다.

1. 서론

인터넷은 사용자의 급속한 증가로 인해 주소고갈 문제에 직면하고 있으며, 새롭게 등장하고 있는 휴대 인터넷, 홈 네트워크 등의 신규 서비스들은 인터넷 주소 고갈을 더욱 앞당길 것으로 예상된다. 이러한 주소 부족문제를 해결하기위해서 IETF 는 IPv6 (Internet Protocol version 6)를 개발하였고 IPv6 이제 그 표준화가 완료되어 도입기에 들어서고 있다[1].

IPv6 는 128 비트 주소 체계를 제공하여 거의 무한한 주소공간을 제공할 수 있다. 또한 단순화된 헤더, 주소 자동설정, 효율적인 이동성 지원 등 다양한 장점들로 인하여하여 기존 인터넷의 기반이 되는 인터넷 프로토콜인 IPv4 를 대체할 차세대 인터넷 프로토콜로 인정받고 있다. 또한, IPv6 는 인터넷의 영역을 사물들에까지 확대하는 차세대 통신패러다임으로 대두되고 있는 유비쿼터스 네트워크를 실현 시킬 기반 기술로서도 그 중요성을 가진다. 그러나 IPv6 는 IPv4 와 호환되지 않고 또한 이미 전 세계를 아우르는 거대한 인터넷이 IPv4 에 기반을 두어 운영되고 있어 점진적인 IPv6 로의 전환이 불가피하다. 이는 상당 기간 동안 IPv4 망과 IPv6 망이 공존하게 됨을 의미한다. 따라서 IPv6 망과 IPv4 망사이의 투명한 연동을 지원하는 IPv6 전환기술의 개발 및 지원이 주요한 이슈로 대두되었고 IETF 의 Ngtrans WG 을 중심으로 다양한 연동기술이 개발되었다[2][3][4]. 이 중에서 기존의 IPv4 네트워크를 이용하여 새로 구성된 IPv6 사이트간의 통신이 필요하게 될 때 이를 자동 터널링 방식을 이용하여 쉽게 통신할 수 있는 6to4 메커니즘이 주목을 받기 시작하였다. 따라서 본 논문에서는 6to4 메커니즘을 기반으로 하여 리눅스 상에서 구현한 6to4 설계 구조 및 운용 방식에 대해서 기술하고자 한다

2. 관련연구

2.1 6to4 의 개요

6to4 는 하나 이상의 글로벌 IPv4 주소를 가지고 있는 IPv6 전용 사이트에 2001:<IPv4 주소>::/48 형태의 단일 IPv6 프리픽스를 할당하여 외부 IPv6 네트워크와의 자동 터널링을 지원하는 메커니즘이다[2]. (그림 1)은 6to4 사이트의 주소형식을 보여준다.

3	13	32	16	64 bits
FP 001	TLA 0x0002	V4ADDR	SLA ID	Interface ID

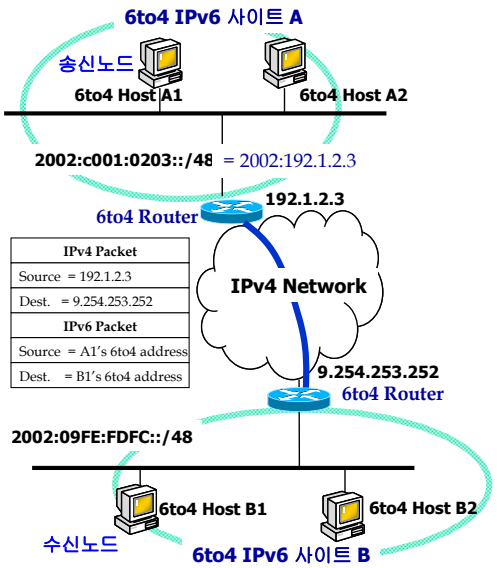
(그림 1) 6to4 주소 형식

6to4 에 의한 IPv6-in-IPv4 터널링에 따라 6to4 사이트에서 다른 6to4 사이트로 향하는 IPv6 패킷들은 6to4 라우터에서 IPv4 프로토콜 타입 41 을 가지는 IPv4 패킷들(RFC 791) 패킷들로 전송되며, IPv4 헤더는 목적지 주소와 송신측 주소를 포함한다. 이들 중 하나는 위에서 언급한 것처럼 구성된 IPv6 프리픽스의 V4ADDR 필드와 동일할 것이다. IPv4 패킷의 Payload 는 IPv6 헤더와 Palyload 를 포함한다. (그림 2)는 6to4 에 의한 IPv6-in -IPv4 터널패킷의 형식을 보여준다. IPv4 TTL 은 캡슐화된 IPv6 Hop limit 이 될 일반 값으로 설정될 것이다.

Version	HL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	**Protocol 41**	Header Checksum	
Source Address			
Destination Address			
Options			Padding
IPv6 header and Payload ·····			

(그림 2) 6to4 에의한 IPv6-in-IPv4 Tunnel 패킷 형식

(그림 3)은 6to4 에 의하여 고립된 IPv6 사이트가 연동하는 시나리오를 보여주고 있다. 이처럼 6to4 는 순수 IPv6 를 지원하지 않는 광역 네트워크에 연결되어 있는 고립된 IPv6 사이트나 호스트가 자동 터널링 방식을 통해 다른 IPv6 도메인이나 호스트와 통신할 수 있도록 지원한다.



(그림 3) 6to4 를 이용한 고립된 IPv6 망간의 연결

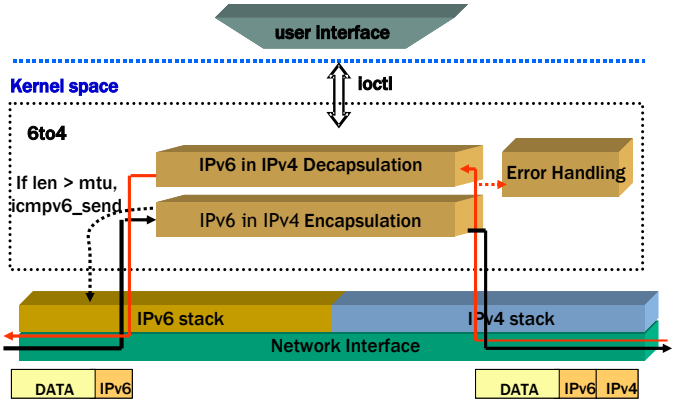
이러한 6to4 는 IPv4 기반의 거대한 인터넷이 주류를 이루고 있는 IPv6 초기 도입기에 IPv4 기반 인터넷을 통한 IPv6 지역망의 연결을 효과적으로 지원할 수 있는 기술로 활용될 것으로 기대된다.

2.2 6to4 의 구현구조 및 운용방법

본 장에서는 리눅스 커널 하에서 구현된 6to4 의 구현구조 및 6to4 의 운용방법을 설명하고자 한다.

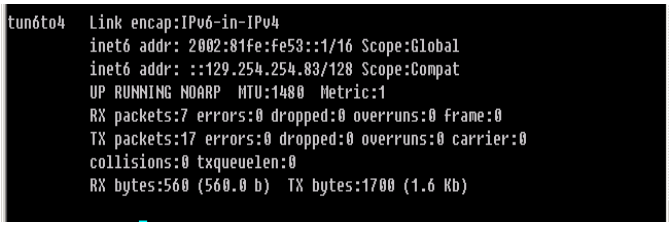
(1) 전체 블록도

(그림 4)는 리눅스 커널 상에서 구현된 6to4 의 구현 구조를 나타낸 그림이다. 6to4 사이트내의 IPv6 단말에서 IPv4 네트워크를 통하여 다른 6to4 사이트 혹은 IPv6 사이트로 향하는 IPv6 패킷의 IPv6-in-IPv4 터널링 수행 및 IPv4 망으로의 전송으로 이어지는 처리과정이 왼쪽에서 오른쪽으로 향하는 화살표로 나타나 있다. 또한 IPv4 망을 통해 수신되는 IPv6-in-IPv4 터널패킷의디 캡슐레이션 (Decapsulation) 및 로컬 6to4 사이트로의 전달과정을 오른쪽에서 왼쪽으로 향하는 화살표의 흐름으로 표시하고 있다. 6TALK 6to4 는 리눅스 커널로 구현되어 있으며 관리자가 CLI(Command Line Interface)를 통한 터널생성 및 제거를 수행할 수 있도록 텍스트 기반의 CLI 를 제공하고 있다. 현재는 6to4 router 로서의 기본 기능만을 수행하도록 구현되었으며 6to4 relay 로서의 기능에 필요한 기능은 아직 지원되지 않는다.



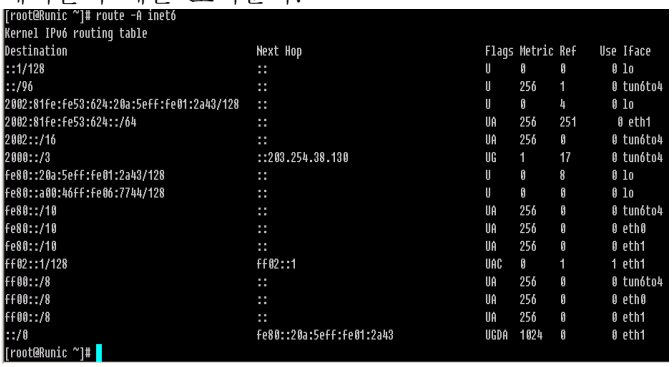
(그림 4) 리눅스 커널상의 6to4 의 구현구조

(2) 6TALK 6to4 에서의 IPv6 패킷 수신 및 IPv6-in-IPv4 터널링



(그림 5) 리눅스 환경에서 생성된 6to4 인터페이스의 예

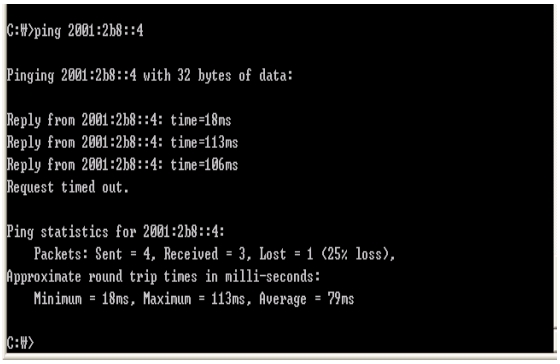
(그림 5)는 리눅스 기반 6to4 인터페이스의 생성 정보를 보여준다. 6to4 의 구동 시 IPv4 망과 연결된 인터페이스는 글로벌한 IPv4 주소를 가지고 있어야하고 tun6to4 인터페이스는 이 IPv4 주소가 내제된 6to4 형식의 IPv6 주소를 가지게 된다. 이러한 인터페이스 생성과 함께 RA(Router Advertisement)를 전송하는 radvd 데몬이 6to4 프리픽스로 RA 를 전송하면 6to4 사이트내의 호스트들은 6to4 주소를 가지게 된다. 단말이 이러한 6to4 라우터를 통해 IPv4 네트워크를 통한 순수한 IPv6 인터넷 망에 접속을 하도록 하기위해서는 6to4 라우터에서 IPv6 인터넷에 접속된 6to4 relay 로의 경로를 가진 라우팅 테이블이 6to4 에 생성되어야 한다. 6to4 라우터의 운영자는 그러한 6to4 릴레이의 주소정보를 설정해 주어야한다. 아래 그림에서는 그러한 6to4 릴레이로 203.254.38.138 을 지정한 경우의 IPv6 라우팅 테이블의 예를 보여준다.



(그림 6) 6to4 와 관련된 IPv6 라우팅 테이블의 예

6to4 라우터의 IPv4 네트워크와 접속된 인터페이스의 IPv4 주소가 129.254.254.83 으로 설정되어 그로 구성된 6to4 프리픽스 2002:81fe:fe53:624::/64 를 RA 로 광고할 경우의 한 6to4 단말에서의 IPv6 주소설정 상태를 (그림 7)이 보여주고 있다.

(그림 7) 6to4 사이트에서의 IPv6 주소설정 예

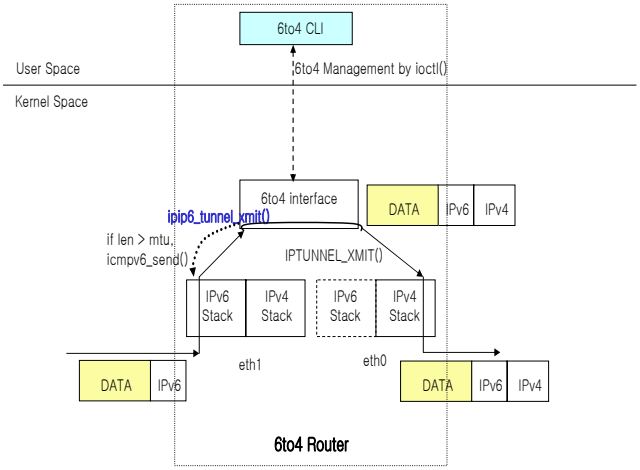


(그림 8) 6to4 사이트에서의 ping6 실행

성공적인 6to4 라우터에 의한 IPv6 망구성이 이루어지면 IPv6 단말에서 IPv6 인터넷 접속이 가능하게 된다. 이는 6to4 라우터에서 IPv6 인터넷으로의 접속을 지원하는 6to4 릴레이까지 IPv4 네트워크를 가로지르는 IPv6-in-IPv4 터널링의 수행을 통해서 가능하게 된다. 아래 (그림 8)은 6to4 단말에서 IPv6 인터넷상의 2001:2b8::4 로의 ping 의 수행 예를 보여주고 있다.

(그림 7) 6to4 사이트에서의 IPv6 주소설정 예

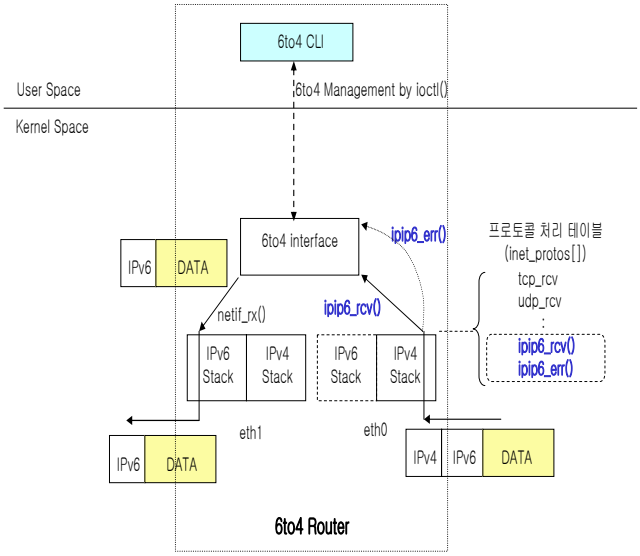
6TALK 6to4 가 활성화되면 tun6to4 라는 이름의 IPv6 in IPv4 IPv6 터널인터페이스가 (그림 5)와 같이 생성된다. 이러한 인터페이스의 전송함수로 ipip6_tunnel_xmit()함수가 등록되고 터널생성함수, 터널인터페이스에 대한 ioctl 를 수행하는 함수, 터널 제거 함수 등이 등록되어 각각의 이벤트 시 호출되게 된다. 6to4 단말로부터 IPv4 네트워크를 가로질러 IPv6 인터넷상의 사이트로 전송되어야 할 IPv6 패킷이 6to4 라우터에 수신되면 IPv6 라우팅에 의해서 tun6to4 인터페이스로 전달되게 되고 ipip6_tunnel_xmit()함수가 이를 수신하여 IPv6-in-IPv4 터널링을 수행 후 IPv4 프로토콜에 의해 처리되도록 IPv4 프로토콜로 전달하게 된다.



(그림 8) 6to4 에서의 IPv6 패킷의 수신 및 IPv6 in IPv4 터널링

(그림 8)은 6to4 를 통한 IPv6 패킷의 수신 및 IPv6 in IPv4 터널링의 과정을 보여주고 있다. 6to4 호스트로부터 전달된 IPv6 패킷은 tun6to4 인터페이스 전송함수로 등록된 ipip6_tunnel_xmit()로 전달되게 되고 tun6to4 의 mtu 보다 클 경우에는 송신측으로 icmpv6_send (ICMPV6_PKT_TOOBIG)을 보내고 그렇지 않은 경우 IPv6 in IPv4 터널헤더를 부가하여 IPTUNNEL_XMIT()를 통해 ip 프로토콜의 전송루틴을 호출하여 IPv4 패킷으로 IPv4 망상으로 전달하게 된다. 이러한 IPv6 in IPv4 터널 패킷은 IPv4 라우팅을 통해 지정한 6to4 relay 라우터 전달되고 6to4 relay 라우터는 IPv6 패킷을 복원하여 해당 목적지로 전달한다.

(3) 6to4 에서의 IPv6 in IPv6 터널 패킷수신 및 처리



(그림 9) 6to4 에서의 IPv6 in IPv4 터널 패킷의 수신 및 처리

(그림 9)는 리눅스 기반 6to4 상에서 수신된 IPv6-in-IPv4 터널 패킷의 처리과정을 보여준다. 6to4 인터페이스는 IPv4 프로토콜로부터 IPv6-in-IPv4 터널 패킷을 전달받기 위해서 IPv4 프로토콜의 프로토콜처리 테이블에 IPv4 헤더의 프로토콜 타입이 IPv6 (41)일 경우 ipip6_rcv() 함수로 전달되도록 함수 핸들러를 등록하고 IPv6-in-IPv4 터널 에러의 경우 처리함수로 pip6_err() 함수 핸들러를 등록하여 해당 이벤트의 처리가 이루어지도록 구현되었다. 이러한 프로토콜 처리 테이블 상으로의 등록 및 제거는 IPv4 프로토콜에서 제공하는 inet_add_protocol()와 inet_del_protocol() API 함수를 통해서 이루어진다. 이러한 방식으로 시스템의 재 실행 없이 tun6to4 인터페이스의 구동 및 제거가 가능하게 된다. IPv6-in-IPv4 터널 패킷이 수신되면 위에서 설명한 것처럼 IPv6-in-IPv4 터널패킷 처리함수인 ipip6_rcv()로 전달되고 IPv6-in-IPv4 터널헤더가 제거된 후 IPv6 프로토콜을 통해 IPv6 라우팅에 의해 6to4 단말로 전달된다. 이러한 방식으로 6to4 사이트가 IPv4 네트워크를 통하여 IPv6 인터넷으로 접속될 수 있게 된다.

(4) 6to4 의 사용자 인터페이스

6TALK 은 관리자에게 웹 CGI 모드 기반 제어 인터페이스 및 텍스트기반 CLI 를 제공한다. 이러한 인터페이스 상의 명령들에 대한 6to4 인터페이스로의 전달은 그림 4 에서도 보여주듯이 ioctl 를 통해 이루어진다. ioctl 코드를 기반으로 6to4 가 관리자에게 지원하는 명령어와 그 기능은 다음과 같다.

명령어	기능	구현방법
down 6to4	6to4 인터페이스삭제	'if6to4ctrl stop'
nodown 6to4 ip_addr4	6to4 인터페이스생성	'if6to4ctrl start ip_addr4'
interface 6to4 relay_addr4	6to4 relay 주소지정	'/sbin/ip -6 route add 2000::/3 via ::relay_addr4 dev tun6to4 metric 1'

III. 결론

본 논문에서는 6to4 연동 메커니즘을 구현하기 위한 리눅스 상에서 설계 및 구현하였고, 그 동작 성을 확인하기 위한 테스트를 수행하였다. 또한 본 논문에서 제시하고 있는 이러한 리눅스 기반의 네트워크 모듈 방식은 터널 링을 기본으로 하는 여러 연동 메커니즘을 손쉽게 구현할 수 있을 것으로 기대한다. 향후 IPv4-in-IPv6 터널링 방식을 이용한 성능 평가에 관한 연구를 진행할 예정이다.

[참 고 문 헌]

[1] Deering, S., and R. Hinden, Editors, "Internet Protocol Version 6 (IPv6) Specification", RFC 1883, December 1995.

[2] B. Carpenter,K. Moore,“ Connection of IPv6 Domains via IPv4 Clouds” ,RFC 3056, February 2001

[3] G. Tsirtsis,P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000

[4] R. Gilligan,E. Nordmark,"Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000