윈도우 NT 커널 환경에서 IPv6 프로토콜 구현 연구

강 신 각[†] · 김 대 영[††]

요 약

월드 와이드 웹(WWW)과 MBone 응용 등의 출현으로 인터넷 이용자가 급속히 증가되면서 인터넷 주소 공간의 확장과 멀티미디어 응용을 지원할 수 있도록 기존 인터넷 프로토콜의 개선이 요구됨에 따라 IETF에서는 차세대 인터넷 망 계층 프로토콜로써 IPv6를 개발하였다.

이 논문에서는 IPv6 프로토콜을 Windows NT의 커널 내부에 프로토콜 드라이버로 구현한 내용을 기술하고 있다. 본 연구에서는 IPv6 호스트로 동작되기 위해 필수적으로 요구되는 IPv6 헤더 처리기능, IPv6 주소 처리기능, 제어메세지 처리기능, 그룹관리 메세지 처리기능, 이웃탐색 기능이 구현 및 시험되었다. 구현된 IPv6 프로토콜 드라이버 모듈은 하부 통신망 접속 카드와 표준 인터페이스인 NDIS를 통해 접속되며, 디스패치 함수와 Lower-Edge 함수를 이용하여 커널 내부에서 동작하는 드라이버 모듈을 상위 사용자 응용 및 하부 NDIS와 접속시키는 형태로 구현하였다. 구현된 IPv6 프로토콜 드라이버는 커널 모드에서 구현됨으로써 향상된 성능을 제공하며, 다이나믹 링크 라이브러리 형태로 사용자 인터페이스를 제공하므로 응용 프로그램 개발자들이 손쉽게 사용할 수 있도록 하였다.

An Implementation of Internet Protocol Version 6 on Windows NT Kernel Environment

Shin-Gak Kang[†] · Dae-Young Kim[††]

ABSTRACT

The next generation internet protocol, IPv6, have been developed by the IETF according to the requirements of enhancement of classic IP protocols to satisfy the lack of Internet address space as well as the support of multimedia applications.

This paper presents an implementation of IPv6 protocols on the Windows NT kernel environment. In this work, we developed and also tested the basic functions, required for operating as an IPv6 host, such as IPv6 header processing, IPv6 address handling, control message processing, group membership processing and neighbor discovery functions. The implemented IPv6 protocol driver module is connected to the lower network interface card through NDIS, a standard network interface. And this driver module that operates within kernel, is implemented as it is connected to upper user applications and lower NDIS using dispatch and lower-edge functions. The developed IPv6 protocol driver can provide not only enhanced performance because it is implemented in kernel mode, but also convenience of usage to the application developers because it gives user interface as a dynamic link library.

† 정 회 원: 한국전자통신연구원 정보통신표준연구센터
†† 정 회 원: 충남대학교 정보통신공학과
논문접수: 1997년 8월 5일, 심사완료: 1997년 9월 23일

1. 개 요

인터넷이 전 세계적으로 급속히 확대되면서 기존의 인터넷 망 계층 프로토콜 버전 4(IPv4)에서[1] 지원하는 인터넷 주소 공간이 얼마 가지 않아 고갈되고 말 것이라는 예측이 1990년 말경에 제기되자 IPv4의 이러한 주소 공간의 제약 문제를 해결 할 수 있는 차세대 인터넷 망 계층 프로토콜(IPng:IP next generation)에 대한 논의가 시작되었다. 이와 함께 새로운 멀티미디어 응용 서비스가 개발, 제공되면서 인터넷 상에서 실시간 멀티미디어 응용을 효과적으로 전송할 수 있는 통신 프로토콜에 대한 요구가 높아지자 기존 IPv4에 주소 공간의 확대 뿐만 아니라 보다 새로운 기능을 추가하기 위한 연구가 본격적으로 추진되게 되었다[2]. 이러한 연구활동 결과로 TUBA(TCP and UDP with Bigger Addresses), IPAE(IP Address Encapsulation), PIP(P Internet Protocol), SIP(Simple Internet Protocol), CATNIP(Common Architecture for the Internet) 등 다양한 차세대 인터넷 프로토콜들이 연구 제안되었으며, 그 후 IPAE, PIP, SIP는 SIPP(Simple Internet Protocol Plus)로 통합되면서 CATNIP, TUBA, SIPP가 IPng 후보로 경합을 벌이게 되었다. 이와 같이 다양한 IPng 후보 프로토콜 들이 제안되자 IETF에서는 1993년에 IPng Area라는 그룹을 구성하여 차세대 인터넷 망 계층 프로토콜 개발 작업을 담당하게 하였다. IPng 그룹에서는 차세대 인터넷 망 계층 프로토콜에 대한 요구사항 및 선정 원칙 등을 규정하는 IPng 기술지침을 개발하였고, 이에 근거해 제안된 여러 후보 프로토콜들을 비교, 평가하여 1994년 말에 SIPP를 IPng의 기본 안으로 최종 채택하였다. 그리고, IPng에 인터넷 망 계층 프로토콜 버전 번호가 6이 할당됨으로써 이 때부터 새로운 차세대 인터넷 프로토콜이 IPv6라 불리게 되었다[3]. IPv6는 기존 IPv4가 제공하는 기능을 그대로 제공하면서 여러 향상된 기능을 제공하는 형태로 개발되었다. IPv6의 주요 특징으로는 인터넷 주소 공간의 확대와 라우팅 기능의 확장, 헤더 형식의 단순화, 확장 헤더 및 옵션의 제공으로 확장성과 유연성 제공, 인증과 데이타 보호기능 제공, 자동 주소 설정 기능 제공, 발신자 요구 경로설정 프로토콜의 지원, 기존 IPv4로부터의 유연한 천이 기능 제공, 다양한 서비스품질 지원 등이 있다[3].

일반적으로 IPv6 라고 부르는 차세대 인터넷 프로토콜은 IP 데이타그램의 헤더 형식 및 송수신과 관련된 기본적인 사항을 규정하는 규격[4], 새로운 형태의 IPv6 주소 구조를 규정하는 규격[5], IP 제어 메세지의 기능 및 형식에 대해 규정하는 규격[6], 그리고 이웃 탐색 프로토콜을 정의하는 규격[7]으로 구성된다. 이밖에, IPv6 데이타그램의 인증 및 보안 기능[8][9][10], IPv6 주소의 자동설정 기능[11], IPv6 호스트 및 라우터의 천이 메카니즘[12] 등에 대한 규격이 차세대 인터넷 프로토콜을 위해 정의되었으며, 기존 IPv4에서 사용되던 일부 규격 들은 IPv6를 위해서도 계속 적용된다.

IETF에서 IPv6에 대한 기본 방향이 설정되고 관련 규격 들이 어느 정도 구체적인 모습을 갖추게 되자 SUN, HP, DEC, Cisco, INRIA 등 유닉스 운영체계와 인터넷 라우터 제품을 생산하는 업체와 인터넷 프로토콜에 대한 연구를 수행하는 대학 및 연구소에서는 앞을 다투어 IPv6 규격에 따른 호스트 및 라우터를 개발하고 있으며, 터널링 기술을 이용하여 6-Bone [13]이라는 가상의 IPv6 망을 구축해 개발된 제품간에 상호운용성 시험을 활발하게 추진하고 있다. 따라서 현재에는 대부분의 유닉스 및 라우터 벤더가 IPv6 규격에 따른 제품을 개발해 보유하고 있으며, 일부 대학 및 연구소에서는 개발된 IPv6 소스 코드를 공개하여 누구나 이용할 수 있도록 제공하고 있다. IPv6 프로토콜 구현 사례를 모아 놓은 Web 홈 페이지에는 현재 IPv6 호스트 기능을 개발한 사례로 22개 기관이, 그리고 IPv6 라우터 제품을 개발한 사례로 9개 기관이 소개되어 있다[14]. 이중 호스트 기능 구현 사례는 대부분 유닉스 계열 운영체계에서 개발된 것이고, Windows 95 및 Windows NT에서 개발된 사례는 각각 1건씩 만이 소개되었으나 자세한 구현 내용에 대해서는 소개되어 있지 않다. 국내에서도 IPv6에 대한 관심이 높아 수년 전부터 대학 및 연구 기관에서 관련 연구가 수행되고 있다. 특히 한국전자통신연구원은 숭실대학교와 공동으로 Free BSD의 커널에서 동작되는 IPv6를 구현하여 동작시험을 완료하였고, 현재 국내 6-Bone 구축을 추진하고 있다[15].

본 연구에서는 윈도우 NT 기반의 차세대 인터넷 멀티미디어 응용 플랫폼을 구축하기 위한 연구의 중

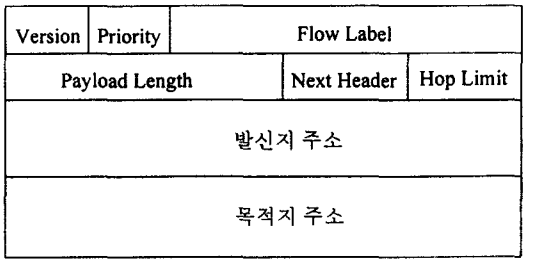
간 단계로써 IETF에서 차세대 인터넷 망 계층 프로토콜 표준으로 채택한 IPv6 프로토콜을 Windows NT 3.5의 커널에 프로토콜 드라이버로써 구현하였고, 개발된 IPv6 API를 이용하는 응용 프로그램을 개발하여 시험함으로써 구현된 IPv6 프로토콜 드라이버의 기능과 동작을 확인하였다. 구현된 기능은 호스트 시스템에서 IPv6의 기본 헤더 및 확장 헤더에 대한 처리기능, IPv6 주소 해석 및 처리기능, 이웃 탐색(Neighbor Discovery) 기능, 제어 메세지 생성 및 처리기능, 그룹관리 메세지 처리기능으로 IPv6 호스트로 동작되기 위해 기본적으로 요구되는 기능이 구현되었다. 라우터에서 요구되는 기능과 주소 자동 설정 기능, 그리고 IP 데이타그램의 인증 및 보안 기능 등은 이번 구현 범위에서 제외 되었다.

2. 차세대 인터넷 망 프로토콜

IPv6는 기존의 IPv4를 대체하도록 설계된 차세대 인터넷 망 계층 프로토콜로써 IPv4와 구별되는 몇 가지 특징을 제공한다. 먼저, IPv6 설계 시 최대 이슈가 되었던 주소 공간을 IPv4가 지원하는 32비트에서 128비트로 확장함으로써 주소 공간의 부족 문제를 해소시킴과 동시에 보다 구조적으로 주소를 지정할 수 있게 되었으며, 주소의 자동 설정 등이 가능하게 하였다. 또한 멀티캐스트 주소에 Scope이라는 영역을 두어 멀티캐스트 라우팅의 기능을 보강하였으며, 애니캐스트 주소라는 새로운 주소 유형을 추가하여 주소 지정 기능을 확장시켰다. 둘째, IPv4의 헤더 형식을 단순화 시켜 IP 데이타그램 처리시 요구되는 비용을 절감시켰고, 주소 영역이 4배로 증가된 것을 헤더 형식의 단순화를 통해 기본 헤더의 전체 길이가 2배만 증가되도록 하였다. 셋째, IP 헤더의 옵션 부분의 부호화 방식을 개선하여 보다 효율적으로 IP 데이타그램이 전달되도록 하였으며, 새로운 방식의 헤더 확장 방식을 도입하여 향후 새로운 기능을 융통성 있게 추가할 수 있도록 하였다. 넷째, 멀티미디어 응용 요구사항을 수용하기 위해 특정 트래픽에 대해 송신자의 QoS 요구사항을 표시할 수 있도록 Flow Label이라는 영역을 IP 헤더에 추가하였다. 송신자 주소와 Flow label을 조합하면 인터넷 상에서 특정 트래픽 흐름에 대한 식별이 가능하므로 이를 이용하여 실시간 또는 멀티미디어 응용 서비스를 인터넷 상에서 보다 용이하게 수용할 수 있을 것으로 예상된다. 다섯째, IP 데이타그램에 인증 및 보안 기능을 지원할 수 있도록 기능이 확장되었다.

2.1 차세대 인터넷 프로토콜(IPv6)

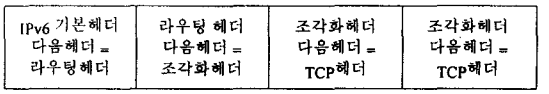
IPv6의 기능은 기본 헤더에 의한 기능과 확장 헤더를 이용하여 제공되는 옵션 기능으로 구분된다. IPv6 기본 헤더는 40바이트의 고정 길이를 가지며, 그 형식은 (그림 1)과 같다. (그림 1)에서 4비트 길이를 갖는 Version 영역은 IP의 버전을 표시하며 IPv6의 경우 값이 6이 된다. 역시 4비트 길이를 갖는 Priority 영역은 전송되는 각 데이타그램의 우선순위를 표시한다. 24비트 길이를 갖는 Flow Label은 송신자가 IPv6 라우터에서 특별하게 처리되기를 원하는 특정 IP 패킷 흐름을 표시할 수 있도록 한다. Payload Length는 16비트 길이를 가지며 전송하고자 하는 사용자 데이타 길이를 표시한다. 8비트 길이를 갖는 Next Header는 현재 헤더의 뒤에 따라오는 확장 헤더의 유형을 지시한다. 8비트 길이의 Hop Limit는 전달되는 패킷의 수명을 표시하는데 패킷이 한 노드를 지날 때 마다 1씩 감소하고 0이 되면 폐기된다. 주소 영역은 각각 16바이트 길이를 가지며 송신측 호스트와 착신측 호스트의 IPv6 주소를 표시한다.



(그림 1) IPv6 기본 헤더
(Fig. 1) IPv6 basic Header

IPv6 확장 헤더는 IPv6 헤더와 상위계층 헤더 사이에 선택적으로 오게 되며, 확장 헤더가 따라오면 상위계층에 부가적인 동작이나 서비스가 제공되게 된다. 확장 헤더는 (그림 2)와 같이 IP 헤더의 Next Header 값에 의해 지시되며, 요구되는 확장 헤더가 계속적으로 IP 데이타그램에 캡슐화 되어 전송될 수

있다. 현재 정의되어 있는 확장 헤더에는 홉간 옵션 헤더, 목적지 옵션 헤더, 라우팅 헤더, 조각화(fragment) 헤더, 인증 헤더, ESP(Encapsulating Security Payload) 헤더가 있다.



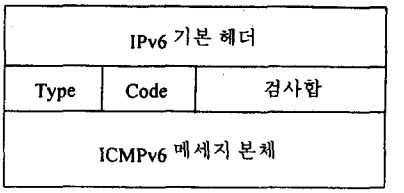
(그림 2) 확장 헤더를 포함하는 IP 패킷의 예
(Fig. 2) An example of IP packet including Extension Headers

IPv6에서 규정하는 최소 링크-MTU(Maximum Transmission Unit)는 576옥텟이며(IPv4의 경우는 68옥텟), 이를 만족할 수 없는 링크에서는 링크 레벨의 조각화 및 재조립 기능이 수행되어야 한다. 송 수신단 간 최소 링크-MTU를 의미하는 path-MTU 보다 큰 패킷을 전송하기 위해서는 조각화 헤더를 사용하도록 하고 있다. 송신측에서 전송할 수 있는 IP 패킷의 크기는 path-MTU 크기에 따라 가변적이지만 IPv6 헤더상의 최대 지원 가능한 패킷 크기는 65,535옥텟이며, 이 이상의 패킷은 점보-Payload 홉간 옵션 헤더를 이용하여 전달할 수 있다.

IPv6 주소는 유니캐스트, 애니캐스트, 그리고 멀티캐스트라는 세가지 유형을 가지며 이들 주소는 인터페이스에 할당되므로 어떤 인터페이스에는 하나 이상의 주소 유형이 할당될 수도 있다. 마찬가지로 다수의 물리적 인터페이스들이 하나의 주소만 할당 받을 수도 있는데 이것은 다수의 물리적 인터페이스에 대한 부하 분산에 유용하다. 주소 유형 중 애니캐스트 주소는 같은 애니캐스트 주소를 갖는 여러 인터페이스 중 하나에게만 전달되며, 유니캐스트 주소 공간으로 부터 할당되었고 유니캐스트 주소 구조를 가지므로 유니캐스트 주소와 구별할 수가 없다.

2.2 차세대 인터넷 제어 메세지 프로토콜(ICMPv6)

IPv6 프로토콜을 위한 제어 메세지의 처리 프로토콜을 규정하는 ICMPv6는 기존 IPv4와 함께 사용되던 ICMPv4 프로토콜[16]과 IGMP(Internet Group Management Protocol) 기능[17]이 포함되어 있다. ICMPv6 메세지는 (그림 1)에 명시된 IPv6 기본 헤더 뒤에 붙어 전달되게 되며, 메세지의 일반 형식은 (그림 3)과 같다. 여기서 Type 영역은 메세지 유형을 표시하며, Code 영역은 메세지 유형에 따라 각각 정의되어 세부적인 메세지 유형을 표시할 수 있다.



(그림 3) ICMPv6 메세지 형식
(Fig. 3) ICMPv6 Message Format

ICMPv6 메세지는 오류 처리용 메세지와 정보용 메세지로 구분되며, 뒤에서 설명할 이웃탐색 프로토콜 메세지도 ICMPv6 메세지를 이용하여 해당 노드로 전달된다. ICMPv6 메세지 유형은 〈표 1〉과 같다.

〈표 1〉 ICMPv6 및 NDP 메세지 유형
〈Table 1〉 Message Types of ICMPv6 and NDP

구분	Code	메세지 유형
ICMPv6 오류 메세지	1	목적지 도달 불가
	2	너무 큰 패킷
	3	시간 초과
	4	매개변수 문제
ICMPv6 정보 메세지	128	에코 요청
	129	에코 응답
	130	그룹 멤버쉽 질의
	131	그룹 멤버쉽 보고
	132	그룹 멤버쉽 삭제
NDP 메세지	133	라우터 찾기
	134	라우터 광고
	135	이웃 찾기
	136	이웃 광고
	137	재지정 메세지

ICMPv6의 주요 동작을 살펴보면 먼저, 정의되어 있지 않은 오류 메세지를 수신하면 이 메세지를 상위 계층에 전달하나, 정보 메세지를 수신하는 경우에는 해당 메세지를 버린다. 둘째, 모든 오류 메세지는 576

옥텟을 넘지 않는 범위에서 오류가 발생한 패킷 정보를 포함한다. 셋째, ICMPv6는 ICMPv6 오류 메세지, 멀티캐스트 패킷, 링크계층 방송으로 전송되는 패킷 등의 오류에 대해서는 응답하지 않는다. 넷째, 오류 메세지의 과다 생성 및 송신을 제한하기 위해 타이머 또는 대역폭에 근거해 오류 메세지의 송신율을 제한한다.

2.3 이웃 탐색 프로토콜(NDP)

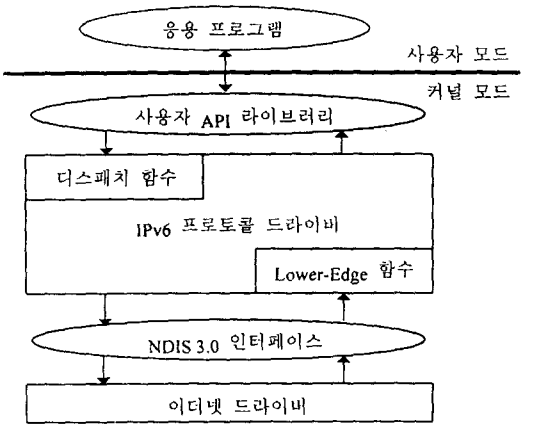
NDP는 기존 IPv4에 근거한 인터넷에서 사용되던 ARP(Address Resolution Protocol) 기능을 수행하며, 또한 호스트가 접속되어 있는 링크상에 존재하는 라우터에 관한 정보와 링크에 할당된 주소의 Prefix에 관한 정보를 찾는 기능, 링크와 관련된 MTU나 최대 홉 한계 등의 전송 매개변수에 대한 정보 찾는 기능, 주소 자동 설정 기능, 목적지 주소에 대해 다음 홉 결정 기능, 이웃 찾기 기능을 제공한다. 이밖에, 동일한 주소가 다른 노드에 할당되었는지 여부를 검출하는 기능과, 기존 ICMPv4에서 수행하였던 재지정(Redirect) 기능을 수행한다. NDP 메세지는 〈표 1〉에 표시된 바와 같이 라우터 찾기(Router Solicitation), 라우터 광고(Router Advertisement), 이웃 찾기(Neighbor Solicitation), 이웃 광고(Neighbor Advertisement), 재지정과 같은 5가지 유형이 있으며, 이들 NDP 메세지들은 ICMPv6 메세지를 이용하여 전달된다. IPv6 호스트는 이웃탐색 기능을 수행하기 위해 최근에 패킷을 송신한 적이 있는 이웃 노드에 대한 정보를 저장하는 Neighbor Cache, 최근 패킷을 송신한 적이 있는 각 목적지에 대한 정보를 저장하는 Destination Cache, 링크 상에 존재하는 주소 집합을 정의하는 Prefix List, 그리고 수신된 패킷에 의한 라우터들의 목록을 표시하는 Default Router(DR) List에 대한 정보들을 보유할 필요가 있다.

NDP의 동작을 간단히 살펴보면 다음과 같다. 먼저, 데이타그램의 송신 요구를 받으면 노드는 Destination Cache와 Prefix List, 그리고 DR List 정보를 이용하여 Next-Hop의 IP 주소를 결정하고, 그 결과는 추후 다시 사용할 수 있도록 Destination Cache에 저장한다. 다음에는, 해당 Next-Hop에 대한 링크 계층 주소를 얻기 위해 Neighbor Cache 정보를 검색하며, 마일 찾는 정보가 없으면 이웃 찾기 기능을 이용하여 원하는 링크 계층 주소를 찾는다. 이밖에 라우터는 재지정 메세지를 이용하여 목적지에 대한 더 좋은 Next-Hop 정보를 호스트에게 알려 줄 수 있으며, 재지정 메세지를 수신하는 호스트는 이 정보를 이용하여 Next-Hop에 대한 정보를 갱신한다.

3. 프로토콜 드라이버의 설계 및 구현

본 연구에서는 IPv6와 ICMPv6 기능을 Windows NT 커널 내부에 네트워크 드라이버 형태로 구현하였다[18][19]. 차세대 인터넷 망 계층 모듈을 커널 내부에 구현하기 위해서는 Windows NT이 제공하는 커널 프로그래밍 기법을 사용하여야 하는데 IPv6 프로토콜 드라이버의 구현 구조는 (그림 4)와 같다.

구현된 IPv6 프로토콜 드라이버 모듈은 하부 통신망 접속 카드와 Windows NT이 제공하는 NDIS(Network Driver Interface Specification) 인터페이스를 통해 접속된다. NDIS는 하위 네트워크 접속 카드의 특성과는 무관하게 NDIS 응용 프로그램이 동작할 수 있게 해 주므로 구현된 IPv6 드라이버는 다양한 네트워크 환경에서 사용이 가능하다. 구현된 IPv6 드라이버는 NCPA(Network Control Panel Applet)을 이용하여 설치하며, 이를 위해 요구되는 스크립트 화일을 작성하였다. 또한 사용자가 원할 때 설치된 드라이버의 로딩과 언 로딩이 가능하도록 하는 기능을 구현하였다.



(그림 4) IPv6 프로토콜 드라이버 구현구조
(Fig. 4) Implementation Architecture of IPv6 Protocol Driver

Windows NT에서는 커널 내부에서 동작하는 드라이버 모듈을 상 하위 계층과 접속시키기 위해 디스패치 함수와 Lower-Edge 함수를 사용한다. 디스패치 함수는 사용자의 호출이 있을 때 미리 지정된 동작을 수행하도록 할 수 있는데 Windows NT의 DDK에서는 이를 위한 기본적인 함수 형태를 제공하므로 이를 변형하여 IPv6 드라이버를 위한 디스패치 함수를 구현하였다. 구현된 디스패치 함수의 종류 및 기능은 〈표 2〉와 같다.

〈표 2〉 디스패치 함수
〈Table 2〉 Dispatch Functions

디스패치 함수	기능
IPv6DispatchOpen	주소 관리 모듈, NEIGHBOR-CACHE, DESTINATION_CACHE, 타이머 모듈 등의 초기화와, 망 접속 카드와의 바인딩 작업 수행.
IPv6DispatchClose	IPv6 드라이버를 위해 할당된 메모리를 모두 해제.
IPv6DispatchReceive	사용자 요구에 따라 할당된 버퍼에 NDIS를 통해 수신된 데이타를 복사.
IPv6DispatchSend	사용자 요구에 따라 데이타를 전송하기 위해 NDIS에 데이타 송신을 요청.
IPv6DispatchIoControl	주소 관리 모듈에 IPv6 주소와 링크 주소를 넘겨 주고, 필요 시 질의가 가능. 또한 이더넷 드라이버에 대한 패킷 필터링 작업 수행.
IPv6DispatchUnload	NDIS 인터페이스로 부터 IPv6 드라이버의 등록을 해제.

드라이버의 초기화 과정에서 디스패치 함수는 NT의 입출력 관리자에 의해 객체에 저장되며, 이를 통해 각 함수는 사용자 요구에 따라 NT 입출력 관리자에 의해 호출될 수 있다. 드라이버 객체와 디스패치 함수의 맵핑 관계는 다음과 같다.

```
DriverObject→MajorFunction[IRP_MJ_CREATE]
=IPv6DispatchOpen;
DriverObject→MajorFunction[IRP_MJ_CLOSE]
=IPv6DispatchClose;
DriverObject→MajorFunction[IRP_MJ_READ]
=IPv6DispatchReceive;
DriverObject→MajorFunction[IRP_MJ_WRITE]
=IPv6DispatchSend;
DriverObject→MajorFunction[IRP_DEVICE_CONTROL]
=IPv6DispatchIoControl;
DriverObject→DriverUnload=IPv6DispatchUnload;
```

Lower-Edge 함수는 NDIS를 통해 네트워크 접속 카드로부터 패킷을 수신하였을 경우나, NDIS에 요청한 송신의 비 동기적 완료를 통보해 주는 경우와 같이 NDIS를 통해 드라이버와 접속 카드 사이에 이벤트가 발생할 때 사용하게 된다. Windows NT의 DDK에서는 디스패치 함수와 마찬가지로 이를 위한 일련의 함수를 제공하므로 이를 이용하여 IPv6 드라이버를 위한 Lower-Edge 함수를 구현하였다. 구현된 Lower-Edge 함수의 종류 및 기능은 〈표 3〉과 같다. 드라이버

〈표 3〉 Lower-Edge 함수
〈Table 3〉 Lower-Edge Functions

Lower-Edge 함수	기능
IPv6OpenAdapterComplete	하부 망 접속 카드가 준비되었음을 비 동기적으로 입출력 서비스에 알림.
IPv6CloseAdapterComplete	하부 망 접속 카드에 대한 사용자의 서비스 종료 요청이 완료되었음을 비 동기적으로 입출력 서비스에 알림.
IPv6SendComplete	데이타 송신이 종료되었음을 비 동기적으로 입출력 서비스에 알림.
IPv6TransferDataComplete	데이타 수신이 종료되었음을 비 동기적으로 입출력 서비스에 알림.
IPv6RequestComplete	사용자의 요청(주소 지정 및 질의, 패킷필터링)이 종료되었음을 동기적, 또는 비 동기적으로 입출력 서비스에 알림.
IPv6ReceiveIndicate	데이타가 망 접속 인터페이스를 통해 도착했음을 비 동기적으로 IPv6 드라이버에 통고해 줌.

가 초기화 될 때 Lower-Edge 함수들은 NDIS 인터페이스에 등록되어 커널 모드로 동작할 수 있게 된다.

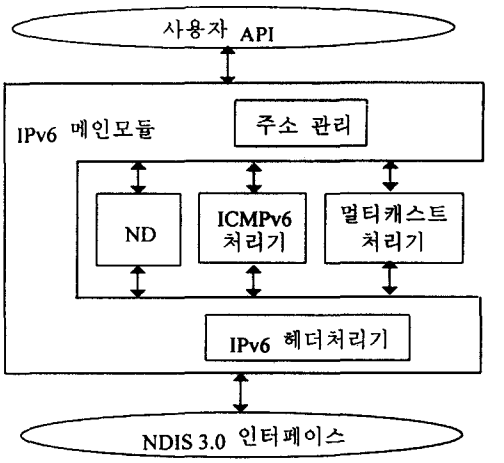
또한 상위 사용자 응용 프로그램이 IPv6 드라이버 모듈을 이용해 데이타를 송 수신할 수 있도록 Windows NT 입출력 서비스 인터페이스에서 제공하는 시스템 함수들을 이용하여 다이나믹 링크 라이브러리(DLL) 형태로 사용자 API를 구현 하였다. Windows NT의 트랜스포트 드라이버는 자신의 디스패치 함수에서 Windows가 지원하는 표준 인터페이스인 TDI(Transport Driver Interface)와 접속되어 사용자가 TDI 함수를 호출함으로써 트랜스포트 드라이버를 이용할 수 있도록 하고 있으나 자체적으로 개발된 IPv6 프로토콜 드라이버를 TDI와 접속시키는 방법이 충분히 공개되어 있지 않아 본 구현에서는 사용자 API를 별도로 정의하고 구현하였다. 그러나 다양한 응용 개발자 및 이용자들이 손쉽게 이용할 수 있도록 하기 위해서는 TDI를 통해 프로토콜 드라이버가 호출되도록 추후 보완될 필요가 있다. 구현된 사용자 API 함수의 종류 및 기능은 〈표 4〉와 같으며, 이들 함수를 이용한 응용 프로그램을 작성하여 구현된 IPv6 프로토콜 드라이버의 기능 및 동작을 시험하였다.

〈표 4〉 사용자 API 함수
〈Table 4〉 User API Functions

사용자 API 함수	기능
IPv6StartUp	IPv6 드라이버의 사용을 위한 다이나믹 링크 서비스를 개시.
IPv6SendData	목적지 주소로 데이타의 전송 요구.
IPv6ReceiveData	망 접속 인터페이스를 통해 수신된 데이타를 사용자에게 넘겨 줌.
IPv6AsciiToAddr6	ASCII 형태의 문자열을 IPv6 주소 형태로 변환.
IPv6SetAddr6	IPv6 주소를 IPv6 드라이버의 주소 관리 모듈에 넘겨 줌.
IPv6SetMulticast	멀티캐스트 모드로 동작할 수 있도록 IPv6 드라이버에게 호스트 그룹으로의 가입, 탈퇴를 지시.
IPv6DefaultRouter	디폴트 라우터의 IPv6 주소를 알려 줌.
IPv6NetworkPrefix	망 주소의 Prefix 정보를 넘겨 줌.

4. 인터넷 망 프로토콜(IPv6) 기능 구현

본 연구에서는 IPv6 데이타그램 처리를 위한 기본적인 기능과 에러 메세지 및 관리용 메세지를 처리해 주는 ICMP 기능, 이웃 노드에 대한 정보를 탐색해 주는 ND(Neighbor Discovery)기능, 그리고 멀티캐스트 데이타의 전송 및 수신을 위한 멀티캐스트 처리기 기능을 IPv6 프로토콜 드라이버로써 구현하였다. 구현된 IPv6 프로토콜 드라이버의 기능 모듈 구조는 (그림 5)와 같으며, 구현 환경으로는 Windows NT 3.5 상에서 Visual C^{++} 2.0 및 Windows NT DDK를 이용하여 커널에 그 기능을 구현하였다.



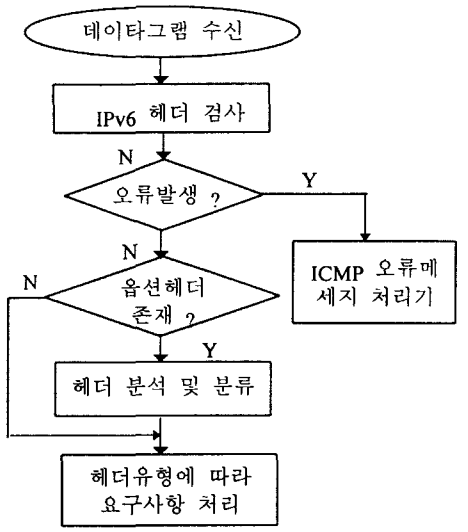
(그림 5) IPv6 프로토콜 드라이버의 기능 모듈 구조
(Fig. 5) Functional architecture of IPv6 protocol driver

4.1 IPv6 기본기능 처리 모듈

IPv6 메인 모듈은 IPv6 데이타그램을 상위 및 하위 계층의 요구에 따라 송 수신하는 기능과, IPv6 패킷의 송 수신과 관련된 제반 제어 및 관리 기능을 요구 대로 처리하도록 관련 기능 모듈들을 총괄한다.

데이타 송신의 경우, 먼저 목적지 주소 정보가 주소 관리 모듈로 전달되고, 주소 관리 모듈은 ND 모듈을 호출하여 Next-Hop의 링크 주소를 결정하며, 그 이후에 이 주소 정보를 이용하여 IPv6 데이타그램을 송신하게 된다. 데이타 수신의 경우, 하위 NDIS 인터페이스로부터 수신된 IPv6 데이타그램을 전달 받게

되면 헤더 처리기 모듈을 호출하여 먼저 기본 헤더 및 옵션 헤더를 검사하고, 이때 오류가 발견되면 ICMP 모듈을 호출하여 오류 메세지를 전송한다. 만일 수신된 헤더에 오류가 없으면 헤더 처리기에서 헤더 정보를 해석한 다음, 각 헤더 유형에 따라 헤더 처리 기능을 수행하게 된다. IPv6 데이타그램 수신 시 그 처리 과정을 간략하게 살펴 보면 (그림 6)과 같다.



(그림 6) 수신 데이타그램 처리과정
(Fig. 6) Procedure for processing of received datagram

주소 관리 모듈은 IPv6 프로토콜이 동작되는 자신의 주소와 데이타가 전달될 단일 목적지 주소, 그리고 멀티캐스트 데이타 전송에 사용되는 멀티캐스트 주소를 관리한다. 또한 이웃 탐색 기능을 수행할 때 요구되는 주소 정보들을 관리한다. 이를 위해 주소 리스트를 정의하였고 다음과 같은 구조체를 사용하여 구현하였다. 이 구조체에서 Addr6는 IPv6 주소로서 주로 망으로 부터 받은 데이타 들의 목적지 주소를 확인할 때 이용한다. Set 영역은 지정된 주소의 설정 방법을 나타내며 AUTO와 MANUAL로 나누어 정의하였다. AUTO는 사용자 접근이 불가능하며 IPv6 프로토콜 드라이버 초기화 시에 설정되고, MANUAL은 사용자에 의해 주소 설정이 이루어지며 사용자 요청에 따라 삽입 및 삭제가 자유롭다.

```
Typedef struct_Add6_Set {
  in_addr6 Addr6;
  int Set;
} Addr6_Set;
Addr6_Set Add6_List[MAX_ADDR6];
```

4.2 ND(Neighbor Discovery) 기능 모듈

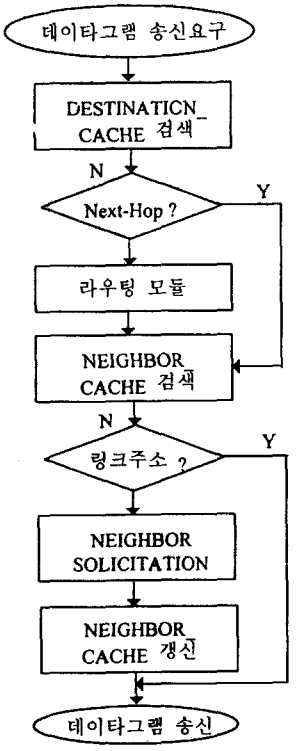
ND 모듈은 목적지 주소로 사용자가 요청한 데이타그램을 전송하기 위해 Next-Hop에 대한 링크 주소를 결정하는 기능을 수행한다. Next-Hop을 결정하기 위해 먼저, 이전의 데이타그램 전송 요구에 따라 이웃 탐색 기능을 수행하여 목적지 주소에 대한 Next-Hop 정보를 찾아 저장해 놓은 것이 있는지 DESTINATION-CACHE를 검색한다. DESTINATION-CACHE에 Next-Hop에 대한 정보가 없는 경우에는 라우팅 모듈을 통해 Next-Hop 정보를 구한다. 일단 Next-Hop이 결정되면 이 주소에 대한 링크 주소를 얻기 위해 NEIGHBOR-CACHE를 검색하나, NEIGHBOR-CACHE에 링크 주소가 등록되어 있지 않은 경우에는 링크 주소 정보를 얻기 위해 Neighbor Solicitation(NS)-메세지를 멀티캐스트 주소를 이용하여 전송하게 된다. 이 때 NS-메세지의 목적지 주소 영역은 Solicited-node 멀티캐스트 주소가 된다. 이 NS-메세지를 받은 호스트는 자신의 이더넷 주소를 Neighbor Advertisement(NA) 메세지에 담아 원래 송신 호스트에 전송하게 되며, NA-메세지가 도착하면 호스트는 다시 ND 모듈을 호출하여 NA-메세지로 부터 링크 주소를 찾아낸 후 NEIGHBOR_CACHE에 이 정보를 저장한다. 이는 추후에 같은 목적지 주소로 전송될 데이타그램에 대해 불필요한 링크 결정과정이 수행되지 않게 하기 위함이다. 이러한 과정을 거쳐 Next-Hop에 대한 링크 주소가 결정되면 사용자가 요청한 데이타그램을 결정된 링크 주소에 따라 전송하게 된다. 링크 주소 결정시 중요한 역할을 하는 NEIGHBOR_CACHE의 구조체는 다음과 같이 구현되었다.

```
Typedef struct_Neighbor_Cache {
  in_addr6 Addr6;
  eth_addr link_addr;
  int Status;
```

} Neighbor_Cache;
Neighbor_Cache Neighbor_Table[MAX_NEIGHBOR];

이 구조체는 Next-Hop의 IPv6 주소와 대응하는 링크 주소를 쌍으로 갖고 있고 이 데이타의 상태를 알려 주는 STATUS를 포함하고 있다. 정의된 STATUS에는 먼저 링크 주소가 유효하며 이웃 노드 인터페이스가 정상적으로 동작함을 나타내는 REACHABLE, 링크 주소를 얻기 위해 NS-메세지를 보냈으나 NA-메세지를 기다리고 있는 상태를 나타내는 INCOMPLETE, 그리고 이웃 노드 인터페이스가 다운된 것으로 간주되는 상태를 나타내는 UNREACH가 있다. ND 모듈에 의해 링크 주소가 결정되어 데이타그램이 송신되는 과정은 (그림 7)과 같다.

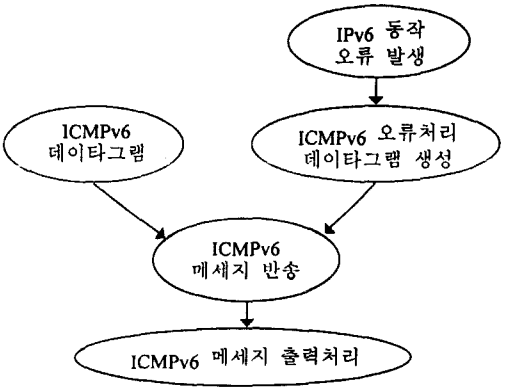
또한 최종 목적지가 국부 망에 존재하지 않고 원거리에 놓여 있다면 호스트는 목적지로 데이타를 전송하기 위해 라우팅 경로를 결정해야 한다. 라우팅 경로를 설정하는 방법으로는 먼저, 사용자가 디폴트 라우터의 주소를 지정해 주는 방식이 있고, 다른 방법으로는 이웃 탐색 프로토콜을 이용해 라우팅 경로를 설정하는 방식이 있다. 자동적으로 라우팅 경로를 설정하는 방법은 앞에서 설명한 Next-Hop의 링크 주소를 찾는 방법과 유사하게 Router Solicitation(RS) 메세지와 Router Advertisement(RA) 메세지를 이용하여 수행된다. 본 구현에서는 호스트에서 동작하는 IPv6 기능 만을 구현 범위로 하였으므로 라우터 정보와 IPv6 주소의 Prefix 정보를 사용자가 직접 사용자 API 함수를 이용해 IPv6 드라이버 모듈에 전달하도록 기능을 구현 하였다.



(그림 7) 링크 주소 결정 및 데이타그램 전송과정
(Fig. 7) Procedure of Link address determination and Datagram send

4.3 IPv6 오류 및 제어 메세지 처리 기능 모듈

ICMPv6는 IPv6 노드간 오류 처리와 시스템 관리상 요구되는 제어 기능을 수행하는 프로토콜로 모든 IPv6 노드 구현 시 반드시 요구되는 중요한 기능이다. 본 연구에서는 IPv6 데이타그램 중 IPv6 헤더의 다음 헤더 영역 값이 58을 갖는 ICMPv6 메세지를 수신하였을 경우 이에 대한 응답 및 국부적 처리를 수행하는 기능과, 또한 IPv6 데이타그램의 전달 과정에서 오류 상태가 발생했을 때 이에 대한 적절한 처리를 위해 ICMPv6 오류처리 메세지를 생성해서 오류를 유발시킨 호스트로 전송하는 기능을 구현 하였다. 구현된 ICMPv6 기능모듈의 구성은 (그림 8)과 같다. 여기



(그림 8) ICMPv6 메세지 처리
(Fig. 8) Processing of ICMPv6 message

서 ICMPv6 반송 모듈은 ICMPv6 메세지에 대한 응답, 또는 ICMPv6 오류 메세지를 오류를 유발시킨 호스트로 전송하는 기능을 수행하며, 이 모듈의 주요 동작으로는 발신지와 목적지 주소의 치환, ICMPv6 메세지에 대한 검사합의 계산 등을 수행한다.

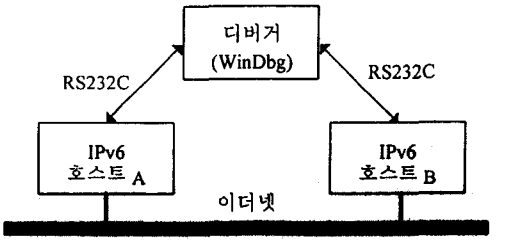
4.4 멀티캐스트 처리기

ICMPv6 규격에서 규정하고 있는 멀티캐스트 데이타그램 전달 기능을 지원하기 위해 멀티캐스트 처리기를 IGMP 모듈과 망 계층 레벨 필터링 모듈, 그리고 IPv6 멀티캐스트 서비스 인터페이스 모듈로 나누어 구현하였다. 가장 중요한 기능은 멀티캐스트 주소 정보를 이용하여 멀티캐스트 그룹에 등록 및 탈퇴를 가능하게 하는 IGMP 기능이다. 그룹 관리 기능은 사용자 API로 정의한 IPv6SetMulticast 함수를 이용하여 멀티캐스트 처리기 내 IPv6 멀티캐스트 서비스 인터페이스로 전달되도록 구현하였고, IGMP 기능 모듈은 사용자 요구를 받아 그룹 멤버쉽 제어 메세지를 생성하여 이웃 라우터로 전달하도록 하였다. 또한 망 계층 레벨 필터링 모듈은 IPv6 멀티캐스트 주소와 이더넷이 제공하는 멀티캐스트 주소 사이의 변환을 통해 IP 계층에 멀티캐스트 서비스를 제공하며, 이들 주소 정보들을 관리하는 기능을 수행한다.

5. 시험 및 결과 고찰

Windows NT 커널 내부에 IPv6 프로토콜의 구현 작업을 원활하게 하기 위해 (그림 9)와 같은 디버깅 환경을 구축하였다. IPv6 프로토콜 드라이버 모듈을 설치한 두 호스트를 이더넷에 접속 시키고, 이들 두 호스트의 동작 상태를 디버깅 하기 위해 디버거를 RS232C로 연결하였다. 디버깅 프로그램으로는 마이크로소프트사가 제공하는 SDK(S/W Development Kit) WinDbg와 DDK(Device Development Kit)가 제공하는 디버깅 함수를 사용하였다. WinDbg를 사용함으로써 두 호스트 상에서 동작하는 IPv6 프로토콜 드라이버의 동작 상태와 메모리 상태의 확인 및 디버깅 작업이 가능하였다.

Windows NT 커널에 구현된 IPv6 프로토콜 드라이버의 동작을 확인하기 위해 DLL 형태로 구현된 사용자 API 함수를 이용하여 네트워크 진단 프로그램인



(그림 9) 디버깅 환경
(Fig. 9) Debugging environments

ping 프로그램과 간단한 텍스트 전송 응용 프로그램을 구현해 기능 시험을 수행하였다. 구현된 IPv6 프로토콜 드라이버의 시험을 위해서는 먼저 Windows NT에 드라이버를 설치해야 한다. 이를 위해 제어판에서 NCPA를 실행시키며 스크립트 화일을 이용하여 프로토콜 드라이버를 설치하면 자동적으로 이더넷 드라이버와의 바인딩 작업을 시켜 준다. 물론 사용자가 임의로 설치된 IPv6 드라이버를 제거할 수도 있으며, 이더넷 드라이버와의 바인딩을 무효화 시킬 수도 있다. 설치시 요구되는 화일은 IPv6 프로토콜 드라이버 화일과 IPv6 드라이버를 사용하기 위해 제공되는 다이나믹 링크 라이브러리 화일이다. 이 과정이 끝나면 안전하게 응용 프로그램을 실행시킬 수 있다.

먼저, 구현된 ping 프로그램을 통해 IPv6 프로토콜에서 중요한 역할을 수행하는 ICMPv6 기능을 시험할 수 있었다. ping 프로그램은 사용자에게 진단을 하고자 하는 다른 호스트의 IPv6 주소를 받아서 일정한 간격으로 다른 호스트에 ICMPv6 에코 요청 메세지를 전송하게 되며, 이 에코 요청 메세지를 받은 호스트는 응답 메세지를 생성해 ICMPv6 메세지를 반송하게 된다. 에코 요청 메세지를 송신한 호스트는 수신된 응답 메세지를 검사하여 이상이 없으면 진단한 호스트가 정상 동작함을 알리는 메세지를 콘솔에 출력하게 된다.

그리고 텍스트 전송 응용 프로그램을 통해 IPv6 프로토콜 드라이버가 정상적으로 이웃 탐색 기능을 수행하여 IPv6 주소로부터 목적지 호스트의 물리적 망 접속 카드의 주소를 찾고, 사용자 데이타가 IPv6 데이타그램으로 패킷화 되어 목적지로 제대로 전송되며, IPv6 헤더 처리 기능이 정상적으로 동작됨을 확인 하

였다. 또한 마지막으로 이더넷 상에 호스트 3대를 접속하여 멀티캐스트 주소를 이용하여 텍스트 전송 응용 프로그램을 실행시켜 봄으로써 데이타그램의 멀티캐스트 전송기능이 제대로 동작하는 것을 확인하였다. 본 시험에서는 멀티캐스트 주소로써 링크 내에서만 유효한 링크-로컬 주소를 사용하였으므로 IPv6 라우터가 없는 환경에서 구현된 IPv6 호스트 기능을 시험할 수 있었다. 또한 기존 IPv4 인터넷 망을 통해 IPv6 프로토콜을 탑재한 타 시스템과 접속할 수 있도록 해 주는 터널링 기술이 본 연구 범위로 구현되지 않았으므로 LAN 환경에서 구현된 호스트 기능이 시험되었고, 6-Bone을 통한 타 벤더 제품과의 상호 운용성 시험은 이루어지지 않았다.

6. 결 론

인터넷 주소 공간의 부족에 대한 대비와 다양한 형태의 멀티미디어 응용을 지원할 수 있도록 기존 인터넷 구조 및 세부 프로토콜에 대한 개선 요구가 높아지면서 인터넷 표준기술을 개발하는 IETF에서는 이러한 요구사항을 만족하는 새로운 차세대 인터넷 망 계층 프로토콜로써 IPv6를 개발하였다. 현재, 대부분의 컴퓨터 업체와 네트워크 장비 제조 업체들은 IPv6를 구현한 호스트 및 라우터 제품을 이미 개발 완료하였고, 6-Bone이라는 가상의 IPv6 망을 구축하여 이들 제품간 상호운용성 시험을 수행하는 단계에 있다. 따라서 이제는 터널링 기술을 통해 기존 IPv4와 함께 IPv6가 실제 인터넷 환경에서 동시에 사용되는 것이 가능해 졌으며, 향후에는 인터넷이 점차 IPv6 망으로 진화되어 갈 것으로 예상된다.

이 논문에서는 최근 실제 인터넷 환경에 적용될 수 있는 단계까지 이른 IPv6 프로토콜의 호스트 기능을 Windows NT의 커널 내부에 프로토콜 드라이버로 구현한 내용을 기술하였다. 이 연구는 Windows NT에 기반을 둔 차세대 인터넷 멀티미디어 응용 플랫폼을 구축하기 위한 일환으로 수행되었으며, IPv6 프로토콜을 사용자 모드가 아닌 커널 모드에서 구현함으로써 향상된 성능을 제공하며, 다이나믹 링크 라이브러리 형태로 사용자 인터페이스를 제공하므로 응용 프로그램 개발자들이 손쉽게 사용할 수 있도록 하였다. IPv6 호스트 기능의 구현사례를 모아 놓은 Web 홈페이지에는 Dassault Electronique Group 만이 Windows NT 상에서 IPv6 호스트 기능을 구현한 것으로 소개하고 있으므로 본 연구를 통해 개발된 구현 결과는 Windows NT 상에서의 프로토콜 구현기술 확보 측면에서 의미 있는 성과를 거두었다고 하겠다.

이번 연구에서는 IPv6 호스트 기능만을 다루었으므로 다음 단계에서는 IPv6 라우터 기능에 대한 연구가 요구된다. 또한 이번 구현 범위에서 향후 고려 사항으로 남겨 둔 IPv6 데이타그램의 인증 및 보안기능, 주소 자동 설정 기능, 그리고 라우터와 연계한 라우터 탐색 기능 및 멀티캐스트 기능 등이 계속해서 연구될 필요가 있다.

참 고 문 헌

[1] J. Postel, "Internet Protocol," RFC 791, September 1981.

[2] Stephen A. Thomas, "IPng and the TCP/IP Protocols," John Wiley Computer Publishing, 1996.

[3] S. Bradner, A. Mankin, "The Recommendation for the IP Next Generation Protocol," RFC 1752, January 1995.

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

[5] R. Hinden, S. Deering, "IPv6 Addressing Architecture," RFC 1884, December 1996.

[6] A. Conta, S. Deering, "Internet Control Message Protocol(ICMPv6) for the IPv6 Specification," RFC 1885, December 1995.

[7] T. Narten, E. Nordmark, W. Simpson, "Neighbor Discovery for the IPv6," RFC 1970, August 1996.

[8] R. Atkinson, "Security Architecture for the Internet Protocol," RFC 1825, August 1995.

[9] R. Atkinson, "IP Authentication Header," RFC 1826, August 1995.

[10] R. Atkinson, "IP Encapsulating Security Payload(ESP)," RFC 1825, August 1995.

[11] S. Thomson, T Narten, "IPv6 Stateless Address

Autoconfiguration," RFC 1971, August 1996.

[12] R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Router," RFC 1993, April 1996.

[13] 6Bone Home Page, http://www. 6bone. net/

[14] IPng Implementations, http://playground.sun.com/pub/ipng/html/ipng-implementations.2.html

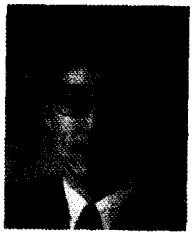
[15] 이왕봉 외, "차세대 인터넷 프로토콜 IPv6의 구현," 한국통신학회 추계학술대회 논문집, 1996.

[16] J. Postel, "Internet Control Message Protocol," RFC 792, September 1981.

[17] S. Deering, "Host Extension for IP Multicasting," RFC 1112, August 1989.

[18] Helen Custer, "Inside Windows NT," Microsoft Press, 1992.

[19] Mark Andrews, "C++ Windows NT Programming," M&T Book, 1994.



강 신 각

1984년 충남대학교 전자공학과 졸업(학사)

1987년 충남대학교 대학원 전자공학과 졸업(공학석사)

1984년~현재 한국전자통신연구원 책임연구원

관심분야:컴퓨터 네트워크, 멀티미디어 통신, 정보보호

김 대 영

1975년 서울대학교 전자공학과 졸업(학사)

1977년 한국과학기술원 전기 및 전자공학과 졸업(공학석사)

1983년 한국과학기술원 전기 및 전자공학과 졸업(공학박사)

1979년~1980년 독일 Aachen 공대, Hannover 공대 연구원

1987년~1988년 미국 University of California, Davis 객원연구원

1983년~1991년 충남대학교 공과대학 전자공학과 교수

1992년~현재 충남대학교 공과대학 정보통신공학과 교수

관심분야:전송부호화 및 모뎀, 컴퓨터 네트워크, 고속통신, 멀티미디어 등