IPv4/IPv6 변환 프로토콜의 설계 및 구현

박 석 천[†] · 이 광 배[††]

요 약

인터넷이 전세계적으로 급속히 확대되면서 IPv4의 32Bit의 주소공간의 얼마가지 않아 고갈될 것으로 예상이 된다. 따라서 주소 고갈 문제를 해결하기 위해 IPv6의 효과적인 전이 방안으로의 기술 전이가 필연적으로 이루어 질 것이다. 인터넷의 적용범위와 거대한 용량으로 인해 IPv6로의 동시 전환은 현실적으로 불가능하지만 전체적으로 빠른 변환 개념을 도입하기 위해서 IPv4와 IPv6의 공존은 특별하고 실용적으로 배열이 되어야 한다. 따라서 본 논문에서는 IP 변환 프로토콜을 도입하여 IPv4에서 IPv6로의 보다 용이한 변환을 제안하고 있으며, IPv4 및 IPv6로 동시 전환을 할 수 있도록 IPv4/IPv6 변환 프로토콜을 설계하고 구현하였다. IPv4/IPv6 변환 프로토콜은 C로 구현하였으며 동작 확인 테스트를 위해 오류 검출능력이 가장 뛰어나고 ISO에 의해서 권고된 국부 시험 방법으로 테스트 시나리오를 작성하고 테스트하여 설계 및 구현한 변환 프로토콜이 정상적으로 동작함을 확인하였다.

Design and Implementaion of IPv4/IPv6 Translation Protocol

Seok Cheon Park[†] · Kwang Bae Lee[††]

ABSTRACT

It is well known that, in the near future, the lifetime of the IPv4 address space will be limited and available 32-bit IP network addresses will not be left any more. In order to solve such IPv4 address space problem in an effective way, the transition to the new version using IPv6 architecture is inevitably required. At present, it is impossible to convert IPv4 into IPv6 at a time, since the coverage and the size of today's Internet is too huge. Therefore, the coexistence of both IPv4 and IPv6 must be arranged in a special and practical fashion for rapid conversion on the whole. IP protocol translation has been proposed to ease the translation of the Internet from IPv4 to IPv6. This paper presents the design and implementation of a transparent transition service that translates packet header as they cross between IPv4 and IPv6 networks. IPv4/IPv6 Translation Protocol is written in c source code and is tested by the local test recommended by ISO, which has the most excellent error detection function. The test was processed with a test scenario and it was found that the results were successful.

키워드 : IPv4/IPv6, 터널링(Tunneling), 변환프로토콜(Translation Protocol)

1. 서 론

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, 인증과 데이터 보호기능 제공 등이 있으며, 빠른 속도로 발전하는 인터넷을 지원하기 위해서는 이와 같이 다양한 특징을 지닌 IPv6 환경으로의 전환이 불가피하다고 할 수 있다[3]. 그러나 현재 사용되고 있는 IPv4 환경에서 차세대 인터넷 프로토콜인 IPv6의 환경으로의 동시 전환은 현실적으로 불가능하기 때문에 IPv4에서 IPv6로의 효과적이고 점진적인 전이 방안이 필요하다. IPv4에서 IPv6로의 전이 방안으로는 IPv4/IPv6 이중 구조의 노드를 구축하거나 IPv4 및 IPv6 단일 노드를 위한 터널링(tunneling) 및 IP 프로토콜의 변환 등 다양한 방안이 제안, 연구되고 있다[1-3].

IPv6의 표준화가 이루어지면서 여러 가지 다양한 플랫폼에서 IPv6 스택을 구현하는 작업이 이루어져왔다. IPv6 구현 코드는 크게 호스트 부분과 라우터 부분으로 나누어 볼 수 있다. 호스트 부분으로는 각 운영체제별로 다양한 구현 코드가 존재할 수 있으며 현재 Linux나 FreeBSD, NetBSD 등의 BSD 계열 및 Apple, SGI, AIX 등 다양한 운영체제에서 구현 및 코드의 개정이 이루어지고 있다. 라우터 부분으로는 3COM, CISCO, BAY Network 등 라우터 장비 개발 회사에서 구현 코드의 개발이 진행 중에 있으며, IPv6로의 전이 방안의 경우 현재 IETF를 중심으로 NAT-PT(Network Address Translation-Protocol Translation), SIIT(Stateless IP/ICMP Translation), AIIH(Assignment of IPv4 Global Address to IPv6 Hosts), DTI(Dynamic Tunnel Interface),

† 종신회원 : 경원대학교 컴퓨터공학과 교수
†† 정 회 원 : 명지대학교 전자공학과 교수
논문접수 : 2001년 2월 6일, 심사완료 : 2001년 10월 5일

BIS(Bump In the Stack) 등 여러 방식이 제안 및 연구되고 있다.

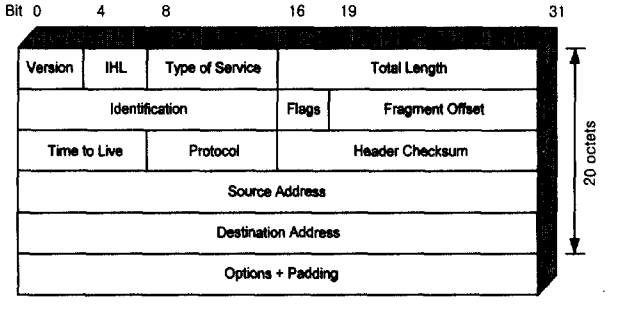
현재 국내 IPv6와 IPv4와의 연동 기술 연구로는 터널링 방식으로 쉽게 테스트 베드를 구축하는 방법이 이루어지고 있으나 순수 IPv6 테스트 베드는 미국을 제외하고는 현재 구성되지 못하고 있는 것이 사실이다. 현재 우리나라를 비롯해 여러 국가들이 참여하는 6Bone 이라는 백본망이 구성되어 이 망을 이용한 많은 테스트들이 이루어지고 있다[4].

본 논문에서는 IPv4 및 IPv6 단일 구조로 구성된 호스트들 간의 원활한 통신을 위한 IPv4/IPv6 변환 프로토콜을 설계하고 구현하였으며, 이를 위하여 인터넷 표준화기구인 IETF의 표준화 동향 및 관련 표준 문서를 연구·분석하고 그 내용을 토대로 IPv4/IPv6 변환 프로토콜을 설계하였다. 그리고 설계한 IPv4/IPv6 변환 프로토콜의 구현을 위해 실제 프로그램에서 사용될 두 프로토콜의 IP 패킷 헤더와 ICMP 메시지에 대한 변환 값과 변환 동작의 절차를 정의하였으며, 이를 토대로 IPv4/IPv6 변환 프로토콜을 구현하고 국부 시험 방법을 통하여 테스트하여 설계 및 구현한 프로토콜이 정상적으로 동작함을 확인하였다.

2. IPv4/IPv6 프로토콜

2.1 IPv4(Internet Protocol version4) 구조

(그림 1)은 IPv4 패킷의 구조를 보여준 것이며, 정상적인 IP 헤더의 길이는 20바이트이다. (그림 1)에서 최상위 비트(most significant bit)는 0으로 표시되어 있으며 최하위 비트(least significant bit)는 31로 표시되어 있다.



(그림 1) IPv4 헤더 구조

IP 패킷은 (그림 1)과 같은 구조를 가진 IP 헤더와 데이터 부분으로 구성된다. IP 헤더를 구성하는 필드들의 정보를 통해 인터넷을 통한 종단간 통신이 이루어지게 된다.

IPv4 헤더 내의 필드에서 IHL(Internet Header Length)은 IP 헤더의 길이를 나타내며 TOS(Type of Service)는 송신중인 IP 서비스의 품질을 나타내며, 상위 프로토콜에 의해 지정된다. Identification, Flags, Fragment Offset은 물리네트워크의 제한 등에 따라 IP 패킷을 분할하여 보낼 필요가 있을 때에 사용하며, 데이터가 원래 속하는 IP 패킷의 식별(ID), 분할된 마지막 데이터인지에 대한 식별(flags)과 원래의 데이터의 위치를 표시한다. 그리고 Source Address는 32비트의 송신측 IP 주소, Destination Address는 최종 목적지인 수신측의 32비트 IP 주소를 의미한다.

2.1.1 IPv4 주소 체계

32비트의 IP 주소는 [네트워크 주소, 호스트 주소]의 조합으로 구성된다. IP 주소의 네트워크 주소는 네트워크 사이에 패킷의 경로를 설정하는데 사용되며 호스트 주소는 특정한 호스트로 패킷을 전송하는데 사용되며 클래스에 따라 네트워크 주소의 길이와 호스트 주소의 길이가 다르다.

32비트의 주소는 192.9.61.113과 같이 각각 1바이트(8비트)씩 4개의 10진수로 각 숫자 사이는 '.'으로 분리하여 표현한다. 각각의 주소 클래스는 주소의 첫 번째 몇 비트로 구별된다. A 클래스는 '0'으로 시작되며, B 클래스는 '10', C 클래스는 '110', 확장 클래스(extended address class)는 '111'로 시작된다.

2.1.2 ICMPv4

IP는 비연결형 네트워크 계층의 프로토콜이며, 패킷을 중계하기 위해 최대한 노력하긴 하지만 패킷이 확실히 전송된다는 보장은 없다. 따라서 라우터나 노드(호스트) 등에서 이상이 발생하여 패킷이 목적지까지 도달하지 못하는 경우도 있다. 이러한 때, 송신측에 대해 상황을 알려 줄 필요가 있으며, 이를 위해 ICMP라는 프로토콜을 사용하여 통지를 한다. 또한 송신측의 노드가 통신을 하기 전에 목적지 노드의 존재 여부를 진단하는 일도 ICMP의 역할이다.

2.2 IPv6(Internet Protocol version6) 구조

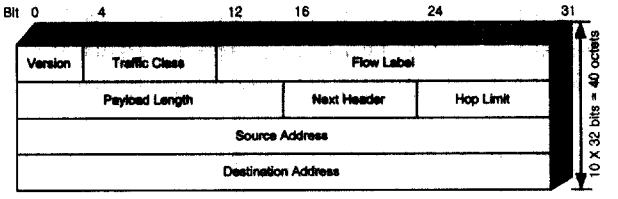
IPv6(Internet Protocol version 6)는 현재의 IPv4를 대체하도록 설계된 차세대 인터넷 프로토콜로서 ATM과 같은 고속망뿐만 아니라 무선망과 같은 저속의 망에서도 효율적으로 동작하도록 설계되었으며 향후 인터넷에서의 새로운 기능을 수행할 수 있도록 개발되었다[1].

IPv6는 IPv4로부터 진화적으로 개발된 것으로 기존의 IPv4의 기능 중 이용되는 것은 계속 이용하고 이용되지 않는 것은 제거되는 방향으로 설계되었다. IPv4에 비해 구별되는 IPv6의 변화는 다음과 같다[5, 6].

- 라우팅과 주소지정 기능의 확장
- 헤더 형식의 단순화
- 옵션기능의 개선
- QoS(Quality of Service) 제어 기능
- 인증(authentication) 및 보안 기능

2.2.1 IPv6 헤더

IPv6의 헤더는 IPv4 헤더의 몇몇 필드가 생략되거나, 옵션으로 추가됨으로써 더 간단해 졌다. 따라서 일반적인 경우 패킷의 처리비용이 줄어들었고, 주소 길이가 IPv4에 비해 크게 늘어났음에도 불구하고 헤더의 대역폭 비용을 제한할 수 있게 되었다. (그림 2)는 IPv6의 헤더 구조를 나타낸 것이다[5].



(그림 2) IPv6 헤더 구조

IPv6 헤더 내의 Flow Label은 송신자가 IPv6 라우터에서 특별하게 처리되기를 원하는 특정 IP 패킷 흐름을 표시하며, Next Header는 IPv6 헤더 다음에 오는 헤더의 형태를 나타내고 사용되는 값은 IPv4 헤더의 프로토콜 필드의 값과 동일하다. 그리고 Source Address는 128비트의 송신측 IP 주소이고 Destination Address는 최종 목적지인 수신측의 128비트 IP 주소이다. IPv6 확장 헤더(extension header)는 IPv6 헤더와 상위 계층 헤더 사이에 선택적으로 붙게 된다. 즉 IPv6 패킷은 이러한 확장 헤더를 포함함으로써 여러 가지 정보와 서비스를 제공할 수 있으며 IPv6의 각 헤더들은 바로 다음에 어떤 종류의 헤더가 있는지 명시해주는 Next Header 필드에 표시된다[5].

2.2.2 IPv6 주소 체계

IPv6에서 할당할 수 있는 주소 공간이 넓어짐에 따라 다양한 주소 할당 방식을 사용할 수 있으며, 여러 종류의 주소들을 포함할 수 있게 되었다. IPv6 주소에서는 주소의 종류 및 서브넷(subnet)을 판별할 때 사용하는 prefix와, 네트워크에 연결되어 있는 각 인터페이스들을 구별해 주는 interface ID로 구성된다. IPv6 망에서 주소를 할당하는 기본 단위는 호스트나 라우터가 아닌 인터페이스 중심이다. 즉 같은 호스트에 동일 링크와 연결되는 여러 개의 인터페이스들이 있을 경우 각 인터페이스마다 다른 주소를 할당받게 된다. 그러나 보통 한 호스트에 하나의 인터페이스로 링크에 연결되므로 interface ID로 호스트를 구분하게 된다. 또한 기존의 IPv4 주소체계에서의 unicast 및 multicast 주소와 다른 새로운 anycast라는 주소의 개념을 도입하였다. 한 라우터가 anycast 주소를 목적지 주소로 가지는 패킷을 수신했을 경우, 라우팅 프로토콜에 따라 같은 anycast 주소를 가지는 여러 노드들 중 가장 가까운 노드로만 패킷을 보내는 방식으로 이용하며 IPv6 주소는 일반적으로 4자리의 16진수 8개를 ':'로 구분해 나열하는 방법으로 표시한다[6].

IPv6 주소 유형 중의 하나인 unicast 주소에는 "IPv6 Addresses with Embedded IPv4 Addresses"라는 형태의 주소가 존재하며, 이 주소는 IPv4 주소가 내장된 IPv6 주소를 나타낸다. IPv4가 내장된 주소의 형태에는 두 가지의 종류가 있다. 첫 번째 형태는 "IPv4-compatible IPv6 address"라는 형태로서 IPv4에서 IPv6로의 전이방안 중 터널링(tunneling) 기법에 사용된다. 두 번째 형태는 "IPv4-mapped IPv6 address"라는 형태로서 IPv6를 지원하지 않는 IPv4 노드를 위해 사용되는 주소 형태이다. 본 논문에서는 IP 헤더를 변환할 때 "IPv4-mapped IPv6 address" 형태의 주소를 이용하였다.

2.2.3 ICMPv6(Internet Control Message Protocol version 6)

ICMPv6는 현재 사용하고 있는 ICMPv4를 수정 및 보완하여 사용되는 프로토콜이며 IPv6 Next Header 필드 값으로 "58"을 갖는다. ICMPv6는 패킷 처리 중에 발생하는 오류를 보고, 진단하는 기능을 수행한다[7]. ICMPv6 헤더의 구조는 ICMPv4의 구조와 마찬가지로 Type, Code 및 Checksum 필드를 가지고 있으며 그 외의 필드들은 메시지에 따라 다른 형태로 구성된다. ICMPv6 메시지는 Error 메시지와 Informational 메시지로 구분되며, Error 메시지는 메시지의 Type 필드 값의 상위 비트에 "0"을 가짐으로서 인식된다. 따라서 Error 메시지는 0부터 127까지(00000000~01111111)의 메시지 형태를 가지며 Informational 메시지는 128부터 255까지(10000000~11111111)의 메시지 형태를 가진다.

3. IPv4/IPv6 변환 프로토콜의 설계

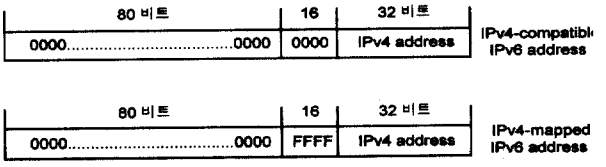
IPv4와 IPv6간의 상호 연동을 위해서는 IP 프로토콜 변환 외에 이중 스택 구조(dual stack)와 터널링(tunneling)이라는 방법이 있다. AIIH(Assignment of IPv4 Global Address to IPv6 Hosts) 방식이 이에 속하며, 이 중 스택 구조는 IP 계층에 IPv4와 IPv6 기능 모두를 설치함으로써 IPv4 호스트와 통신할 경우와 IPv6 호스트와 통신할 경우로 나누어 각각 IP 계층 내의 기능 분할을 통하여 통신을 수행한다. 즉 IPv6 스택을 갖춘 호스트 및 라우터들은 IPv4 스택도 함께 갖추게끔 하여 기존 IPv4 인터넷과의 통신을 가능하게 하고, IPv4 인터넷상에서의 IPv6를 통한 통신에서 보조 역할을 담당하게 한다. 이 방안은 모든 호스트의 IP 계층 프로토콜 스택에 대한 수정이 불가피하기 때문에 비용이 많이 드는 단점을 지니고 있다[1]. 터널링은 IPv6 패킷을 IPv6로 통신이 가능한 두 지점의 끝점을 IPv6 헤더로 캡슐화하여 실제로 IPv4 인터넷 환경 상에서 IPv6 패킷의 전달을 가능하게 하는 역할을 담당하며 DTI(Dynamic Tunnel Interface)와 같은 방식이 이에 속한다. 이는 IPv6 패킷 위에 IPv4의 헤더가 덧붙여지는 방식이기 때문에 모든 IPv4 라우팅 하부 구조 상에 IPv6 패킷의 터널링 기법을 추가해야 하므로 구현이 쉽지 않으며 동작 과정 역시 복잡한 단점이 있다[1]. 상기에서 언급한 방안에 비하여 본 논문에서 설계 및 구현되는 IP 프로토콜 변환은 변환 방식이 투명하고 변환 절차가 간단하기 때문에 비교적 구현이 용이하다는 장점을 지닌다. 또한 IP 변환 프로토콜을 IPv4 네트워크나 IPv6 네트워크의 라우터에 장착함으로써 기존의 IPv4 호스트나 향후 보급될 IPv6 호스트의 프로토콜 스택에 대한 수정이 없이 원활한 통신을 지원할 수 있다.

3.1 IPv4/IPv6 프로토콜의 변환

프로토콜 변환의 기본적인 동작은 원래의 패킷의 IP 헤더를 다른 버전의 IP 헤더로 교체하는 것이다. IP 헤더 내의 주

요 필드들에 대한 변환은 양 프로토콜의 대부분의 필드가 직접적으로 대응된다. 헤더 내의 몇몇 필드들의 경우 의미가 다르거나 크기가 다른 경우가 있으며 이를 제외하고는 IPv4 및 IPv6 헤더는 대부분 유사하다. 변환 프로토콜은 필드들을 직접 복사하거나 변환, 폐기 등의 기능을 수행하며 (그림 3)에 IPv4 및 IPv6 헤더 필드간의 관계를 나타내었다.

(그림 3)과 같이 대부분의 필드들은 특별한 제한 사항이 없이 변환된다. 예를 들면, IPv4로부터 IPv6로 변환될 때 IPv4 헤더의 TTL 필드는 직접 IPv6 헤더의 Hop Limit 필드로, Protocol 필드는 Next Header 필드로 복사된다. 또한 IPv4와 IPv6간의 어드레스 변환은 IPv6를 지원하지 않는 IPv4 노드를 위해 "IPv4-mapped IPv6 address" 방법으로 설계하였다. 본 논문에서 정의한 IPv4 주소가 내장된 IPv6 주소의 구조는 (그림 4)와 같다.



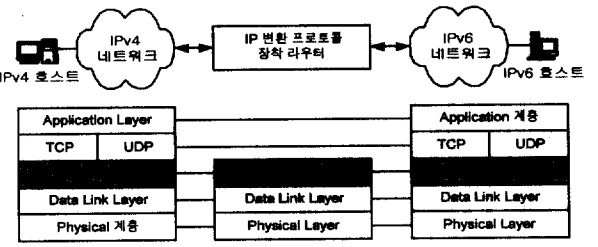
(그림 4) IPv4 주소가 내장된 IPv6 주소

IPv4와 IPv6간의 변환은 IP 패킷 크기를 변화시키며, 이는 IP 헤더 크기의 차이에서 기인한다. 그러나 IPv6를 지원하는 라우터는 IP 패킷을 분할(fragmentation)할 수 없기 때문에, 본 논문에서는 변환 프로토콜을 거치는 IP 패킷은 IPv6의 최대 전송 단위(MTU : Maximum Transmission Unit)인 1,280바이트보다 작은 크기를 갖도록 제한하였다. 만일 이러한 제한이 없이 용량이 큰 패킷을 전송하기 위해서는 네트워크의 종단간에 전송할 수 있는 패킷의 크기를 전송하기 전에 미리 파악할 수 있는 path MTU discovery라는 방법을 사용하게 되는데 이는 네트워크상의 장비들에 대한 비용부담 및 구현의 복잡성이 증가하므로 본 논문에서는 path MTU discovery는 고려하지 않는다[8]. IPv4/IPv6 프로토콜 변환은 ICMP의 변환을 수반한다. 2장에서 기술한 바와 같이 ICMP는 IP 패킷의 전송 상태에 대한 통지를 담당하기 때문에 IPv4 라우터나 IPv6 라우터로부터 요청, 반송되는 ICMP 메시지들에 대한 적절한 변환이 필요하게 된다. ICMP의 메시지들은 버전간에 유사한 부분은 Type과 Code 필드를 재설정함으로써 변환을 수행하며, 해당하지 않는 메시지들은 폐기된다.

3.2 IPv4/IPv6 변환 프로토콜의 설계

3.2.1 IPv4/IPv6 변환 프로토콜의 구성

본 논문에서 설계 및 구현되는 IPv4/IPv6 변환 프로토콜은 IPv4 호스트와 IPv6 호스트 사이의 통신을 위한 망 계층 프로토콜이며 본 프로토콜 및 네트워크 구성을 (그림 5)에 나타내었다.

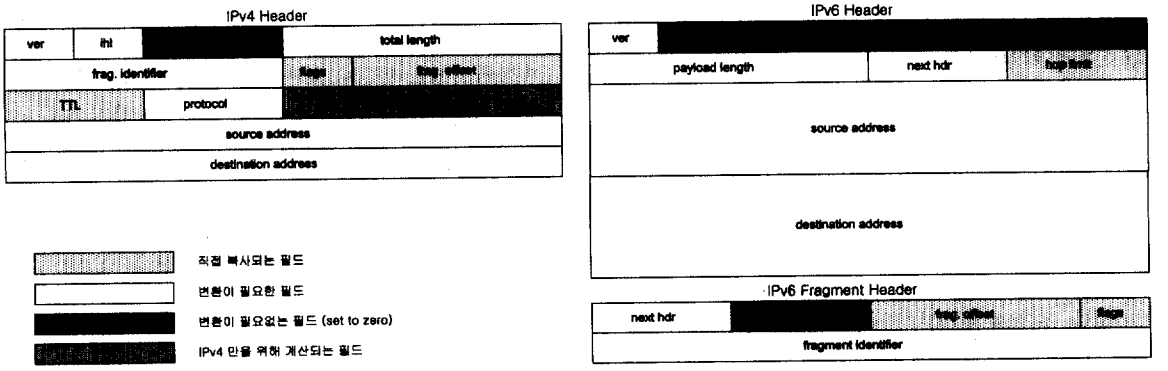


(그림 5) IPv4/IPv6 변환 프로토콜 구성

(그림 5)에서 나타낸 바와 같이 변환 프로토콜은 IPv4의 프로토콜만을 지원하는 호스트와 IPv6의 프로토콜만을 지원하는 호스트간의 원활한 통신을 위해 설계되는 프로토콜이다. 따라서 기존의 호스트에 내재된 프로토콜 스택에 대해 어떠한 수정도 필요하지 않으며, 네트워크 상에 변환 프로토콜을 포함하는 라우터를 위치시킴으로서 IPv4에서 IPv6로의 효과적인 전이방안을 제공할 수 있다.

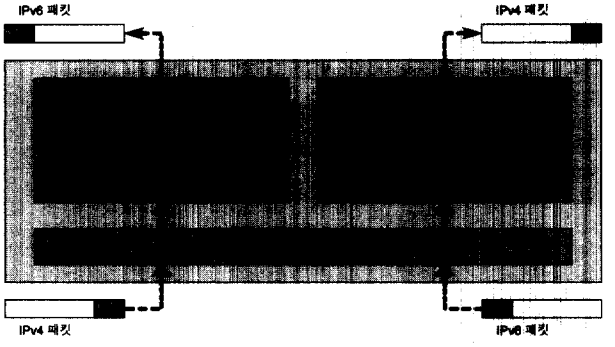
본 논문에서 설계 및 구현되는 변환 프로토콜은 (그림 6)과 같이 크게 IPv4에서 IPv6로의 변환 기능부와 IPv6에서 IPv4로의 변환 기능부 및 IP 패킷 검사부의 세 가지 기능부로 구성된다.

IP 패킷 검사부는 IP 패킷의 버전을 검사한 후 각 버전에 따라 IP 패킷을 IP 프로토콜 변환 기능부로 전달하는 역할을 담당한다. IP 패킷의 버전에 따른 분류가 수행된 후 IP 패킷 헤더의 변환을 통한 프로토콜 변환이 수행되며 이는 IPv4-to-IPv6 프로토콜 변환 기능부와 IPv6-to-IPv4 프로토콜 변환 기능부를 통하여 이루어진다. 각각의 IP 프로토콜



(그림 3) IPv4 및 IPv6 헤더 필드

변환 기능부는 ICMP에 대한 변환 기능부를 포함하고 있으며 ICMP 헤더 변환을 통한 ICMP 메시지의 변환을 처리한다.



(그림 6) IPv4/IPv6 프로토콜 변환모듈의 구조

3.2.2 IPv4-to-IPv6 변환 프로토콜의 설계

(1) IP 헤더 변환

<표 1>은 IPv4에서 IPv6로의 변환에 있어서의 IP 헤더 필드간 변환을 기술한 것이며, 이에 따라 IPv4에서 IPv6로의 각 필드간 변환이 이루어지게 된다. IPv4의 Option 헤더들이 패킷 내에 존재하면, 이들은 무시되며 변환을 위한 시도는 일어나지 않는다. IP 변환 프로토콜은 라우터에 구축되기 때문에 TTL 필드로부터 얻는 값에서 1을 감산한 값으로 변환이 이루어지게 된다.

(2) ICMP 변환

ICMPv4와 ICMPv6는 유사한 헤더 형식(Type, Code, Checksum 필드로 구성되는)을 가지고 있으며 대부분의 메시지들의 의미가 동일하므로 직접적인 변환이 이루어진다. ICMP의 변환에 있어서 중요하게 고려해야 할 점은 Checksum 필드에 대한 처리이다. ICMPv6는 ICMPv4와 달리 pseudo-header checksum이 존재하기 때문에 ICMP를 변환할 경우 모든 ICMP 메시지들은 ICMP Checksum 필드를 수정해야 할 필요가 있다. 또한 모든 ICMP 패킷들은 Type 값과 Code 값을 변환시킴으로서 ICMP 메시지에 대한 변환을 처리하며 특히 ICMP Error 메시지의 변환은 IP 헤더의 변환을 수반하는 경우를 고려해야 한다.

IPv4에서 IPv6로 패킷이 전송될 경우 ICMPv4의 메시지들이 ICMPv6로 변환되는 유일한 ICMP Query 메시지들은 Echo Request와 Echo Reply 메시지이며, 나머지 메시지들은 모두 폐기된다. 본 논문에서 설계되는 변환모듈은 이들 ICMP Query 메시지에 대해 원래의 ICMP 패킷을 복사하고, 다음에 Type 및 Checksum 필드를 수정함으로써 메시지를 변환한다.

변환되는 ICMP Error 메시지들은 Destination Unreachable, Time Exceeded 및 Parameter Problem 메시지들이다. 이들 역시 ICMP Query 메시지의 변환과 같은 방식으로 변환된다. ICMP Error 메시지는 Error를 유발시킨 패킷의 IP 헤더에 대한 변환을 수반해야 할 경우가 있다. 변환된 패킷이 IPv6 라우터에 도착했으나 목적지 IPv6 호스트에 도착하지 못했을 경우 IPv6 라우터는 ICMP Error 메시지와 함께 IP 패킷을 송신측으로 반송하게 되며, 이 패킷은 변환기를 거치게 되기 때문에 본 논문에서는 이러한 경우의 ICMP Error 메시지 변환 역시 고려하였다.

(3) IPv4-to-IPv6 프로토콜 변환 기능부의 동작 절차

IPv4-to-IPv6 프로토콜 변환 기능부는 IPv4 패킷을 입력받아 일련의 변환과정을 거친 후 IPv6 패킷을 반환한다. (그림 7)은 본 논문에서 설계한 IPv4-to-IPv6 프로토콜 변환 기능부의 동작 절차를 나타낸 것이며 일곱 단계로 구분하였다.

(가) 1단계 : IP 패킷 전처리 단계

IP 헤더를 변환하기 전에 검사해야 하는 단계로서 IPv4 패킷이 입력되면 우선 주소 필드에 대한 검사를 토대로 IP 주소가 형식에 맞는지를 판별한다.

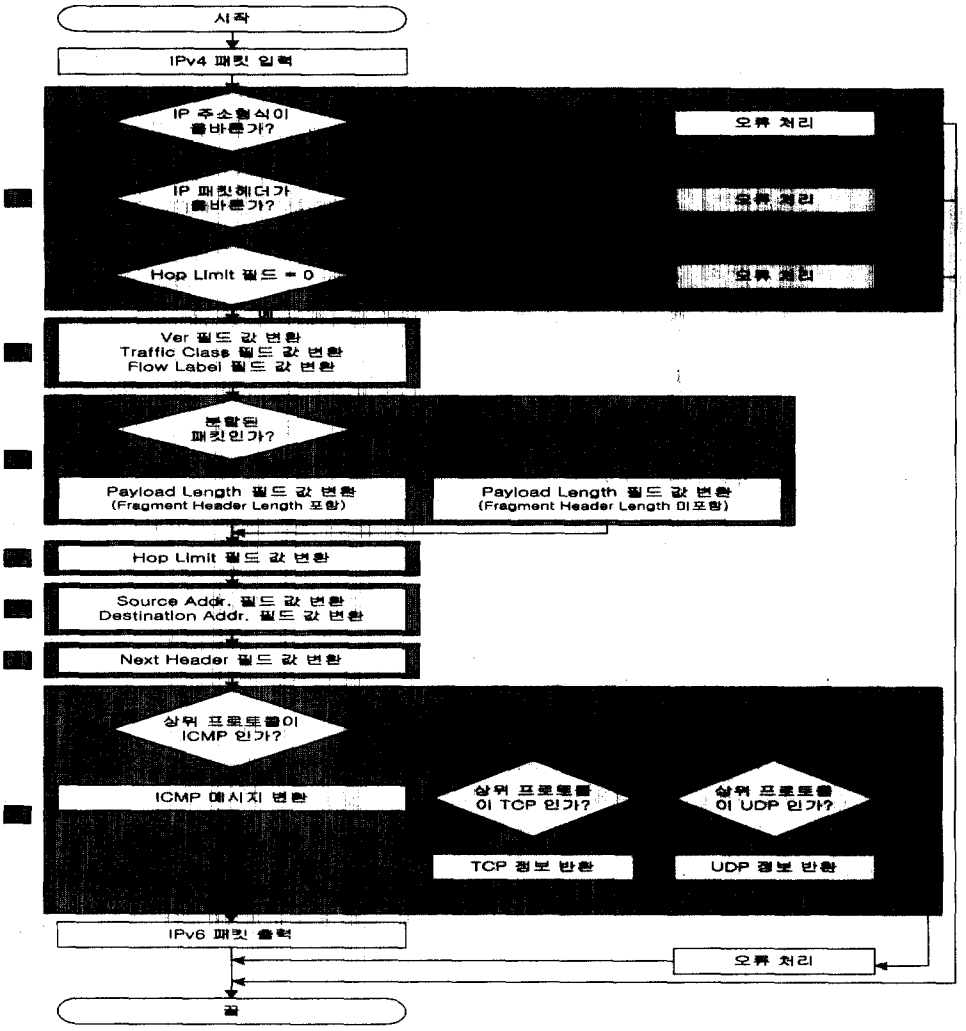
본 논문에서 설계한 IPv4/IPv6 변환 프로토콜은 IP 프로토콜을 변환하기 전에 이러한 전처리 과정을 거침으로서 입력되는 IP 패킷에 대한 오류를 미연에 제거함과 동시에 IP 패킷 헤더의 정보를 미리 추출함으로써 효과적인 프로토콜 변환 과정을 제공한다.

(나) 2단계 : Version, Traffic Class 및 Flow Label 필드 변환 단계

IPv4 패킷을 입력받으면 먼저 IPv6의 Version, Traffic Class 및 Flow Label 필드를 생성하게 된다. Version 필드는 '6'으로 변환되며 Flow Label 필드는 IPv6 표준 명세 상에 아직 정의되어 있지 않기 때문에 모든 비트를 '0'으로 설정한다. IPv6의 Traffic Class 필드는 IPv4의 Type of Service 필드와 대응되며 양쪽의 비트 수가 같기 때문에(8비트) 그대로 복사한다.

〈표 1〉 IP 헤더 필드 변환(IPv4-to-IPv6)

필 드 명	변환되는 값 (내용)
Version	IPv6의 버전 (6)
Traffic Class	IPv4의 TOS 필드값 복사, 일반적으로 0으로 설정 (0)
Flow Label	아직 정의되지 않은 필드 (0)
Payload Length	IPv4 헤더의 Total Length 필드값에서 IPv4 헤더의 IHL 필드값을 뺀 값 복사
Next Header	IPv4 헤더의 Protocol 필드값 복사. IPv4 헤더의 Protocol 필드값이 1(ICMPv4)이면, 이를 58(ICMPv6)로 대체
Hop Limit	IPv4의 TTL 필드값에서 1을 뺀 값 복사
Source Address	상위 96비트는 IP-mapped prefix (: : ffff : 0 : 0/96), 하위 32비트는 IPv4 헤더의 Source Address
Destination Address	상위 96비트는 IPv4-mapped prefix (: : ffff : 0 : 0/96), 하위 32비트는 IPv4 헤더의 Destination Address



(그림 7) IPv4-to-IPv6 변환 프로토콜 기능부의 동작 절차

(다) 3단계 : Payload Length 필드 변환 단계

Payload Length를 구하기 위해서는 입력된 패킷이 분할되었는지에 대한 여부의 확인이 필요하다. 만일 패킷이 분할되었다면 IPv6 헤더 뒤에 Fragment 확장 헤더가 붙기 때문에 Payload Length는 IPv4의 Total Length 필드 값에서 IHL 필드 값을 뺀 다음 Fragment Header Length 값을 더한 값이 되며, 분할되지 않았으면 Total Length 필드 값에서 IHL 필드 값을 뺀 값이 된다.

(라) 4단계 : Hop Limit 필드 변환 단계

Hop Limit 필드 값은 IPv4 헤더의 TTL 필드 값에서 '1'을 뺀 값을 복사한다.

(마) 5단계 : Source Address 및 Destination Address 필드 변환 단계

IPv6 표준 명세에 정의된 IPv4 주소가 내장된 IPv6 주소의 형태 중 하나인 IPv4-mapped Address를 사용한다.

(바) 6단계 : Next Header 필드 변환 단계

2단계의 수행을 통해서 패킷의 분할 여부에 대한 정보를 알 수 있으며, 만일 패킷이 분할되었다면 Next Header 필드 값은 '0'으로 설정되며 이는 IPv6 헤더 다음에 오는 헤더가 Fragment 확장 헤더라는 것을 의미한다.

(사) 7단계 : ICMP 메시지 변환 단계(ICMPv4-to-ICMPv6)

ICMP 메시지의 변환을 처리하기 전에 먼저 IPv4의 Protocol 필드 값을 토대로 상위 프로토콜의 종류를 판별한 후 상위 프로토콜이 ICMP라면 이에 대한 변환을 수행한다. 만일 상위 프로토콜이 TCP나 UDP라면 이와 관련된 처리를 해주면 된다.

3.2.3 IPv6-to-IPv4 변환 프로토콜의 설계

(1) 헤더 변환

<표 2>는 IPv6에서 IPv4로의 변환에 있어서의 IP 헤더 필드간 변환을 기술한 것이다.

만일 IPv6 패킷 내에 Hop-by-Hop Option 헤더, Destination Option 헤더 혹은 Routing 헤더가 있다면 이들은 무시되며, 변환을 위한 시도가 일어나지 않는다.

〈표 2〉 IP 헤더 필드 변환(IPv6-to-IPv4)

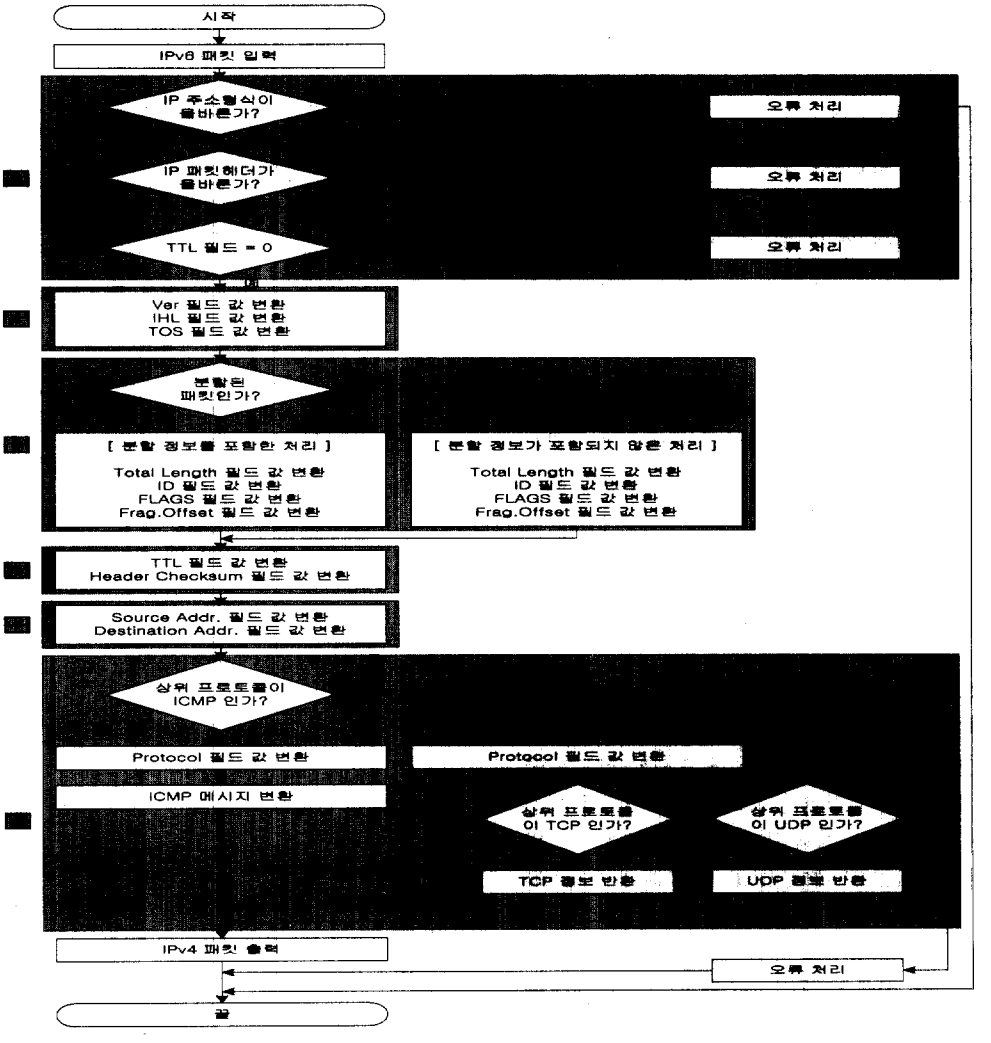
필 드 명	변환되는 내용
Version	IPv4의 버전 (4)
IHL	기본값으로 설정 (5, IPv4 Option들이 없을 경우)
TOS	IPv6 헤더의 Class 필드 값을 복사
Total Length	IPv6 헤더의 Payload Length 필드값에 IPv4의 크기를 더한 값 복사
Identifier	모두 0으로 설정
Flags	More Fragment Flag는 0으로, Don't Fragment Flag는 1로 설정
Fragment Offset	모두 0으로 설정
TTL	IPv6 헤더의 Hop Limit 필드 값에서 1을 뺀 값 복사
Protocol	IPv6 헤더의 Next Header 필드 값을 복사, IPv6 헤더의 Next Header 필드 값이 58이면 1로 대체
Header Checksum	IPv4 헤더가 생성될 때 한 번 계산
Source Address	IPv6 Source Address의 하위 32비트를 복사
Destination Address	IPv6 Destination Address의 하위 32비트를 복사

(2) ICMP 변환

IPv6에서 IPv4로 패킷이 전송될 경우 ICMPv6의 메시지들이 ICMPv4로 변환되는 유일한 ICMPv6 Informational 메시지들은 Echo Request와 Echo Reply 메시지이며, 나머지 메시지들은 모두 폐기된다. 본 논문에서 설계하는 변환모듈은 이들 ICMPv6 informational 메시지에 대해 원래의 ICMP 패킷을 복사하고, 다음에 Type 및 Checksum 필드를 수정함으로서 메시지를 변환한다. 그리고 ICMPv6 Error 메시지들 역시 ICMPv6 Informational 메시지의 변환과 마찬가지로 Type 필드 및 Code 필드의 변환을 통하여 메시지를 변환한다.

(3) IPv6-to-IPv4 프로토콜 변환 기능부의 동작 절차

IPv6-to-IPv4 프로토콜 변환 기능부는 IPv6 패킷을 입력



(그림 8) IPv6-to-IPv4 프로토콜 변환 기능부의 동작 절차

받아 일련의 변환 절차를 거친 후 IPv4 패킷을 반환하도록 설계하였으며, 패킷의 분할과 ICMP에 대한 처리가 포함되어 있다. (그림 8)은 본 논문에서 설계한 IPv6-to-IPv4 프로토콜 변환 기능부의 동작 절차를 나타낸 것으로서 다섯 단계로 구분하였다.

(가) 1단계 : IP 패킷 전처리 단계

IP 패킷 헤더를 변환하기 전에 IP 패킷에 대한 검사를 처리하는 단계이며 IP 주소 검사, IP 패킷 헤더 검사 및 TTL 필드에 대한 검사를 토대로 IP 패킷이 정확하게 프로토콜 변환 기능부로 전달되도록 처리한다.

(나) 2단계 : Version, IHL 및 TOS 필드 변환 단계

IPv6 패킷을 입력받으면 먼저 IPv4의 Version, IHL 및 TOS 필드를 생성하게 된다.

(다) 3단계 : Total Length, ID, Flags 및 Fragment Offset 필드 변환 단계

이들 필드 값을 변환하기 위해서는 IP 패킷이 분할에 대한 정보를 보유하고 있는지의 여부를 확인해야 한다.

(라) 4단계 : TTL 및 Header Checksum 필드 변환 단계

TTL 필드 값은 IPv6 헤더의 Hop Limit 필드 값에서 '1'을 뺀 값을 복사하고, Header Checksum 필드 값은 IPv4 헤더가 생성될 때 한 번 계산한다.

(마) 5단계 : Source Address 및 Destination Address 필드 변환 단계

IPv6의 Source Address와 Destination Address는 IPv4-mapped Address를 사용하므로 이들 필드 값의 하위 32비트를 생성되는 IPv4 헤더의 Source Address 및 Destination Address 필드에 복사한다.

(바) 6단계 : ICMP 메시지 변환 단계(ICMPv6-to-ICMPv4)

IPv4 Protocol 필드 값은 IPv6의 Next Header 필드 값으로부터 변환되며, 상위 프로토콜이 ICMPv6이면 ICMP 메시지 변환을 수행하고, 만일 상위 프로토콜이 TCP나 UDP라면 이와 관련된 처리를 수행한다.

4. IPv4/IPv6 변환 프로토콜의 구현 및 테스트

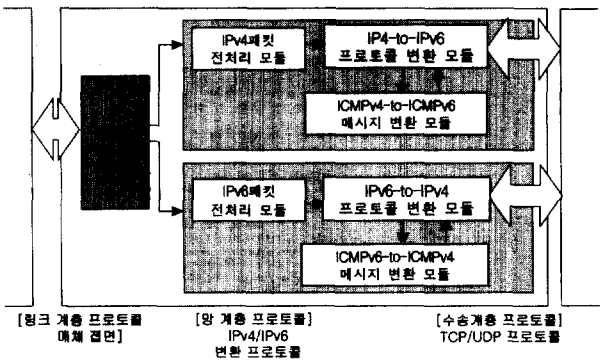
4.1 IP 변환 프로토콜 구현

본 논문에서의 IPv4/IPv6 변환 프로토콜에 대한 실제 구현은 기존에 정의된 Winsock와 Winsock2에 정의된 데이터 구조 및 API들과의 효과적인 연계를 고려하여 C 언어를 이용하여 코딩하였다. (그림 9)는 실제 프로그램에서 전송되는 버전별 IP 패킷 헤더 및 ICMP 헤더, IPv6 확장 헤더 클래스를 정의한 것이다.

본 논문에서는 IPv4/IPv6 변환 프로토콜을 구현하기 위해 위와 같이 정의된 기본 클래스들을 사용하였고 그 외에 ICMP의 Type 및 Code 필드값, IP 상위 프로토콜 및 IPv6 확장 헤더 등을 정의한 부가 클래스들을 사용하였다. IPv4-to-IPv6 프로토콜 변환 기능부 및 IPv6-to-IPv4 프로토콜 변환 기능부는 3장에서 도시한 (그림 7)및 (그림 8)의 동작 처리절차에 따라 변환을 수행하게 된다. 3장에서 도시한 IP 변환 동작 절차 및 메시지 변환 동작 절차에 따라 본 장에서는 변환 모듈의 알고리즘을 생략하였으며 (그림 10)은 본 논문에서 구현한 IPv4/IPv6 변환 프로토콜의 기능 모듈 구조를 나타낸 것이다.

```
typedef struct IPv4Header {
  uchar ipv4h_ver ;       // IPv4 헤더 Version 필드
  uchar ipv4h_lhleng ;    // IPv4 헤더 Internet Header Length 필드
  uchar ipv4h_tos ;       // IPv4 헤더 Type of Service 필드
  ushort ipv4h_tlen ;     // IPv4 헤더 Total Length 필드
  ushort ipv4h_fid ;      // IPv4 헤더 Fragment Identification 필드
  uchar ipv4h_flag ;      // IPv4 헤더 Flags 필드
  ushort ipv4h_foffset ;  // IPv4 헤더 Fragment Offset 필드
  uchar ipv4h_ttl ;       // IPv4 헤더 Time to Live 필드
  uchar ipv4h_protocol ;  // IPv4 헤더 Protocol 필드
  ushort ipv4h_hchecksum ; // IPv4 헤더 Header Checksum 필드
  ip4addr ipv4h_srcaddr ;  // IPv4 헤더 Source Address 필드
  ip4addr ipv4h_desaddr ;  // IPv4 헤더 Destination Address 필드
                }
typedef struct ICMPv4Haeder {
  uchar icmpv4_type ;     // ICMPv4 헤더 Type 필드
  uchar icmpv4_code ;     // ICMPv4 헤더 Code 필드
  ushort icmpv4_checksum ; // ICMPv4 헤더 Checksum 필드
                }
typedef struct IPv6Header {
  uchar ipv6h_ver ;       // IPv6 헤더 Version 필드
  uchar ipv6h_tclass ;    // IPv6 헤더 Traffic Class 필드
  uchar ipv6h_flabel ;    // IPv6 헤더 Flow Label 필드
  ushort ipv6h_plen ;     // IPv6 헤더 Payload Length 필드
  uchar ipv6h_nheader ;   // IPv6 헤더 Next Header 필드
  uchar ipv6h_hlimit ;    // IPv6 헤더 hop Limit 필드
  ipv6addr ipv6h_srcaddr ; // IPv6 헤더 Source Address 필드
  ipv6addr ipv6h_desaddr ; // IPv4 헤더 Destination Address 필드
                }
typedef struct ICMPv6Haeder {
  uchar icmpv6_type ;     // ICMPv6 헤더 Type 필드
  uchar icmpv6_code ;     // ICMPv6 헤더 Code 필드
  ushort icmpv6_checksum ; // ICMPv6 헤더 Checksum 필드
                }
typedef struct IPv6FragmentHeader {
  uchar NextHeader ;      // IPv6 Fragment 헤더 Next Header 필드
  uchar Reserved ;        // IPv6 Fragment 헤더 Reserved 필드
  ushort OffsetFlag ;     // IPv6 Fragment 헤더 Offset Flag 필드
  ulong Id ;              // IPv6 Fragment 헤더 ID 필드
```

(그림 9) 확장 클래스 정의



(그림 10) IPv4/IPv6 변환 프로토콜의 기능 모듈구조

4.2 실험 모델의 테스트

4.2.1 IPv4/IPv6 프로토콜 변환 시험

IPv4/IPv6 프로토콜 변환 시험에서는 사용자 인터페이스를 통해 IPv4와 IPv6간의 프로토콜 변환에 대한 기본 동작 및 변환 절차에 대하 시험을 수행하였다. 시험의 시나리오는 다음과 같다.

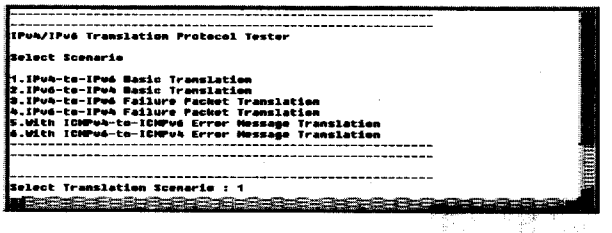
- IPv4와 IPv6간의 정상적인 변환 절차 시험

- 임의의 IP 패킷 오류 발생에 의한 IP 프로토콜 변환 절차 시험
- ICMP Error 메시지 발생 시의 IP 프로토콜 재변환 절차 시험

<표 3>은 이러한 시험 시나리오를 IPv4 측에서의 변환 요구의 경우와 IPv6 측에서의 변환 요구로 구분하여 나타낸 것이다. IPv4 및 IPv6 표준 패킷을 생성하여 변환 프로토콜을 거쳤을 경우 결과로 생성되는 IP 패킷이 역시 표준 패킷인지를 검사하는 시험은 IP 변환 프로토콜의 기능을 확인하기 위한 가장 기본적인 시험이며, IP 패킷이 패킷 정보 오류를 내재하고 있을 경우 IP 변환 프로토콜이 이를 인식하고 대처할 수 있는가에 대한 시험 역시 필요하다.

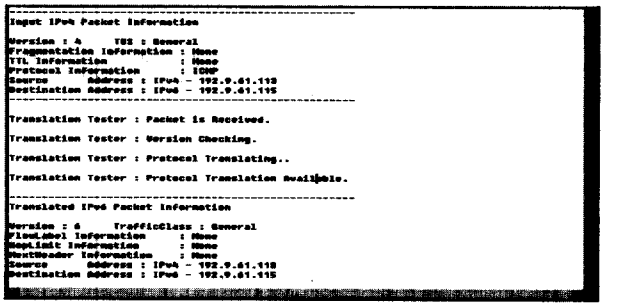
또한 ICMP Error 메시지가 발생하였을 경우 3장에서 언급한 바와 같이 일단 IP 패킷이 변환 프로토콜을 경유하여 수신측 망의 라우터에 도달한 뒤에 오류가 발생하는 경우이므로 IP 프로토콜에 대한 재처리가 필요하기 때문에 이에 대한 능력을 확인하기 위한 시험이다.

각 시험에 대한 전체적인 진행 사항 및 변환 절차를 구성하는 각 클래스에서 송·수신되는 패킷 정보를 화면에 표시함으로써, 시험의 결과를 확인할 수 있었다. 다음에 IPv4 호스트측에서 IPv6로의 변환을 요구한 경우의 수행 결과의 예를 나타내었다. 시험 시스템의 초기 화면은 (그림 11)과 같다.



(그림 11) 변환 프로토콜 시험 시스템의 초기 화면

IPv4/IPv6 변환 프로토콜 시험 시스템을 이용한 시험 결과 화면을 (그림 12)에 나타내었다.



(그림 12) 변환 프로토콜 시험 결과 화면

4.2.3 시험 결과 및 검토

본 시험 모델에서는 사용자 인터페이스를 통하여 IP 패킷의 정상적인 변환 결과를 알 수 있었다. 또한 시험단계에서는 IP 패킷 변환의 정확성을 살펴보기 위해 실제 입력 값을 디스플레이 해 보았다.

본 논문은 IP 프로토콜, 즉 망 계층의 부분을 대상으로 하고 있기 때문에 상·하위 프로토콜이 지원된 완전한 망에서 IPv4 호스트와 IPv6 호스트 종단간의 테스트는 수행할 수 없으나 본 시험의 결과를 통하여 정상적인 프로토콜 변환의 역할을 수행함을 추정할 수 있다.

IP 패킷에 대한 정상적인 변환 시험의 경우 IP 패킷 헤더 내에 있는 각각의 필드값이 정확하게 변환되었으며, 상위 프로토콜인 ICMP 메시지 역시 표준 명세에 의거한 결과 메시지로의 변환이 수행되었다. 임의의 오류를 발생시킨 경우 변환 프로토콜 내부의 오류 처리 루틴에서 이를 발견하고 이에 따른 예외처리를 발생시켰으며 송신측으로 IP 패킷 재전송을 요구하는 메시지를 보내주도록 이상 없이 수행되었다. ICMP Error 메시지를 포함한 경우 본 논문에서 고려한 바와 같이 IP 패킷에 대한 재변환 처리를 이상 없이 수행되었다.

본 실험 모델은 IP 프로토콜 계층을 중심으로 IP 패킷 헤

〈표 3〉 IP 변환 시험 시나리오(패킷 입출력)

IPv4 노드	정상적인 IPv4 패킷 입력 • 정상적인 헤더 형식 및 크기 • 분할(fragmentation) 정보 없음 • TTL 필드값 오류 없음	정상적인 IPv6 패킷 출력 • 정상적인 헤더 형식 및 크기 • IP 패킷 헤더내의 각 필드에 대한 정상적인 변환 수행
	임의의 IPv4 패킷 오류 • Flag 필드에 대한 오류 발생(비트 0, 1, 2에 오류값 입력)	IP 패킷 헤더 내의 오류 필드 추출 해당 필드에 대한 오류 정보 반송
	ICMP Error 메시지 발생 • Destination Unreachable(Type 3, Code 0)	ICMP Error 메시지 처리 IPv4 헤더에 대한 재변환 처리
IPv6 노드	정상적인 IPv6 패킷 입력 • 정상적인 헤더 형식 및 크기 • 분할(fragmentation) 정보 없음 • Hop Limit 필드 오류 없음	정상적인 IPv4 패킷 출력 • 정상적인 헤더 형식 및 크기 • IP 패킷 헤더내의 각 필드의 정상적인 변환 수행
	임의의 IPv6 패킷 오류 • Next Header 필드 오류 발생(정의되지 않은 헤더값 입력)	IP 패킷 헤더 내의 오류 필드 추출 해당 필드에 대한 오류 정보 반송
	ICMP Error 메시지 발생 • Time Exceeded(Type 2, Code 0)	ICMP Error 메시지 처리 IPv6 헤더에 대한 재변환 처리

더의 변환 모듈을 구현하고 시험하였으며 향후 상·하위 프로토콜을 구성함으로써 인터넷을 구성하는 DNS나 라우터, 게이트웨이와의 연동을 통한 실제 망에서의 실험을 지속적으로 수행함으로써 실제 망에 대한 적용성을 시험해야 할 것으로 사료된다.

5. 결 론

기존의 인터넷 망 계층 프로토콜인 IPv4는 기술적으로 안정성을 가지고 있으나 차세대 인터넷이 추구하는 응용 서비스 관점에서 볼 때 가용한 주소가 모자라고 QoS, 보안 문제 등을 지니고 있다. 이러한 환경에서 IPv6 환경으로의 동시 전환은 현실적으로 불가능하며 IPv4에서 IPv6로의 효과적이고 점진적인 전이 방안에 대한 연구가 필요하다. 특히 이러한 전이 방안 중 IP 프로토콜의 변환은 IPv4, IPv6 각각 단일 구조로 구성된 호스트들간에 효과적이고 원활한 상호 통신을 제공할 뿐만 아니라 기존의 수많은 호스트들에 대한 IP 프로토콜을 수정할 필요가 없다는 점에서 매우 효율적인 방안이라고 할 수 있다.

이에 따라 본 논문에서는 IPv4 및 IPv6 단일 구조로 구성된 호스트들간의 원활한 통신을 위해 IETF(Internet Engineering Task Force)의 표준 문서 및 관련 기술을 기반으로 하는 IPv4/IPv6 변환 프로토콜을 설계하고 구현하였다.

IPv4/IPv6 변환 프로토콜의 구현을 위해 IETF의 IPv4 및 IPv6 관련 표준문서에 따라 IP 프로토콜 헤더를 분석하고 프로토콜 헤더간의 변환 상태를 정의하였으며 이를 토대로 프로토콜 변환 동작 절차를 IPv4에서 IPv6로의 변환과 IPv6에서 IPv4의 변환으로 구분하여 각각 설계하였다. IP 패킷의 효과적인 변환을 위해 패킷 선처리부를 두었으며 IP 패킷 변환 기능부 내에 ICMP 메시지 변환 기능부를 포함하여 IP 프로토콜뿐만 아니라 ICMP 메시지에 대한 변환 역시 수행이 가능하도록 설계하였다. 또한 설계된 동작 절차 및 프로토콜 표준을 토대로 IPv4/IPv6 변환 프로토콜을 구현하였으며 국부 시험 방법에 의한 적합성 시험을 통하여 IPv4/IPv6 변환 프로토콜이 이상 없이 동작함을 확인하였다.

본 논문에서 설계 및 구현한 IPv4/IPv6 변환 프로토콜에 관한 결과들은 향후 차세대 인터넷 환경으로의 효과적인 전이 방안을 위한 기반 기술로 활용될 수 있을 것으로 생각되며, 초고속 인터넷 환경을 위한 효율적인 진화 전략을 수립하는데 일익을 담당할 수 있을 것으로 사료된다.

참 고 문 헌

[1] T. Larder, "Transition Scenarios and Solutions," Internet-Draft, IETF, April, 1999.

[2] K. Yamamota, K. Sumikawa, "Categorizing Translator between IPv4 and IPv6," Internet-Draft, IETF, January, 1999.

[3] R. E. Gilligan, Erik Nordmark, "Transition Mechanism for IPv6 Hosts and Routers," Internet-Draft, IETF, May, 31, 1999.

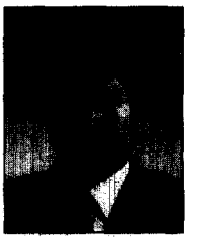
[4] 김성일, "http : //www.ngi.or.kr", IPv6 개념 및 현황, 테마특강.

[5] S. Deering, R. Hinden "Internet Protocol, Version (IPv6) Specification," RFC 2460, IETF, December, 1998.

[6] R. Hinden, S. Deering "Internet Protocol Version 6 Addressing Architecture," RFC 2373, IETF, July, 1998.

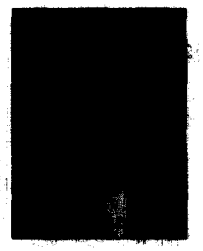
[7] A. Conta, S. Deering "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version (IPv6) Specification," RFC 2463, IETF, December, 1998.

[8] J. McCann, S. Deering, J. Mogul, "Path MTU Discovery for IP version 6," RFC 1981, IETF, August, 1996.



박 석 천

e-mail : scpark@mail.kyungwon.ac.kr
1977년 고려대학교 전자공학과(공학사)
1982년 고려대학교 컴퓨터공학(공학석사)
1989년 고려대학교 컴퓨터공학(공학박사)
1979년~1985년 금성통신연구소
1991년~1992년 Univ. of California, Irvine Post Doc.
1992년~1994년 경원대학교 산업기술연구소장
1988년~현재 경원대학교 컴퓨터공학과 정교수
관심분야 : 차세대 인터넷, 멀티미디어 통신, 위성 통신, 통신망 관리, IMT-2000 등



이 광 배

e-mail : kblee@wh.myongji.ac.kr
1979년 고려대학교 전자공학과(공학사)
1981년 고려대학교 전자공학과(공학석사)
1981년~1982년 삼성반도체
1982년~1983년 금성사
1984년~1986년 Univ. of Southern California, Computer Engineering(공학석사)
1986년~1991년 Arizona state Univ., Computer Engineering (공학박사)
1992년~1998년 명지대학교 전자공학과 교수
1998년~1999년 Worchestor Polytechnic Institute 대학 WINLAB 연구실 방문 연구 교수
1999년~2000년 Columbia 대학 COMET 연구실 방문 연구 교수
2000년~현재 명지대학교 전자공학과 교수
관심분야 : 이동 무선 인터넷망, 멀티미디어, 위성 통신, 컴퓨터 구조