IPv6-IPv4 프로토콜 변환기의 하드웨어 설계 및 구현

이경렬° 공인엽 이중렬 이정태
부산대학교 컴퓨터공학과
{idking96°, leafgirl, blueirix, jtlee}@pusan.ac.kr

Hardware Design and Implementation of IPv6-IPv4 Protocol Translator

KyongYeol Lee° InYeup Kong JoongLyul Lee JungTae Lee
Dept. of Computer Engineering, Pusan National University

요 약

IPv6 도입 단계에서는 IPv4 Network와 IPv6 Network가 혼재하게 되는데, 이 때 IPv4 Network와 IPv6 Network 간의 통신을 가능하게 하는 IPv6-IPv4 프로토콜 변환기가 요구된다. 그러나 성능 분석 결과에 따르면, 기존에 구현된 프로토콜 변환기는 운영 체제를 기반으로 한 소프트웨어로 구현되어 있어서 Network 간의 모든 트래픽을 처리하기에는 성능 상의 한계가 있다. 이에 본 논문에서는 기존 소프트웨어 프로토콜 변환기의 성능 인자 분석 연구를 토대로 하여, 하드웨어 기반의 고성능 64Translator를 제안하였다. 64Translator는 하드웨어 TCP/IPv6와 TCP/IPv4를 내장하고 개선된 메모리 엑세스 방식을 사용함으로써 기존 구현 방식에 비해 성능을 개선하였다. 구현된 하드웨어 모듈에 대해서는 소프트웨어 시뮬레이션과 시험망상에서의 테스트를 수행함으로써 그 기능을 검증하였다.

1. 서 론

인터넷 사용의 급증과 정보 가전의 보급 확산에 따라 IP 주소의 부족 현상이 중요한 문제로 부각되었다. 이에 IETF(Internet Engineering Task Force)에서는 기존의 32비트 IP 주소 체계를 128비트로 확장시킨 차세대 인터넷 프로토콜인 IPv6를 제안하게 되었으며 IPv6를 도입하기 위한 많은 연구와 노력이 각 연구 단체와 대학을 중심으로 진행되고 있다[1].

IPv6와 관련된 기술 중에서 새로 구축되거나 IPv6로 업그레이드 되는 Network가 기존의 IPv4 기반 Network와 원활하게 통신하기 위해서는 IPv6-IPv4 프로토콜 변환기와 관련된 연구가 필수적이다. 그러나 이를 위해 구현된 기존의 IPv6-IPv4 프로토콜 변환기의 경우, 운영 체제에 종속적인 소프트웨어 모듈로 구현되어 있어 성능 측정 결과에 따르면 Network 간의 트래픽을 처리하기에는 성능 상의 한계가 있다[2].

따라서 이러한 문제를 해결하기 위해서는 운영 체제에 독립적인 고성능의 IPv6-IPv4 프로토콜 변환기가 필요하다. 이에 본 논문에서는 고성능 IPv6-IPv4 프로토콜 변환기의 구현 방법에 대한 연구를 토대로 하여 하드웨어 기반의 프로토콜 변환기인 64Translator를 제안하였다. 하드웨어로 구현된 64Translator는 소프트웨어 시뮬레이션에 의해 기본적인 기능을 검증하였고, 일반 호스트들로 구성된 시험망에서 테스트를 수행하였다.

본 논문의 전체 구성은 다음과 같다. 먼저 2장에서는 NAPT-PT/SIIT 변환 기술에 대해 살펴보고, 3장에서는 64Translator의 설계 및 구현, 테스트 환경 및 방법에 대한 내용을 설명하였고, 4장에서는 결론 및 향후 연구 과제에 대한 내용을 기술하였다.

2. 관련 연구

기존에 구현된 IPv6-IPv4 프로토콜 변환기는 NAT-PT(Network Address Translation-Protocol Translation)를 기반으로 하는데 비해, 본 논문에서 제안한 64Translator에서는 프로토콜 변환 기법으로서 NAPT-PT(Network Address Port Translation-Protocol Translation)를 사용하였다. NAPT에 대해서는 2.1절에서 다루며, Protocol Translation에 해당하는 SIIT에 대해서는 2.2절에서 다룬다.

2.1 NAPT

NAPT는 64Translator의 핵심 변환 기술로서, 하나의 IP 주소를 사용하고, 각 세션마다 고유의 포트 번호를 할당하여 다중화하여 IP 주소의 효율의 높이는 것을 특징으로 한다[3]. 이러한 특징을 가지는 NAPT는 내부의 IP 어드레스를 외부 Network와 구별하기 위해 보안 수단으로서 사용되거나 다수의 사용자에 대한 하나의 IP 주소로 서비스하고자 할 때 사용하게 된다.

그림 1에서 보는 바와 같이 IPv6-IPv4 프로토콜 변환기에서 NAPT를 사용하는 경우, 각각의 세션은 IPv4 Network의 관점에서는 고유의 포트 번호(6000번, 6700번, 7600번)로 구분되고, IPv6 Network 내부에서는 각각에 해당하는 IPv6 호스트로 맵핑된다. 이를 위해서는 IPv6-IPv4 프로토콜 변환기에서는 포트 번호 할당과 TCP/UDP 헤더의 포트 번호 필드 변환, 그리고 필드 수정으로 인한 체크섬 재계산 기능이 추가적으로 구현되어야 한다.

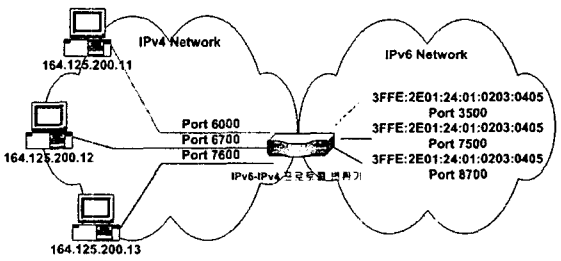

그림 1. NAPT의 포트 변환 원리

2.2 SIIT

SIIT(Stateless IP/ICMP Translation)는 기본적으로 IPv4와 IPv6 헤더 간의 변환, ICMPv4와 ICMPv6 헤더 간의 변환 규칙을 정의하고 있다[4]. 더불어 SIIT는 TCP, UDP, ALG(Application Level Gateway) 부분에서의 고려 사항에 대해 표 1에 정리된 바와 같이 제시한다.

표 1. 프로토콜에 따른 변환 내용

구분	변환 내용
IP	IPv4 Header To IPv6 Header
	IPv6 Header To IPv4 Header
ICMP	ICMPv4 Header To ICMPv6 Header
	ICMPv6 Header To ICMPv4 Header
TCP/ UDP	Checksum Update
	Port Translation
ALG	FTP-ALG
	DNS-ALG
	SIP-ALG

3. 64Translator의 설계 및 구현

3.1 전체 모듈 구성 및 상세 설계

64Translator는 IPv6-IPv4 프로토콜 변환기의 기능을 하드웨어로 구현한 것으로서, 고성능 IPv6-IPv4 프로토콜 변환기의 구현에 관한 기존 연구를 기반으로 하고 하드웨어 TCP/IPv4 Core, TCP/IPv6 Core를 사용하여 그림 2와 같이 설계되었다[5].

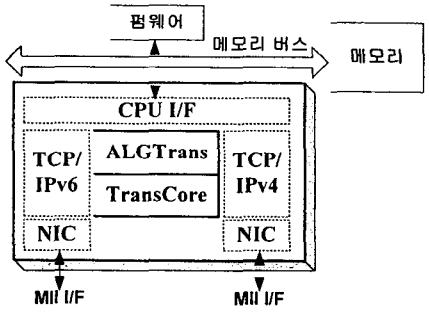

그림 2. 64Translator의 Top-Model

TransCore 모듈은 프로토콜 변환에 필요한 필드 정보의 관리, 주소 및 포트 번호의 변환, IP와 ICMP 프로토콜의 변환, 펌웨어와의 통신을 위한 I/F 등의 다양한 기능을 제공한다. ALGTrans 모듈은 DNS, FTP에 대한 하드웨어 ALG 변환 모듈을 사용한다[6]. 펌웨어는 기본적인 하드웨어 초기화와 DHCP를 이용한 주소 설정[2], 및 모니터링 기능 등을 수행한다. TCP/IPv4 Core와 TCP/IPv6 Core는 기구현된 모듈을 사용하였다. CPU I/F를 포함한 TransCore 모듈의 세부 모듈 구성은 그림 3과 같다.

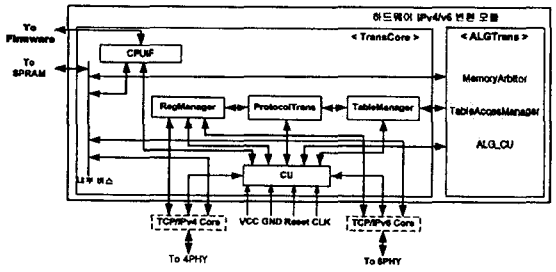

그림 3. TransCore 모듈의 Top-Model

.그림 3에서 보는 바와 같이 TransCore 모듈은 기능에 따라 CPUIF, CU, ProtocolTrans, TableManager, RegManager의 세부 모듈로 구성된다. 각 모듈의 기능은 표 2에 제시한 바와 같다. 여기서 RegisterMAP은 헤더 필드 정보를 유지 및 관리하는 내부 레지스터 모듈이다.

표 2. TransCore 모듈의 세부 모듈 기능

모듈 명	기 능
CPUIF	- RegisterMAP을 정의 - 펌웨어에 대한 인터페이스를 제공 - 펌웨어가 DNS, FTP에 대한 ALG 변환, DHCP 주소 할당, 모니터링 기능을 할 수 있도록 지원.
CU	- 다른 세부 모듈을 제어하는 모듈
ProtocolTrans	- NAPT-PT/SIIT의 구현 - IP, ICMP, TCP, UDP 헤더에 대한 변환을 수행.
RegManager	- 패킷의 변환에 필요한 필드 정보를 저장 - 저장된 정보를 유지, 관리하는 기능.
TableManager	- 프로토콜 변환, ALG 변환 시에 필요한 주소 및 포트 번호에 대한 정보를 관리하고, 제공.

표 2의 기능을 수행하는 TransCore 모듈의 설계 및 구현에 대하여 CPUIF 모듈은 그림 4과 같이 정의된다.

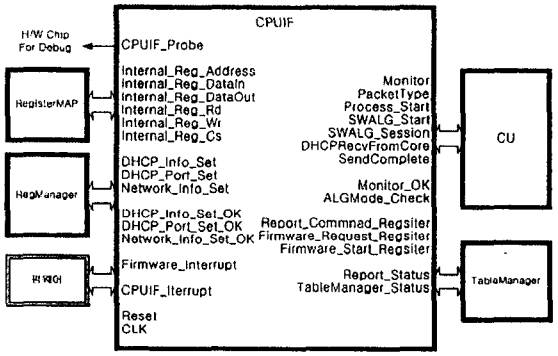

그림 4. CPUIF 모듈의 블록도

그림 4에 제시된 바와 같이 CPUIF 모듈은 RegisterMAP, RegManager, 펌웨어, CU 및 TableManager 모듈과 연결되는데, CPUIF 모듈의 인터페이스 중에서 CU와의 인터페이스는 표 3과 같이 정의된다.

표 3. CPUIF 모듈과 CU 모듈 간의 인터페이스 정의

Pin Name	I/O	Bit	Description
Monitor	I	1	현재 상태의 Monitor 요청
Packet_Type	I	4	모니터할 Packet을 알려준다 0000 : No Type, 0001 : ETH-IP-TCP, 0010 : ETH-IP_UDP, 0011 : ETH-IP-ICMP, 그외 Reserved
Process_Start	I	2	ALG 변환 모듈을 선택 00 : Nothing, 01 : 펌웨어, 10 : H/W, 11 : Reserved
SWALG_Start	I	2	변환할 ALG를 알려준다 00 : idle, 01 : DNS, 10 : FTP. 11 : SIP
SWALG_Session	I	2	세션의 방향을 표시 00 : idle, 01 : IPv4→IPv6, 10 : IPv6→IPv4,. 11 : Nothing
DHCPRecvFromCore	I	1	DHCP 패킷 수신시 알려준다

표 3. CPUIF 모듈과 CU 모듈 간의 인터페이스 정의 (계속)

Pin Name	I/O	Bit	Description
SendComplete	I	1	Packet Send의 완료 시 알림
Monitor_OK	O	1	Monitor 완료를 CU모듈에 알림
ALGMode_Check	O	1	동작 초기에 ALG Mode를 알림 (0 : H/W ALG, 1 : F/W ALG)

표 3의 인터페이스 정의에 따라 CPUIF 모듈의 Entity를 선언하였고, 동작 시나리오를 구상하였다.

CPUIF 모듈의 동작 시나리오는 그림 5와 같다. 하드웨어가 Reset되면, 가장 먼저 펌웨어가 동작을 시작한다. 이 때 펌웨어는 DHCP 프로토콜을 통해서 동적으로 IPv4 주소를 할당 받고, 초기 네트워크 정보를 설정한다[2]. 이후, 펌웨어는 할당 받은 주소를 포함한 네트워크 정보를 재설정하고 CPUIF 모듈에 알리면 변환을 위한 동작을 시작한다. 이후, 패킷이 수신되면 프로토콜 변환과 병행하여 ALG 변환이 수행되고, 펌웨어에서 ALG 변환을 수행하는 경우는 CPUIF 모듈과 통신하여 ALG 변환을 수행한다. 변환이 완료되면, 변환 결과를 펌웨어를 통해서 모니터하고, 완료되면 변환된 패킷을 송신하고, 송신 완료되면 다음 변환할 패킷을 기다린다. 위의 과정을 요약하여 그림으로 표현하면 그림 5와 같다.

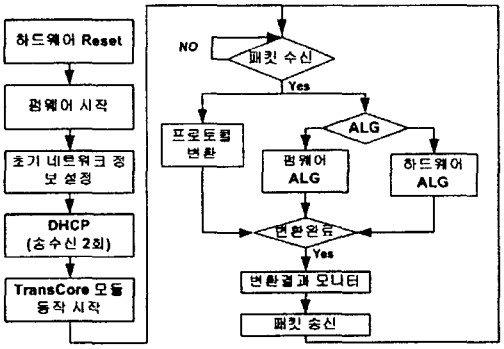

그림 5. CPUIF 모듈의 동작 시나리오

그림 3에 제시된 TransCore 모듈의 나머지 세부 모듈들도 CPUIF 모듈와 동일한 설계 방법을 적용하여 각각에 대해 인터페이스 및 동작 시나리오를 정의하였다.

3.2 구현 및 테스트

그림 3의 각 세부 모듈은 VHDL로 구현하여 통합하였고, 이에 대해 소프트웨어 시뮬레이션을 수행하였다. 또한 그림 6과 같은 테스트 환경을 구축하고, 구현된 모듈을 FPGA 보드에 탑재하여 기능을 검증하였다.

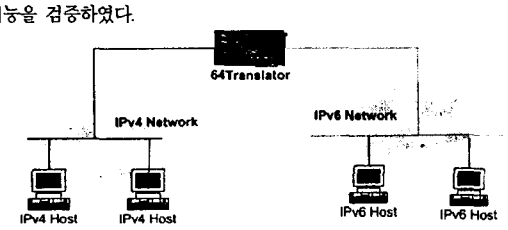

그림 6. 64Translator의 테스트 환경

IPv6 호스트와 IPv4 호스트는 각각 IPv6, IPv4 전용 호스트로서, 윈도우 XP 기반으로 설정되었다. 64Translator는 FPGA 보드에 탑재되고, 기존의 IPv4 Network와 Native IPv6 Network의 중간에 위치하여 프로토콜 변환 및 ALG 변환 기능을 수행한다. DNS 주소를 획득하기 위해서는 IPv4 Network의 DNS Server를 이용했다. 64Translator의 기능을 테스트하기 위한 테스트 시나리오는 표 4의 내용과 같으며 이에 따라 기능 검증을 수행하였다.

표 4. 64Translator의 기능 테스트 시나리오

Test 구분	내용	테스트 프로토콜
Ping Test	6호스트와 4호스트 상호 Ping Message를 송수신	ETH-IPv4-ICMPv4 ETH-IPv6-ICMPv6
Ping을 통한 DNS Test	Domain Name을 이용한 Ping Message 송수신	ETH-IPv4-UDP (ALG) ETH-IPv6-UDP (ALG)
FTP Test	6호스트에서 4호스트로 FTP 접속 및 데이터 송수신	ETH-IPv4-TCP (ALG) ETH-IPv6-TCP (ALG)
Web Test	6호스트에서 4호스트의 Web Page를 Download	ETH-IPv4-TCP ETH-IPv6-TCP
E-Mail Test	E-Mail의 송수신	ETH-IPv4-UDP ETH-IPv6-UDP

4. 결 론

IPv4 Network의 일부에 IPv6 Network가 점진적으로 구축되면서 IPv6 Network와 IPv4 Network가 혼재하는 Network의 형태가 오랜 기간 존재할 것으로 예상되고 있다. 이러한 환경에서 IPv6의 원활한 도입을 위한 기술이 IETF에 의해 제안되었고, 그 중 일부는 소프트웨어로 구현되었다. 그러나 기존에 구현된 운영 체제 기반의 IPv6-IPv4 프로토콜 변환기는 운영 체제 자체의 오버헤드와 비효율적인 메모리 엑세스 방식으로 인하여 성능이 저하되는 문제점이 있다.

이에 본 논문에서는 고성능 IPv6-IPv4 프로토콜 변환기인 64Translator를 제안하였고, 이에 대한 상세 설계 및 구현 내용을 제시하였다. 또한 구체적인 세부 모듈의 시나리오, 전체 테스트 환경 그리고 테스트 방법에 대한 내용도 제시하였다.

본 논문에 의한 고성능 64Translator는 IPv6의 도입을 앞당기는 기반 기술로서의 가치가 있다. 향후 연구 과제로는 다양한 테스트 환경을 구성하여 변환 기능을 안정화하고, 코드 및 알고리즘을 최적화하는 연구를 진행할 예정이다.

참고 문헌

[1] 신명기, "IPv4/IPv6 주소 및 프로토콜 변환기", ETRI, 2001.
[2] 공인엽 외 2명, "Design and Implementation of IPv6-IPv4 Protocol Translation System using Dynamic IP address", 2nd HS@I Conference, LNCS2713, pp.496-506, 2003.
[3] Tsirtsis, G. and P. Srisuresh, "Network Address Translation □ Protocol Translation (NAT-PT)", RFC 2766, 2000.
[4] E. Nordmark, "Stateless IP/ICMP Translation Algorithm (SIIT)," IETF RFC 2765, 2000.
[5] 공인엽 외 3명, "고성능 IPv6-IPv4 프로토콜 변환기의 구현에 관한 연구", 한국정보과학회 2003년 추계학술대회 제출.
[6] 이중렬 외 3명,, "IPv6-IPv4 프로토콜 변환기를 위한 ALG 모듈의 H/W 설계 및 구현", 한국정보과학회 2003년 추계학술대회 제출.