Hybrid TCP/IP Offload Engine 프로토타입의 설계 및 구현

(Design and Implementation of a Hybrid TCP/IP Offload Engine Prototype)

장 한 국 † 정 상 화 †† 오 수 철 †††
(Han-Kook Jang) (Sang-Hwa Chung) (Soo-Cheol Oh)

요 약 최근 TCP/IP 프로토콜을 네트워크 어댑터 상에서 처리함으로써 호스트 CPU의 부하를 줄이는 TOE (TCP/IP Offload Engine)에 대한 연구가 활발히 진행되고 있다. TOE의 구현 방안으로는 임베디드 프로세서를 사용하여 TCP/IP를 처리하는 소프트웨어적인 구현 방법과 TCP/IP의 모든 기능을 하드웨어로 구현하는 방법이 제안되어 왔다. 본 논문에서는 하드웨어적인 접근 방법과 소프트웨어적인 접근 방법을 결합한 Hybrid TOE 구조를 제안한다. Hybrid TOE는 많은 작업 부하로 인하여 임베디드 프로세서 상에서 성능을 확보하기 어려운 기능들은 하드웨어로 구현하고, 연결 설정과 같이 통신의 성능에 영향을 크게 끼치지 않는 기능들은 임베디드 프로세서 상에서 소프트웨어로 처리한다. 이 방법은TCP/IP의 모든 기능을 하드웨어로 구현하는 방법에 근접하는 성능을 제공할 수 있으며, 새로운 기능을 추가하거나 TCP/IP를 기반으로 하는 상위 계층 프로토콜까지 오프로딩하는 것이 가능하므로 구조의 유연성 측면에서 장점을 가진다. 본 논문에서는 Hybrid TOE의 프로토타입을 개발하기 위해 FPGA와 ARM 프로세서를 탑재한 프로토타입 보드를 개발하였고, 하드웨어 모듈과 소프트웨어 모듈을 각각 FPGA와 ARM 프로세서 상에 구현하였다. 또한 하드웨어 모듈과 소프트웨어 모듈의 연동 메커니즘을 개발하였다. 실험을 통해 Hybrid TOE 프로토타입이 호스트 CPU 상에 발생하는 부하를 줄여줌을 입증하고, 하드웨어/소프트웨어 연동 구조의 효과를 분석하였다. 그리고, Hybrid TOE의 완성을 위해 필요한 요소들을 분석하였다.

키워드 : TOE, TCP/IP Offload Engine, TCP/IP, 리눅스, 임베디드 리눅스

Abstract Recently TCP/IP Offload Engine (TOE) technology, which processes TCP/IP on a network adapter instead of the host CPU, has become an important approach to reduce TCP/IP processing overhead in the host CPU. There have been two approaches to implementing TOE: software TOE, in which TCP/IP is processed by an embedded processor on a network adapter; and hardware TOE, in which all TCP/IP functions are implemented by hardware. This paper proposes a hybrid TOE that combines software and hardware functions in the TOE. In the hybrid TOE, functions that cannot have guaranteed performance on an embedded processor because of heavy load are implemented by hardware. Other functions that do not impose as much load are implemented by software on embedded processors. The hybrid TOE guarantees network performance near that of hardware TOE and it has the advantage of flexibility, because it is easy to add new functions or offload upper-level protocols of TCP/IP. In this paper, we developed a prototype board with an FPGA and an ARM processor to implement a hybrid TOE prototype. We implemented the hardware modules on the FPGA and the software modules on the ARM processor. We also developed a coprocessing mechanism between the hardware and software modules. Experimental results proved that the hybrid TOE prototype can greatly reduce the load on a host CPU and we analyzed the effects of the

· 이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음
† 학생회원 : 부산대학교 컴퓨터공학과
hkjang@@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
(Corresponding author임)
††† 정 회 원 : 한국전자통신연구원 디지털홈연구단
ponylife@etri.re.kr
논문접수 : 2005년 10월 11일
심사완료 : 2006년 1월 25일

coprocessing mechanism. Finally, we analyzed important features that are required to implement a complete hybrid TOE and we predict its performance.

Key words : TOE, TCP/IP Offload Engine, TCP/IP, Linux, Embedded Linux

1. 서 론

최근 네트워크 기술이 급속히 발전함에 따라 컴퓨터 시스템들을 연결하는 네트워크의 속도도 크게 향상되고 있다. 인터넷을 비롯한 많은 영역에서 사용되고 있는 대표적인 컴퓨터 네트워크 기술인 이더넷(Ethernet)은 이미 1 Gbps의 대역폭이 일반화되고 있으며, 더 나아가 10 Gbps의 대역폭을 제공하는 10 Gigabit Ethernet으로의 발전이 진행되고 있다. 이러한 네트워크 상에서 표준으로 사용되는 통신 프로토콜인 TCP/IP는 컴퓨터 시스템의 호스트 CPU가 전담하여 처리하는 것이 일반적이었는데, 이는 호스트 CPU 상에 막대한 부하(load)를 발생시켜 전체 시스템의 성능을 저하시킨다는 문제점을 가진다. 특히, 네트워크의 물리적 대역폭이 Gbps급을 넘어서 10 Gbps급으로 발전함에 따라 호스트 CPU에 발생하는 부하도 급격히 증가하게 된다[1,2]. 이러한 문제점을 해결하기 위해 TCP/IP를 호스트 CPU 대신 네트워크 어댑터에서 처리하는 TOE(TCP/IP Offload Engine) 기술이 제안되었다. TOE를 사용하여 TCP/IP를 네트워크 어댑터 상에서 처리하게 되면 호스트 CPU에 가해지는 부하가 줄어들고, 호스트 CPU가 프로토콜 처리 이외의 실질적인 작업에 전념함으로써 전체 시스템의 성능이 향상되는 효과를 얻을 수 있다.

TOE를 개발하는 데에는 두 가지 접근 방법이 제안되어 왔다. 첫 번째 방법은 네트워크 어댑터에 탑재한 임베디드 프로세서 상에서 소프트웨어로 TCP/IP를 처리하는 방안으로, TCP/IP를 하드웨어로 구현하는 방안에 비해 구현이 쉽다는 장점을 가진다[3]. 그러나 임베디드 프로세서는 호스트 CPU에 비해 성능이 낮으므로, 네트워크의 성능이 떨어지는 단점이 있다[4]. 두 번째 방법은 TCP/IP를 처리하는 전용 ASIC을 개발하는 방안이다[5-7]. 이 방안은 네트워크의 성능을 보장할 수 있지만[8-10], 새로운 기능의 추가나 변경이 어려우므로 구조에 유연성이 없다는 단점을 가진다. 또한 TCP/IP를 기반으로 하는 상위 수준 프로토콜을 네트워크 카드에서 같이 처리하고자 할 때 이에 대응하기가 어렵다.

본 연구진은 선행 연구[11]에서 리눅스 기반 TCP/IP 프로토콜 스택 내의 각 함수를 처리하는 데 걸리는 시간을 측정하여 호스트 CPU에 많은 부하를 발생시키는 요인들을 분석하였다. 그리고, 이러한 분석을 바탕으로 하드웨어 기반의 TOE 구현 방안과 소프트웨어 기반의 TOE 구현 방안을 결합한 Hybrid TOE 구조를 제안하였다. Hybrid TOE 구조는 많은 작업 부하로 인하여 임베디드 프로세서 상에서 성능을 확보하기 어려운 기능들은 하드웨어로 구현하고, 연결 설정과 같이 통신의 성능에 영향을 크게 끼치지 않는 기능들은 임베디드 프로세서 상에서 소프트웨어로 처리한다. 그 결과 Hybrid TOE는 하드웨어 기반의TOE에 근접하는 성능을 제공할 수 있으며, 새로운 기능을 추가하거나 TCP/IP를 기반으로 하는 상위 프로토콜까지 네트워크 어댑터에서 처리할 수 있으므로 구조의 유연성 측면에서 장점을 가진다.

본 논문에서는 Hybrid TOE의 프로토타입을 구현하기 위해 FPGA와 ARM 프로세서를 탑재한 프로토타입 보드를 개발하였다. 이 보드는 컴퓨터 시스템의 64bit/66MHz PCI 슬롯에 장착되어 호스트 CPU에 연결된다. Hybrid TOE 프로토타입의 하드웨어 모듈은 FPGA 상에 구현되고, 소프트웨어 모듈은 ARM 프로세서 상에서 구현된다. 또한 하드웨어 모듈과 소프트웨어 모듈의 연동 메커니즘을 개발하였다. 본 논문에서는 실험을 통해 Hybrid TOE의 프로토타입이 호스트 CPU 상에서 발생하는 부하를 감소시킴을 입증하고, 하드웨어/소프트웨어 연동 구조의 효과를 분석하였다. 마지막으로 Hybrid TOE의 완성을 위해 필요한 요소들을 분석하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를 소개하고, 3장에서는 Hybrid TOE에 기반한 네트워크 어댑터의 구조를 설명한다. 4장에서는 Hybrid TOE 프로토타입의 구현에 대해 설명한다. 5장에서는 실험 결과와 분석을 제시한다. 마지막으로 6장에서는 결론과 향후 연구를 제시한다.

2. 관련 연구

TOE를 개발하는 기존의 방안 중에서 소프트웨어 기반 TOE 구현의 사례로는 Intel사의 PRO/1000T IP Storage Adapter[3]가 있다. 이 제품은 기존의 Gigabit Ethernet 어댑터에 Intel 80200 StrongARM(200MHz) 프로세서를 장착하였으며, TCP/IP와 iSCSI(SCSI over IP) 프로토콜을 소프트웨어로 처리한다. 콜로라도 대학에서 이 어댑터를 사용하여 실험한 결과[4]에 따르면 단방향 대역폭이 최대 30 MB/s를 넘지 못하고 있는데, 이는 TCP/IP를 사용하는 일반적인 Gigabit Ethernet 어댑터가 최대70 MB/s 정도의 단방향 대역폭을 보여주는 것과 비교했을 때 성능이 2배 이상 낮은 것이다. 이

렇게 소프트웨어 기반 TOE 구현은 임베디드 프로세서의 성능이 호스트 CPU에 비해서 낮으므로 통신의 성능 측면에서 약점을 가지며, 네트워크의 속도가 빨라질수록 그 차이가 더욱 커진다. 반면에, 소프트웨어 기반 TOE 구현은 TCP/IP의 모든 기능을 소프트웨어로 처리하므로 구현이 쉽다는 장점을 가지며, 기능을 개선하거나 TCP/IP 이외의 프로토콜을 처리하는 기능을 추가하는 것이 용이하다.

하드웨어 기반 TOE의 구현 사례로는 Alacritech사의 SLIC[5], Adaptec사의 NAC-7711[6], QLogic사의ISP 4010[7] 등이 있으며, 이들 모두가 Gigabit Ethernet을 지원하고 있다. 제작사들의 발표 자료에 의하면 하드웨어 기반 TOE 제품들은 대체로 100 MB/s 정도의 단방향 대역폭을 보유하고 있으며[8-10], 이는 Gigabit Ethernet의 최대 성능에 근접한 것이다. 그러나, 하드웨어 기반 TOE 구현은 전용 ASIC 구현에 많은 시간과 비용이 소모되고, 구현된 하드웨어에서 수정하거나 개선할 사항이 발생할 때마다 새로운 ASIC을 개발해야 한다는 단점을 가진다. 특히 가까운 미래에 IPv6 기반의 TOE에 대한 요구가 증가할 것으로 예상되는데, 현재까지 개발된 IPv4 기반의 TOE 중에서 하드웨어 기반 TOE의 경우 이에 신속히 대응하기가 어렵다. 또한 RDMA(Remote Direct Memory Access) 등과 같은 새로운 상위 프로토콜까지 네트워크 어댑터에서 처리하려는 요구가 발생할 때에도 이에 효과적으로 대응하기 어렵다.

그리고, 소프트웨어 기반 TOE 구현과 하드웨어기반 TOE 구현 사례들을 제외하면 본 논문에서 제안하는 바와 같이 하드웨어 모듈과 범용 임베디드 프로세서 상에서 운용되는 소프트웨어 모듈을 결합하여 TCP/IP 프로토콜을 처리하는 방식의 Hybrid TOE는 아직까지 개발된 사례가 없다.

3. Hybrid TOE 어댑터

그림 1은 Hybrid TOE [11]에 기반한 네트워크 어댑터 (Hybrid TOE 어댑터)의 구조를 나타낸다. Hybrid TOE 어댑터는 Hybrid TOE Module, TOE Interface, Memory Controller 및 Gigabit Ethernet Controller로 구성된다. Hybrid TOE Module은 Hybrid TOE 어댑터의 핵심 모듈로서, 두 개의 임베디드 프로세서 코어와 한 개의 하드웨어 모듈로 구성된다. 임베디드 프로세서는 소프트웨어로 처리할 TCP/IP 기능을 구현하며, 두 개의 임베디드 프로세서를 사용하여 송수신 과정을 분담하여 처리함으로써 호스트 CPU에 비해 성능이 떨어지는 임베디드 프로세서의 단점을 극복할 수 있다. 또한 이를 통해 송신 프로세스와 수신 프로세스 사이의 스케

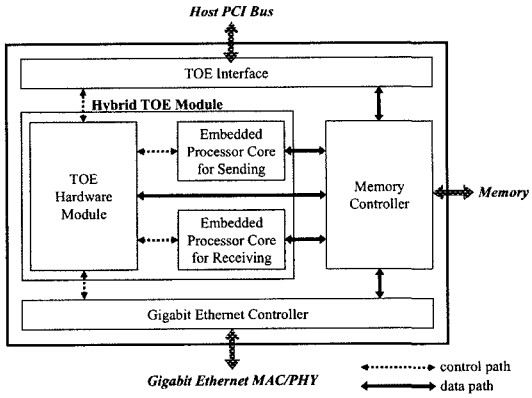


그림 1 Hybrid TOE 어댑터의 구조

줄링에 의한 작업 전환 오버헤드를 제거할 수 있다. TOE Hardware Module은 하드웨어로 처리할 TCP/IP 기능을 구현하며, TOE Interface는 호스트 CPU와의 인터페이스를 담당한다. Memory Controller는 패킷 버퍼의 운용에 필요한 버퍼 메모리를 관리하는 역할을 담당한다. 패킷 버퍼는 TCP, IP, 그리고 MAC 헤더들을 저장하기 위한 헤더 영역과 송수신 데이타를 저장하기 위한 데이타 영역으로 구성되며, 송수신 패킷을 저장하는 데 사용된다. Gigabit Ethernet Controller는 Gigabit Ethernet MAC/PHY 칩과의 인터페이스를 담당한다.

Hybrid TOE 어댑터를 사용한 통신의 진행 과정은 다음과 같다. 호스트 CPU의 사용자 프로그램이 Hybrid TOE를사용한 통신을 요청하면 이 요청은 호스트 운영체제의 TCP/IP 프로토콜 스택을 거치지 않고 TOE Interface로 직접 전달된다. TOE Interface는 이 요청을 Hybrid TOE Module로 전달하고, Hybrid TOE Module에서 하드웨어와 소프트웨어를 연동하여 요청된 작업을 처리한다. 요청된 작업이 데이타의 송신인 경우 Hybrid TOE Module은 패킷 버퍼를 생성한 후 원격 노드로 전송할 데이타를 호스트 CPU의 메인 메모리에서 패킷 버퍼의 데이타 영역으로 DMA를 사용하여 가져온다. Hybrid TOE Module은 패킷 버퍼의 헤더 영역에 TCP/IP 헤더와 MAC 헤더를 생성하여 패킷을 완성하고, 패킷 생성이 끝나면 Gigabit Ethernet Controller에 패킷의 전송을 요청한다. 마지막으로 Gigabit Ethernet Controller의 요청을 받은 Gigabit Ethernet MAC이 DMA를 사용하여 패킷 버퍼로부터 그 패킷을 가져가서 수신 노드로 전송한다. 수신 과정에서는 송신 과정의 역순으로 수신 패킷이 처리된다. 요청된 작업의 처리가 끝나면 TOE Interface가 호스트 CPU에 인터럽트를 발생시켜 처리가 완료되었음을 알리고, 호스트 CPU는 TOE Interface에 저장된 처리 결과를 읽어가서

이를 사용자 프로그램에 반환한다.

TCP/IP에서 수행하는 주요 기능으로는 (1) 연결 설정, (2) 패킷 버퍼의 생성 및 해제, (3) TCP/IP 헤더 생성 및 처리, (4) ACK (Acknowledgement) 패킷 생성 및 처리, (5)흐름 제어 등이 있다. 선행 연구[11]에서 리눅스의 TCP/IP 프로토콜 스택을 분석한 결과에 따르면 (2), (3), (4)와 리눅스의 작업 스케줄링에서 TCP/IP 처리 시간의 약 70~90%를 소모하는 것으로 나타났다. 따라서, TOE를 구현하기 위해서는 이 기능들의 최적화가 필수적인 것으로 분석되었다. Hybrid TOE 구조에서는 이러한 연구 결과를 바탕으로 (2), (3), (4)는 하드웨어로 구현하고, 나머지 TCP/IP 기능 및 작업 스케줄링은 임베디드 프로세서 상에서 소프트웨어로 구현한다.

4. Hybrid TOE 프로토타입

4.1 Hybrid TOE 프로토타입 보드

본 논문에서는 Hybrid TOE의 최종 구현에 앞서 FPGA와 단일 임베디드 프로세서를 장착한 Hybrid TOE 프로토타입 보드를 제작하였다. 그림 2는 Hybrid TOE 프로토타입 보드의 구조를 보여준다. 이 보드에는 FPGA, 임베디드 프로세서, Gigabit Ethernet MAC/PHY, SDR AM, FLASH 메모리 등이 탑재되어 있다. 하드웨어 모듈들이 구현될 FPGA는 Xilinx사의 Virtex-II Pro (XC 2VP30)를 채택하였고, 소프트웨어 모듈을 운용할 임베디드 프로세서는 삼성전자의 ARM 프로세서(S3C2410X)를 채택하였다. Gigabit Ethernet MAC/PHY chipset으로는 National사의 DP82820/DP83865을 사용하였다. SD RAM과 FLASH 메모리는 각각 64MB와 16MB의 용량을 가진다.

FPGA는 TOE Interface, TOE Hardware Module, HW/SW Interface 및 Gigabit Ethernet Controller로 구성된다. TOE Interface는 호스트 CPU와 Hybrid TOE Module 사이의 인터페이스를 담당하며, TOE Hardware Module은 송신 과정에서 패킷 버퍼와 관련된 세 가지 기능을 처리한다. HW/SW Interface는 하드웨어/소프트웨어 연동 메커니즘을 구성하는 핵심 요소로서 TOE Hardware Module과 ARM 프로세서 사이의 인터페이스를 제공한다. Gigabit Ethernet Controller는 Gigabit Ethernet MAC 칩을 제어하는 역할을 담당한다.

호스트 CPU와 Hybrid TOE 프로토타입 보드는 64bit/66MHz의 호스트 PCI 버스를 통해 연결되고, FPGA 상의 하드웨어 모듈들은 66MHz PCI 버스 클럭을 사용하여 동작한다. ARM 프로세서는 135MHz의 core clock을 사용하여 동작하고, 67.5MHz의 system clock으로 동작하는 메모리 버스를 통해 SDRAM, FLASH 메모리, FPGA에 접근한다.

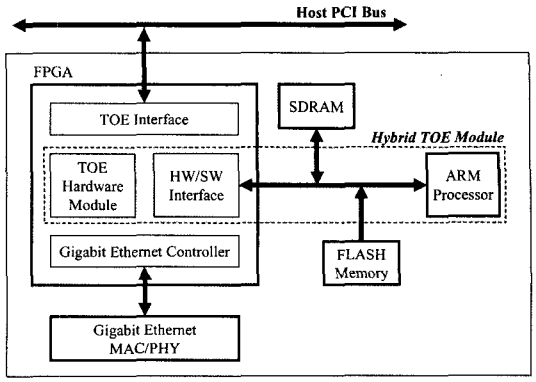


그림 2 Hybrid TOE 프로토타입 보드의 구조

4.2 Hybrid TOE를 위한 프로토콜 스택과 TOE Interface

그림 3은 Hybrid TOE 프로토타입을 사용하기 위한 프로토콜 스택의 구조를 보여준다. 본 프로토콜 스택은 호스트 CPU의 BSD 소켓 계층이 TCP/IP 계층을 거치지 않고 Hybrid TOE 프로토타입 보드의 디바이스 드라이버로 직접 연결되는 구조를 가진다. 그리고, 디바이스 드라이버는 호스트 PCI 버스를 통해 Hybrid TOE 프로토타입의 TOE Interface에 직접 접근한다.

TOE Interface는 호스트 PCI 버스와의 연결을 위해 PCI Controller를 내장하고, Command Buffer, Completion Buffer, Send Buffer, Receive Buffer를 통해 BSD 소켓 계층의 TCP/IP 함수를 지원하는 소켓 기반 인터페이스를 제공한다. Command Buffer에는 호스트 CPU에서 호출한 소켓 함수에 대응하는 Command Request가 저장되어 Hybrid TOE Module로 전달된다. Completion Buffer에는 Hybrid TOE Module이 처리 결과를 보고하는 Completion Result가 저장되어 호스트 CPU로 전달된다. Send Buffer는 ARM 프로세서가 SDRAM으로 복사할 데이타를 메인 메모리에서 DMA (Direct Memory Access)로 읽어와서 저장하는 데 사용되며, Receive Buffer는 DMA를 사용하여 메인 메모리로 전송될 수신 데이타를 저장하는 데 사용된다.

본 프로토콜 스택의 동작 과정은 다음과 같다. 호스트 CPU의 사용자 프로그램에서 BSD 소켓 함수를 사용하여 TCP/IP 통신을 요청하면, BSD 소켓 계층에서 곧바로 디바이스 드라이버를 호출한다. 디바이스 드라이버는 소켓 함수에 대응하는 Command Request를 생성하여 TOE Interface 내부의 Command Buffer에 저장한다. Hybrid TOE Module에서는 Command Request를 해석하여 해당되는 작업을 수행한다. 요청된 작업의 처리가 끝나면 Hybrid TOE Module은 처리 결과를 반환하

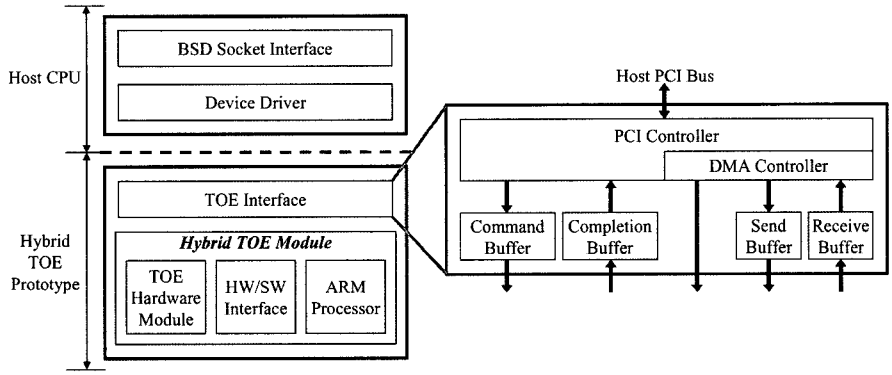


그림 3 Hybrid TOE 프로토타입의 프로토콜 스택

는 Completion Result을 생성하여 TOE Interface 내부의 Completion Buffer에 저장하고, TOE Interface는 호스트 CPU에 처리가 끝났음을 알리는 인터럽트를 발생시킨다.

4.3 TOE 소프트웨어 모듈

그림 4와 같은 구조를 가지는 TOE 소프트웨어 모듈은 ARM 프로세서 상에서 임베디드 리눅스를 사용하여 개발하였다. 임베디드 리눅스는 다른 운영체제에 비해서 네트워크 성능이 우수하고, TCP/IP 프로토콜 스택을 포함한 커널의 수정이 용이하기 때문에 선택되었다. 또한 임베디드 리눅스를 사용하면 TCP/IP를 기반으로 하는 상위 프로토콜을 네트워크 어댑터에서 오프로딩하는 방향으로 TOE를 응용할 때 편리한 개발 환경을 제공할 수 있다.

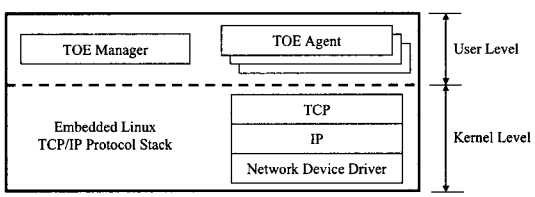


그림 4 TOE 소프트웨어 모듈의 구조

TOE 소프트웨어 모듈에서는 사용자 수준 쓰레드로 구현된 TOE Agent들을 사용하여 통신을 수행하고, TOE Manager를 사용하여 TOE Agent들을 관리한다. TOE Agent는 호스트 CPU의 사용자 프로그램에서 TCP/IP 통신을 위해 하나의 연결을 생성할 때마다 이에 대응하여 ARM 프로세서 상에서 하나씩 생성되며, Command Request를 해석하여 관련된 소켓 함수를 호출함으로써 통신을 수행한다. TOE Manager는 연결 설정 및 해제시 TOE Agent의 생성과 소멸을 관리하고, 호스트 CPU의 데이타 송수신 명령이 발생할 경우, 해당 TOE Agent를 찾아 호출하는 역할을 담당한다. 디바이스 드라이버는 Gigabit Ethernet Controller의 사용을 지원하기 위해 개발되었다.

TOE 소프트웨어 모듈에서 데이타의 송신을 처리하는 과정은 다음과 같다. 그림 3의 TOE Interface에 Command Request가 저장되면 HW/SW Interface가 이를 감지하여 ARM 프로세서에 인터럽트를 발생시킨다. 인터럽트를 받은 ARM 프로세서에서는 인터럽트 핸들러가 TOE Manager를 호출하고, TOE Manager는 Command Request를 처리할 TOE Agent를 찾아서 호출한다. TOE 소프트웨어 모듈에서 인터럽트 핸들러는 리눅스 커널에서 관리하고, TOE Manager와 TOE Agent는 사용자 수준 쓰레드이기 때문에 이들 사이의 호출은 작업 전환을 통해 수행된다. 이는 TOE 소프트웨어 모듈에서 많은 오버헤드를 유발하는 원인이 되는데, Hybrid TOE의 최종 구현에서는 이러한 작업 전환을 제거한 메커니즘을 개발할 예정이다.

TOE Agent에서 데이타 송신을 위한 첫 단계는 TCP 계층에서 소켓 버퍼를 생성하는 것이다. 이 소켓 버퍼 내에서 헤더와 데이타를 저장할 메모리 영역으로는 4.4절에서 설명할 TOE 하드웨어 모듈에서 제공하는 패킷 버퍼를 사용한다. 소켓 버퍼를 생성한 후 TOE 하드웨어 모듈은 전송할 데이타를 메인 메모리로부터 패킷 버퍼로 복사하고, TOE 소프트웨어 모듈은 TCP/IP 헤더를 SDRAM에 생성한 후 패킷 버퍼로 복사하여 패킷을 완성한다. 이후 디바이스 드라이버가 Gigabit Ethernet Controller에 패킷의 전송을 요청하면 Gigabit Ethernet MAC이 패킷 버퍼에 저장되어 있는 패킷을 DMA로 가져가서 전송한다.

TOE 소프트웨어 모듈에서 수신을 처리하는 과정은 다음과 같다. 수신된 패킷이 Gigabit Ethernet Controller의 수신 버퍼(RX buffer)에 저장되면, Gigabit Ethernet Controller는 인터럽트를 사용하여 ARM 프로세서

에 이를 알려준다. 인터럽트를 받은 ARM 프로세서에서는 디바이스 드라이버가 소켓 버퍼를 생성한 후 Gigabit Ethernet Controller의 수신 버퍼에 저장되어 있는 패킷을 소켓 버퍼로 복사한다. ARM 프로세서 내부의 TCP/IP 모듈에서TCP/IP 수신 처리가 끝나면 TOE Agent가 수신 데이타를 TOE Interface의 Receive Buffer로 복사한다. TOE Interface는 DMA 쓰기를 통해 수신 데이타를 메인 메모리 상의 사용자 버퍼로 전송하고, DMA가 끝나면 호스트 CPU에 인터럽트를 보내 데이타의 수신을 알린다.

4.4 TOE 하드웨어 모듈 및 하드웨어/소프트웨어 연동 메커니즘

본 논문에서는 Hybrid TOE 프로토타입을 위해 송신 과정에서 패킷 버퍼의 생성과 사용에 관련된 세 가지의 하드웨어 유닛을 개발하였고, TOE 하드웨어 모듈(TOE Hardware Module)과 ARM 프로세서 사이에서 하드웨어/소프트웨어 연동 메커니즘을 지원하는 HW/SW Interface를 개발하였다.

그림 5는 TOE 하드웨어 모듈과 HW/SW Interface의 구조를 나타낸다. TOE 하드웨어 모듈에서MAU (Memory Allocation Unit)는 데이타 및 헤더를 저장하는 패킷 버퍼를 생성하기 위해 패킷 버퍼 메모리를 관리한다. MAU는 패킷 버퍼 메모리를 2048 bytes 크기의 슬롯(slot) 단위로 관리하며, 이를 통해 Ethernet의 MTU 크기인 1514 bytes를 지원하기에 충분한 크기의 메모리 공간을 제공한다. DFU(Data Fetch Unit)는 TOE Interface 내부의 DMA 엔진을 사용하여 메인 메모리에서 패킷 버퍼의 데이타 영역으로 송신 데이타를 복사한다. 이 때 데이타는 TOE Interface의 Send Buffer를 거치지 않고 직접 패킷 버퍼로 복사된다. PCU(Partial Checksum Calculation Unit)는 DFU에 의해 데이타가 복사되는 동안 데이타의 부분 체크섬(partial checksum)을 계산한다. 부분 체크섬은 TCP 헤더가 완성된 후에 TCP 헤더의 체크섬과 함께 TCP

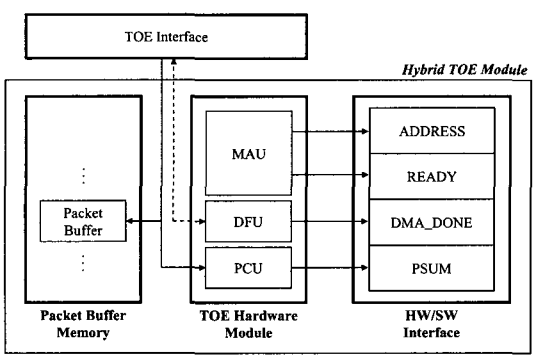


그림 5 TOE 하드웨어 모듈과 HW/SW Interface의 구조

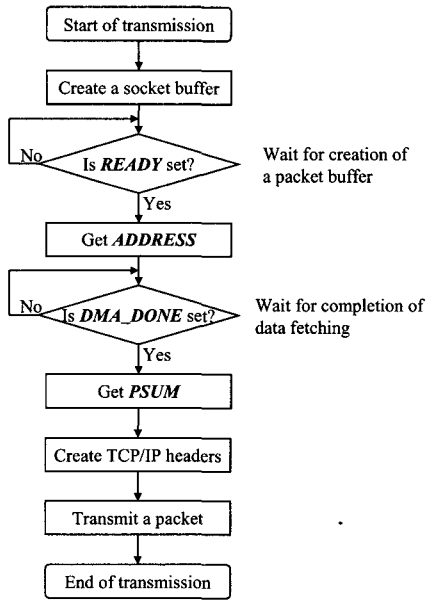


그림 6 ARM 프로세서 상에서의 데이타 송신처리 흐름도

패킷 전체의 체크섬을 계산하는 데 사용된다. 이 때 PCU에 의한 부분 체크섬 계산은 DFU에 의한 데이타 복사와 동시에 수행되기 때문에 통신의 지연시간 (latency)에 영향을 끼치지 않는다.

HW/SW Interface에서 ADDRESS 필드는 MAU에 의해 할당된 패킷 버퍼의 주소를 저장하고, READY 필드는 패킷 버퍼가 준비되었는지의 여부를 나타낸다. DMA_DONE 필드는 DFU에 의한 데이타 복사가 끝났는지의 여부를 나타내고, PSUM 필드에는 데이타의 부분 체크섬이 저장된다.

그림 6은 하드웨어/소프트웨어 연동 메커니즘에서 ARM 프로세서가 HW/SW Interface의 필드들을 사용하여 송신을 수행하는 과정을 흐름도로 보여준다. ARM 프로세서는 소켓 버퍼 구조체를 생성할 때 READY 필드를 폴링하여 MAU에서 패킷 버퍼의 생성이 완료되었는지 검사한다. READY 필드가 세팅되면 ARM 프로세서는 ADDRESS 필드를 읽어서 패킷 버퍼의 주소를 획득한다. 이후 DMA_DONE 필드를 통해 데이타 복사와 부분 체크섬 계산의 완료를 인지하면 PSUM 필드에서 부분 체크섬을 읽어서 소켓 버퍼에 저장한다. 데이타 복사가 끝나면 ARM 프로세서는 TCP/IP 헤더 및 MAC 헤더를 생성하여 패킷 버퍼의 헤더 영역으로 복사하고, 생성된 패킷은 수신 노드로 전송된다.

5. 실험 및 분석

본 장에서는 Hybrid TOE 프로토타입의 성능을 측정

하였다. 두 노드는 프로토타입 보드를 탑재하고, 스위치를 거치지 않고 직접 연결하였다. 각 노드는 1.8 GHz Intel Xeon CPU, 512 MB의 메인 메모리 및 64 bit/66 MHz PCI bus를 탑재하고 있다. 호스트 CPU의 운영체제로는 리눅스 커널 2.4.7-10을 사용하였다. 성능 비교를 위한 일반 Gigabit Ethernet 어댑터로는 Intel사의 PRO/1000MT Server Adapter를 사용하였다.

5.1 CPU 점유율

본 절에서는 TCP/IP 기반 통신을 수행할 때 Hybrid TOE 프로토타입(Hybrid TOE)을 사용한 경우와 일반 Gigabit Ethernet 어댑터(GBE)를 사용한 경우의 호스트 CPU 점유율을 비교하였다. 그림 7에서 제시하는 바와 같이 GBE를 장착한 시스템의 경우 호스트 CPU 점유율은 데이타의 크기에 따라 50~80% 정도인 반면에, Hybrid TOE를 장착한 시스템의 경우 호스트 CPU 점유율은 약 9%로 일정하며 GBE를 장착한 시스템보다 훨씬 낮은 것을 알 수 있다. 이러한 결과는 Hybrid TOE가 호스트 CPU에서 TCP/IP를 처리하는 부하를 크게 감소시키는 효과를 입증하는 것이다.

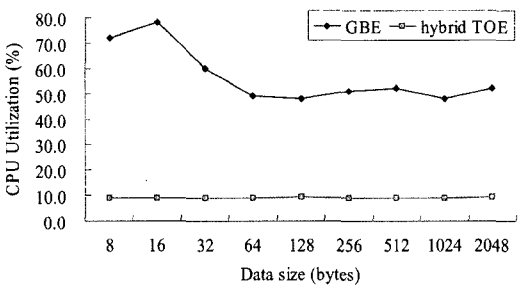


그림 7 Hybrid TOE와 GBE의 CPU 점유율

5.2 Hybrid TOE와 Software TOE

본 절에서는 Hybrid TOE 프로토타입에 적용된 하드웨어/소프트웨어 연동 메커니즘의 유용성을 검증하기 위해 Hybrid TOE 프로토타입과 소프트웨어 TOE의 성능을 비교하였다. 소프트웨어 TOE는 Hybrid TOE 프로토타입에서 구현된 세 개의 하드웨어 유닛을 포함한 모든 TCP/IP 함수들을 ARM 프로세서 상에서 소프트웨어로 처리한다.

표 1은 Hybrid TOE 프로토타입과 소프트웨어 TOE에서 송신 과정을 처리할 때 소모하는 시간을 분석한 결과를 제시하며, 하드웨어 구현에 의해 Hybrid TOE 프로토타입에서 성능이 향상된 기능들을 구체적으로 보여준다. 이러한 결과는 하드웨어 구현과 함께 하드웨어/소프트웨어 연동 메커니즘을 사용함으로써 Hybrid TOE 프로토타입이 소프트웨어 TOE보다 좋은 성능을 가짐을 보여준다.

표 1 데이타 전송시 소요시간 분석 (단위 : μs)

Label	관련 operation	Hybrid TOE	Software TOE
S1	Enter TCP layer from interrupt handler	469.2	469.2
S2	Create socket buffer	20.2	20.2
	Create packet buffer	**1.0**	**10.0**
S3	**Fetch data from main memory**	**7.7**	**69.6**
S4	Create TCP header	48.9	48.9
S5	Enter IP layer from TCP layer	52.6	52.6
S6	Create IP header	49.0	49.0
S7	Enter device driver from IP layerand Create MAC header	32.7	32.7
S8	**Copy packet to TX buffer**	**12.2**	**690.5**
S9	Complete transmission	104.2	104.2
송신 처리 시간		797.8	1547.4

Hybrid TOE 프로토타입에서 하드웨어 구현으로 인해 성능이 향상된 부분은 Create socket buffer (S2), Fetch data (S3) 및 Copy packet to TX buffer(S8)이다. S2는 송신과정에서 소켓 버퍼를 생성하는 시간으로, 소켓 버퍼에 포함된 패킷 버퍼의 생성이 하드웨어로 구현되어 처리시간이 10μs에서 1μs로 감소되었기 때문이다. S3과 S8을 처리하는 시간이 Hybrid TOE 프로토타입에서 크게 감소한 이유는 소프트웨어 TOE에서는 SDRAM상에 패킷 버퍼가 유지되는 반면에 Hybrid TOE 프로토타입에서는 FPGA 상에 패킷 버퍼가 유지됨으로 인해 데이타 복사 회수가 줄었기 때문이다. 소프트웨어 TOE의 경우, 메인 메모리의 데이타는 TOE Interface의 Send Buffer를 거쳐 SDRAM으로 복사된다(S3). ARM 프로세서는 SDRAM 상에서 패킷을 생성한 후, 생성된 패킷을 Gigabit Ethernet Controller 내부의 송신 버퍼(TX buffer)로 복사한다(S8). 송신 버퍼에 저장된 패킷은 Gigabit Ethernet MAC에 의해 수신 노드로 전송된다. 반면, Hybrid TOE 프로토타입에서는 FPGA 내부에 패킷 버퍼를 유지하기 때문에 소프트웨어 TOE와 같이 SDRAM으로 데이타를 복사할 필요가 없으며, 따라서 데이타 복사에 소요되는 시간이 감소하여 성능이 향상된다.

5.3 성능 예측

본 절에서는 Hybrid TOE의 완성을 위해 최적화해야 할 요소들을 표 2와 같이 분석하고, 이 분석을 바탕으로 하여 Hybrid TOE가 완성되었을 때의 성능을 예측한다. 표 2에서 F1~F3은 송신 과정에서 하드웨어로 구현해야 할 기능이며, F4~F5는 소프트웨어를 최적화하여 구현할 기능들이다. 수신 처리 과정은 송신 처리 과정과 유사한 방법으로 최적화할 수 있다.

패킷 버퍼의 생성에 걸리는 시간은 Software TOE의

표 2 Hybrid TOE의 완성을 위해 최적화할 요소들 (단위 : μs)

Label	최적화 요소	표 1에서 관련된 요소	Hybrid TOE 프로토타입의 처리 시간	Hybrid TOE의 처리 시간
F1	Manage socket buffer by hardware	S2	21.2	1.0
F2	Create TCP, IP, and MAC headersby hardware	S4	48.9	1.5
		S6	49.0	1.5
		S7	32.7	1.5
		S8	12.2	0.0
F3	Optimize completion flow	S9	104.2	10.0
F4	Enter TCP layer directlyfrom interrupt handler	S1	469.2	3.6
F5	Remove overheadon entering every layer	S5	52.6	3.6
		S7	32.7	3.6
합계			822.7	26.3

경우 약 10μs가 소요되었다. 이를 하드웨어로 구현한 Hybrid TOE 프로토타입에서는 MAU가 패킷 버퍼 메모리에서 비어있는 슬롯을 확보하여HW/SW Interface의 ADDRESS 필드와 READY 필드를 세팅한 후 ARM 프로세서에서 이들을 읽어가는 데 약 1 μs (70 clocks)가 필요하였다. 차후 완전한 Hybrid TOE에서는 패킷 버퍼뿐만 아니라 이를 포함하는 소켓 버퍼 전체를 MAU가 관리하도록 구조를 개선할 예정이다 (F1). 이 경우 소켓 버퍼를 생성하는 시간은 프로토타입에서 패킷 버퍼를 생성하는 시간과 동일하므로, 소켓 버퍼 생성에는 약 1 μs가 걸릴 것으로 예상된다.

완전한 Hybrid TOE에서 TCP 헤더, IP헤더, MAC 헤더를 생성하는 과정을 하드웨어로 구현하여 처리하는 경우 (F2)의 성능은 다음과 같이 예상할 수 있다. 이러한 헤더들을 생성하기 위해서는 TCP/IP 통신의 연결 정보들을 관리하는 메모리에서 해당되는 연결 정보를 획득하는 과정과 임시 헤더 버퍼의 각 필드들을 채워 넣는 과정, 그리고 임시 헤더 버퍼의 내용을 패킷 버퍼로 복사하는 과정이 필요하다. TCP, IP, MAC 헤더를 생성하는 모듈들은 MAU의 처리 시간과 비슷하게 각각 100 클럭 이내에 작업을 완료할 수 있을 것으로 보이며, 따라서 모든 헤더 생성에 4.5 μs 이내의 시간이 소요될 것으로 예상된다. 또한, 완전한 Hybrid TOE에서 모든 헤더들을 패킷 버퍼 상에서 하드웨어로 직접 생성하게 되면 임베디드 프로세서가 SDRAM 상에서 헤더를 생성하여 Gigabit Ethernet Controller의 송신 버퍼로 복사하는 오버헤드가 제거된다.

Completion Result를 생성하는 과정을 하드웨어로 구현할 경우 MAU와 마찬가지로 수십 클럭 이내에 처리가 가능하므로 처리에 약 1 μs의 시간이 소모될 것으로 예상된다. 그리고, 호스트 CPU에 인터럽트를 발생시켰을 때 호스트 CPU가 반응하여 Completion Result를 읽어가는 시간은 실험 결과 약 9 μs가 걸렸다. 따라서, Completion Result를 생성하는 과정을 하드웨어로 구현하여 TOE Interface에서 호스트 CPU에 인터럽트를 발생시키는 과정과 연동하면 호스트 CPU에서 Completion Result를 읽어갈 때까지 약 10 μs의 시간이 걸릴 것으로 보인다(F3).

F4는 Command Buffer에 호스트 CPU의 요청이 도착했을 때 인터럽트 핸들러, TOE Manager, TOE Agent를 거쳐서 임베디드 프로세서 내부의 TCP/IP 프로토콜 스택이 호출되는 과정을 최적화하는 것이다. 이 과정은 세 번의 프로세서 호출을 필요로 하고 각 호출은 Linux 스케줄러에 의해 동작하므로 수행시간이 매우 크다. 차후의 완전한 Hybrid TOE에서는 인터럽트 핸들러가 직접 임베디드 프로세서 내부의TCP/IP 프로토콜 스택을 호출하도록 하며, 이러한 호출 과정도 스케줄러가 아닌 함수 호출에 의해서 이루어지도록 하여 최적화를 수행할 것이다. 실험 결과 인터럽트 핸들러에서 커널 내부의 함수를 호출하는 데에는 약 3.6 μs가 걸렸으며, 따라서 F1에서도 이 정도의 시간이 걸릴 것으로 예상된다.

F5에서는 프로토콜 스택의 각 계층 사이에서 발생하는 불필요한 기능들과 스케줄링을 제거하고자 하며, 이 경우 계층간 이동 시간은 F4에서와 유사하게 3.6 μs가 걸릴 것으로 예상한다. 따라서, 표 1의 S6과 S8에서 발생하는 2번의 계층간 이동에 임베디드 프로세서 상에서는 7.2 μs의 시간이 소요될 것으로 예상된다.

차후 수신 과정을 송신 과정과 동등한 수준으로 최적화하여 Hybrid TOE을 완성한 경우 1024-byte의 데이타를 전송할 때 통신의 지연 시간은 다음과 같이 예상할 수 있다. Hybrid TOE에서 송신을 처리하는 시간은 표 2의 26.3 μs와 표 1의 S3에서 걸리는 7.7 μs를 합쳐서 약 34 μs가 걸릴 것으로 보인다. 수신 과정을 송신 과정과 동등하게 최적화하면 결국 송신과 수신을 처리하는 데 약 70 μs가 걸릴 것으로 예상된다. 지연 시간을 구성하는 요소들 중에서 Gigabit Ethernet 네트워크

상의 propagation delay와 호스트 CPU에서 Command Request를 생성하여TOE Interface에 저장하는 시간은 실험 결과 약 40 μs가 걸렸으며, 이는 Hybrid TOE의 구현이 완성되어도 변하지 않는 시간이다. 따라서, 1024-byte의 데이타를 전송하는 데 걸리는 지연 시간은 약 110 μs가 될 것으로 예상된다. 이와 같은 지연시간은 하드웨어로 TCP/IP를 처리하는 Adaptec사NAC-7711의 지연시간인 90 μs [10]에는 미치지 못하지만 일반 Gigabit Ethernet 어댑터의 지연 시간인 120 μs [10]보다는 우수한 성능을 보유할 것으로 기대된다.

6. 결론 및 향후 과제

본 논문에서는 Hybrid TOE 구조를 제안하고, FPGA와 ARM 프로세서를 탑재한 Hybrid TOE 프로토타입을 개발하였다. Hybrid TOE 프로토타입의 사용을 위해 호스트 CPU에서는 소켓 계층이 TCP/IP를 거치지 않고 디바이스 드라이버 계층에 직접 연결되는 프로토콜 스택을 개발하고, 프로토타입 보드에서는 호스트 CPU와의 인터페이스를 담당하는 TOE Interface를 개발하여 FPGA 상에 구현하였다. 그리고, FPGA를 사용하여 Hybrid TOE 프로토타입의 하드웨어 모듈들을 구현하였고, ARM 프로세서 상에서 TOE 소프트웨어 모듈을 개발하였다. 또한 하드웨어와 소프트웨어의 연동 메커니즘을 개발하고, 이를 지원하기 위한 HW/SW Interface를 FPGA 상에 구현하였다. 실험 결과 Hybrid TOE 프로토타입의 CPU 점유율은9% 이하였으며, 호스트 CPU에서TCP/IP를 처리할 때의 CPU 점유율은 50%를 초과하였다. 이는 Hybrid TOE 프로토타입이 호스트 CPU에서발생하는 부하를 크게 감소시키는 효과를 입증한다. 또한, 하드웨어/소프트웨어를 연동한 Hybrid TOE 프로토타입이 소프트웨어 TOE에 비해서 성능이 우수함을 보였다. 이러한 실험 결과를 바탕으로 하여 Hybrid TOE의 개발이 완료되었을 때, 1024-byte의 데이타 전송시 약 113 μs의 지연시간을 보일 것으로 예상하였으며, 이 예측 결과는 기존의 하드웨어 기반 TOE 및 Gigabit Ethernet 어댑터와 유사한 성능을 보이는 것으로 기대된다. 향후 과제로는 제시한 요소들을 모두 구현하여 Hybrid TOE를 완성하고자 한다.

참 고 문 헌

[1] N. Bierbaum, "MPI and Embedded TCP/IP Gigabit Ethernet Cluster Computing," *Proceedings of 27th Annual IEEE Conference on Local Computer Networks 2002*, pp. 733-734, 2002.

[2] E. Yeh, H. Chao, V. Mannem, J. Gervais, and B. Booth, "Introduction to TCP/IP Offload Engine (TOE)," *10 Gigabit Ethernet Alliance*, 2002.

[3] Intel Corporation, "Intel PRO/1000T IP Storage Adapter," Data Sheet, 2003, available at http://www.intel.com

[4] S. Aiken, D. Grunwald, A. R. Pleszkun, and J. Willeke, "A Performance Analysis of the iSCSI Protocol," *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp. 123-134, 2003.

[5] Alacritech, Inc., "SLIC Technology Overview," Technical Review, 2002, available at http://www.alacritech.com

[6] Adaptec, Inc., "Adaptec TOE NAC 7711," Data Sheet, 2003, available at http://graphics.adaptec.com

[7] QLogic Corporation, "iSCSI Controller," Data Sheet, 2003, available at http://download.qlogic.com

[8] Lionbridge Technologies, Inc., "Alacritech SES1001T: iSCSI HBA Competitive Analysis," VeriTest Benchmark Report, 2004, available at http:// www.veritest. com

[9] Adaptec, Inc., "Unleashing File Server Potential with Adaptec GigE NAC 7711," Benchmark Report, 2003, available at http://graphics.adaptec.com

[10] H. Ghadia, "Benefits of full TCP/IP offload (TOE) for NFS Services," *Proceedings of 2003 NFS Industry Conference*, 2003, available at http://nfsconf.com

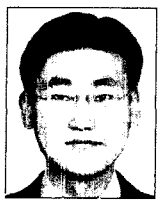
[11] S.-C. Oh, H. Jang, and S.-H. Chung, "Analysis of TCP/IP protocol stack for a Hybrid TCP/IP Offload Engine," *Proceedings of the 5th International Conference on Parallel and Distributed Computing*, Applications and Technologies, pp. 406-409, 2004.

장 한 국

1999년 부산대학교 컴퓨터공학과 학사 2001년~부산대학교 컴퓨터공학과 석사 수료. 2001년~현재 부산대학교 컴퓨터공학과 석박사 통합과정. 관심분야는 컴퓨터 구조, 클러스터 시스템, TOE, RDMA

정 상 화

1985년 서울대학교 전기공학과 학사. 1988년 Iowa State Univ. 컴퓨터공학과 석사. 1993년 Univ. of Southern California 컴퓨터공학과 박사. 1993년~1994년 Univ. of Central Florida 컴퓨터공학과 조교수. 1994년~현재 부산대학교 컴퓨터공학과 교수, 컴퓨터및정보통신연구소 연구원. 2002년~2003년 Oregon State Univ. 컴퓨터공학과 초빙교수. 관심분야는 클러스터 시스템, 병렬처리, TOE, RDMA, VOD, RFID



오 수 철

1995년 부산대학교 컴퓨터공학과 학사 1997년 부산대학교 컴퓨터공학과 석사 1997년~1998년 LG전자 멀티미디어연구소 연구원. 1998년~2003년 부산대학교 컴퓨터공학과 박사. 2003년~2004년 (주) 아이온테크 연구원. 2004년~2005년 부산대학교 BK21사업단 기금교수. 2005년~현재 한국전자통신연구원 선임연구원. 관심분야는 클러스터 시스템, TOE, VOD