고성능 경량 TCP/IP를 이용한 소프트웨어 기반 TCP/IP 오프로드 엔진 구현

(Implementation of a TCP/IP Offload Engine Using High Performance Lightweight TCP/IP)

전 용 태 † 정 상 화 †† 윤 인 수 †††

(Yong-Tae Jun) (Sang-Hwa Chung) (In-Su Yoon)

요 약 최근 이더넷 기술은 기가비트급의 대역폭을 넘어서 10 기가비트급으로 빠른 속도로 발전하고 있다. 이러한 고속 네트워크 환경에서는 호스트 CPU가 운영체제내의 TCP/IP를 처리하는 기존의 방식은 호스트 CPU에 많은 부하를 야기하며, 그 결과 실제 수행되어야 할 사용자 응용 프로그램에 충분한 컴퓨팅 파워를 제공하지 못한다. 이러한 문제점의 해결을 위해 네트워크 어댑터에서 TCP/IP를 처리하도록 하는 TCP/IP Offload Engine (TOE)이 연구되고 있다. 본 논문에서는 TOE를 위한 고성능의 경량 TCP/IP를 구현하였으며, 이를 임베디드 시스템에 실제 적용하여 검증 및 실험을 수행하였다. 본 논문에서 구현한 고성능의 경량 TCP/IP는 기존 TCP/IP의 기본적인 기능들인 흐름제어, 혼잡제어, 재전송, 지연 ACK, Out-of-Order 패킷처리 등을 지원한다. 또한 본 논문에서 구현한 고성능의 경량 TCP/IP는 기가비트 이더넷 MAC에서 하드웨어적으로 지원하는 TCP segmentation offload(TSO), Checksum offload(CSO), 인터럽트 coalescing 기능 등을 이용하도록 구현하였다. 그리고 데이타를 전송할 때, 호스트 사용자 메모리에서 네트워크 어댑터의 메모리로 데이타를 복사하는 부하를 제거하였다. 또한 재전송해야 할 경우를 대비해 전송한 데이타에 대한 복사본을 네트워크 어댑터의 메모리에 저장하는 방법을 개선하여 지연시간 및 대역폭 성능을 향상시켰다. 본 논문에서 구현한 고성능의 경량 TCP/IP를 이용한 소프트웨어 기반 TOE는 6% 이하의 호스트 CPU 사용률과 453Mbps의 최대 대역폭을 보인다.

키워드 : TCP/IP Offload Engine, 임베디드 시스템, TCP/IP, 기가비트 이더넷

Abstract Today, Ethernet technology is rapidly developing to have a bandwidth of 10Gbps beyond 1Gbps. In such high-speed networks, the existing method that host CPU processes TCP/IP in the operating system causes numerous overheads. As a result of the overheads, user applications cannot get the enough computing power from the host CPU. To solve this problem, the TCP/IP Offload Engine (TOE) technology was emerged. TOE is a specialized NIC which processes the TCP/IP instead of the host CPU. In this paper, we implemented a high-performance, lightweight TCP/IP (HL-TCP) for the TOE and applied it to an embedded system. The HL-TCP supports existing fundamental TCP/IP functions; flow control, congestion control, retransmission, delayed ACK, processing out-of-order packets. And it was implemented to utilize Ethernet MAC's hardware features such as TCP segmentation offload (TSO), checksum offload (CSO) and interrupt coalescing. Also we eliminated the copy overhead from the host memory to the NIC memory when sending data and we implemented an efficient DMA mechanism for the TCP retransmission. The TOE using the HL-TCP has the CPU utilization of less than 6% and the bandwidth of 453Mbps.

Key words : TCP/IP Offload Engine, Embedded Systems, TCP/IP, Gigabit Ethernet

---

· 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/차세대물류IT기술연구사업단)

논문접수 : 2006년 11월 21일
심사완료 : 2008년 3월 17일

† 정 회 원 : 부산대학교 컴퓨터공학과
ytjun@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
(Corresponding author임)
††† 학생회원 : 부산대학교 컴퓨터공학과
isyoon@pusan.ac.kr

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제4호(2008.6)

1. 서 론

최근 이더넷 기술은 기가비트급의 대역폭을 넘어서 10 기가비트급으로 빠른 속도로 발전하고 있다. 이러한 고속 네트워크 환경에서는 호스트 CPU가 운영체제내의 TCP/IP를 처리하는 기존의 방식은 호스트 CPU에 많은 부하를 야기한다. 결국 이는 전체 시스템의 성능을 저하시킨다. 이러한 문제점을 극복하기 위해 네트워크 어댑터에서 TCP/IP를 처리하는 TOE기술이 활발히 연구되고 있다.

TOE는 크게 두 가지 방법으로 개발이 이루어지고 있다. 첫 번째 방법은 TCP/IP를 하드웨어적으로 처리하는 전용 칩(ASIC)을 개발하는 것이다. 이 방법은 통신 성능이 우수한 반면, 하드웨어로 구현되었기 때문에 추후 발생할 수 있는 TCP/IP의 변경사항들 및 iWARP와 같은 TCP/IP를 기반으로 하는 새로운 상위수준 프로토콜을 추가하기 어렵다. 즉, 구조의 유연성 측면에서 단점을 보인다. 두 번째 방법은 임베디드 시스템에 탑재된 범용 프로세서를 사용하여 소프트웨어적으로 TCP/IP를 처리하는 것이다. 이 방법은 하드웨어 방식에 비해 성능이 떨어지는 단점이 있지만, 상대적으로 구현이 쉽고 유연성이 뛰어난 장점을 가진다.

소프트웨어 기반 TOE는 임베디드 운영체제를 이용하여 구현하는 방법과 운영체제 없이 TCP/IP를 처리하는 전용 소프트웨어를 이용하여 구현하는 방법이 있다. 먼저 임베디드 운영체제를 이용하는 방법은 운영체제내의 TCP/IP 스택과 기존의 소켓 라이브러리를 사용하여 TOE를 구현하는 것이다. 그러나 운영체제를 사용함으로써 문맥 전환, 프로세스 대기 및 활성화 그리고 운영체제 자체의 부하로 인해 성능이 높지 못하다. 그래서 운영체제를 사용한 TOE의 단점을 극복하고, 소프트웨어 기반 TOE의 장점인 유연성을 가질 수 있는 방법으로써 TCP/IP를 처리하는 전용 소프트웨어를 이용한 TOE가 연구되고 있다. 본 논문에서는 운영체제 없이 TCP/IP를 처리할 수 있는 전용 소프트웨어인 HL-TCP (High Performance Lightweight TCP/IP)를 개발하여 소프트웨어 기반 TOE를 구현하였다. 그리고 실험을 통해 호스트 CPU사용률과 지연시간 및 대역폭 성능을 비교, 분석하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를 소개하고, 3장에서는 HL-TCP를 이용한 TOE의 구조 및 성능 향상을 위한 개선사항들을 설명한다. 그리고 4장에서는 HL-TCP를 이용한 TOE의 실험 결과를 분석한다. 마지막으로 5장에서는 결론과 향후 연구를 제시한다.

2. 관련연구

호스트 CPU의 부하를 줄이기 위한 TOE 관련 연구들이 지금까지 널리 행해져 왔다[1-4]. TOE 성능을 향상시키기 위한 여러 가지 메커니즘들을 소개한 연구[4], 에뮬레이션 기법을 통해 프로토콜 오프로딩을 할 때의 성능을 예측한 연구[1], 기존에 호스트 CPU에서 관리되던 연결 정보들을 프로그램 가능한 이더넷에 핸드 오프시켜 성능향상을 도모한 연구[5], 그리고 TOE를 위해 필요한 리눅스 커널의 지원 방법에 대해 다룬 연구들도 있다[6,7].

기존의 TOE 구현연구들은 일반적으로 ASIC[8] 또는 FPGA등과 같은 전용 하드웨어를 통해 TCP/IP를 처리하였다. 인텔은 수신되는 TCP/IP 패킷들을 10 기가비트급의 고속 이더넷 링크에서 처리할 수 있는 프로토타입[8]을 발표하였다. 그러나 이 프로토타입은 단지 패킷의 헤더만을 특별한 명령어 셋을 이용하여 처리하는 기능만 있고, 패킷의 데이타에 대한 전송 및 호스트와 TOE 간의 DMA 지원에 대한 메커니즘들은 제공하지 못하고 있다. 인텔 외의 몇몇 기업들도 이미 10 기가비트 이더넷 제품들을 발표하고 있으나, 이들 중 소수만이 현재 시장에 공급되고 있으며, TOE의 구현 방법에 대한 자세한 정보들은 공개적으로 제공되지 않고 있는 실정이다. 비록 이러한 하드웨어 기반의 방법들은 고성능이지만, 새로운 기능을 가지며 진화하고 있는 TCP/IP의 변화를 수용하기에는 유연성 면에서 단점이 있고, 또한 개선된 점을 반영하는 새로운 하드웨어를 디자인하는데 많은 시간이 소요된다.

이러한 단점을 극복하고자 TOE를 하이브리드 방법으로 구현한 연구들[9-11]이 있다. 이 방법들은 시간이 많이 걸리는 부분은 FPGA와 같은 하드웨어 로직이 담당하며, 하드웨어로 처리하기에 부적합한 부분들은 ARM과 같은 임베디드 프로세서를 통해 소프트웨어적으로 처리한다. 관련 연구들을 살펴보면 다음과 같다. IP, ARP, ICMP와 같은 프로토콜들은 하드웨어가 처리하고, TCP는 임베디드 프로세서가 처리하도록 하는 연구[11], 그리고 FPGA 기반 개발보드에 TOE를 구현하고, 이 TOE와 호스트 운영체제 간의 인터페이스에 관한 연구[10]가 있다. 그러나 성능과 유연성을 모두 추구하고 있는 하이브리드 방법은 여전히 하드웨어 부분에 대한 구현이 복잡하며, 하드웨어 부분과 소프트웨어 부분간의 연동에 많은 오버헤드를 가진다는 문제점이 있다.

유연성을 최대한 유지하면서도 고성능을 가지는 TOE를 구현하기 위해 우리는 선행 연구로서 임베디드 리눅스를 사용하여 TOE를 구현[12]하였다. 그러나 TCP/IP, 디바이스 계층간을 이동하면서 발생하는 데이타 복사 문제와 임베디드 커널과 응용 프로그램 사이의 문맥 전환 오버헤드로 인해 높은 성능을 내지 못 하였다. 이러한 운영체제를 사용함으로써 발생하는 부하를 없애기

위해 운영체제 없이 TCP/IP를 처리하는 전용 소프트웨어를 사용하여 TOE를 구현하는 방법이 있다. 이러한 전용 소프트웨어들 중에서 소스코드가 공개되어 있는 대표적인 것으로 lwIP (Lightweight TCP/IP)[13]를 들 수 있다. 원래 lwIP는 자원이 제한된 소형 임베디드 시스템을 위하여 개발되었으며[14], 이를 본 논문의 선행 연구로서 소프트웨어 기반 TOE구현에 적합하도록 개선한 관련연구[15]가 있다. 앞으로 본 논문에서는 이러한 lwIP를 이용한 TOE를 lwIP-TOE라고 명한다.

이러한 연구들을 기반으로 본 논문에서는 소프트웨어 기반 TOE가 가지는 유연성이라는 장점을 유지하면서 보다 더 고성능을 낼 수 있는 HL-TCP를 구현하였다. 그리고 HL-TCP를 이용한 TOE는 선행 연구인 lwIP를 이용한 TOE와 동일한 임베디드 시스템에서 구현되었으며, HL-TCP 성능의 우수성을 본 논문에서 실험을 통해 밝힌다.

3. HL-TCP의 구현 및 고성능화

본 장에서는 HL-TCP의 구현과 고성능화에 대해 크게 세 부분으로 나누어 설명한다. 먼저 HL-TCP가 지원하는 TCP/IP 기능들과 호스트 측의 구현사항 및 개발환경에 대해 설명한다. 그 다음으로 HL-TCP를 이용한 TOE에서의 데이타 전송함수 처리과정을 설명하고, 마지막으로 HL-TCP의 고성능화를 위한 개선사항들을 설명한다.

3.1 HL-TCP의 구현 및 개발환경

HL-TCP는 다음과 같은 TCP/IP의 기본적인 기능들을 지원할 수 있도록 구현하였다. HL-TCP는 흐름제어, 혼잡제어, 재전송, 지연 ACK, Out-of-Order 패킷처리 등을 지원하며 혼잡제어는 AIMD(Additive Increase, Multiplicative Decrease) 알고리즘을 채택하여 구현하였다. 그리고 매 수신 패킷마다 ACK패킷을 보내는 것은 네트워크의 효율성이나 성능측면에서 좋지 않기 때문에 본 논문에서는 지연 ACK를 구현하였다.

TOE를 구현하기 위해서는 소켓 응용 프로그램들이 호스트 운영체제내의 TCP/IP를 거치지 않고, 직접 TOE에 구현된 TCP/IP기능들을 호출할 수 있는 메커니즘이 필요하다. 이를 위해서는 호스트 운영체제의 수정이 필요하며, 이러한 호스트 운영체제의 수정 사항은 본 연구의 선행 연구로서 수행한 임베디드 리눅스를 이용한 TOE[12]와 lwIP를 이용한 TOE[15]에 공통적으로 사용되며, 그 메커니즘이 설명되어 있다.

HL-TCP를 구현하기 위한 개발환경으로 Cyclone Microsystems사의 PCI-730 카드 [16]를 사용하였다. 그림 1은 PCI-730 카드의 블록 다이어그램을 나타낸다. PCI-730 카드는 Intel사의 600MHz XScale CPU와 Intel사의 82544 기가비트 이더넷 MAC을 탑재하고 있다. 이 기가비트 이더넷 MAC은 TCP/IP기능 중 세그멘테이션과 체크섬 계산을 하드웨어적으로 처리하는 TSO, CSO 기능을 가진다. 그리고 이 MAC은 고속 네트워크 환경에서 매 송수신 패킷마다 인터럽트를 처리함으로써 발생하는 부하를 줄이기 위해 인터럽트 coalescing기능을 지원한다. 그리고 PCI-730카드는 기가비트 이더넷 MAC과 연결된 Secondary PCI버스와 호스트 메모리에 접근할 수 있는 Primary PCI버스를 연결해주는 PCI-to-PCI 브릿지가 있다. 이 브릿지를 이용하여 기가비트 이더넷이 직접 호스트 메모리를 DMA할 수 있다.

3.2 HL-TCP에서의 데이타 전송함수 처리

본 절에서는 HL-TCP를 이용한 TOE에서 데이타 전송 함수의 처리과정을 설명한다. 먼저 호스트의 사용자 응용프로그램이 send()소켓함수를 호출하면 호스트 운영체제의 TCP/IP 스택을 거치치 않고 디바이스 드라이버에 구현된 toe_send() 함수가 호출된다. toe_send()는 전송할 데이타가 저장된 호스트 메모리내의 각 페이지들의 물리주소와 길이를 구한다. 그리고 toe_send()는

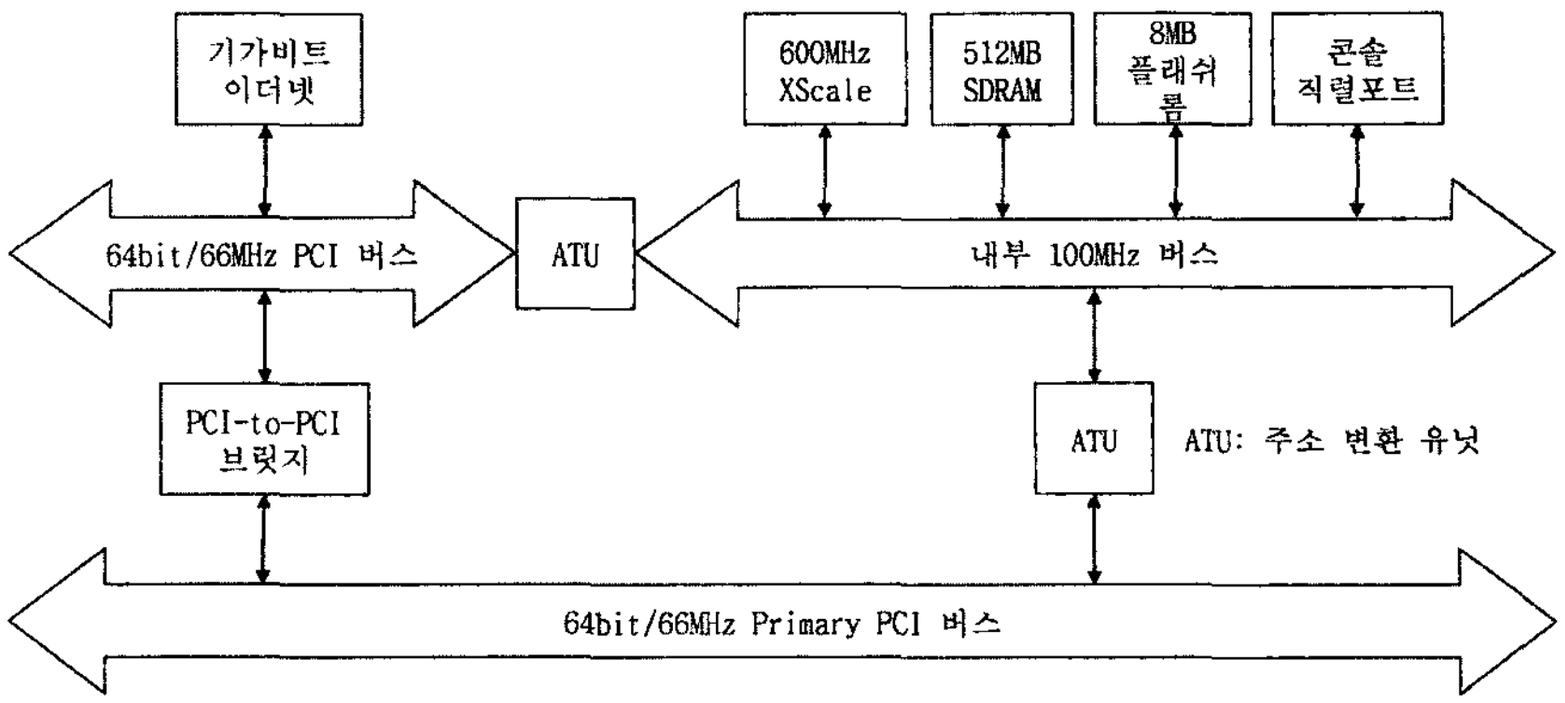


그림 1 PCI-730의 블록 다이어그램

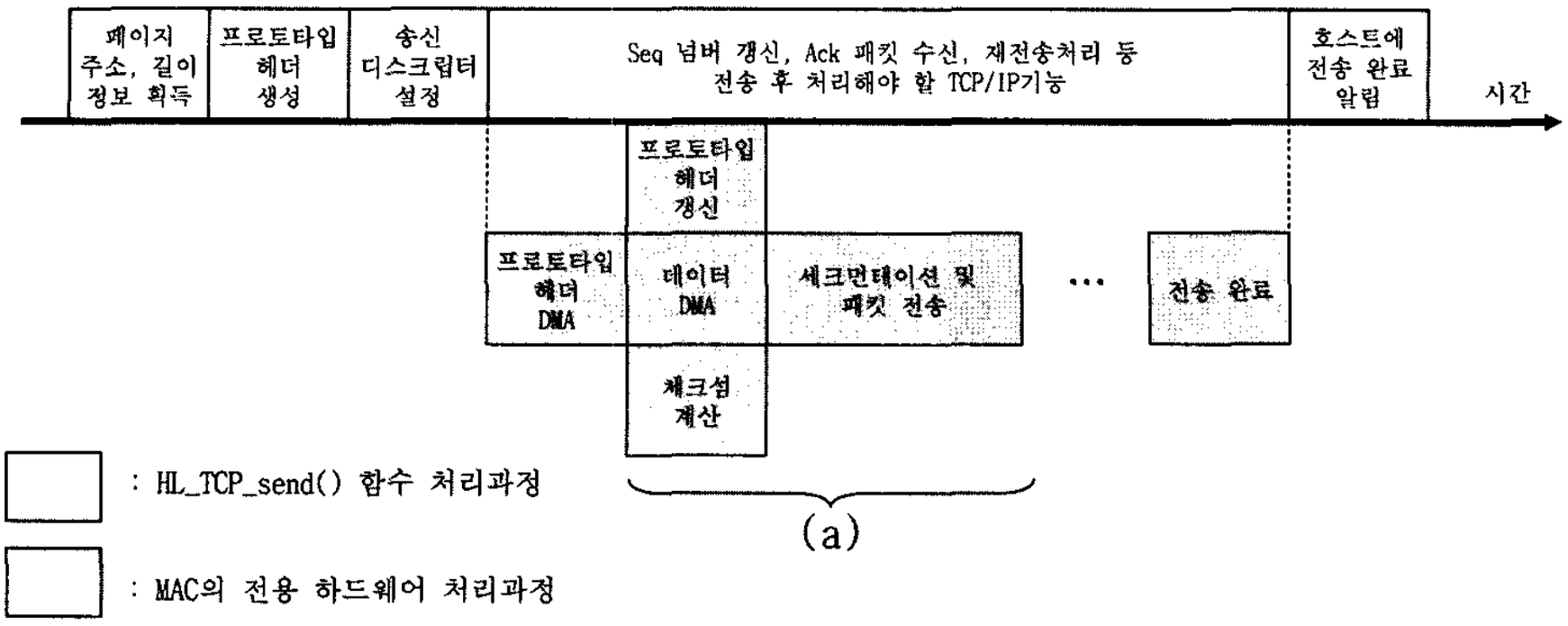


그림 2 HL_TCP_send함수 처리과정

HL-TCP가 관리하는 네트워크 어댑터의 메모리에 send() 함수임을 구분할 수 있는 구분자와 위에서 구해진 각 페이지의 물리주소와 길이를 기록한다. 그리고 HL-TCP에 인터럽트를 보내어 기록된 정보를 처리할 것을 알려준다. HL-TCP의 인터럽트 핸들러는 toe_send()로부터 받은 구분자를 확인하고 HL_TCP_send()함수를 호출한다.

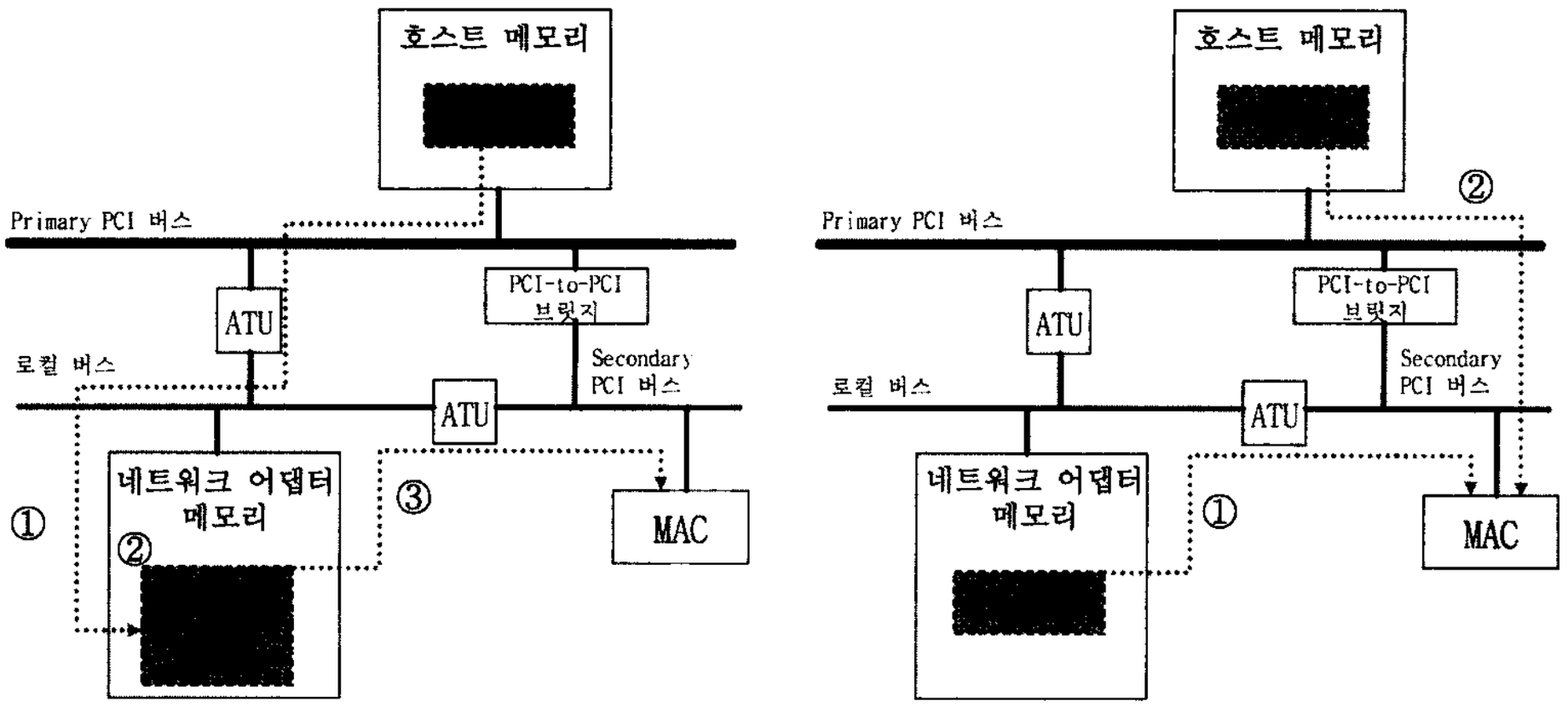
HL_TCP_send()함수의 처리과정은 그림 2를 통해 설명한다. 먼저 HL_TCP_send()함수는 전송할 데이타가 있는 호스트 메모리의 페이지정보를 얻는다. 그리고 MAC이 TSO기능을 처리하기 위해 필요한 TCP/IP 프로토타입 헤더를 만들고 TSO, CSO기능을 사용할 수 있도록 MAC을 설정한다. 그 다음으로 MAC이 데이타와 TCP/IP 프로토타입 헤더를 DMA할 수 있도록 송신 디스크립터를 설정한다. 그 후 MAC은 TCP/IP 프로토타입 헤더를 DMA한다. 이것을 기반으로 MAC은 각 패킷의 헤더를 만들고, 그와 동시에 데이타를 DMA하면서 체크섬 계산을 한다. 그리고 읽어온 데이타를 세그먼테이션 및 패킷화하여 전송하는 과정을 반복한다. 즉, 그림 2의 (a)과정을 데이타의 전송을 완료할 때까지 반복한다. 이렇게 데이타 전송이 끝나면 호스트에게 send()함수의 결과 값을 인터럽트를 통해 알려준다.

3.3 HL-TCP의 고성능화

본 절에서는 HL-TCP의 성능향상을 위해 적용한 세 가지 개선사항을 설명한다. 첫 번째로는 MAC자체에서 하드웨어적으로 지원하는 오프로딩 기능들을 HL-TCP가 사용할 수 있도록 하였다. 최근 기가비트 이더넷 MAC은 TCP segmentation offload(TSO), Checksum offload(CSO), 인터럽트 coalescing등을 지원한다. TSO와 CSO는 TCP/IP의 기능인 세그멘테이션과 체크섬 계산을 MAC내의 전용 하드웨어에서 처리해 주는 기능이다. 그리고 인터럽트 coalescing은 고속 네트워크 환경에서 매 송수신 패킷마다 인터럽트를 처리할 때 발생하는 부하를 줄이기 위해, 일정 시간 동안 발생한 인터럽트를 모아 한 번에 처리하도록 하는 것이다. 하위 이더넷 MAC이 이러한 기능들을 지원하면 HL-TCP 내의 TCP 세그멘테이션 및 체크섬 계산 루틴은 동작하지 않고 이더넷 하드웨어가 처리하도록 HL-TCP를 구현하였다. 그 결과 하위 하드웨어의 오프로딩 기능의 지원 유무에 상관없이 항상 소프트웨어적으로 세그멘테이션 및 체크섬을 수행하던 기존의 방법보다 높은 성능을 얻을 수 있었고, 이를 4.2절의 실험 결과에서 보인다.

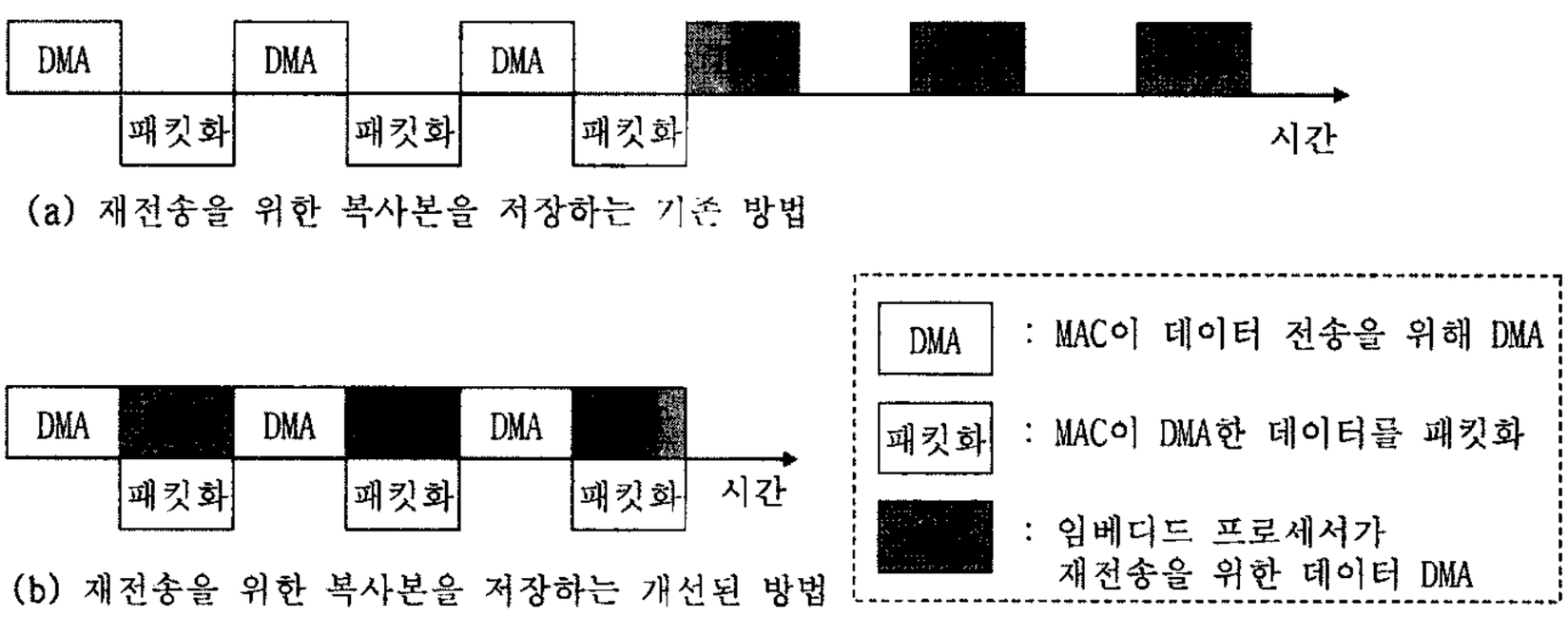
두 번째로는 데이타를 전송할 때, 호스트 사용자 메모리에서 네트워크 어댑터의 메모리로 복사하는 부하를 제거하였다. 그림 3에 표시된 원 문자를 기준으로 이러한 메커니즘을 설명한다. 그림 3의 (a)와 같이 일반적인 TOE에서는 호스트 메모리에 있는 데이타를 네트워크 어댑터의 메모리에 복사한 후(①), 데이타의 앞에 TCP/IP 헤더를 만들고(②), MAC이 이를 DMA하여 전송한다(③). 이러한 방식은 복사 문제로 인해 시스템의 성능을 저하시킨다. 이를 해결하기 위해, 본 논문에서는 그림 3의 (b)와 같이 TCP/IP 헤더와 데이타가 각각 분리되어 있어도 전송할 수 있도록 HL-TCP를 구현하였다. 즉, TCP/IP 헤더는 네트워크 어댑터의 메모리로부터 DMA하고(①), 데이타는 호스트 메모리로부터 DMA하도록 구현하였다(②). 그 결과, 호스트 사용자 메모리에 있는 데이타가 네트워크 어댑터로 복사될 필요가 없어 통신 성능을 향상시킬 수 있었다.

세 번째로는 재전송해야 할 경우를 대비해 전송한 데이타에 대한 복사본을 네트워크 어댑터의 메모리에 저장하는 방법을 개선하였다. 기존의 방식은 앞 단락에서 설명하였듯이, MAC이 복사 없이 데이타를 전송한 후에, 재전송을 위한 데이타를 다시 DMA를 통해 네트워크 어댑터 메모리로 복사한다. 그림 4의 (a)와 같이 순차적으로 이루어지는 두 번의 DMA 전송방식은 MAC이 DMA로 가져온 데이타를 실제로 패킷화하여 네트워



(a) 데이타와 TCP/IP 헤더를 DMA하는 기존방법 (b) 데이타와 TCP/IP 헤더를 DMA하는 개선된 방법

그림 3 데이타와 TCP/IP헤더 DMA하는 방법



(a) 재전송을 위한 복사본을 저장하는 기존 방법

(b) 재전송을 위한 복사본을 저장하는 개선된 방법

그림 4 재전송을 위한 복사본 저장 방법

크로 내보는 시간 동안 PCI 버스가 유휴 상태에 있도록 하며, 이는 PCI 버스의 사용률을 낮추어 성능상의 단점으로 작용한다. 따라서 본 논문에서는 이를 개선하기 위해, 그림 4의 (b)와 같이 MAC 내의 전용 하드웨어가 DMA한 데이타를 패킷화하는 동안, 재전송을 위한 DMA 전송작업들이 병렬화되도록 구현하여 PCI버스의 사용률을 높였다. 그 결과, 재전송을 위해 네트워크 어댑터의 메모리로 데이타를 복사함으로써 통신성능이 저하되는 문제를 해결할 수 있었다.

4. 실 험

본 장에서는 HL-TCP를 이용한 소프트웨어 TOE의 성능을 측정 하였다. 먼저 일반 기가비트 어댑터를 사용하였을 때와 본 논문에서 구현한 HL-TCP를 이용한 TOE를 사용하였을 때의 CPU사용률을 측정하였다. 다음으로 HL-TCP를 이용한 TOE와 2장에서 설명한 lwIP를 이용한 TOE[15]인 lwIP-TOE의 대역폭을 측정하여 성능을 비교, 분석하였다. 그리고 3.3절에서 설명한 HL-TCP의 개선사항들을 차례로 적용하였을 때 대역폭과 지연시간을 실험하고 분석하였다.

실험 환경으로 1.8GHz Intel Xeon 프로세서, 512MB 메인 메모리 및 64bit/66MHz PCI 슬롯을 가진 두 대의 컴퓨터를 사용하였다. 그리고 3COM사의 SuperStack3 스위치를 사용하여 노드들을 연결하였다. 호스트 컴퓨터의 운영체제로는 리눅스 커널 2.4.27을 사용하였다. 성능 비교를 위한 일반 기가비트 이더넷 어댑터로는 Intel사의 PRO/1000MT Server Adapter를 사용하였고, TOE 네트워크 어댑터로는 HL-TCP가 탑재된 Cyclone Microsystems사의 PCI-730카드를 사용하였다. 실험 방법은 호스트의 사용자 응용 프로그램에서 send() 함수의 호출로부터 데이타가 원격 노드의 수신버퍼에 들어갈 때까지의 지연시간과 대역폭을 측정하였다. 측정치는 4바이트에서 256K 바이트까지의 데이타 전송을 100회 반복한 평균치이다.

4.1 호스트 CPU의 사용률

그림 5는 일반 기가비트 이더넷 어댑터를 장착한 시스템과 HL-TCP를 이용한 TOE를 장착한 시스템의 호스트 CPU 사용률을 비교하고 있다. 그림 5의 x축에 해당하는 크기의 데이타를 전송할 때, 호스트 CPU 사용률은 아래 식을 통해 구해진다.

$$\text{CPU 사용률} = \frac{\text{호스트 CPU의 사용시간}}{\text{지연시간}} \times 100$$

즉, 4 바이트의 데이타를 보낼 때, 지연시간이 100μs였고, 그 때 호스트 CPU가 전송을 위해 50μs의 시간을 사용하였다면 CPU 사용률은 50%가 되는 것이다. 그림 5에서 보는 바와 같이 일반 기가비트 이더넷 어댑터를 장착한 시스템의 경우 데이타 크기에 따라 호스트 CPU 사용률은 최소 32%에서 최대 69%까지 측정되었다. 이것은 일반 기가비트 이더넷 어댑터를 사용하면 호스트 CPU가 운영체제내의 TCP/IP를 처리하기 때문이다. 반면에, HL-TCP를 이용한 TOE를 장착한 시스템의 경우 CPU 사용률은 약 6% 이하로 측정되었다. 또한 데이타 크기가 커질수록 TOE의 CPU 사용률은 거의 0%에 가까워지는 것을 볼 수 있다. 이는 데이타 크기가 커질수록 지연시간은 길어지는데 반해, 호스트 컴퓨터가 호스트 CPU를 사용하는 시간은 거의 변화가 없기 때문이다. 왜냐하면 TOE를 장착한 시스템에서 호스트 CPU가 하는 일은 사용자 영역 데이타에 대한 가상 주소를 물리 주소로 변환하고 이를 TOE 알려주는 작업인데, 이러한 작업은 데이타의 크기 변화에 따라 그 소요되는 시간의 증가분이 크지 않기 때문이다.

4.2 HL-TCP를 이용한 TOE 대역폭

본 절에서는 3.3절에 설명한 개선사항들을 모두 적용한 HL-TCP를 이용한 TOE와 lwIP-TOE의 대역폭을 비교하였다. 그림 6에서 보는 바와 같이 lwIP-TOE의 성능은 194Mbps로 낮은 성능을 보였다. 이러한 낮은 성능의 원인 크게 두 가지이다. 첫 번째 이유는 MAC자체에서 하드웨어적으로 지원하는 오프로딩 기능들을 사용하지 못했기 때문이고, 두 번째 이유는 데이타를 전송할 때, 네트워크 어댑터의 메모리로 복사하는 부하가 발생하기 때문이다. 이에 반해 HL-TCP는 구현 시에 3.3절에 설명한 고성능을 위한 개선 사항들을 모두 적용하여 구현하였기 때문에 453Mbps의 최대 대역폭을 보이고 있다.

4.3 HL-TCP의 개선을 통한 TOE의 성능

본 절에서는 3.3절에서 제시한 각 개선사항들을 HL-TCP를 이용한 TOE에 적용하였을 때의 지연시간과 대역폭을 측정하고 분석하였다. 먼저 HL-TCP에 TCP segmentation offload(TSO), Checksum offload(CSO)를 적용하였을 때의 대역폭과 지연시간을 측정하였다. 다음으로 데이타를 전송할 때 호스트 사용자 메모리에서 네트워크 어댑터의 메모리로 복사하는 부하를 제거였을 때의 대역폭과 지연시간을 측정하였다. 또한 MAC과 재전송을 위한 DMA 전송작업들을 병렬화한 개선사항과 인터럽트 coalescing 사용하는 개선사항 등을 차례로 적용하여 각각의 대역폭과 지연시간을 측정하였다.

표 1은 각각의 개선 방법을 HL-TCP를 이용한 TOE에 적용하였을 때의 대역폭을 보여준다. 이 실험에서는 대역폭 측정을 위해 256KB 데이타를 사용했다. HL-TCP에 MAC의 TSO와 CSO기능을 적용하였을 경우 최대 대역폭은 306Mbps를 보였다. 그리고 전송할 때 네트워크 어댑터로 데이타를 복사하지 않도록 개선하였을 경우 최대 대역폭이 350Mbps로 약 45Mbps 정도의 향상되었다. 그리고 MAC과 재전송을 위한 DMA 전송작업들이 병렬화되도록 구현하여 PCI버스의 사용률을 높임으로써 재전송을 위해 네트워크 어댑터로의 데이타

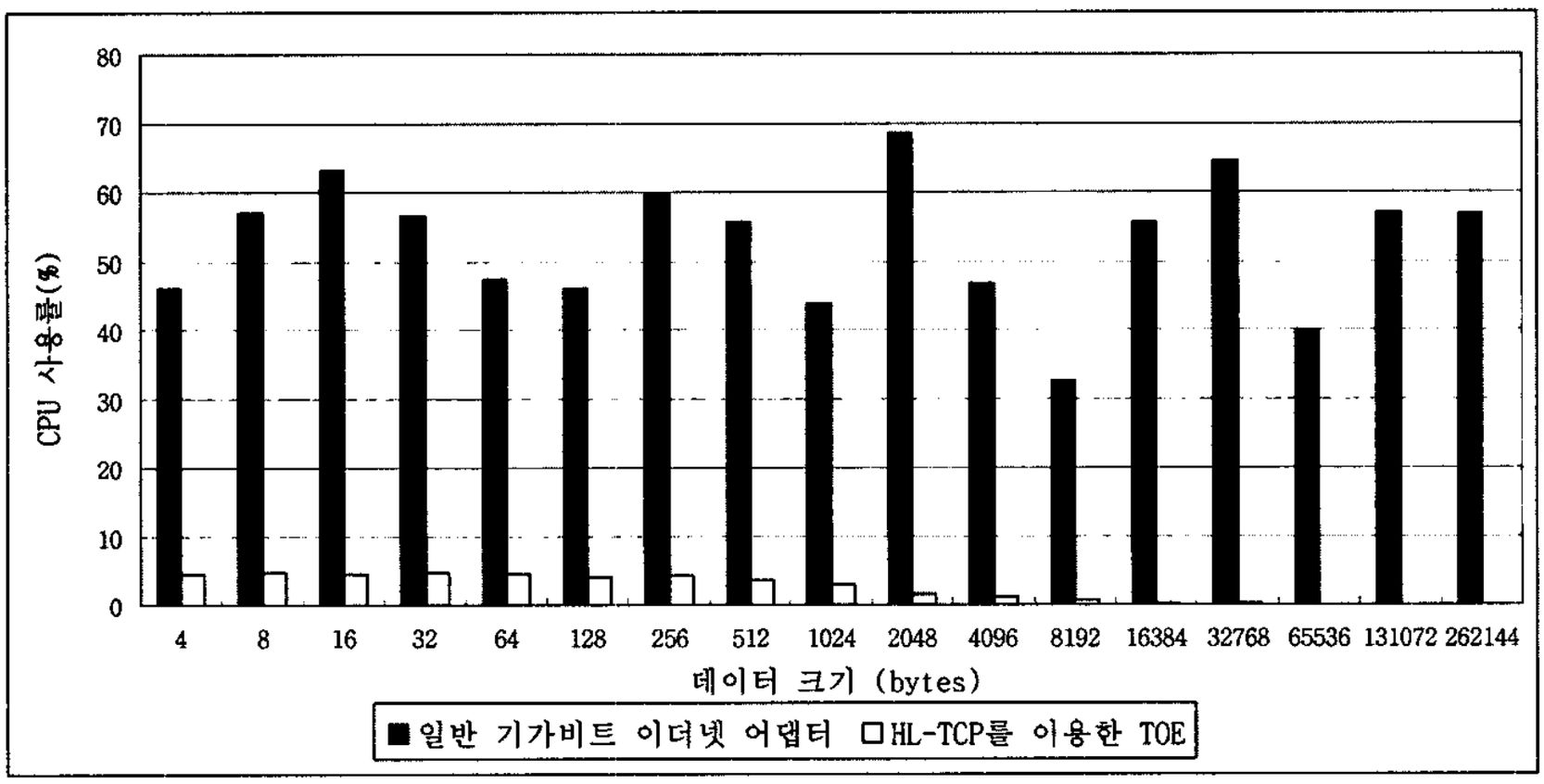


그림 5 일반 기가비트 어댑터와 HL-TCP를 이용한 TOE의 호스트 CPU 사용률

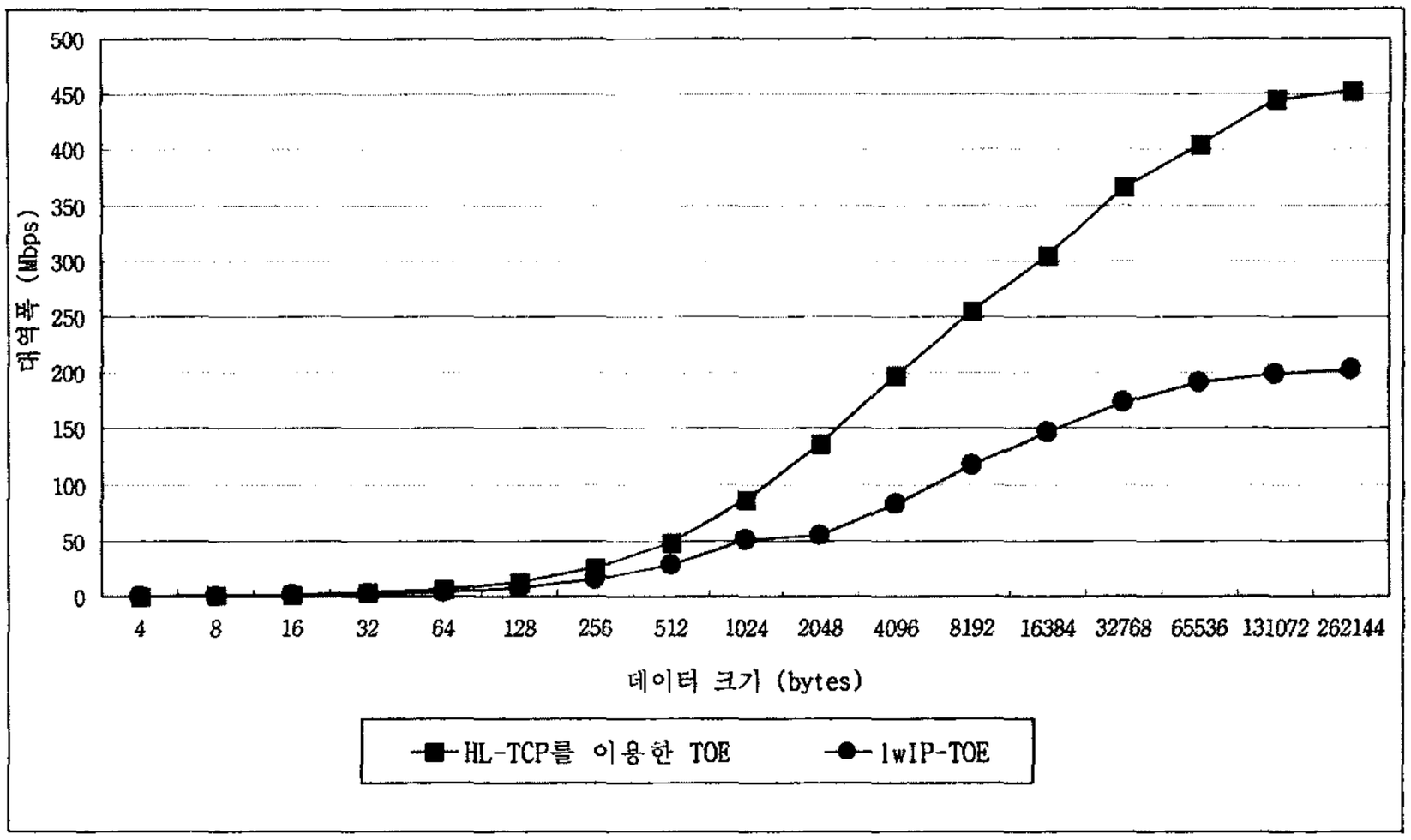


그림 6 lwIP-TOE와 HL-TCP를 이용한 TOE의 대역폭 비교

표 1 각 개선사항들을 HL-TCP를 이용한 TOE에 적용하였을 때의 대역폭 (256KB)

HL-TCP의 개선 방법	대역폭 (Mbps)
HL-TCP+TSO+CSO	306
HL-TCP+TSO+CSO+메모리 복사 제거	350
HL-TCP+TSO+CSO+메모리 복사 제거+재전송을 위한 DMA병렬화	345
HL-TCP+TSO+CSO+메모리 복사 제거+재전송을 위한 DMA병렬화+Interrupt Coalescing	453

복사가 발생하여도 대역폭은 거의 줄어들지 않았다. 마지막 개선사항으로써 인터럽트 coalescing기능을 적용하였을 때 최대 대역폭은 453Mbps를 보였다. 이와 같이 약 100Mbps정도 최대 대역폭 향상을 볼 수 있었던 이유는 매 송수신 패킷마다 인터럽트를 처리하지 않고, 일정 시간 동안 발생한 인터럽트를 모아 한 번에 처리하기 때문이다.

표 2는 각각의 개선사항들을 적용하였을 때 HL-TCP를 이용한 TOE의 최소 지연시간을 보여준다. 이 실험에서 지연시간 측정을 위해 4byte 데이타를 사용했다. HL-TCP에 TSO와 CSO를 적용하였을 때 지연시간은 59.45μs를 보였다. 추가 개선사항으로써 메모리 복사를 제거하였을 때는 54.21μs로 5.24μs의 지연시간이 감소되었다. 그리고 재전송을 위한 네트워크 어댑터로의 데이타 복사가 발생하여도 최소 지연시간은 0.22μs만 증가하였다. 이는 MAC과 재전송을 위한 DMA 전송작업들이 병렬화되도록 구현하여 PCI버스의 사용률을 높였기 때문이다. 이에 추가 개선사항으로써 인터럽트 coalescing 기능을 적용하였을 때는 최소 지연시간이 54.43μs에서 67.23μs으로 12.80μs증가하는 결과를 보였다. 지연시간이 증가한 이유는 매 송수신 인터럽트를 일정시간 모아서 한번에 처리하기 때문이다. 인터럽트 coalescing기능을 적용하였을 때 12.80μs의 지연시간 길어지지만, 이 기능을 적용하면 표 1에서 설명하였듯이 대역폭 성능은 약 100Mbps정도 증가하는 결과를 보였다.

그림 7은 인터럽트 coalescing 기능 사용시 수신 인터럽트 지연시간에 따른 최대 대역폭을 비교하고 있다. 수신 인터럽트 지연시간이 13.31μs일 때 최대 대역폭 성능이 가장 높았다. 수신 인터럽트 지연시간이 13.31μs보다 작을 때는 빈번한 인터럽트 발생에 의한 부하가 발생하여 대역폭이 줄어드는 것을 확인할 수 있었다.

표 2 각 개선사항들을 HL-TCP를 이용한 TOE에 적용하였을 때의 지연시간 (4byte)

HL-TCP의 개선 방법	지연시간 (μs)
HL-TCP+TSO+CSO	59.45
HL-TCP+TSO+CSO+메모리 복사 제거	54.21
HL-TCP+TSO+CSO+메모리 복사 제거+재전송을 위한 DMA병렬화	54.43
HL-TCP+TSO+CSO+메모리 복사 제거+재전송을 위한 DMA병렬화+Interrupt Coalescing	67.23

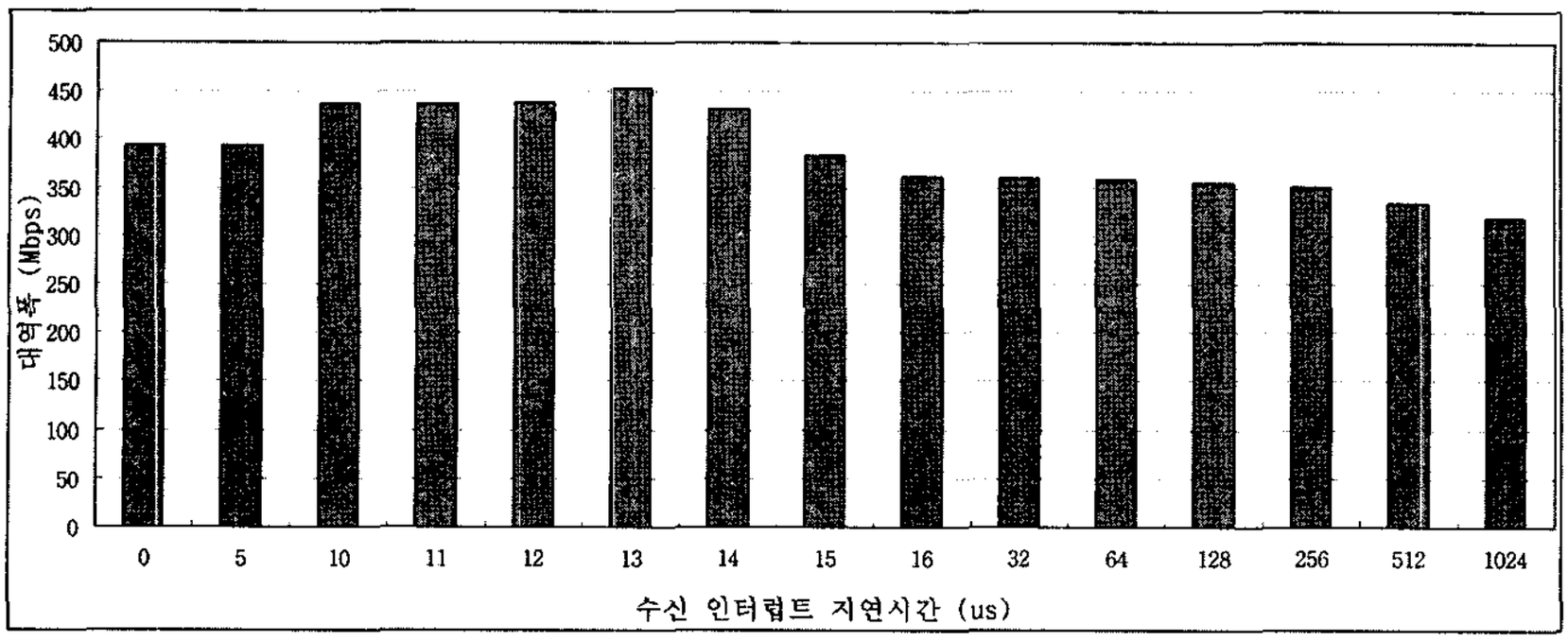


그림 7 수신 인터럽트 지연시간에 따른 최대 대역폭 (256KB)

5. 결론 및 향후 과제

본 논문에서 구현한 HL-TCP를 이용한 소프트웨어 기반 TOE는 MAC에서 하드웨어적으로 제공하는 TSO, CSO, 인터럽트 coalescing기능을 사용하도록 구현하였다. 그리고 데이타를 전송할 때, 호스트 사용자 메모리에서 네트워크 어댑터의 메모리로 복사하는 부하를 제거하였다. 또한 MAC과 재전송을 위한 DMA 전송작업들이 병렬화되도록 구현하여 PCI버스의 사용률을 높였다. 그 결과, 통신 대역폭은 최대 453Mbps를 보였고, CPU 사용률은 6%를 넘지 않았다. 향후 연구로서 HL-TCP와 실제 인터넷 어플리케이션을 결합하여 실험할 것이며, 이를 통해 HL-TCP의 안정성 및 다중연결 상황에서의 성능을 측정할 것이다.

참 고 문 헌

[1] R. Westrelin, N. Fugier, E. Nordmark, K. Kunze, E. Lemoine, "Studying network protocol offload with emulation: approach and preliminary results," In Proceedings of the 12th Annual IEEE Symposium on High Performance Interconnects, pp. 84-90, Aug. 2004.

[2] Soo-Cheol Oh, Hankook Jang and Sang-Hwa Chung, "Analysis of TCP/IP Protocol Stack for a Hybrid TCP/IP Offload Engine," Lecture Notes in Computer Science, Vol.3320, pp. 406-409, Dec 2004.

[3] Paul Willmann, Hyong-youb Kim, Scott Rixner and Vijay S. Pai, "An Efficient Programmable 10 Gigabit Ethernet Network Interface Card," In Proceedings of the International Symposium on High-Performance Computer Architecture, pp. 96-107, Feb 2005.

[4] Wen-Fong Wang, Jun-Yau Wang, Jin-Jie Li, "Study on Enhanced Strategies for TCP/IP Offload Engines," In Proceedings of the 11th International Conference on Parallel and Distributed Systems, Vol.1, pp. 398-404, July 2005.

[5] Hyong-youb Kim, Scott Rixner, "TCP Offload through Connection Handoff," In Proceedings of EuroSys 2006, pp. 279-290, Apr. 2006.

[6] Dong-Jae Kang, Chei-Yol Kim, Kang-Ho Kim, Sung-In Jung, "Design and implementation of kernel S/W for TCP/IP offload engine(TOE)," In Proceedings of the 7th International Conference on Advanced Communication Technology, Vol.1, pp. 706-709, Feb. 2005.

[7] Soo-Cheol Oh, Seong-Woon Kim, "An Efficient Linux Kernel Module supporting TCP/IP Offload Engine on Grid," In Proceedings of Fifth International Conference on Grid and Cooperative Computing, pp. 228-235, Oct. 2006.

[8] Y. Hoskote, B. A. Bloechel, G. E. Dermer, V. Erraguntla, D. Finan, J. Howard, D. Klowden, S. G. Narendra, G. Ruhl, J. W. Tschanz, Sriram Vangal, V. Veeramachaneni, H. Wilson, Jianping Xu, N. Borkar, "A TCP offload accelerator for 10 Gb/s Ethernet in 90-nm CMOS" IEEE Journal of Solid-State Circuits, Vol.38, Issue 11, pp. 1866-1875, Nov. 2003.

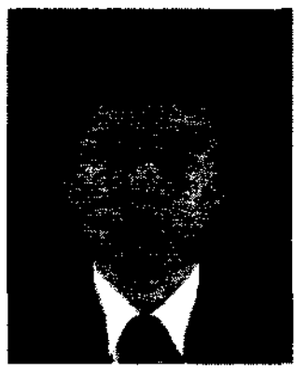
[9] Hankook Jang, Sang-Hwa Chung and Soo-Cheol Oh, "Implementation of a Hybrid TCP/IP Offload Engine Prototype," Lecture Notes in Computer Science, Vol.3740, pp. 464-477, October 2005.

[10] Liu Tian-Hua, Zhu Hong-Feng, Zhou Chuan-Sheng, Chang Gui-Ran, "Research and Prototype Implementation of a TCP/IP Offload Engine Based on the ML403 Xilinx Development Board," In Proceedings of the 2nd International Conference on Information and Communication Technologies, Vol. 2, pp. 3163-3168, April 2006.

[11] Zhong-Zhen Wu, Han-Chiang Chen, "Design and Implementation of TCP/IP Offload Engine System over Gigabit Ethernet," In Proceedings of the 15th International Conference on Computer Communications and Networks, pp. 245-250, Oct. 2006.

[12] In-Su Yoon and Sang-Hwa Chung, "Implementation and Analysis of TCP/IP Offload Engine and

RDMA Transfer Mechanisms on an Embedded System," Lectures Note in Computer Science, Vol. 3740, pp. 818-830, Oct 2005.
[13] A Lightweight TCP/IP stack [online]. Available: http://savannah.nongnu.org/projects/lwip
[14] A. Dunkels, "Full TCP/IP for 8-Bit Architectures," In Proceedings of the 1st International Conference on Mobile Applications, Systems and Services, pp. 85-98, May 2003.
[15] 윤인수, 정상화, 최봉식, 전용태, "임베디드 시스템상에서 Lightweight TCP/IP를 이용한 TCP/IP Offload Engine의 구현", 정보과학회논문지, 제33권, 제7호, pp. 413-420, 2006년.
[16] PCI-730: Intelligent Gigabit Ethernet Controller [online]. Available: http://www.cyclone.com/products/pci730.php



전 용 태

2005년 울산대학교 컴퓨터공학과 학사
2007년 부산대학교 컴퓨터공학과 석사
2007년~현재 대우조선해양주식회사 정보기술 R&D팀. 관심분야는 클러스터 시스템, 병렬처리, 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, TOE, RDMA, WMN, RFID

윤 인 수

2001년 부산대학교 컴퓨터공학과 학사
2001년~현재 부산대학교 컴퓨터공학과 석박사 통합과정. 관심분야는 클러스터 시스템, 병렬처리, TOE, RDMA, WMN