네트워크 프로세서를 활용한 리눅스용 고속 네트워크 프로토콜 설계 및 구현

김주홍[0] 조혜영 성종우 정성인* 김대영
한국정보통신대학교, 한국전자통신연구원*
(scarlet[0], hycho, email, kimd)@icu.ac.kr, sijung@etri.re.kr*

Design and implementation of high speed Linux network protocol using network processor

Ju-Hong Kim[0] Hye-Young Cho Jong-Woo Sung Sung-In Jung* Daeyoung Kim
Dept. of Computer Science & Engineering, Information and Communications University, ETRI*

요 약

기가급 이상의 고속 인터넷 접속이 필수적인 고성능 리눅스 서버 시스템에 사용되는 네트워크 프로토콜의 성능향상을 위하여, 기존의 소프트웨어 성능향상 기술과 고속 패킷 처리 전용의 네트워크 프로세서를 활용하여 고속 네트워크 프로토콜을 설계 및 구현하였다. 네트워크 정합 장치로는 인텔의 IXP1200 네트워크 프로세서가 탑재된 네트워크 카드를 사용하였고, 리눅스 커널의 TCP/UDP/IP 네트워크 프로토콜 스택 중 계층 3 IP 프로토콜 이하를 네트워크 프로세서가 처리하게 함으로써 커널의 오버헤드를 줄이고, 고속의 프로토콜 처리가 가능하게 되었다.

1. 서 론

초고속 인터넷 구축을 위한 네트워크 관련 기술들이 테라 급으로까지 급속히 발전한데 비하여, 상대적으로 컴퓨터의 네트워크 정합 기술은 네트워크의 발전 속도를 따라잡지 못하고 있다. 그 원인은 버스와 같은 공유하드웨어의 병목현상과 소프트웨어로 구현된 네트워크 프로토콜에 의한 커널에서의 병목현상 때문이다. 특히, 대용량의 고품질 멀티미디어 정보를 네트워크를 통해 다수의 클라이언트에게 공급해야 하는 고성능 미디어서버의 경우 이러한 병목현상을 줄이는 고속 네트워크 정합기술은 필수적이다. 현재 일반적으로 사용되고 있는 고속 이더넷 카드의 경우 CRC검사 같은 일부 기능만을 하드웨어로 처리하고, 디바이스 드라이버, IP/ICMP, TCP/UDP 등의 네트워크 프로토콜은 커널에서 소프트웨어로 처리됨으로 인해 병목현상이 발생하고 있다. 본 논문에서는 커널의 오버헤드를 줄이기 위한 개선방안으로 현재 고속 라우터나 이더넷 스위치 등에 많이 활용되고 있는 네트워크 프로세서를 이용하여 프로토콜 스택 중 일부를 네트워크 프로세서가 처리하게 하여 성능을 향상시키는 방안을 제안한다.[1]

2. 관련연구

기존의 순수 소프트웨어로만 구현된 네트워크 프로토콜 스택에 존재하는 병목현상의 원인들 중 가장 비중이 높은 것은 패킷의 메모리 대 메모리로의 복사와 체크 섬으로 인해 발생하는 오버헤드 등이다. 이를 해결하기 위한 여러 가지 방안 중 대표적인 예는 다음과 같다.

2.1 제로 복사 (Zero Copy)

제로 복사 이전의 네트워크 프로토콜 처리기술에서는, 수신된 데이터는 커널 영역의 메모리로 복사되고, 또 다시 프로토콜간 그리고 최종적으로 사용자 영역의 메모리로 다중 복사되어지는 구조를 가지고 있다. 이러한 구조는 커널에서의 데이터 처리 속도를 저하시키고, 메모리 사용량을 증대 시킨다. 제로 복사 기술은 네트워크 정합장치로부터 수신한 데이터를 프로토콜간 공유를 가능하게 함으로써, 커널의 오버헤드와 메모리 사용량을 줄인다. 이미 이 기술은 현재 상용 운영체제에 많이 적용되고 있으며, 주목할 만한 성능향상을 가져왔다. 하지만, 이러한 소프트웨어적인 해결 방법만으로는 고속 네트워크 트래픽을 처리하는 데 한계가 있다. [2][3][4]

2.2 하드웨어를 이용한 기법

하드웨어를 이용한 기법으로는 기존의 소프트웨어로 수행되던 프로토콜을 ASIC화 하여 처리 속도를 높이는 방식이 있으며, 이 예로는 TCP/IP 전용 칩이 있다. 또한 소프트웨어 프로토콜 처리에서 오버헤드의 큰 부분을 차지하는 체크섬 등을 하드웨어에서 수행시킴으로써 성능을 향상시키는 기법도 널리 사용되고 있다.

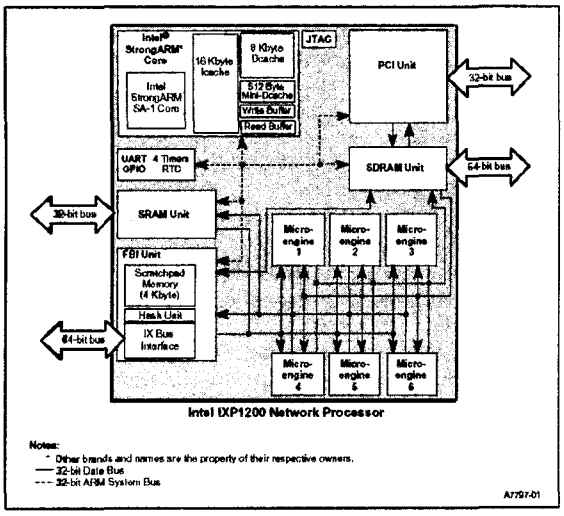
2.3 Infiniband를 이용한 기법

최근에는 Infiniband를 이용한 제로 복사 기술이 연구되고 있다. Infiniband Trade Association에서 표준화 작업이 진행중인 SDP(Socket Direct Protocol)프로토콜이 그것으로 이는 Infiniband라는 새로운 버스 기술에 근간을 둔 제로 복사 기술이다. 이 기술의 기본 개념은 커널을 거치지않고 데이터를 직접 사용자 버퍼에 저장하는 것이다. 주로 서버 팜 (Server Farm)을 고속의 Infiniband 버스로 연결한 환경에서 입출력 병목현상을 줄이기 위해 사용된다. [5]

본 연구에서는 기본적으로 제로 복사의 개념을 근간으로 하여 소프트웨어 구조를 개선하고, 네트워크 프로세서를 이용한 고속 패킷처리를 통하여 높은 성능을 가지는 네트워크 프로토콜를 설계하고 구현한다. 이를 위해 인텔의 IXP1200 네트워크 프로세서를 사용하며, 호스트 프로세서의 소프트웨어는 리눅스 운영체제를 기반으로 프로토콜 스택과 커널의 일부분을 수정한다. 즉, 운영체제(소프트웨어)와 하드웨어 구조의 변경을 결합하여

고성능의 네트워크 정합 기술을 구현한다.

3. 네트워크 프로세서 IXP1200의 구조



< 그림 1 > 인텔 IXP1200의 구조

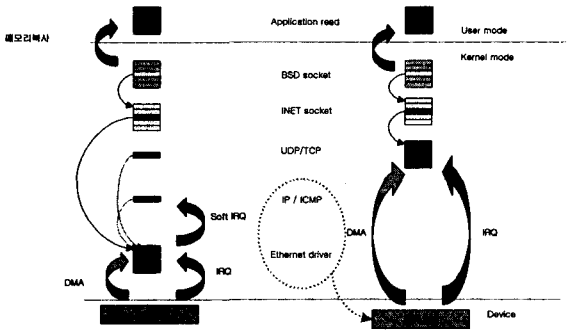
<그림1>에서 보는 바와 같이 IXP1200은 여섯 개의 독립적인 RISC 구조의 패킷처리 전용 프로세서로(Microengine) 구성되어 있고, 각각은 병렬적으로 그리고 파이프라인 방식으로 동작하여 고속의 패킷처리를 가능하게 하며, 각각의 마이크로 엔진에는 네 개의 하드웨어 쓰레드가 동시에 수행된다. 또한 각각의 쓰레드는 자신의 고유 레지스터를 사용함으로써 제로 문맥 교환 시간을 실현 할 수 있으며, 또한 쓰레드간 공유메모리를 사용해 제로 복사 기법을 구현 할 수 있다. 이와 같이 네트워크 프로세서는 네트워크 정합 장치의 성능향상을 위한 소프트웨어적 기법을 마이크로 엔진으로 구현할 수 있도록 하는 독특한 하드웨어 구조를 가지고 있다. [6]

본 논문에서는 IXP1200프로세서를 탑재한 Radisys ENP-2506p 보드를 이용하여 고성능의 네트워크 정합장치를 구현한다. Radisys ENP-2506p 보드는 두개의 기가비트 이더넷 포트를 통해 외부 네트워크와 연결되며, 호스트 프로세서와는 64bit/66Mhz PCI 버스를 통해 연결된다.

4. 네트워크 프로세서를 이용한 리눅스 네트워크 프로토콜(NP^2)

리눅스 NP^2(Network Processor based Network Protocol)프로토콜은 기존의 인터넷 종단 시스템의 네트워크 프로토콜과 호환되도록 설계했으며, IP계층 이하의 프로토콜 스택이 네트워크 프로세서에서 수행 된다. 호스트의 리눅스 커널에서는 네트워크 계층 4 이상의 프로토콜을 담당하며, 계층 3 이하의 프로토콜은 Radisys ENP-2506p 보드에서 처리하게 된다. 호스트와 ENP-2506p보드의 연결은 PCI 인터페이스를 사용한다. 이 구조는 기존에 네트워크 카드가 단순히 CRC체크만을 담당하고, 디바이스 드리이버 이상의 상위 계층 프로토콜 처리는 호스트의 커널이 담당했던 구조에 비교해 커널의 오버헤드를 상당히 줄일 수 있고, 그동안 문제가 되어 왔던 커널에서의 병목 현상을 해결할 수 있다.

4.1 리눅스 NP^2 프로토콜 스택의 설계 개념



< 그림 2 > 리눅스 NP^2 프로토콜 패킷 처리 비교

<그림 2>의 왼쪽 패킷 흐름도는 기존의 네트워크 프로토콜 스택을 사용했을 때의 패킷 처리 과정이고, 오른쪽은 IXP1200 네트워크 프로세서를 이용한 리눅스 NP^2 사용했을 때의 패킷 처리 과정이다. 각 방식의 처리과정을 살펴보면 다음과 같다.

가. 기존 리눅스 네트워크 프로토콜의 패킷 처리

네트워크 정합장치로 패킷이 들어오면, 인터럽트가 발생하고 DMA(Direct Memory Access)에 의해 커널 영역의 소켓 버퍼에 패킷이 복사된다. 이 후 소프트웨어 인터럽트에 의해 상위 프로토콜인 TCP/UDP/IP 에 의한 패킷 전달이 일어난다. 그리고, INET/BSD socket 처리 과정을 거치며, 이러한 커널의 수행과정에서 메모리의 복사는 일어나지 않으며, 소켓버퍼의 내용을 그대로 활용한다. 즉, BSD 소켓 구조체는 INET 소켓 구조체를 링크하며, INET 소켓 구조체는 다시 소켓 버퍼 구조체를 링크하는 방식을 사용한다. 그리고, 마지막으로 사용자 영역으로 패킷을 복사한다. 이때, 또 한번의 메모리 복사가 일어나게 된다.

나. 리눅스 NP^2 프로토콜 패킷 처리

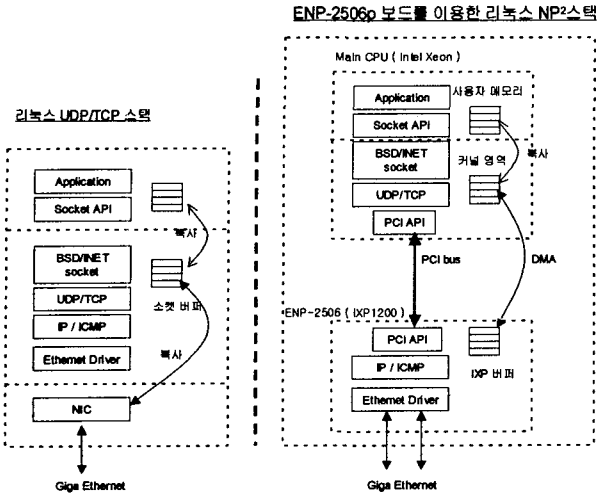
<그림 2>에서 보듯이 기존의 커널내의 프로토콜 스택이 담당하던 소켓버퍼를 이용한 패킷 처리 과정을 IXP1200을 이용하여 사용함으로써 커널의 오버헤드를 줄인다. 일반적인 네트워크 정합장치를 이용할 경우 커널에서 처리되던 패킷 처리가 IXP1200의 여섯 개 마이크로 엔진에서 병렬 처리된다. 각 마이크로 엔진은 4개의 하드웨어 쓰레드를 지원하므로 총 24개의 하드웨어 쓰레드가 패킷 처리를 위해 멀티쓰레딩 된다. 따라서, 대용량의 고속 패킷 처리 시 성능 향상을 기대할 수 있으며, 하드웨어 쓰레드 각각은 자신의 고유의 레지스터 영역을 확보함으로써 문맥 교환 시간을 크게 줄일 수 있다. IXP1200의 6개 마이크로엔진들과 StrongARM은 256MB의 SDRAM을 공유할 수 있으므로 프로토콜 계층간의 패킷 처리 시 복사가 필요 없으므로 성능 향상을 기대할 수 있다.

4.2 리눅스 NP^2 프로토콜 스택 구조

<그림 3>의 왼쪽 그림은 리눅스 운영체제에서의 일반적인 네트워크 정합장치를 사용했을 때의 프로토콜 스택구조를 나타내고, 오른쪽 그림은 본 연구에서 제안한 네트워크 프로세서를 탑재한 네트워크 정합장치에 사용되는 NP^2 프로토콜 스택의 구조를 나타낸다.

가. 일반적인 TCP/UDP 프로토콜 스택

네트워크 정합장치를 통해 들어온 패킷은 인터럽트에 의해



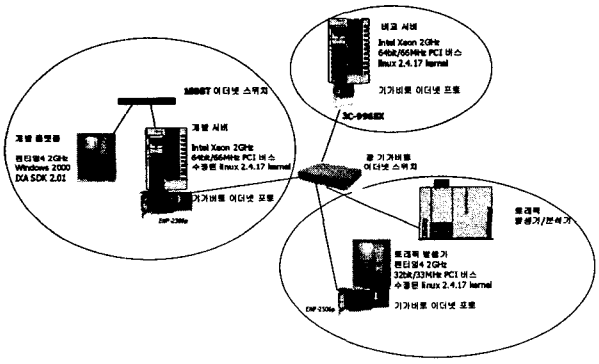
< 그림 3 > 리눅스 NP^2 프로토콜 스택

이더넷 디바이스 드라이버로 전달된다. 이후 리눅스 커널에서 구현되어 있는 IP/UDP/TCP 프로토콜 스택을 거치면서 헤더 검사, 체크섬, 목적지 주소, 포트정보, 분리, 재조합, 버전정보 등이 커널 내에서 검사 된다. 이때 사용자 응용 소프트웨어는 소켓 API 함수를 사용해 패킷 수신을 위한 대기 상태에 놓이게 된다.

나. 네트워크 프로세서를 이용한 NP^2 스택

<그림 3>에서 보는 바와 같이 기존 리눅스 커널 영역에서 운영되던 IP/ICMP 프로토콜, 이더넷 드라이버가 ENP-2506p 네트워크 정합 카드의 마이크로엔진과 StrongARM에 구현된다. 그리고, 서버의 상위 네트워크 프로토콜 스택과 통신하기 위해, PCI를 통해서 연결된다. 제 3장 IXP1200의 구조에서 살펴보았듯이 IXP1200과 서버의 프로세서가 통신할 수 있는 통로는 PCI 버스 뿐이다. 따라서, 리눅스 NP^{2}은 PCI 버스를 이용하여 통신하며, 이를 위한 API를 제공한다.

5. 개발 및 분석환경



< 그림 4 >리눅스 NP^2 프로토콜 개발과 성능 측정을 위한 플랫폼

리눅스 NP^2 프로토콜 개발과 성능 측정을 위하여, <그림 4>와 같은 플랫폼을 본 연구에서는 갖추고 있다. Radisys ENP-2506p 보드는 64비트/66MHz PCI 인터페이스를 가지는 Xeon 서버에 탑재되며, 개발된 리눅스 NP^2 프로토콜의 성능은 동일한 플랫폼에 기가비트 서버 네트워크 카드인 3COM의 3C-996SX를 장착한 서버의 성능과 비교 평가될 것이다.
성능 측정 방법으로는 마이크로엔진의 한 사이클 당 필요로 하는 수행 속도와 마이크로엔진 명령어 수를 계산하는 분석적인 방법과 인공적으로 생성된 데이터와 실제 네트워크 상의 데이터를 입력하여 패킷 처리 속도를 측정하는 방법을 사용한다. 패킷 생성기는 상용 네트워크 패킷 발생기를 사용하거나, IXP1200을 패킷 발생기 전용으로 개발하여 사용한다.

6. 결론 및 향후 연구과제

본 논문에서 구현하는 고성능 네트워크 정합장치의 기본 설계 개념은 병목현상의 주 원인인 커널의 오버헤드를 줄이는 것이다. 이를 위해 본 연구에서는 네트워크 프로세서를 사용하여 기존의 소프트웨어로 처리되는 프로토콜 일부분을 마이크로 엔진의 24개 하드웨어 쓰레드와, 네트워크 프로세서 내부에 있는 마이크로프로세서가 처리하도록 함으로써 호스트 커널의 오버헤드를 경감시켜 성능향상을 도모하였다.

이를 위해 네트워크 프로세서를 이용한 리눅스 NP^2 프로토콜 스택을 설계하였고, 현재 Radisys ENP-2506p 네트워크 프로세서 보드를 장착한 Xeon 서버에서 구현 중에 있다.

향후 계획으로는 1차 구현결과의 성능을 분석을 할 것이며, 분석 결과를 바탕으로 최적화된 성능을 가지도록 수정,보완할 것이다. 또한, 실시간 미디어 스트리밍을 위한 RTP 프로토콜을 포팅하여, 실시간 비디오 전송을 필요로 하는 멀티미디어 서버에 사용할 것이다.

참고문헌

[1] A. T. Campbell, S. Chou, M. E. Kounavis, V. D. Stachtos and J. Vicente, "NetBind: A Binding Tool for Constructing Data Paths and Virtual Routers in Network Processor-based Routers", Comet Group Software Release, December, 2001.
[2] Chu, H.-K.J, " Zero-copy TCP in Solaris," Proceedings of the USENIX 1996 Annual Technical Conference, Pages 253-264, 1996.
[3] Guo Chuanxiong, Zheng Shaoren, "Analysis and evaluation of the TCP/IP protocol stack of LINUX," WCC 2000 - ICCT 2000. 2000 International Conference on Communication Technology Proceedings, Page 444-453 Volume 1, 2000
[4] Skevik, K.-A. , Plagemann, T., Goebel, V., Halvorsen, P, "Evaluation of a zero-copy protocol implementation," Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey, Pages 324-330, 2001.
[5] InfiniBand^{SM} Trade Association, InfiniBand InfiniBandTM Architecture Specification Volume 1 Release 1.0.a Annex A4 Sockets Direct Protocol - April 2, 2002.
[6] "IXP1200 Hardware Reference Manual", Intel, Dec, 2001.
[7] Dunning, D., Regnier, G., McAlpine, G., Cameron, D., Shubert, B., Berry, F., Merritt, A.M., Gronke, E., Dodd, "The Virtual Interface Architecture," IEEE Micro, Volume 18, Issue 2, Pages 66-76, 1998.