사용자 수준 프로토콜을 사용한 파일 전송 성능 향상*

조원범°, 강다원, 조유근
서울대학교 컴퓨터공학과

Improved File Transmission Using User Level Protocol

Won-bum Cho, Da-won Kang, Yoo-kun Cho
Department of Computer Engineering, Seoul National University

요 약

본 논문은 Mach 마이크로 커널 운영체제에서 네트워 프로토콜과 운영체제의 오버헤드를 줄이는 사용자 수준의 프로토콜의 구현을 제시한다. 기존 사용자 수준의 프로코콜의 구현들은 그 성능에 있어서 커널내 프로토콜의 그것에 비해 낮은 성능을 보여왔다. 이에 비해 본 논문에서는 더욱 높은 성능 향상을 보이는 사용자 수준의 프로토콜 구조의 설계와 구현을 제시한다.

또한 Mach의 공유 메커니즘을 쓴 사용자 프로토콜과 BSD UNIX 서버와의 공유 메모리를 사용하여 디스크에 저장된 파일을 네트웍 패킷으로 보낼 경우의 전송 능력을 향상시켰다 이는 HTTP와 FTP 같이 최근 그 사용이 증대되고 있는 응용의 성능을 향상시킨다.

1. 개 요

사용자 수준라이브러리 연구의 핵심 기반은 네트워크에 대한 응용 프로그램의 인터페이스는 운영체제로부터 독립적이며 운영체제의 인터페이스로부터 분리될 수 있다는 사실이다. 네트워크 서비스는 공유 라이브러리와 운영체제 서버 프로세스에 의해 제공되며, 공유 라이브러리는 성능에 큰 영향을 미치는 서비스 부분을 구현하고 , 운영체제 서버는 응용 프로그램에서 구현하기 힘든 서비스나 성능에 영향을 거의 미치지 않는 부분을 구현한다 운영체제와 네트워크 서비스의 인터페이스를 분리함으로써, 표준 응용 프로그램 인터페이스에 의해 정의된 운영체제 기능들의 시멘틱은 유지하면서, 응용프로그램과 네트워크간의 빠른 경로를 제공해 줄 수 있다.

최근, 인터넷의 사용이 급속도로 증가하고 HTML 문서나 MPEG과 같은 대용량의 멀티미디어 자료 전송 요구가 늘어남에 따라 네트워크를 통한 파일 전송 기능이 시스템의 운용에 큰 비중을 차지하게 되었으며, 많은 서비에 있어서 이러한 기능이 시스템 전체의 성능에 큰 영향을 미치게 되었다. 이러한 응용을 위해 사용자 수준 프로토콜과 BSD 서버 사이에 공유메모리를 구현하여 파일 전송 성능을 향상시켰다.

* 이 연구는 특정목적기초연구 사업의 지원을 받아 이루어졌습니다

2. 사용자 수준 프로토콜의 설계

2.1. 구성 요소

사용자 수준 프로토콜을 구성하고 있는 요소들은 다음과 같다.

- 운영체제 서버 : 연결 설정, 해제, 예외적인 네트워크 패킷의 처리와 같이 성능에 큰 영향을 미치지 않는 네트워크 작업을 담당한다. 또한, 라우팅 정보와 TCP 포트 이름공간과 같이 오래 지속되는 공유 프로토콜 상태들을 유지한다.
- 라이브러리 : 각각의 응용프로그램 내부에 있는 라이브러리는 프로토콜 스택을 구현하며 특히 데이터 전송과 관련된 부분을 구현한다.
- 네트워크 인터페이스 : 네트워크 하드웨어 바로 위에서 소프트웨어 층을 제공한다.

2.2. 프로토콜의 구조

네트워크 서비스는 공유 라이브러리와 운영체제 서버 프로세스를 통해 제공된다. 공유 라이브러리는 성능에 큰 영향을 미치는 서비스 부분을 구현하고, 운영체제 서버는 응용 프로그램에서 구현하기 힘든 서비스나 성능에 영향을 거의 미치지 않는 부분을 구현한다. 운영체

제와 네트워크 서비스의 인터페이스를 분리함으로써, 표준 응용 프로그램 인터페이스에 의해 정의된 운영체제 기능들의 시멘틱은 유지하면서, 응용 프로그램과 네트워크간의 빠른 경로를 제공해 줄 수 있다 구체적으로 말하자면, 응용 프로그램의 주소공간에 있는 라이브러리 코드가 네트워크 프로토콜을 구현하여, 데이터를 네트워크로 전송하거나 네트워크로부터 받는다. 반면에 운영체제 서버는 전송과 수신 외에 응용 프로그램이 네트워크를 조작할 때 필요로 하는 기능들을 제공하고 이들을 관리한다.

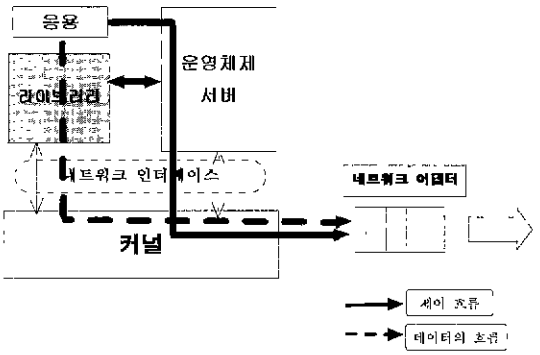

그림 2.1 새로운 프로토콜 구조

위와 같은 방식을 사용하여 얻을 수 있는 장점은 다음과 같다.

1. 효율성 : 임계적 프로토콜 경로를 응용 프로그램 주소공간에 둠으로써 보호 구역 경계 교차, 데이터 복사, 불필요한 소프트웨이 층의 구현 등을 피할 수 있다.
2. 유연성 : 사용자 수준의 네트워킹 소프트웨어는 운영체제의 다른 부분들과 독립적으로 개발, 구성, 특성화할 수 있으며, 운영체제가 아닌 응용 프로그램이 네트워크 프로토콜의 동작을 정의할 수 있다.

그 외 본 사용자 수준 프로토콜의 부차적인 목표는 다음과 같다.

1. 기존 프로토콜 코드의 재사용
2. 기존의 프로토콜을 이용하는 응용 프로그램과의 소스 수준의 호환성

2.3. 프로그래밍 인터페이스

본 설계는 기본적으로 소켓 인터페이스를 유지하며 커널의 프로토콜과 유사한 자료구조를 가진다. 사용자 수준 라이브러리는 BSD UNIX 소켓 인터페이스를 지원하기 위해 프로토콜의 상태를 운영체제 서버와 응용 프로그램의 주소공간에 전달하기 위한 대리자 구조를 가진다. 대리자는 응용 프로그램의 주소공간에 상주하는 코드의 모듈이다.

대리자는 기존에 운영체제가 제공하던 시스템 호출 인터페이스를 프로시져 호출 형태로 구현하여 사용자의 응용 프로세스에 제공한다. 소켓과 관련된 응용 프로그램의 시스템 호출은 처음에 대리자 모듈에게 전달되고, 대리자 모듈이 처리하여 네트워크 인터페이스에 그대로 전달하거나 운영체제 서버의 시스템 호출로 변형한다. 그림 2.2은 라이브러리 내에 대리자 모듈로 구현된 프로시져 호출과 이러한 메커니즘에 의한 시스템 호출들의 흐름을 보여준다.

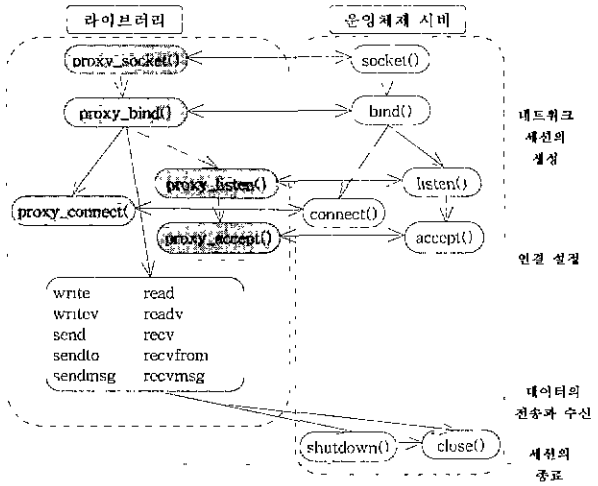

그림 2.2 라이브러리의 프로시져 호출과 시스템 호출의 흐름

read, write와 같은 높은 성능이 요구되는 기능은 사용자 프로세스에 링크된 라이브러리가 처리하며, 기타의 기능은 모듈을 통해 운영체제 서버에 전달된다

2.4. 메모리 관리

기존의 TCP/IP 구현은 패킷 헤더의 처리와 사용량을 줄이기 위해 복잡한 메모리 관리 시스템을 이용한다. 이러한 요구 사항을 충족시키는 것 중 하나가 Berkeley UNIX의 mbuf 자료 구조이다.

본 논문에서 구현한 프로토콜 구조에서는 사용자 버퍼와 mbuf사이에서 데이터 복사를 수행하는 호출부를 변경하여 사용자 버퍼를 직접 프로토콜 스택에 전달한다. 사용자 버퍼의 자료구조는 mbuf에서의 경우와는 달리 버퍼의 체인을 다루는 부분은 존재하지 않으며, 단일의 연속 공간에 데이터를 저장하고 처리할 수 있다. 이 기법을 사용하면 mbuf와 사용자 버퍼와의 복사가 발생하지 않으며, 버퍼 체인을 처리하기 위한 포인터 연산이 필요 없으므로 작은 데이터를 처리하는 경우 성능의 향상을 예상할 수 있다

3. 파일 시스템과의 공유메모리 설계

본 절에서 설명하는 공유메모리는 디스크에 저장된 파일의 내용을 사용자 수준 프로토콜을 사용하여 네트워크를 통해 전송하는 기능을 제공한다. 이는 BSD 서버를 사용할 경우 디스크 접근시에 일어나는 성능 지하를 방지한다.

3.1. 공유메모리 구조

Mach에서 기존의 UNIX 기능을 제공하는 BSD 서버를 수정하여 서버와 사용자 프로세스의 프로토콜 간에 공유 메모리를 제공한다. 사용자로부터 전송이 발생하면 Mach의 IPC를 통해 BSD 서버에 요청이 전달되며 서버는 자체의 버퍼 블록을 사용하지 않고 바로 공유된 영역으로 디스크의 내용을 복사한다. 단, 서버 내의 버퍼에 사용자가 원하는 파일의 내용이 있을 경우에는 불필요한 디스크로의 접근을 막기 위해 버퍼의 내용을 복사하여 준다. 복사를 마치고 사용자 프로

세스로 복귀하면, 사용자 프로세스는 자신의 주소공간에 존재하는 사용자 수준 프로토콜을 사용하여 네트워크 디바이스를 통해 자료를 전송한다.

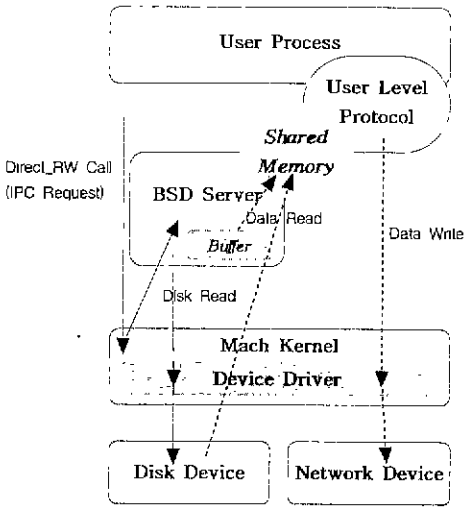

그림 3.1 사용자 수준 프로토콜을 사용한 공유메모리 구조

3.2. 인터페이스

공유메모리를 사용하기 위해 다음과 같은 사용자 호출을 제공한다.

direct_rw(int source, int bytes, int target)

source : 자료을 읽어올 파일의 descriptor
bytes : 읽어올 자료의 크기
target : 자료를 전송할 소켓의 descriptor

위의 함수는 source가 지정하는 파일에서 bytes만큼의 자료를 읽어 target이 지정하는 소켓으로 자료를 전송한다

4. 구현 및 성능평가

성능 평가에 사용한 시스템의 구성은 다음과 같다

- Pentium 200Mhz CPU, 32M Ram, 10Mbps Ethernet, EIDE HDD

4.1. 사용자 수준 프로토콜 성능 평가

사용자 수준 프로토콜과, 일반 모노리딕 FreeBSD 커널을 mbuf를 사용한 경우와 그렇지 않을 경우로 나누고 Mach를 사용한 OS 서버와도 비교해 보았다. 성능 측정에는 네트워크 벤치마크 프로그램인 netperf 2.1을 사용하였으며, 평가 결과 mbuf를 사용하지 않은 사용자 프로토콜이 모든 크기의 패킷에서 최상의 성능을 보였다.

4.2. 파일 전송 성능 평가

다양한 크기의 파일을 디스크에서 읽어 전송을 마칠 때 까지의 시간을 측정하였다. 측정 대상은 왼쪽에서부터 모노리딕 FreeBSD, 사용자 수준 라이브러리, BSD 서비이며, 진송 데이터의 크기를 다르게

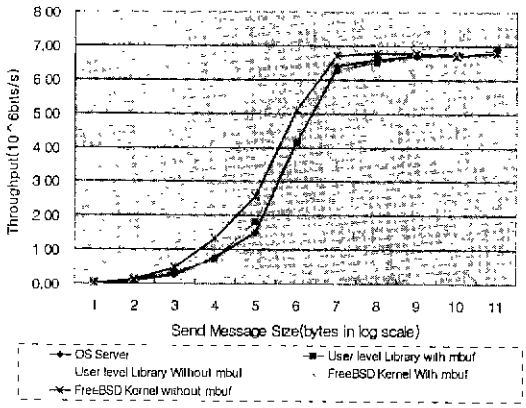

그림 4.1 사용자 수준 프로토콜 성능 측정 비교

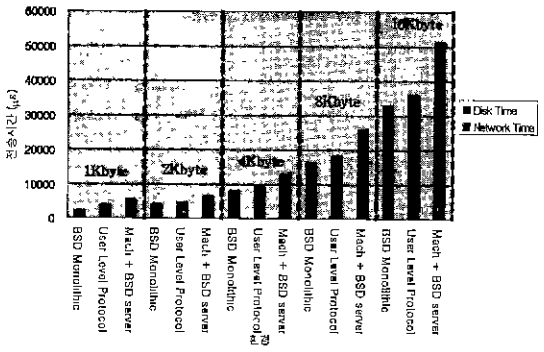

그림 4.2 파일 전송 성능 평가

하여 각각의 성능을 측정하였다. 측정 결과 사용자 수준 프로토콜을 사용한 경우 기존의 BSD 서버보다 39-41% 정도의 성능이 향상되었다.

참고문헌

[1] Chandramohan A. Thekkath. Thu D. Nguyen, Evelyn Moy, Edward D. Lazowska. Implementing Network Protocols at User Level. *IEEE/ACM Transactions on Networking* 1(5):554-565, OCT 1993

[2] Maeda, C. And Bershad, B N. Networking Performance for Microkernels. *Proceedings of the Third Workshop on Workstation Operating Systems*, pp. 154-159, April 1992

[3] R.M. Watson and S. A Mamrak. Gaining efficiency in transport services by appropriate design and implementation choices. *ACM Transactions on Computer Systems*. 5(2):97-120, May 1987.

[4] Shin-Yuan Tzou and David P. Anderson. the performance of message-passing using restricted virtual memory remapping Software-Practice and Experience, 21:251-267, March 1991.