2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

멀티미디어 서버의 커널 메커니즘 설계

남상준, 이병래, 장경아, 김태윤
고려대학교 컴퓨터학과
e-mail: sjnam@netlab.korea.ac.kr

The Design of Kernel Mechanism for Multimedia Server

Sang-Jun Nam, Byung-Rae Lee, Kyung-Ah Chang, Tai-Yun Kim
Dept. of Computer Science and Engineering, Korea University

요 약

멀티미디어 서비스에 대한 사용자의 요구가 증가되고 있으나 서버 시스템은 이러한 멀티미디어 데이터에 대해 사용자들에게 효율적으로 공급하지 못한다. 본 논문에서는 기존의 운영체제에 멀티미디어 데이터 전송을 효율적으로 만족시키기 위한 방안으로 멀티미디어 데이터 처리를 SIO(Special Input/Output)메커니즘을 커널에 내장하는 방법을 제안한다. SIO 메커니즘은 일반적인 서버 시스템에서 발생하는 사용자 모드와 커널 모드 사이의 데이터 복사와 문맥 교환을 줄이고, 효율적 멀티미디어 데이터 전송 구조를 지원한다.

1. 서론

멀티미디어 서버 시스템의 애플리케이션은 저장 장치와 네트워크 하부 시스템 사이에서 주기적으로 데이터를 주고받는다. 따라서 기대되는 작업에 대해 서버 시스템은 높은 성능으로 디자인되고 전송되어야 한다. 여러 방안이 제시되고 있으나 저장 장치와 네트워크 하부 시스템 사이의 효과적인 경로는 제어하지 못한다[2]. 대부분의 멀티미디어 응용 프로그램들은 사용자 모드와 커널 모드에서 과도한 데이터 복사와 문맥 교환이 발생한다[3].

본 논문에서는 멀티미디어 데이터 전송을 위한 SIO(Special Input/Output) 메커니즘을 커널에 내장함으로써 비디오와 오디오 같은 실시간 서비스에 만족하는 효율적 전송 메커니즘을 제안한다.

제안된 SIO(Special Input/Output) 메커니즘은 빠른 전송을 수행하기 위해 UDP 전송 메커니즘을 변경하여 수행하고 SIO 메커니즘을 커널에 내장하여 이를 기반으로 리눅스 환경에서 구현과 성능 평가를 한다. 이러한 SIO 메커니즘은 멀티미디어 데이터 전송의 서비스 품질을 향상시키고 입출력(Input/Output) 시스템의 성능을 향상시킨다.

본 논문의 구성은 2장에서 관련 연구를 분석하고, 3장에서 새로운 네트워크 메커니즘인 SIO 시스템 콜을 설계, 구현하고 일반적 서버와의 비교, 분석한다. 4장에서는 본 논문의 결과 및 향후 과제에 대해 기술한다.

2. 관련 연구

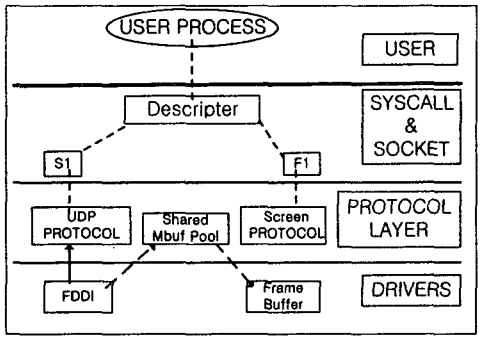
2.1 스플라이스(Splice) 메커니즘

먼저 일반적인 서버 시스템은 사용자 공간에서 네트워크 장치까지 두 단계의 데이터 복사가 사용자 모드와 커널 모드 사이에서 발생한다.

첫 번째 복사는 read 콜이다. 이것은 커널의 버퍼 캐쉬에서 사용자 버퍼 공간까지 데이터를 옮기는 경우이다. 두 번째 복사는 write 콜이다. 이 write 콜은 사용자 버퍼 공간의 데이터를 네트워크 버퍼로 보내는 역할을 한다. 이러한 서버 시스템은 멀티미디어 데이터(오디오, 비디오, 애니메이션 등)와 같은 캐싱 속성을 가진 데이터들은 기대하는 것만큼의 효과를 보지 못한다[5]. 또한 이 서버 시스템은 메모리 공간을 많이 사용한다. 데이터는 종종 이것이 재 사용될 수 있음에도 불구하고 대체될 수 있다.

일반적인 서버 전송 메커니즘의 단점인 잦은 데이터 복사의 보완이 스플라이스(Splice)[4] 메커니즘이

다. 스플라이스 메커니즘은 I/O 디바이스들 사이의 데이터 경로를 직접적으로 다루는 커널을 설정하기 위해 사용하는 시스템 콜이다. 스플라이스에 관계하여 어떤 애플리케이션은 메모리 주소를 참조하지 않는 식별자(Descriptors)를 사용하여 운영체제에 관계하여 직접적으로 데이터 근원지와 목적지 사이의 관계를 표현한다. <그림 1>를 보면 UNIX에서 구현한 스플라이스 메커니즘의 모습이다.



<그림1> 스플라이스 메커니즘

소켓 버퍼가 준비가 되었을 때 사용자 프로세스는 소켓 버퍼 속에 있는 데이터를 그 자신의 연속적인 사용자 주소 공간까지 복사하면서 그것의 커널 내부에서 수행을 계속하도록 알려준다.

스플라이스 메커니즘의 단점은 다음과 같다.

- 식별자(Descripter) 포인터를 두어 연산을 하므로 추가의 포인터 연산으로 인해 비효율이 발생한다.
- 들어오는 패킷마다의 추가의 식별자 플래그를 세팅하여야 한다.
- 새로운 식별자로 인해 다른 장치와의 호환성이 결여된다.

데이터 복사를 최소화하려는 시도는 이전부터 연구되어 왔다. 보호 영역(protection domains)을 통해 물리적 데이터 이동을 최소화하는 기술인 fbufs[6], mmbufs(Multimedia Memory Buffer)라는 새로운 구조체를 만들어 커널 영역에서 데이터 전송을 하는 MARS[5] 메커니즘, 위의 Splice[4] 메커니즘 등이 있다. 이러한 연구들이 데이터 복사를 줄임으로써 데이터 전송 속도를 높이려는 것이다.

2.2 멀티미디어 서버 시스템의 입출력 메커니즘

최근에 인터넷상에서 멀티미디어 서비스들의 전송에 대한 과부하와 실시간 전송을 위한 연구에 관심이 집중되고 있다. RTP 프로토콜과 같은 멀티미디어 데이터 처리는 전송 계층과 네트워크 계층의 상위에서 작동하도록 디자인 된 프로토콜이다[1].

사용자 모드에서 멀티미디어 데이터와 함께 만들어진 멀티미디어 데이터 헤더는 그 하위 계층인 UDP 프로토콜로 전달되고 네트워크 인터페이스로 전달된다. 바로 이러한 점에서 멀티미디어 데이터의 생성과 하부 계층으로의 전송을 무수히 반복하는 경우는 과도한 데이터 복사와 문맥 교환이 발생한다.

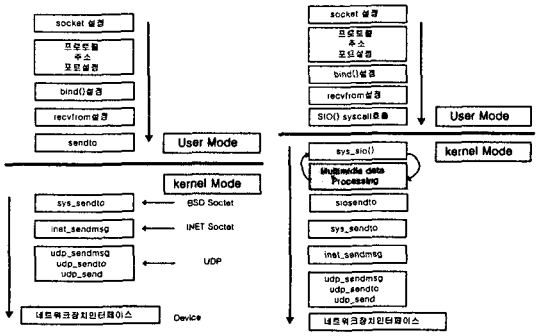
현재 멀티미디어 데이터는 사용자 애플리케이션에서 작업이 이루어진다. 멀티미디어 데이터 전송을 운영체제의 커널 영역에 두면 멀티미디어 데이터를 처리할 때 빠르게 작업이 이루어 질 수 있다. 이러한 스택의 변화를 본 논문에서 제안한다.

3. 제안된 SIO 메커니즘 설계

3.1 SIO(Special Input/Out) 메커니즘

SIO 메커니즘은 애플리케이션과 커널 영역 사이의 데이터 복사와 문맥 교환의 오버헤드를 줄이기 위해 일반적인 서버 시스템에 존재하는 데이터 전송 단계를 줄임으로써 전송상의 향상을 보고자 한다[3].

오디오나 비디오 데이터와 같은 멀티미디어 데이터는 변경이 많이 일어나지 않는 연속적인 스트림의 전송이다. 이와 같은 전송은 항상 새로운 데이터를 생성해서 전송하는 것이 아니라 저장 장치로부터 읽어온 데이터를 연속적으로 전송하면 된다. 일반적으로 멀티미디어 데이터는 UDP 전송 계층을 이용하므로 이러한 오버헤드가 발생한다. 리눅스 기반의 일반적인 UDP 전송 메커니즘의 커널 경계 부근에서 데이터 전송 경로는 <그림 2>의 (a)와 같다. <그림 2>의 (a)는 애플리케이션에서 UDP 프로토콜을 거쳐 네트워크 인터페이스로 데이터 전송 경로를 나타낸 것이다. <그림 2>의 (b)는 멀티미디어 데이터 전송을 효율적으로 지원하기 위한 패킷 처리를 두었다. <그림 2>의 (b)와 같은 모델을 가진 운영체제는 현재 존재하고 있지 않다. 이와 같은 모델로 사용자 모드와 커널 모드 사이의 데이터 복사와 문맥 교환을 줄여 전송 효율을 증대시키는 동시에 기존의 커널 메커니즘 최소한의 변경으로 호환성도 유지하였다. 이러한 SIO 메커니즘을 통해 멀티미디어 데이터를 커널에서 처리해 네트워크 인터페이스로 보내는 SIO 메커니즘은 <그림 2>의 (b)이다.



(a) (b)

<그림 2> 멀티미디어 데이터 전송 메커니즘

본 논문에서의 제안하는 SIO 메커니즘은 멀티미디어 데이터 처리를 SIO 메커니즘을 이용해 커널로 내장하는 것이다. 멀티미디어 데이터 패킷이 UDP 프로토콜 기반으로 전송이 수행되므로 UDP 기반의 SIO 전송 메커니즘과 잘 호환된다[7].

다음은 커널 내부로 옮긴 멀티미디어 데이터 처리와 SIO 메커니즘의 이점은 다음과 같다.

- 멀티미디어 데이터 패킷 처리
  멀티미디어 데이터 패킷을 생성함에 있어, 기존의 방식은 사용자 애플리케이션에서 이루어지는 것을 커널 내부에 내장함으로써 처리 속도의 향상과 멀티미디어 데이터 응용 애플리케이션의 인터페이스를 단순하게 만들 수 있게 하였다.
- 커널 메커니즘의 수용[8]
  최소한의 커널 메커니즘의 수정으로 효율적인 멀티미디어 데이터의 서비스를 지원한다.

3.2 SIO 구현 및 성능평가

<그림 2> (a)와 같은 메커니즘을 가진 서버 시스템은 현재 존재하고 있지 않다. 멀티미디어 데이터 전송에 적합한 전송 메커니즘을 가지고 있지 않는 리눅스 시스템의 커널에 새로운 SIO 전송 메커니즘을 적용하여 구현하고 비교한다.

리눅스 시스템에 멀티미디어 데이터 전송을 커널 영역에 내장하였다. 리눅스의 UDP 전송 메커니즘은 <그림 2>의 (a)와 같이 동작하기 때문에 데이터 전송시 커널 모드와 사용자 모드 사이에서 데이터 복사와 문맥 교환이 발생한다.

위의 서버 애플리케이션의 알고리즘을 보면 커널에 내장한 SIO 메커니즘을 sio 시스템 콜로 호출한다. 그리고 sio 함수에서 전송하고자 하는 파일의 포인터와 멀티미디어 데이터 패킷의 정보를 보낸다.

넘겨준 정보와 포인터를 이용해 커널 영역에서는 모든 멀티미디어 서비스를 수행하고 결과값만을 리턴하게 된다. 이때의 애플리케이션은 리턴 값이 오는 동안 휴지 상태로 머물게 된다. 이로써 SIO 메커니즘은 멀티미디어 데이터 전송시 SIO 메커니즘과 함께 멀티미디어 패킷 전송시 사용자 모드와 커널 모드의 데이터 복사의 횟수를 줄여서 효율적 전송을 기대하고, 애플리케이션의 단순화로 인해 사용자 영역의 부하를 줄이게 되었다.

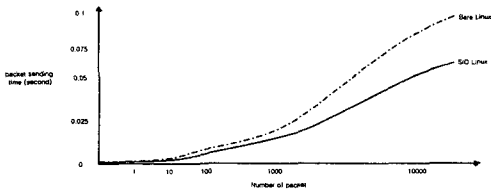
3.3 SIO 메커니즘의 성능 평가

리눅스 시스템에 SIO 메커니즘을 커널에 내장하였다. 실험 환경의 구성을 위해 2개의 리눅스 시스템을 사용한다. 한쪽은 서버 시스템으로 멀티미디어 데이터 패킷을 전송하고 다른 한쪽은 이 패킷을 수신함으로써 성능을 비교하였다. 두 시스템의 시스템적 하드웨어 사양은 다음과 같다.

- 서버 주소: 163.152.39.115
- 클라이언트 주소: 163.152.39.120
- Linux Kernel Version: 2.2.12
- GCC Version: 2.91.66
- Libc6 Version: 2.1.1

실험에 대한 세부 사항과 수행 시나리오는 다음과 같다.

- 수행 시나리오는 일반적인 UDP 전송과 동일하게 하고 일반적 리눅스와 SIO 메커니즘 리눅스의 패킷 전송 시간을 비교하였다.
- 클라이언트의 요청을 받은 서버 시스템은 1개, 10개, 100개, 1000개, 10000개의 패킷을 각각 따로 전송하여 서버 측면에서의 전송 완료 시간을 비교하였고 실험 결과는 <그림 3>과 같다.



<그림 3> 일반적 리눅스와 SIO 메커니즘 리눅스의 패킷 수에 따른 전송 속도 비교

실험 결과는 각각 100회씩 반복하여 평균을 구한 값이다. <그림 3>에서 보는 바와 같이 각각의 패킷 개수에 대해 일반적 리눅스의 전송 시간보다 SIO 메커니즘의 리눅스가 속도면에서 향상되는 그래프를 볼 수 있다. <그림 3>에서 패킷 한개의 전송에서부터 전송 패킷의 수가 많아질수록 전송 속도 차이는 점점 향상됨을 알 수 있다.

<그림 3>에서 커널 내부에 SIO 메커니즘으로 UDP 전송을 수행한 리눅스의 전송횟수를 보면, 일반적 리눅스보다 약 6% ~ 23% 의 성능향상을 보였다. 더욱이 현재 비디오나 오디오 스트림이 많은 대용량 멀티미디어 데이터 전송을 고려할 때 성능 향상의 폭이 증가함을 알 수 있다.

4. 결론 및 향후 연구 과제

사용자의 요구를 충족시키기 위해 서버 시스템과 네트워크 대역폭이라는 두 가지 방향으로 연구가 진행되고 있다. 본 논문에서는 이러한 요구에 의해 멀티미디어 데이터의 네트워크 전송 지원을 위한 SIO 메커니즘을 제안하였다. 이 SIO 메커니즘은 커널 공간에서 사용자 공간으로의 데이터 복사가 필요하지 않고, 이를 수행하기 위한 멀티미디어 데이터 전송의 사용자 프로세스를 스케줄링할 필요가 없는 성능상의 향상이 있었다.

향후 연구 과제는 현재의 일반적인 운영체제 네크워크 전송 단계는 기존의 텍스트 위주의 데이터을 처리할 수 있도록 제작되어 있어서 멀티미디어 데이터를 처리하기에는 적합하지 않다. 이러한 속도면에서 향상된 SIO 메커니즘을 실시간 패킷에 우선 순위를 지원하는 스케줄러를 연구하여 합치면 애플리케이션 데이터 전송에 적합한 멀티미디어 서버 시스템의 핵심 기술로 활용할 수 있을 것이다.

참 고 문 헌

[1] H. Schulzrinne and S. Casner, "RTP: the Real-time Transport Protocol", Audio-Video Transport Working Group, RFC 1889, January 1996.

[2] Jose C. B., "Effects of Data Passing Semantics and Operating System Structure on Network I/O Performance", Rice University, 1997

[3] Moti N. Thadani and Yousef A. Khalidi, "An Efficient Zero-Copy I/O Framework for UNIX", Technical Report, SMLI TR-95-39, Sun Microsystems Lab, Inc., May 1995.

[4] Kevin Fall and Joseph Pasquale, "Improving Continuous-Media Playback Performance with In-kernel Data Paths", California University, Computer System Lab., Dept. Computer Science and Engineering. 1994.

[5] Milind M. B., Dakang W., Guru M.P. and Xin J.C., "Enhancements to 4.4 BSD UNIX for Efficient Networked Multimedia in Project MARS", Washington University Dept. of Computer Sci.

[6] Peter Druschel and Larry L. Peterson, "Fbufs: A High-Bandwidth Cross-Domain Transfer Facility", Proceedings of 14th SOSP, Dec. 1993.

[7] J. Rosenberg and H. Schulzrinne, "Issues and Options for an Aggregation Service within RTP", Internet Draft, Internet Engineering Task Force, Nov. 1996, Work in progress.

[8] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly, January 2001.