리눅스 기반 DVR시스템 통신모듈 설계 및 성능 측정

김지훈[O] 홍승욱 안종석

동국대학교 컴퓨터공학과

{whitefan[O], swhong, jahn}@dgu.edu

Estimation of a Linux based Communication Module for DVR System

JiHoon Kim[O] SeongWook Hong JongSuk Ahn

Dept. of Computer Engineering, DongGuk University

요 약

DVR은 아날로그(analog) 테이프 레코더(tape recorder)에 비해 여러 가지 장점을 가지고 있어 빠른 속도로 아날로그 테이프 레코더를 대치하고 있다[1]. 지금까지 대부분의 DVR시스템은 MS사의 윈도우즈(MS-windows)에 기반하여 개발되었으나 MS윈도우즈 운영체제의 비공개 소스 코드(source code)로 인해 효율적인 시스템을 개발하기 어려운 문제점이 있다. 이에 본 논문에서는 운영체제의 코드가 오픈 되어 있어 누구나 원하는 소프트웨어를 개발할 수 있는 리눅스(Linux) 운영체제를 기반으로 한 DVR시스템의 통신 모듈을 설계한다. 또한 본 논문에서는 리눅스 운영체제의 스케쥴링에 의한 실시간 프로그램의 부하를 측정하고, 본 논문에서 구현한 통신 모듈을 사용하여 프로세스의 개수에 따른 DVR시스템의 성능을 측정하였다. 측정 결과 구현한 통신 모듈은 20Mbps이상의 전송 속도로 수행되는 것으로 나타났다.

1. 서 론

DVR시스템은 디지털 비디오의 품질의 우수성, 다양한 기능의 제공, 그리고 제어의 용이성으로 인해서 빠른 속도로 아날로그 테이프 레코더를 대치하고 있다. 또한 DVR시스템은 개발비용을 적게 사용하면서도 개발 시간을 단축할 수 있는 임베디드(embedded) 시스템으로 구현된다. 일례로 어느 DVR시스템은 TriMedia사의 DSP(Digital Signal Processing)을 장착한 보드와 리눅스를 운영체제로 사용하는 PC로 구성된다.

리눅스 운영체제는 오픈 되어있어 누구나 원하는 소프트웨어를 개발할 수 있다는 장점이 있기 때문에 많은 임베디드 시스템에서 리눅스를 운영체제로 사용하고 있다. 그러나 리눅스 운영체제는 실시간(real-time) 운영체제가 아니기 때문에 DVR과 같은 실시간 응용 프로그램을 수행할 때에는 성능에 영향을 미치게 된다. 즉 리눅스에서의 프라이어티(priority)에 의한 균등 시간 배분 스케쥴링 (scheduling) 방식은 실시간 프로그램 수행에 적합하지 않다.

이러한 리눅스 운영체제의 문제점에도 불구하고 많은 실시간 응용 시스템들은 리눅스 운영체제를 실시간 운영체제로 사용하고 있다. 이는 시스템의 부하를 줄여 항상 실시간 응용 프로그램의 요구를 충족시키는 일종의 오버 프로비젼(overprovision) 방식이다. 본 논문에서는 리눅스 시스템을 임베디드 시스템으로 사용할 때 스케쥴링으로 인한 응용 프로그램의 성능에 미치는 영향을 평가한다. 구체적으로 리눅스 시스템에서 프로세스의 개수가 증가할 때, 하나의 프로세스가 수행하기 위해 기다려야 하는 시간을 평가한다. 또한 본 논문에서는 리눅스 운영체제 하에서 본 연구에서 개발하고 있는 DVR시스템의 하부 통신 모듈의 성능을 측정한다.

본 논문의 구성은 다음과 같다. 2장에서는 구현중인 실제 DVR시스템의 하부 통신 구조를 기술하며, 3장에서는 리눅스 운영체제의 스케쥴링에 의한 실시간 프로그램의 부하를 측정한다. 4장은 2장에서 기술한 하부 통신 구조의 성능을 평가하고 마지막으로 5장에서는 실험 결과를 요약하고 향후 연구 과제를 기술한다.

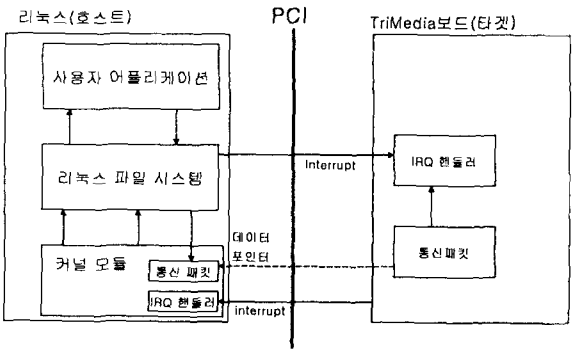
2. 리눅스 기반의 DVR시스템 구조

(1) TriMedia 보드를 사용한 DVR시스템

본 절에서는 본 논문에서 리눅스(이하 호스트, host) 기반의 DVR시스템을 구현하기 위하여 사용 한 TriMedia보드(이하 타겟, target)의 구조를 기술한다.

TriMedia사의 DVR용 보드는 자체적으로 DSP를 장착하고 있고, IRQ(Interrupt Request) 핸들러(handler)와 PCI(Peripheral Component Interconnect)를 사용하여 PC와 통신을 하게 된다[2]. 멀티미디어 데이터(오디오, 비디오 데이터)가 TriMedia보드의 입력단을 통하여 입력되면 인터럽트(interrupt)를 발생시켜 데이터가 입력되었음을 PC로 알려준다.

<그림 1>는 TriMedia보드와 리눅스를 운영체제로 사용하는 컴퓨터와의 통신방법 및 전체 시스템 구성도를 나타낸다. TriMedia보드를 사용하는 DVR시스템은 TriMedia사에서 구현되어 있는 세 가지 하부 통신 방식, 즉 이벤트(event), 메시지(message), 그리고 공유 메모리(shared memory)방식을 기반으로 구축된다. 각각의 하부 통신 방식을 설명하면 다음과 같다.



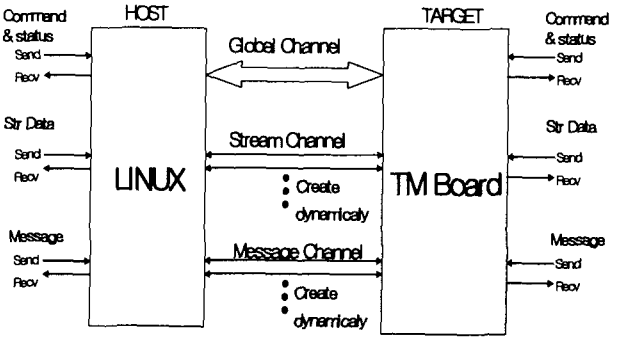
<그림 1. TriMedia기반의 DVR시스템>

이벤트 방식은 호스트 또는 타겟에서 PCI를 통한 인터럽트 또는 호스트나 타겟에서 어떤 사건(채널의 생성· 소멸, 데이터의 입·출력)이 발생하였을 때 사용하는 통신 방식이다. 메시지 방식은 호스트와 타겟간에 통신용 패킷을 이용하여 간단한 제어 명령이나, 32 바이트 이하의 간단한 데이터를 주고받을 때 사용되는 방식이다. 공유 메모리 방식은, 호스트와 타겟간의 실질적인 멀티미디어 자료 송수신에 관련된 방식으로, 대용량의 데이터를 교환할 때 사용되는 방식이다. 호스트에서는 물리적인 메모리의 주소를 사용하고, 타겟에서는 가상의 메모리 주소를 사용한다. 호스트와 타겟간에 멀티미디어 데이터를 전송하고자 할 때는 먼저 공유 메모리를 설정하고, 공유메모리에 전송하고자 하는 데이터를 저장한다. 그 후 메모리의 주소를 서로에게 알려주면 호스트와 타겟간에 이 공유 메모리를 이용하여 멀티미디어 데이터의 송수신이 이루어진다. 공유 메모리 방식을 사용할 때에는 하나의 프로세스만이 중요 데이터에 접근할 수 있도록 하는 세마포어(semaphore) 메카니즘(mechanism)을 지원해야 한다.

(2) 리눅스 기반의 DVR시스템

본 논문에서 구현한 리눅스 기반의 DVR시스템은 TriMedia에서 구현한 세 가지 하부 통신 방식을 기반으로 하였으며, 호스트와 타겟간의 고속의 데이터 통신을 제공하기 위하여 채널 개념과 DMA(Direct Memory Access)방식을 사용하였다. 채널은 TriMedia의 커널(kernel) 드라이버(driver)에서 제공하는 세 가지 하부 통신 방식과, 두 개의 공유 순환(circular) 큐(queue)로 구현된다. 즉, 호스트와 타겟간의 통신은 모두 이 채널을 통하여 이루어지며 채널은 전송되는 데이터의 특성에 따라 세 가지로 나누어지며, 제어, 스트림(stream), 메시지(message)채널이 본 논문에서 구현한 DVR시스템에서 사용된 논리적 채널의 세 가지 종류이다.

제어 채널은 TriMedia에서 제공하는 이벤트와 메시지 방식을 이용하며 호스트와 타겟 프로그램이 시작될 때 생성되고 프로그램이 종료될 때 까지 사용되는 채널이다. 또한 제어 채널을 통하여 메시지와 스트림 채널의 생성, 소멸, 관리, 전송 상태 등에 관련된 명령어들이 교환되도록 구현하였다. 스트림 채널은 호스트와 타겟간에 필요에 의해 능동적으로 생성, 소멸되는 채널이며, 스트림 채널은 경계 없는 데이터의 전송을 위하여 사용된다. 메시지 채널은 스트림 채널과 같이 능동적으로 생성, 소멸되는 채널이며, 32바이트 이하의 메시지 단위로 교환되는 데이터의 전송을 위해서 사용된다. <그림 2>는 본 논문에서 구현한 리눅스 기반의 DVR시스템의 구조도를 나타낸다.



<그림 2. 리눅스 기반의 DVR시스템>

스트림 채널과 메시지 채널은 TriMedia에서 제공하는 이벤트와 두 개의 순환 큐로 구현되었는데, 이벤트는 데이터가 전송되었다는 것을 알려주기 위하여 사용되고, 두 개의 순환 큐는 각각 두 개의 포인터를 가지고 있어 새로운 데이터의 위치와 사용가능한 버퍼의 위치를 알려주는데 사용된다. 즉, 스트림과 메시지 채널을 통하여 데이터가 전송되는 방법은 데이터를 전송하고자 할 때 먼저 호스트(또는 타겟)으로 데이터 전송을 알리는 이벤트를 만들어 주고, 이 이벤트에 대한 응답이 도착하면 순환 큐의 두 개의 포인터에 의해서 새로운 데이터가 위치한 곳을 알아내고, 이 데이터를 사용가능한 버퍼에 저장하는 방식으로 이루어진다.

3. 리눅스 시스템의 스케쥴링에 의한 부하측정

본 절에서는 리눅스 시스템의 스케쥴링이 실시간 응용 프로그램의 성능에 미치는 영향을 측정한다. 리눅스 스케쥴링은 기본적으로 프로세스의 우선순위(priority)에 의해 수행시간을 비 균등 적으로 배분하는 간단한 방식이다[3][4].

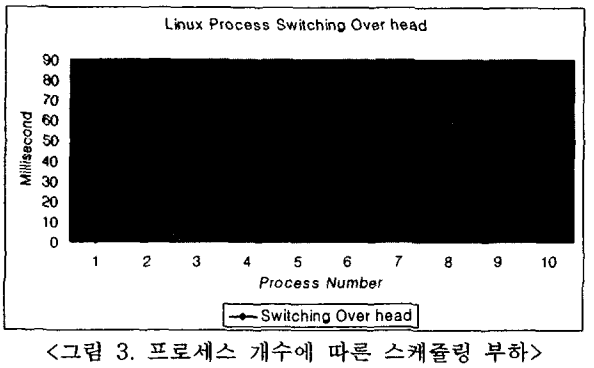
IBM PC를 기준으로 리눅스 시스템은 일반적인 프로세스의 기본 할당시간(base time quantum)값을 약 210 ms(milliseconds)로 설정하고 있으며, 이 기본 할당시간 값을 기준으로 우선순위에 따라 많거나 적게 CPU시간 값을 부여하게 된다.

또한 우선순위가 낮은 프로세스의 수행을 보장하기 위해서 높은 우선순위를 가지고 수행하는 프로세스는 한 스케쥴링 주기(epoch)가 지남에 따라 계속적으로 그 우선순위가 낮추어 지도록 하는 방법을 같이 병행하여 사용하고 있다. 스케쥴링이 발생하였을 때 수행 중이던 프로세스는 다음 스케쥴링에 의해 CPU시간을 할당 받기 전까지 실행을 멈추고 실행 큐(run-queue)에서 대기하게 된다. 스케쥴러는 실행 큐에서 대기하는 프로세스 중 가장 높은 우선순위를 가진 프로세스 중에 가장 오래전에 수행되었던 프로세스를 선택하여 실행되게 한다. 이러한 Linux 운영체제에서의 스케쥴링은 다음 두 곳에서 발생한다[3][5].

- 실행중인 프로세스에게 할당된 CPU시간이 다 소모된 경우.(Timer interrupt)
- I/O 발생시.(I/O interrupt)

본 논문에서는 실행 중이던 프로세스가 스케줄러에 의해 수행대기 큐로 들어간 후 다시 CPU를 할당 받아 수행을 시작하기 전까지 대기하는 시간(실행 큐에서의 대기 시간)을 측정하였다. 이러한 대기 시간은 특히 실시간 응용 프로그램의 성능에 많은 영향을 미치게 되므로, 실시간 응용 프로그램의 성능에 영향을 미치지 않고 각각의 응용 프로그램에서 요구하는 최대한의 성능을 낼 수 있는 프로세스 개수를 측정하였다. <그림 3>은 인텔사의 펜티엄 III 프로세서(Intel Pentium III processor)를 장착한 컴퓨터에서 *Linux(kernel version 2.4.18)* 응용 프로세스 수의 증가에 따른 스케줄링에 의한 부하를 측정한 것이다. 즉, 각 프로세스가 실행 중에 스케줄링으로 인해 수행을 멈추고 기다리게 되는 시간이 프로세스의 개수가 증가함에 따라 어떻게 변화하는지 측정하여 그래프로 나타낸 것이다. 이때 순수한 스케줄링에 의한 부하를 측정하기 위해서 모든 프로세스는 I/O가 없는 프로그램을 수행한다고 가정한다.

측정 방법은 수행 중인 프로세스가 스케줄링 되어서 CPU에서 수행대기 큐로 내려가는 시점에 시스템의 시간을 기록해 놓고, 다시 스케줄링이 되어 CPU에서 수행을 시작하게 되면 이전에 저장해 놓은 시간과 현재시간을 비교하여 수행대기 큐에서 얼마동안 대기했는지를 측정하고, 5분 단위로 프로세스의 개수를 증가시켜 프로세스의 개수별로 단위 시간당 대기시간의 평균값을 구하였다. 이 평균값을 그래프로 표현한 것이 <그림 3>이다.



<그림 3. 프로세스 개수에 따른 스케줄링 부하>

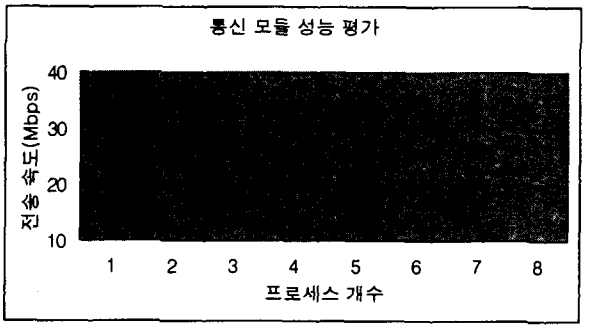
<그림 3>에서 알 수 있듯이 프로세스의 개수가 6개 까지 일 경우에는 스케줄링 부하가 높지 않게 나타났다. 이 결과를 바탕으로 본 논문에서는 다수의 프로세스를 쓰레드(thread)로 구현하여 DVR시스템의 통신 모듈을 수행하였다[6].

4. DVR시스템 통신모듈 성능 측정

본 절에서는 본 논문에서 구현한 통신 모듈을 이용하여 TriMedia보드를 사용하여 리눅스를 운영체제로 사용하는 DVR시스템을 구현하였을 때의 성능에 대하여 기술한다.

본 논문에서 구현한 통신 모듈의 성능을 측정한 방법은, 호스트와 타겟간에 제어 채널을 생성후 스트림과 메시지 채널을 각 10개씩 생성하여, 호스트에서 타겟으로 10만번 데이터를 전송한 후, 전송받은 데이터를 다시 타겟에서 호스트로 전송한다. 이 때 총 수행시간을 측정하고 총 송 수신 받은 데이터의 양을 측정하여 통신 모듈의 전송속도를 측정하였다. DVR 시스템은 1개 이상의 입력단을 가지고 있으므로 다중 프로세스 환경 하에서 성능을 측정하였다. 즉, 1개의 제어 채널을 공유하는 각각의 프로세스를 통하여 각 프로세스가 스트림과 메시지 채널을 생성하여 채널을 통하여 메시지를 송 수신하는 방법을 사용하였다.

<그림 4>는 프로세스의 증가에 따른 통신 모듈의 성능을 도표로 나타낸 그래프이다.



<그림 4. 통신 모듈 성능 평가>

5. 결론 및 향후 과제

본 논문에서는 TriMedia사의 DVR시스템용 보드를 사용하여 리눅스 운영체제 하에서 DVR시스템을 구현 하였다.

현재 대부분의 DVR시스템에서는 MPEG-4(Moving Picture Experts Group standards 4) 표준을 사용하여 동영상을 압축, 복원한다. MPEG-4 표준을 사용하면 최소 2Mbps정도의 전송속도로 동영상 데이터를 전송해 주어야 깨끗한 화질의 영상을 제공할 수 있다.

본 논문에서 구현한 DVR시스템은 평균 전송속도가 20MBps 이상으로 MPEG-4압축기법을 사용하는 시스템에서는 깨끗한 영상을 제공할 수 있다. 또한, 다중 프로세스 환경 하에서도 성능에 큰 영향을 받지 않고 수행되는 통신 모듈을 구현하였다.

본 논문에서 구현한 DVR시스템을 기반으로 향후 인터넷을 통해 분산된 DVR시스템을 원격으로 통합, 제어, 관리하는 통신 모듈 개발과, 리눅스와 DSP보드 사이의 멀티프로세스 환경을 위한 고속의 통신 모듈 개발을 향후 과제로 한다.

6. 참고 문헌

[1] http://dvr.sourceforge.net/, DVR System Project-
[2] TriMedia Manual Book
[3] "Understanding the LINUX KERNEL", Daniel P.bovert, Marco Cesati O'REILLY 2001
[4] "커널 프로그래밍", 조유근, 최종무, 홍지만, 2002
[5] "Linux Kernel Programming", M Beck, M Dziadzks, U Kunitx, ADDISON WESLEY 2002
[6] "Programming with POSIX Threads", David R. Butenhof, Addison-Wesley 2001