Q+ 실시간 운영체제에서 동작하는 미디어 재생기의 구현

조 창 식† · 마 평 수††

요 약

ADSL, ISDN 등과 같은 초고속 인터넷 접속 서비스가 발전함에 따라 일반 가정에서 인터넷을 이용하여 영화나 음악을 감상하는 것이 가능하게 되었다. 또한 정보가전의 활용 범위가 확대됨에 따라 다양한 서비스를 제공하는 정보가전의 개발이 가속화되고 있으며 정보가전을 위한 운영체제 개발 및 실시간 운영체제를 탑재한 단말장치에서의 스트리밍 서비스가 중요한 개발 목표가 되고 있다. 본 논문에서는 실시간 운영체제인 Q-에서 동작하는 미디어 재생기의 구현 기술과 경험에 대하여 설명한다. 미디어 재생기는 서버에서 전송된 MP3, MPEG-1, MPEG-4 데이터를 소프트웨어로 디코딩하여 사용자에게 보여준다. 미디어 재생기는 저가의 CPU가 장착된 디지털 TV 셋탑박스에서 동작하며, Q+ 운영체제의 커널 및 라이브러리를 이용하여 구현되었다. 따라서 하드웨어와 실시간 운영체제의 특성을 고려한 프로그래밍 기법 및 성능 향상 기법이 요구된다. 본 논문에서는 Q+ 운영체제에서 동작하는 미디어 재생기 구현과 관련하여 프로그래밍 상의 기법 및 미디어 재생기의 성능 향상 방법에 대하여 설명한다.

The Implementation of a Media Player on Q+ Real-time Operating System

Chang-Sik Cho† · Pyeong-Soo Mah††

ABSTRACT

Due to a recent advance on the internet access technologies such as ADSL and ISDN, it becomes possible to watch movies or listen to music at home through the Internet. In this paper, we propose implementation technique and experience we learned in the development of media players operating on Q+ real-time operating system. The media player can manipulate MP3, MPEG-1, and MPEG-4 streams, and the decoding routine is implemented by software. The media player is operated on the digital TV set-top-box, and is implemented by using Q+ kernel and Q+ libraries. In this paper, we focus on programming technique on Q+ real-time operating system and performance enhancement technique.

1. 서 론

ADSL, ISDN 등과 같은 초고속 인터넷 접속 서비스가 제공됨에 따라 일반 가정에서 인터넷을 이용하여 영화나 음악을 감상하는 것이 가능하게 되었다. 또한 인터넷의 발전과 함께 다양한 서비스를 제공하는 정보가전의 개발이 가속화되고 있으며, 정보가전을 위한 운영체제 개발 또한 중요한 목표가 되고 있다[1]. 정보가전용 기기는 데스크탑 컴퓨터와는 달리 시스템 자원이 한정되어 있기 때문에 기존의 데스크탑 운영체제와는 다른 요구사항을 가지고 있다. 이러한 환경을 위한 운영체제가 실시간 운영체제(RTOS, Real-Time Operating System) 또는 내장형 운영체제(Embedded Op-

† 정 회 원 한국전지통신연구원 선임연구원
†† 정 회 원 한국전지통신연구원 컴퓨터 소프트웨어기술연구소 책임연구원
논문접수 2000년 7월 11일, 심사완료 : 2000년 10월 23일

erating System)이다[2, 3].

기존의 상용화된 실시간 운영체제에는 VxWorks[4], VRTX[5], OS-9[6], WindowsCE[7] 등이 있다[8]. 상용화된 운영체제는 운영체제를 사용하기 위한 도구의 가격이 통상 만 달러에 이르며, 실제 제품으로 만들어질 때 요구하는 런타임 로열티 등으로 인하여 경제적으로 부담이 된다. 또한 원천 코드가 공개되어 있지 않아 기능 추가 요인이 발생했을 때 운영체제에 적용되는 시간이 길어지게 되어 상용화 경쟁에서 불리하다. 이러한 이유로 최근에는 Linux 운영체제를 변형하여 실시간 환경에 적용한 Embedded Linux[9, 10]에 관하여 많은 연구가 있지만, 현재까지는 관련 도구들이나 개발 환경이 미약한 현실이다.

본 연구에서는 정보가전용 실시간 운영체제인 Q-를 개발하고 있다[23]. 또한 개발된 운영체제의 활용도를 높이고 안정성을 검증하기 위해 Q+에서 작동하는 미디어 재생기, 웹 브라우저, 전자우편, 디지털 방송 지원 소프트웨어 등의 응용 프로그램도 개발하고 있다. Q+가 목표로 하는 정보가전용 기기는 디지털 TV 셋탑박스이며, Q+ 운영체제에서의 응용 개발을 지원하는 개발도구도 동시에 개발되고 있다.

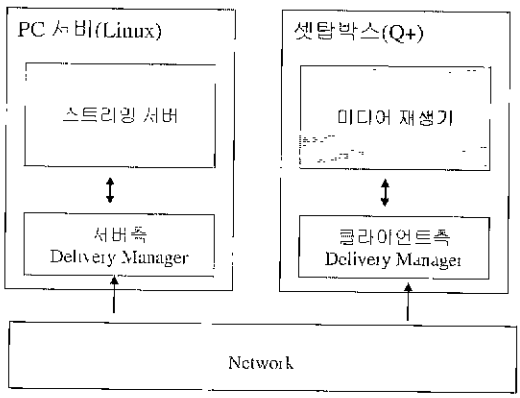
디지털 TV 셋탑박스는 일반 아날로그 TV를 이용하여 디지털 TV 방송을 수신할 수 있으며, 인터넷 서비스 및 멀티미디어 서비스를 가능하게 한다. 디지털 TV 셋탑박스가 일반 가정에서 사용되기 위해서는 기능적인 요소도 중요하지만 저렴한 가격으로 제공되는 것이 중요하다. 일반적으로 디지털 TV 셋탑박스에는 경제적인 요인으로 인해 저가의 CPU가 사용되기 때문에 하드웨어와 실시간 운영체제의 특성을 고려한 성능 향상 기법이 요구된다.

본 논문에서는 Q-에서 동작하는 미디어 재생기의 구현 기술과 경험에 대하여 설명한다. 미디어 재생기는 서버에서 전송되는 멀티미디어 스트림을 네트워크를 통하여 전송받아서 셋탑박스에서 디코딩하여 보여준다. 미디어 재생기는 MP3 Layer 1, 2, 3[12], MPEG-1[11-13], MPEG-4[14-20]를 지원하며, 디코딩을 소프트웨어로 지원한다. 단말장치의 성능 향상 및 고속의 유무선 네트워크의 보급에 따라 실시간 운영체제를 탑재한 단말장치에서의 스트리밍 서비스는 중요한 개발 요소가 되고 있다. 현재는 DirectX를 탑재한 WindowsCE[7] 및 OS-9의 DAVID[1]가 스트리밍 서비스에 관한 API를 제공하고 있다. WindowCE는 기존의 Windows에서 작동하는 DirectX를 WindowsCE에서 사용하게 함으로써 스트리밍 서비스에 필요한 라이브러리를 지원하고 있지만, WindowsCE가 고성능의 CPU를 요구하고 운영체제의 크기가 크다는 단점이 있다. DAVID는 디지털 TV 셋탑박스를 위한 통합 API의 일부로 스트리밍 서비스에 관계된 API를 제공하지만, 디지털 TV 신호인 MPEG-2 데이터를 처리하는 것을 주로 하고 있어 네트워크를 이용한 실시간 VOD 스트리밍 환경과는 다르다.

2절에서는 미디어 재생기의 구조에 대하여 설명하고, 3절에서는 2절에서 설명한 구조에 대한 Q+에서의 구현 사항을 기술하였다. 구현에 관계된 Q+ 커널 및 라이브러리의 사용 방법 및 전략을 포함한다. 4절에서는 성능향상 방법 및 실행 결과를 보여준다.

2. 미디어 재생기의 작동 환경

미디어 재생기는 클라이언트 셋탑박스에서 작동한다. 미디어 재생기는 스트리밍 서버[21,22]에서 전송되는 미디어 데이터를 디코딩하여 화면이나 스피커로 재생하고, VCR 연산과 같은 사용자 상호작용을 수행한다. 스트리밍 서버와 미디어 재생기 간의 데이터와 제어의 처리는 Delivery Manager에 의해 이루어진다. 스트리밍 서버, Delivery Manager, 미디어 재생기의 작동 환경은 (그림 1)과 같다.



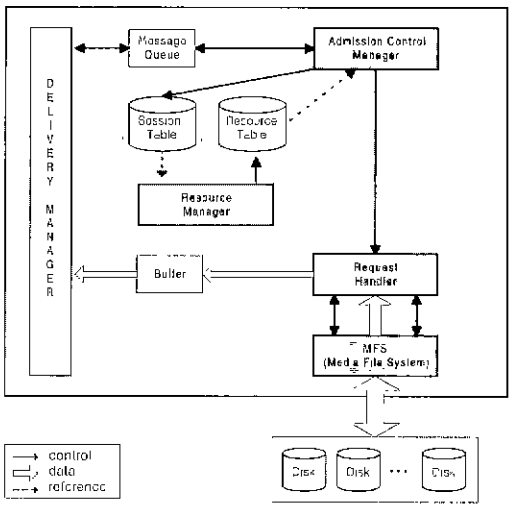
(그림 1) 미디어 재생기 작동 환경

2.1 스트리밍 서버

스트리밍 서버는 Linux 운영체제를 탑재한 PC 서버에서 구현되었다[24]. (그림 2)은 스토리지 서버의 구

조를 나타낸다. 스트리밍 서버는 Admission Control Manager, Request Handler, Resource Manager, MFS로 구성된다

Admission Control Manager는 클라이언트의 요구에 대한 허용제어를 수행하며, 세션을 설정, 관리한다. 허용 제어가 성공되면 세션이 하나 추가되고, 서비스를 요청한 파일에 대한 열기 및 네트워크 전송을 위한 이중 버퍼가 할당된다. Request Handler는 세션에서 요구하는 데이터를 버퍼링한다. Request Handler는 스트라이핑 되어 있는 디스크의 정보를 이용하여 MFS에게 데이터 읽기를 요구하고, MFS에서 받은 데이터를 이중 버퍼에 쓰게 된다. Resource Manager는 세션 테이블을 주기적으로 검사하여 자원 테이블을 갱신하여 세션 정보와 자원 정보간의 일치성을 보장한다. MFS는 각 Request Handler가 요구한 입출력 명령을 실질적으로 수행하는 부분이 된다. MFS에서는 Linux의 Ext2 파일시스템을 수정하여 한번에 읽어들이는 비디오 데이터의 read unit 양을 크게 하고, 파일시스템의 전송 속도를 높이기 위해 버퍼 캐쉬를 거치지 않고 바로 디스크에서 메모리로 데이터를 전송하는 Direct I/O를 구현하고 있다.

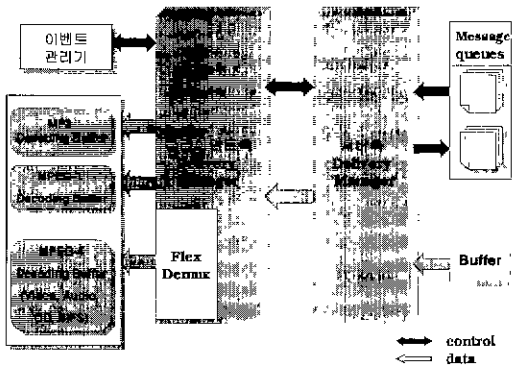


(그림 2) 스트리밍 서버의 구조

2.2 Delivery Manager

Delivery Manager는 미디어 재생기와 스트리밍 서버간의 통신을 담당한다. Delivery Manager는 스트리밍 서버와 클라이언트 양쪽에 하위 계층으로 존재하며, 상위 계층에 네트워크 투명성을 제공한다. Delivery Manager는 MPEG-4의 네트워크 표준인 DMIF[17]를 기반으로 하고 있다. (그림 2)는 Delivery Manager의 구조를 나타낸 것이다.

미디어 재생기와 스트리밍 서버간의 제어 전송과 데이타 전송은 서로 다른 네트워크 채널로 분리되어 이루어진다. Delivery Manager는 미디어 재생기의 이벤트 관리기로부터 받은 제어를 제어 전송용 네트워크 채널을 이용하여 스토리지 서버의 메시지 큐에 저장한다 또한 데이타 전송용 네트워크 채널을 통해 스토리지 서버의 버퍼로부터 미디어 재생기의 디코딩 버퍼에 데이타를 전달하게 된다 제어 전송은 신뢰성을 보장해야 하므로 TCP를 통해 이루어지고, 데이터 전송은 신뢰성을 보장하지는 않지만 전송 속도가 빠른 UDP를 통해 이루어진다.



(그림 3) Delivery Manager의 구조

본 연구에서는 스트리밍 서버가 MP3, MPEG-1, MPEG-4 스트림을 모두 처리하기 위해 DMIF에 기반하여 인터페이스를 작성하였는데, MP3, MPEG-1에서는 Channel과 TransMux 개념이 없다. 따라서 MP3, MPEG-1는 각 스트림에 대해 하나의 논리적 Channel과 하나의 TransMux를 가지고 있는 것으로 구현하였다. MPEG-4 스트림에 대해서는 포함하는 각 미디어 스트림에 대하여 Channel이 할당되고 전송을 위해 하나의 TrandMux가 사용된다. 클라이언트측 Delivery Manager의 FlexDemux는 다중화된 Flexmux MPEG-4 스트림을 각각의 미디어 스트림으로 분리하여 디코딩 버퍼에 쓴다.

스트리밍 서버에 저장된 MPEG-4 파일을 재생하는 과정은 다음과 같다 미디어 재생기는 Delivery Man-

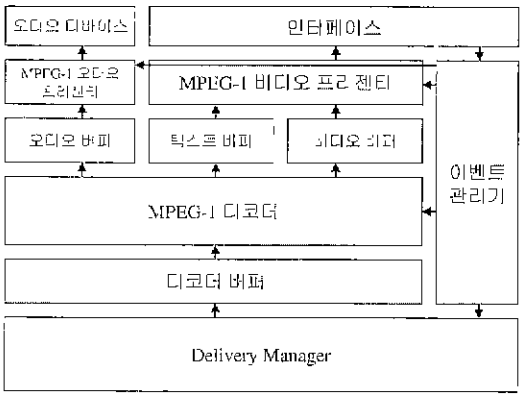
ager를 통해 서비스를 요청하게 되면 스트리밍 서버는 세션을 설정하고 파일에 대한 열기를 수행한다. 세션 설정과 동시에 Delivery Manager는 물리적인 경로인 TransMux를 설정하게 되며, 스트리밍 서버는 Delivery Manager의 FlexDemux에게 파일에 대한 정보와 클라이언트측 역다중화에 필요한 정보를 전송하게 된다. 미디어 재생기가 해당 스트림에 대해 채널 추가를 요청하면 스트리밍 서버는 해당 데이터를 Delivery Manager를 통해 미디어 재생기에 전달하게 된다 미디어 재생기는 재생중에 일시정지, 점프, 정지 등의 VCR 연산을 요청할 수 있고, 스트리밍 서버는 그에 해당되는 파일 연산을 수행한다. 클라이언트가 파일 재생의 종료를 위해 서비스 종료를 요청하면 미디어 재생기와 스트리밍 서버간의 세션이 종료된다.

이러한 기능을 제공하기 위해 Delivery Manager는 세션을 설정하는 ServiceAttach, 생성된 세션을 해제하는 ServiceDetach, 개별 미디어에 대한 논리적인 채널을 설정하는 ChannelAdd, 채널을 해제하는 Channel Delete, 전송시 미디어에 대한 물리적 경로를 지정하는 TransMuxSetup, 물리적 경로를 해제하는 TranseMux Release, VCR 연산을 제어하는 UserCommand의 메시지를 처리한다

2.3 미디어 재생기

미디어 재생기는 MP3 재생기, MPEG-1 재생기, MPEG-4 재생기로 구성되어 있다 MPEG-1 재생기의 구조는 (그림 4)와 같다. MPEG-1 재생기는 디코더 버퍼, MPEG-1 디코더, 비디오/오디오/텍스트 버퍼, 이벤트 관리기, MPEG-1 비디오 프리젠터, MPEG-1오디오 프리젠터, 인터페이스로 구성된다.

MPEG-1 재생기는 Delivery Manager를 통해 MPEG-1 미디어 스트림을 전송 받아 디코더 버퍼에 저장한다. MPEG-1 디코더는 시스템 디코더, 비디오 디코더, 오디오 디코더로 나누어진다. 시스템 디코더는 디코더 버퍼의 내용을 역다중화하여 비디오, 오디오 내부 버퍼에 저장한다. 비디오 디코더와 오디오 디코더는 디코딩을 수행하고 결과를 비디오/오디오 버퍼에 적는다. MPEG-1비디오 프리젠터와 오디오 프리젠터는 비디오/텍스트/오디오 버퍼의 내용을 확인하여 프리젠테이션 시간이 되면 인터페이스와 오디오 디바이스를 통해 내보낸다. 인터페이스를 통해 VCR 연산에 관한 이론을 누르면 이벤트 관리기는 디코딩을 중단시키고 Delivery Manager를 통해 서버에 전송 중단 및 재전송을 요청한다.



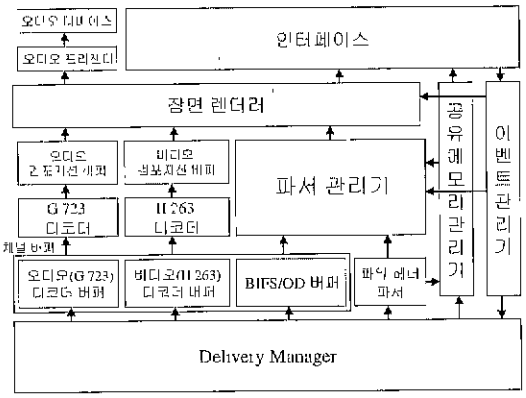
(그림 4) MPEG-1 재생기의 구조

MP3 재생기는 오디오 버퍼, MP3 디코더, MP3 오디오 프리젠터, 이벤트 관리기, 인터페이스로 구성된다 Delivery Manager를 통해 MP3 미디어 스트림을 전송받아 디코더 버퍼에 저장한다 MP3 디코더는 디코더 버퍼의 내용을 읽어 디코딩 한 다음 오디오 디바이스를 통해 내보낸다 MP3 디코더는 MPEG-1 디코더의 오디오 디코딩 부분과 동일하다. MPEG-1에서는 시스템 부분에서 역다중화 결과가 오디오 내부 버퍼로 들어가서 디코딩되고, MP3 재생기에서는 Delivery Manager에서 전송된 스트림이 디코딩 버퍼에 저장되고 곧바로 디코딩되는 점만 다르다.

MPEG-4 재생기는 채널 버퍼(오디오, 비디오, BIFS/OD 버퍼), H.263[18] 디코더, G.723[19] 디코더, 오디오/비디오 컴포지션 버퍼, 파서관리기, 장면렌더러, 오디오 프리젠터, 공유메모리 관리기, 이벤트 관리기, 파일 헤더 파서, 인터페이스로 구성된다. MPEG-4 재생기의 구조는 (그림 5)와 같다.

본 연구에서는 미디어의 효율적 관리 및 네트워크 전송의 오버헤드를 줄이기 위해 미디어 데이터를 하나의 MPEG-4 파일로 통합하여 사용하였다. MPEG-4 파일 포맷에 대한 표준이 없기 때문에 표준 초안[20]에 근거하여 파일을 설계하였으며, 유연하고 확장 가능한 형식으로 설계하여 향후 MPEG-4 파일에 쉽게 적용되게 하였다 파일에 들어가는 데이터로는 IOD 정보, OD 정보, BITS로 표현되는 장면 기술(Scene Description) 정보, 및 비디오, 오디오, 이미지 등의 미디어 데

이터가 있다[14, 20]. OD 정보는 실제 파일이 관리하는 미디어 데이터에 대한 정보를 포함하며, BIFS은 각 미디어가 구성되는 장면 규칙을 기술한다. 본 연구의 MPEG-4 재생기가 지원하는 미디어 타입은 2D, GIF 이미지, 텍스트, H.263 비디오, G.723.1 오디오이다



(그림 5) MPEG-4 재생기의 구조

MPEG-4 재생기의 작동 시나리오는 다음과 같다. 먼저 세션이 설정되면 스트리밍 서버는 파일의 헤더에 존재하는 IOD 정보를 보낸다. IOD 정보는 파일 헤더 파서에 의해 해석되며, Delivery Manager의 FlexDemux에서 역다중화 정보로서 사용된다. Delivery Manager의 FlexDemux는 전송되는 스트림을 역다중화해서 BIFS, OD 스트림 및 각 미디어 스트림으로 분리해서 채널 버퍼에 저장한다. 파서관리기는 BIFS, OD 스트림을 해석하여 장면 그래프로 만든다. 채널 버퍼에 저장된 각각의 미디어는 해당되는 디코더에 의해 디코딩되고 결과가 컴포지션 버퍼에 쓰여진다. 장면 렌더러는 만들어진 장면그래프를 참조하여 컴포지션 버퍼에 있는 미디어로 장면을 구성한다. 구성된 장면은 비디오, 오디오, 그래픽, 이미지, 텍스트의 형태로 오디오 프리젠터와 인터페이스를 통해 재생된다

3. 구현 사항 및 프로그래밍 기법

본 절에서는 2절에서 설명한 미디어 재생기를 Q+ 운영체제에 구현하면서 생긴 문제 및 해결 방안을 기술한다. 미디어 재생기는 실시간 운영체제의 특성을 고려해서 설계 및 구현되었다. 태스크 관리 측면, 사용자 인터페이스 측면, 동기화 및 네트워크 측면에서의 쟁점 사항 분석과 Q+에서의 프로그래밍 기법을 설명한다.

3.1 태스크 관리

태스크 관리 측면에서 주로 고려한 사항은 태스크의 할당, 라운드 주기와 우선순위의 조정, 사용자 인터페이스에 관계된 태스크의 처리, 태스크간 동기화 처리 등이다. Q+에서는 FIFO 및 라운드 로빈 스케쥴링을 지원하며, 우선순위에 기반하여 완전 선점형으로 스케쥴링된다. 즉, 실행중인 태스크는 항상 READY 상태의 태스크들 중 가장 높은 우선순위를 갖는다. 태스크에 관계된 API는 태스크 생성 및 종료, 태스크 실행 제어, 태스크 스케쥴러 제어, 태스크 인식에 관한 것이 있다. 본 연구에서 사용한 태스크 관련 API는 다음과 같다

- task_create(char name, int (addr)(void*), void* arg, int flags)
- task_xcreate(name, addr, arg, int flags, int prio, void *stackbase, int stacksize)
- task_delete(tid), task_exit()
- task_suspend(taskid_t tid) task_resume(tid),
- task_delay(ticks), task_pause(tid)
- task_get_priority(tid), task_set_priority(tid),
- task_get_tslice(tid), task_set_tslice(tid),
- sched_yield(), sched_lock(), sched_unlock(),
- task_name(tid), task_getid(), task_name_to_id(const char* name)

본 응용에서는 모든 태스크가 라운드 로빈 방식으로 작동하게 된다. 응용에서 사용되는 태스크는 Delivery Manager에서 제어와 데이터를 위해 두개의 태스크가 사용되며, 디코딩은 각각의 미디어에 대하여 하나의 태스크가 할당되고 있다. MP3에서는 오디오 디코더를 위해 하나의 태스크가 존재하고, MPEG-1에서는 시스템, 비디오, 오디오를 위해 3개의 태스크가, MPEG-4에서는 OD, BIFS, 시스템, 비디오, 오디오, 이미지, 텍스트를 위해 6개의 태스크가 사용된다. MPEG-4 시스템은 실제 사용되는 미디어의 갯수가 응용에 따라 가변적이기 때문에 만약 미디어가 추가되면 추가된 미디어 갯수만큼 태스크가 증가하게 된다

기술한 바와 같이 태스크들은 라운드 로빈 방식으로 스케쥴링되며, 각각의 태스크들은 중요도에 따라 우선순위가 결정된다. 예를 들어 MPEG-1 시스템에서는

오디오가 가장 높은 우선순위를 가지는데 이것은 동기화의 기본이 오디오이고, 비디오는 오디오를 처리하고 남는 시간에 맞추어 선택적으로 디코딩된다. 또한 각각의 태스크들은 작업량에 따라 라운드 주기가 달리 설정되어 있다. 비디오, 오디오 데이터의 디코딩은 실제 CPU 작업량이 많기 때문에 라운드도 다른 태스크에 비하여 길게 설정되어 있다.

사용자 인터페이스를 위한 태스크로는 윈도우와 오디오 드라이버 구동이 있다. 윈도우 시스템에서의 이벤트 처리는 우선순위가 높게 설정되어 있기 때문에, 윈도우 이벤트가 발생하면 항상 Preemption이 일어나게 되고, 윈도우 메니저에게 프로세스가 할당된다. 오디오 사용 루틴도 마찬가지 경우로 오디오 작업이 일어나면 항상 Preemption이 일어나게 된다.

미디어 재생기에서는 사용자의 VCR 연산 요청시 태스크 간 동기화를 위해 세마포를 사용한다. 이벤트 관리기는 시스템 태스크와 디코딩 태스크에 세마포 연산을 보낸다. 태스크들이 세마포를 POST하면 Delivery Manager의 스트림 전송 태스크를 중지시키게 된다 이벤트 관리기는 각 태스크들이 중지되면 Delivery Manager를 통해 서버에 VCR 연산을 전송하고 스트리밍 서버는 해당하는 작업을 수행한다 아래는 본 응용에서 사용한 세마포 API 들이다

- sema_create(int value, int policy)
- sema_delete(sid)
- sema_wait(sid)
- sema_post(sid)
- sema_timed_wait(sid, long ticks)
- sema_value(sid)

3.2 사용자 인터페이스

사용자 인터페이스는 그래픽과 오디오로 나누어진다. Q+에서는 윈도우 컨트롤을 생성, 삭제, 제어하고, 윈도우에 관계된 이벤트를 처리하는 기능을 제공한다 미디어 재생기에서 사용되는 윈도우 컨트롤로는 재생, 정지, 점프 등의 사용자 상호작용을 제어하는 윈도우 컨트롤, 비디오 재생을 위한 재생 화면, 텍스트 재생을 위한 컨트롤, 그리고 로고를 위한 윈도 컨트롤이 존재한다. Q+에서 제공하는 그래픽은 8비트(256 색상) 기반으로 작성되었다. 따라서 본 시스템에서 사용하는 시스템은 디더링을 필요로 한다 이를 위해 시스템 팔레트를 새로 조정하고, 프레임에 대해 디더링 알고리즘을 적용하게 된다.

오디오 드라이버는 8/16비트, 모노/스테레오 데이터를 재생할 수 있다 사용되는 주파수는 미디어의 종류에 따라 다르다. 실제 오디오 재생 루틴은 실시간성을 보장하기 위해 8KB의 데이터 연속적으로 오디오 드라이버에 보내야 한다 본 응용에서는 8KB로 구성된 이중버퍼를 사용함으로써 디코딩과 오디오 드라이버 사용시의 버퍼 충돌을 해결하였다. 또한 디코딩 루틴 중 오디오의 프레임 버퍼로 데이터를 전송하는 기간 중에 Sched_lock 상태를 만들어 인터럽트 서비스 루틴 중에는 태스크 스케쥴링이 일어나지 않도록 만들었다.

3.3 네트워크 인터페이스

미디어 재생기에서는 Delivery Manager에서 서버와 클라이언트의 데이터 및 제어 전송을 위해 소켓을 사용하게 된다. 스트리밍 데이타는 4KB씩 전송되며, Delivery Manager가 미디어 데이터를 전송하는 시간은 전체 시스템에서 차지하는 비중은 크지 않다.

Delivery Manager에서 제어를 위한 명령이는 소켓 데이터를 바로 읽고 처리하기 때문에 제어 전송이 없을 경우 자동으로 커널과 연계되어 태스크가 SUSPEND된다. 그러나 데이터 전송에서는 전송된 데이터를 읽은 다음 디코딩 버퍼에(MPEG-4에서는 채널 버퍼에) 쓰는 과정에서 디코딩 버퍼가 락킹되어 있으면 데이터 전송 태스크는 버퍼가 사용가능할 때까지 기다리게 (busy waiting) 된다 이러한 경우는 디코더에서 디코딩 버퍼를 접근하고 있을 때 Delivery Manager가 디코딩 버퍼에 데이터를 쓰려고 하는 경우이거나, 디코딩 버퍼에 오버플로우가 발생하여 더 이상 데이터를 버퍼에 쓰지 못하는 경우이다. 이 경우는 디코딩 버퍼의 락킹 기능에 의해 버퍼에 대한 접근이 통제되지만, Delivery Manager의 데이터 전송을 담당하는 태스크가 계속 READY 상태가 일어나게 되고 불필요한 CPU 사용 지연이 발생하게 된다. 따라서 Delivery Manager에서는 버퍼가 완전히 차게 되면 강제적으로 스케쥴링을 조정함으로써 불필요한 CPU 오버헤드를 줄일 수 있다 즉 라운드 주기에 의해 태스크가 자발적으로 CPU를 떠나기 전에 sched_yield를 사용하여 태스크를 강제로 큐에서 제거함으로써 다른 태스크가 프로세스를 차지할 수 있도록 한다. 또한 이러한 경우가 나오는 것을 조정하여 라운드 주기를 적절히 조정

하여 짧은 라운드가 데이터 전송에 사용되게 하였다. 차후에는 세마포를 사용하여 버퍼의 접근을 동기화 시킴으로써 빈번한 스케쥴링 오버헤드를 줄이고자 한다.

4. 성능 향상 기법 및 실행결과

미디어 재생기가 동작하는 디지털 TV 셋탑박스는 일반 사용자에게 저렴한 가격으로 제공되어야 한다. 본 연구에서 사용하는 디지털 TV 셋탑박스는 경제성을 위하여 저가의 CPU인 strongarm SA110을 사용하며, 일반적으로 21 MIPS의 성능으로 알려지고 있다. 실제로 MP3, MPEG-1 스트리밍 데이터를 소프트웨어로 디코딩 하기에는 CPU의 성능이 많이 떨어진다. 본 과제가 경제적인 면을 고려하여 낮은 CPU에 기반하여 인터넷 서비스를 구현하는 것을 목적으로 하였기 때문에 제공하는 미디어 데이터에 대한 약간의 제약이 존재한다.

실제 구현 결과 스트림 서비스의 기능상의 문제점은 없으나, 재생 속도가 문제가 되었다. 성능 측정 결과로는 MP3 스트림의 경우 Layer I, II, III에 대하여 표준이 정하는 모든 스트림을 실시간으로 재생할 수 있었다. 즉 가장 CPU 성능이 많이 요구되는 Layer III, 16 비트 샘플, 스트레오, 48Khz 데이터에 대하여 실시간 스트리밍 서비스가 가능하였다. MPEG-1의 경우는 주로 쓰이는 Layer II, 44.1Khz, 스테레오인 오디오 프레임은 실시간으로 모두 재생 가능하고, 비디오 데이터는 352×240 화질의 비디오 데이터를 초당 5 프레임으로 재생 가능하다. MPEG-4의 경우는 지원하는 미디어의 수를 작게 줄임으로써 실시간 재생이 가능하다. MPEG-4 재생기에서는 비디오와 오디오, 이미지, 텍스트 데이터가 디중화된 파일을 실시간으로 처리할 수 있게 구현되었으며, CPU 성능 때문에 장면의 변경이나 장면에서의 사용자 상호작용 지원은 클라이언트측의 상호작용만 포함되었다.

본 구현에서 성능을 올리기 위해 사용한 방법 및 향후 고려 사항은 다음과 같다.

- 각 태스크 별 프로세싱 량을 계산하여 라운드 시간을 조정하였다. 디코딩 태스크들이 가장 많은 라운드 시간을 가지며, Delivery Manager가 가장 적은 라운드 시간을 가지게 하였다. 이러한 근거로는 소켓 연산이 커널상에서 높은 우선순위를 가지기 때문에 데스크 지연이 생기지 않으며, 데이터를 메모리에서 읽어서 버퍼에 쓰는 과정은 CPU 작업에 크게 영향을 주지 않는다. 따라서 버퍼가 일정하게 채워져 있는 상태를 만들도록 하도록 비디오, 오디오, 시스템 태스크에 라운드 주기를 계산하였다. 버퍼는 네트워크에서 시스템 디코더로 들어가는 양과 시스템 역다중화 후에 비디오 데이터, 오디오 데이터용 데이터 버퍼의 양을 동시에 고려하였다.
- 기존에 존재하던 MP3 디코딩 알고리즘을 수정하여 실수형 연산으로 디코딩하는 것을 정수형 연산으로 대체하였다. STRONGARM CPU가 실수형 연산에 대해 취약한 점을 고려하여 64비트의 실수 표현을 32 비트 혹은 64 비트의 정수 표현으로 고쳐서 값을 할당하고 특정 정수에 대해 가상적인 소수점을 프로그램 코드 상에 두고 계산하였다. 최종적으로 오디오 디코더의 출력이 정수이기 때문에 알고리즘 상에서만 실수 부분을 정수 표현으로 프로그래밍 하였다. 정수형 연산은 실수형 연산에 비해 오디오 디코딩 태스크에서는 백배 정도의 성능 향상이 있었으며, 미디어 재생기 전체적으로는 다른 태스크들이 일정 부분 CPU를 차지하기 때문에 5배 정도의 성능 향상 효과가 있었다. 실제로 실수 연산의 알고리즘으로는 MP3 오디오를 실시간으로 재생할 수 없었으나, 정수형 연산으로는 가능하였다 G.723.1 오디오 디코딩 알고리즘은 표준에서 정수형 연산으로 처리하도록 되어 있기 때문에 알고리즘에 대한 변경은 필요하지 않았다
- 태스크에서 IDLE 상태가 최소가 되도록 IDLE 상태에서는 재스케줄링이 일어나게 하였다. 이는 네트워크에서 데이터를 받는 과정에서 버퍼의 적정량을 계산하여 필요이상으로 네트워크에서 프로세싱 시간이 지연되는 것을 방지하였다
- MPEG-1 시스템에서는 SCR에 기반한 동기화 방법이 아닌, 시스템 디코딩에서 역다중화하는 루틴을 오디오의 재생 속도에 맞추고, 역다중화하는 과정에서 비디오 프레임을 버림으로써 비디오와 오디오의 동기화를 맞추었다 제시한 동기화는 SCR에 기반하여 tick 연산을 이용하여 구현한 동기화 방법과 기능상의 큰 차이가 없었기 때문에 tick 연산으로 동기화 하는 부분은 포함하지 않았다. 향후에 MPEG-1 비디오 부분을 하드웨어로 처리하는 방법도 고려하고 있다.

- 비디오 재생 화면은 재생 속도를 높이기 위하여 그래픽의 프레임 버퍼에 바로 쓰는 Direct Draw를 사용하였다.
- 본 시스템이 8비트 색상을 지원하기 때문에 비디오에서 나오는 데이터를 디더링하는 과정이 필요하다. 디더링 과정이 CPU의 성능을 많이 차지 하기 때문에 그래픽의 오버레이 기능을 이용하여 RGB 데이터 대신 4:2 0 YUV 데이터를 그래픽 드라이버로 바로 출력하고있다.

(그림 6)는 정보가전용 실시간 운영체제인 Q+를 탑재한 디지털 TV 셋탑박스에서 동작하는 MPEG-1 재생기의 실제 실행 화면이다. 미디어 재생기의 출력은 일반 아날로그 TV로 나타나며, 256 칼라의 640×480의 해상도를 가진다.

(그림 6) MPEG-1 재생기의 실행 화면

5. 결 론

본 논문에서는 실시간 운영체제인 Q+에서 작동하는 미디어 재생기의 구현 기술과 경험에 대하여 설명하였다. 본 시스템은 서버로 Linux 환경의 PC 서버를 사용하며, 클라이언트로는 Q+가 탑재한 셋탑박스 시스템을 대상으로 개발되었다

본 미디어 재생기의 주요 특징은 다음과 같다

- 본 미디어 재생기는 자체 개발한 Q+ 실시간 운영체제에서 작동하고 있으며, Q+ 운영체제의 특징을 포함하는 프로그래밍 기법이 사용되었다 운영체제가 달라지면서 크게 변경되는 테스크 관리와 동기화 기법, 인터페이스 처리, 네트워크 처리 등을 효율적으로 처리하는 기법을 적용하였다.
- 본 미디어 재생기는 경제성을 고려하여 저가의 CPU를 기반으로 한 디지털 TV 셋탑박스에서 작동하도록 개발되었다. 셋탑박스가 프로세싱 능력이 한정되어 있는 점을 고려하여 성능을 향상시키는 여러 기법을 사용하였다. 라운드 주기와 우선순위 고려, 디코딩 알고리즘의 최적화, 태스크의 불필요한 시간 낭비 방지, Direct Draw의 사용을 들 수 있다.
- VOD 서비스를 위해서 미디어 데이터를 서버에서 실시간으로 전송 받을 수 있도록 하였다. 디코딩 루틴을 포함하여 모든 미디어 재생기를 소프트웨어로 구현함으로서 하드웨어 디코딩에서 필요한 칩을 셋탑박스에서 배제시킴으로써 경량화를 가능하게 하고 가격을 낮출 수 있다. 또한 소프트웨어로 작성되었기 때문에 유지, 보수가 쉽고 업그레이드가 용이하다.
- 디코딩 연산시 연산 속도를 빠르게 하기 위해 실수형 연산을 정수형으로 대체하여 작업하였다. STRONGARM CPU가 실수형 연산에 대해 현저하게 프로세싱 능력이 떨어지기 때문에 64비트의 실수 표현을 32 비트 혹은 64 비트의 정수 표현으로 고쳐서 값을 할당하고 특정 정수에 대해 가상적인 소수점을 프로그램 코드 상에 두고 계산하였다. 정수형 연산은 실수형 연산에 비하여 5배의 성능향상 효과가 있었다.
- CPU 성능에 따라 다중화 루틴에서 탄력적으로 비디오 프레임을 디코딩한다. MPEG 파일의 특성을 고려하여 초기 일정 시간동안 재생 가능한 프레임 수를 결정한 다음, 이후의 데이터에 대해서는 오디오 데이터는 실시간으로 재생하고 비디오 데이터에 대해서는 CPU 성능에 맞게 프레임을 버리는 방식을 채택하였다.

본 응용은 자체 개발한 실시간 운영체제인 Q+ 커널과 라이브러리를 사용한다 미디어 재생기 및 기타 응용 프로그램의 정상적인 동작은 Q+ 운영체제의 커널과 라이브러리의 안정성을 검증하는 한 수단이 되며, 응용을 개발하면서 발생한 여러 문제점을 Q+ 개발에 반영하였다. 또한 자체 개발한 운영체제 개발도구를 사용함으로써 개발 기간을 단축시켰다. 응용의 컴파일, 셋탑박스로의 응용 적재, 응용 프로그램의 디버깅 과정에 개발도구를 사용함으로써 미디어 재생기를 빠르게 개발할 수 있었으며, 미디어 재생기를 개발하면서

필요성을 인식한 기능을 개발도구에 추가하도록 하여 도구의 활용도를 높였다.

앞으로는 MPEG 데이터를 더 많이 실험하여 태스크의 라운드 주기 조정과 우선순위 조정, 스케줄링 방법 개선을 거쳐 재생되는 프레임 수를 늘리고자 한다. CPU을 많이 사용하는 루틴에 대해서는 STRONGARM CPU의 특성을 고려하여 최적화된 어셈블리 코드로의 변환도 고려중이다. 또한 미디어 재생기가 돌아가는 단말장치를 다양화하여 유/무선을 장착한 PDA, Hand Held-PC 등의 단말장치 등에서도 작동하게 할 계획이다. 무선 환경은 유선에 비하여 네트워크 대역폭이 한정되어 있기 때문에 실제적 접속으로 미디어 데이터를 전송하는 것이 아니라 서버에서 브로드캐스팅을 수행하고 단말장치에서 해당되는 비디오 스트림을 받는 방식으로 개발하고자 한다.

참 고 문 헌

[1] John Washburn, DAVID 22, A Superior Foundation for Digital Television Solutions, Microware Systems Corporation, 1998.

[2] Philip A. Laplante, "Real-Time Systems Design and Analysis-An Engineer's Handbook, 2 Ed.," IEEE Computer Society Press, 1997.

[3] Jean Bacon, "Concurrent Systems : An Integrated Approach to Operating Systems, Database and Distributed Systems," Addison-Wesley Publishing Company, 1993

[4] Wind River, "VxWorks Programmer's Guide, 5.3.1 Edition 1," Wind River Systems, Inc, 1997.

[5] http : //www.mentor.com/index.html, Embedded software development solutions Mentor Graphics Embedded Software Division.

[6] http : //www.microware.com/, Microware Systems Corporation Homepage

[7] http : //www.microsoft.com/windows/embedded/, Windows Embedded home page

[8] http : //www.cs.umd.edu/~fwmuller/etc/realtime/comm.html, Real-Time Systems Commercial Operating Systems.

[9] http : //luz.cs.nmt.edu/rtlinux.new/index.html, RTLinux.org Home Page

[10] http : //www.ittc.ukans.edu/kurt/, KURT The KU Real-Time Linux

[11] ISO/IEC, Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s Part 1 : System, IS 11172-1, 1993

[12] ISO/IEC, Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s Part 2 : Audio, IS 11172-2, 1993.

[13] ISO/IEC, Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s Part 3 : Video, IS 11172-3, 1993.

[14] ISO/IEC, Information Technology-Coding of Audio-Visual Objects Part 1 : Systems, IS 14496-1, July 2000

[15] ISO/IEC, Information Technology-Coding of Audio-Visual Objects Part 2 : Visual, IS 14496-2, May 1998.

[16] ISO/IEC, Information Technology-Coding of Audio-Visual Objects Part 3 : Audio, IS 14496-3, May 1998

[17] ISO/IEC, Information Technology-Coding of Audio-Visual Objects Part 6 : Delivery Multimedia Integration Framework, IS 144496-6, May 1998.

[18] ITU-T, Recommendation H.263 : "Video coding for low bitrate communication," 3. 1996

[19] ITU-T, Recommendation G.723.1 : "Dual rate speech coder for multimedia communication transmitting at 5.3 & 6.3 kbit/s." 3. 1996.

[20] ISO/IEC, Information Technology-Coding of Moving pictures and audio-MPEG-4 Systems : Intermedia Format (MP4) VM text, N2612p4, Dec, 1998.

[21] D J Gemmel, H. M. Vin, D. D. Kandlur, P V. Rangan, and L A Rowe, "Multimedia Storage Servers : A Tutorial," IEEE Computer, Vol.28, No. 5, pp.40-49, May 1995

[22] D. D. Kandlur, M S Chen, and Z. Y. Shae, "Design of a Multimedia Storage Server," IS&T/SPIE Int. Symposium on Electronic Imaging : Science and Technology, San Jose, pp.164-178, Feb. 1994.

[23] 조립형 실시간 OS 개발 수행계획서, 한국전자통신연구원, 1998 11

[24] 마평수,조창식, "초기 대기시간 감소를 위한 디스크 재스케쥴링 방법", 한국정보처리학회 추계학술대회 제6권 제2호, 1999. 10

조 창 식

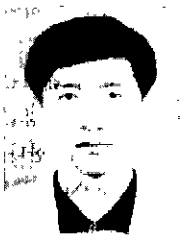
e-mail cscho@etri.re.kr

1993년 경북대학교 전자계산학과 (학사)

1995년 경북대학교 전자계산학과 (석사)

1995년~현재 한국전자통신연구원 선임연구원

관심분야 : 실시간운영체제, 멀티미디어 저장서버, 객체지향 기술, 웹 기술 등



마 평 수

e-mail pmah@etri.re.kr

1985년 서울대학교 식물병리학과 졸업(학사)

1992년 City University of New York, USA 전산학과 (석사)

1995년 Wright State University, USA 전산학과(박사)

1985년~1989년 시스템공학연구소 연구원

1989년~1990년 (주)태양금속 정보산업연구실 대리

1996년~현재 한국전자통신연구원 컴퓨터 소프트웨어 기술연구소 책임연구원

관심분야 : 멀티미디어 저장서버, 멀티미디어 검색, 웹 기술 등