

실시간 운영체제인 Q+에서 작동하는 미디어 재생기 개발

조창식, 마평수
한국전자통신연구원
e-mail : { cscho, pmah }@etri.re.kr

The Development of Q+ Media Player

ChangSik Cho, PyeongSoo Mah
Electronics and Telecommunications Research Institute

요 약

ADSL, ISDN 등과 같은 초고속 인터넷 접속 서비스가 제공됨에 따라 일반 가정에서 인터넷을 이용하여 영화나 음악을 감상하는 것이 가능하게 되었다. 또한 인터넷의 발전과 함께 정보가전의 활용 범위가 확대됨에 따라 다양한 서비스를 제공하는 정보가전의 개발이 가속화되고 있으며, 정보가전을 위한 운영체제 개발 또한 중요한 목표가 되고 있다. 본 논문에서는 실시간 운영체제인 Q+에서 작동하는 미디어 재생기에 대하여 설명한다. 단말장치가 지원하는 미디어로는 MP3, MPEG-1, MPEG-4 이며, Q+의 커널 및 라이브러리를 이용하여 구현하였다. 미디어 재생기는 서버와 Delivery Manager, 클라이언트로 구성된다. 서버는 멀티미디어 파일에 대한 스트리밍 서비스를 수행하며, Delivery Manager 는 서버와 클라이언트의 네트워크 투명성을 제공한다. 본 논문에서는 미디어 재생기를 Q+로 구현하면서 운영체제가 달라지면서 변경되는 프로그래밍 상의 기법 및 미디어 재생기의 성능 향상 방법에 대하여 설명한다.

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 정보가전의 활용 범위가 확대됨에 따라, 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 Operating System)이다[2].

기존의 상용 실시간 운영체제에는 VxWorks[3], VRTX[4], pSoS[5], WindowsCE[6] 등이 있고, 최근에는 Linux 운영체제를 변형하여 실시간 환경에 적용한 Embedded Linux[7,8]가 사용되고 있다. 상용화된 운영체제는 운영체제를 사용하기 위한 도구의 가격이 통상 만 달러에 이르며, 실제 제품으로 만들어질 때 요구하는 런타임 로열티 등으로 인하여 사용하기에 경제적으로 부담이 된다. 또한 커널에 대한 원천 코드가 공개되어 있지 않기 때문에 커널 수준의 기능 추가 요인이 발생했을 때 운영체제에 빨리 적용되지 않으므로써 Time-to-market 시간이 길어지게 되어 상용화 경쟁에서 불이익을 당하게 된다. 이러한 이유로 최근 코드가 공개된 Embedded Linux 에 관하여 많은 연구가 있지만, 현재까지는

관련 도구들이나 개발 환경이 미약한 현실이다.

전자통신연구원에서는 실시간 운영체제를 국산화하기 위한 과제를 수행하고 있는데, 셋탑박스를 사용하여 인터넷 접속 및 멀티미디어 서비스, 디지털 데이터 방송을 가능하게 하는 응용서비스 개발을 목표로 하고 있다. 셋탑박스 단말 장치에 Q+ 실시간 운영체제, 각종 라이브러리와 응용 API, 최종 응용 프로그램을 개발하고, 응용 프로그램의 개발을 지원하기 위한 Q+ 개발도구를 개발하고 있다. 최종 응용 프로그램으로는 디지털 방송 지원, 웹 브라우저, 전자우편, MP3, MPEG-1, MPEG-4 재생기 등을 포함하고 있다.

본 논문에서는 실시간 운영체제인 Q+에서 MP3(Layer 1,2,3 포함)[9], MPEG-1[9], MPEG-4[10] 재생을 지원하는 미디어 재생기 개발에 대하여 설명한다. 미디어 재생은 서버에서 전송되는 스트림을 네트워크로 받아서 사용자 인터페이스에서 보여지게 되는데, 각 미디어에 대한 디코딩은 소프트웨어적으로 이루어지게 된다. 제시하는 응용 서비스는 Q+ 운영체제의 안정성을 측정하는 방법이며, 셋탑박스의 서비스로 개발됨으로써 곧바로 상용화가 가능한 수준에서 작업되었다.

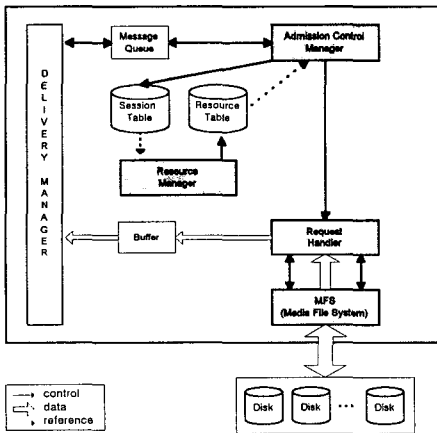
2 절에서는 응용 프로그램을 개발하기 위한 Q+ 개발 환경 및 미디어 재생기의 구조에 대하여 설명한다. 3 절에서는 2 절에서 설명한 구조를 실제로 구현하면서 생기는 문제점과 성능향상 방법 및 실행 결과를 보여준다. 구현에 관계되어 Q+ 커널 및 라이브러리의 사용 방법을 기술한다. 마지막 으로 결론이 있다.

2. 미디어 재생기의 구조 및 Q+ 개발환경

본 절에서는 미디어 재생기의 구조 및 개발환경에 대하여 설명한다. 미디어 재생기의 실행 환경은 미디어 전송을 위한 스트리밍 서버와 재생기가 작동하는 셋탑박스로 구성된다. 미디어 재생기를 위한 개발 환경으로는 응용이 작동하는 셋탑박스와 응용 개발에 필요한 컴파일, 링크, 다운로드, 디버깅을 위한 도구가 돌아가는 호스트 환경으로 구성된다.

2.1 미디어 재생기의 구조

미디어 재생기는 서버, Delivery Manager, 클라이언트로 구성되어 있다. 미디어 서버는 Linux 운영체제 상의 PC 서버에서 구현하였다. (그림 1)은 구현된 스토리지 서버의 구조를 나타낸다. 서버는 클라이언트의 허용제어를 수행하고 각 세션을 관리하는 Admission Control Manager, 서버 시스템의 자원을 관리하고 허용제어의 기준을 제시하는 Resource Manager, 클라이언트에게 데이터를 공급하는 Request Handler, 멀티미디어 파일에 대한 처리를 담당하는 MFS로 구성된다.



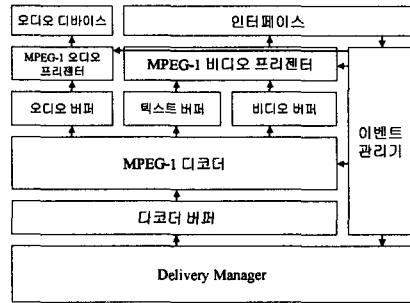
(그림 1) 스토리지 서버의 구조

Delivery Manager는 미디어 프리젠테이션과 스토리지 서버간의 통신을 담당하며, MPEG-4 표준인 DMIF[10]를 기반으로 하고 있다. 미디어 프리젠테이션과 스토리지 서버간의 제어 전송과 데이터 전송은 서로 다른 네트워크 채널로 분리되어 이루어진다. 제어 전송은 신뢰성을 보장해야 하므로 TCP를 통해 이루어지고, 데이터 전송은 신뢰성을 보장하지는 않지만 전송 속도가 빠른 UDP를 통해 이루어진다. Delivery Manager는 세션을 설정하는 ServiceAttach, 생성된 세션을 해제하는 ServiceDetach, 개별 미디어에 대한 논리적인 채널을 설정하는 ChannelAdd, 채널을 해제하는 ChannelDelete, 전송시 미디어에 대한 물리적 경로를 지정하는 TransMuxSetup, 물리적 경로를 해제하는 TranseMuxRelease의 메시지를 처리한다. 이중 TransMuxSetup, TranseMuxRelease, 및 클라이언트에서 이를 해석하는 FlexDemux는 MPEG-4로 다중화된 스트림에만 적용된다.

클라이언트 프로그램은 MP3 재생기, MPEG-1 재생기, MPEG-4 재생기로 구성되어 있다. 각각의 미디어 재생기는 동일한 인터페이스를 가지지만 독립적인 어플리케이션으로 동작한다. MP3, MPEG-1, MPEG-4 재생기는 재생, 중단, 정지 버튼을 가지며, 재생되는 스트림의 진행 상태를 보여주는

상태바를 가진다.

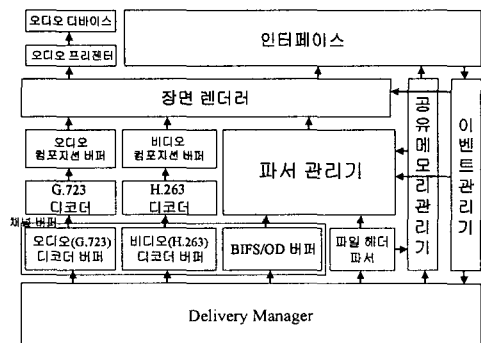
MPEG-1 재생기의 구조는 (그림 2)와 같다. MPEG-1 재생기는 디코더 버퍼, MPEG-1 디코더, 비디오/오디오/텍스트 버퍼, 이벤트 관리자, MPEG-1 비디오 프리젠테이션, MPEG-1 오디오 프리젠테이션, 오디오 버퍼, 텍스트 버퍼, 비디오 버퍼, MPEG-1 디코더, 디코더 버퍼, Delivery Manager로 구성된다.



(그림 2) MPEG-1 재생기의 구조

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-1 비디오 프리젠테이션과 오디오 프리젠테이션은 비디오/텍스트/오디오 버퍼의 내용을 확인하여 프리젠테이션 시간이 되면 인터페이스와 오디오 디바이스를 통해 내보낸다. 인터페이스를 통해 VCR 연산에 관한 버튼을 누르면 이벤트 관리기는 디코딩을 중단시키고 Delivery Manager를 통해 서버에 전송 중단 및 재전송을 요청한다. MP3 재생기는 MPEG-1과 거의 유사하며, MPEG-1의 오디오 디코더 부분은 MP3 디코더와 동일하다.

MPEG-4 재생기는 채널 버퍼(오디오, 비디오, BIFS/OD 버퍼), H.263 디코더, G.723 디코더, 오디오/비디오 컴포지션 버퍼, 파서관리기, 장면렌더러, 오디오 프리젠테이션, 공유메모리 관리기, 이벤트관리기, 파일헤더 파서, Delivery Manager로 구성된다. MPEG-4 재생기의 구조는 (그림 3)과 같다.



(그림 3) MPEG-4 재생기의 구조

본 연구에서는 네트워크 전송의 오버헤드를 줄이기 위해 미디어 데이터를 하나의 MPEG-4 파일로 통합하여 사용하였다. 파일에 들어가는 데이터로는 IOD 정보, OD 정보,

BIFS로 표현되는 장면 기술(Scene Description) 정보, 및 비디오, 오디오, 이미지 등의 미디어 데이터가 있다. 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 그래픽, GIF 이미지, 텍스트, H.263 비디오, G.723.1 오디오이다.

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 프리젠테이션 인터페이스를 통해 재생된다.

2.2. 개발 환경

본 연구에서 사용하는 셋탑박스의 CPU는 STORNGARM SA110 칩을 사용하며, 그래픽 및 오디오로 IGST CYBERPRO 5050 칩을 사용한다. 또한 사용자의 입력을 받아들일 수 있는 우선 마우스와 키보드를 제공한다. 셋탑박스는 아날로그 TV 신호를 처리할 수 있는 것과 디지털 TV 신호를 처리할 수 있는 것 두개가 나누어진다. 데이터 전송을 위해 아날로그 신호로는 NTSC에 기반한 VBI 서비스를 제공하며, 디지털 신호는 ATVEF 신호를 처리할 수 있다.

Q+ 커널은 크게 HAL, Primitive Kernel, Component Kernel, Device Driver Interface (DDI)로 구성되어 있다. HAL 부분은 하드웨어의 특정 부분과 운영체제를 연결하는 것으로서 운영체제가 특정 하드웨어에 종속되는 것을 방지해 주며, DDI는 주변 디바이스를 연결해 준다. Primitive kernel에서 커널의 기본 기능들이 구현되고 파일시스템, IPC, 네트워크, 메모리 관리 등은 상위 커널 계층인 Component Kernel에서 지원된다. 이는 개발 응용의 종류에 따라 Component Kernel의 특정 모듈을 조합함으로써 운영체제의 크기를 조절할 수 있게 해 준다. 프로그래머는 커널의 구조에 관계하지 않고 커널 API를 통하여 특정 기능을 구현할 수 있다.

사용자개발도구는 자체 개발한 ESTO를 사용한다. ESTO는 GCC 컴파일러, 원격지(호스트)에서 타겟보드에 적재된 프로그램을 디버깅하는 C 언어원격디버거, 타겟의 메모리에 목적 프로그램을 적재/제거하거나, 타겟에 적재된 커널과 라이브러리, 응용 API의 함수를 호출하고 상호작용을 하는 대화형셸, 타겟시스템의 상태와 OS 자원에 대한 정보를 원격에서 제공하는 자원모니터, 실시간 커널과 목적프로그램에 메시지를 전달하거나 감시하는 디버그 에이전트로 구성되어 있다. ESTO는 통합 개발환경을 제공하며, 응용을 개발하면서 필요한 기능이 추가 보완되었다.

3. 구현

본 절에서는 Q+에서 미디어 재생기를 개발하면서 생긴 문제점과 해결 방안 및 성능 향상을 위해 사용한 기법을 설명한다. 또한 실행 결과 화면을 보여준다. 구현 부분에서는 태스크 관리 측면, 사용자 인터페이스 측면, 동기화 및 네트워크 관련 문제점을 기술한다.

3.1. Q+ 미디어 재생기 구현

Q+에서의 스케줄링 정책은 정적 우선순위 기반의 완전 선점형 스케줄링이다. 즉 실행중인 태스크는 항상 READY 상태의 태스크들 중 가장 높은 우선순위를 갖는다. 또한 FIFO 및 Round-Robin을 지원한다. 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 디코딩 태스크, MPEG-1에서는 시스템, 비디오, 오디오 디코딩을 위한 3개의 태스크, MPEG-4에서는 OD, BIFS, 시스템, 비디오, 오디오, 이미지, 텍스트를 위해 6개의 태스크가 사용된다. Mpeg-4 시스템은 실제 사용되는 미디어의 개수가 응용에 따라 가변적이기 때문에 미디어가 추가되면 추가된 미디어 개수만큼 태스크가 증가하게 된다.

각각의 태스크들은 중요도에 따라 우선순위가 결정된다. 예를 들어 MPEG-1 시스템에서는 오디오가 가장 높은 우선순위를 가지는데 이것은 동기화의 기본이 오디오이고, 비디오 재생은 오디오 동기화에 따라서 맞추어지게 된다. 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 윈도우 시스템과 오디오 드라이버 구동이 있다. 윈도우 시스템에서의 이벤트 처리는 우선순위가 높게 설정되어 있기 때문에, 윈도우 이벤트가 발생하면 항상 Preemption이 일어나게 되고, 윈도우 매니저에게 프로세스가 할당된다. 오디오 사용 루틴도 마찬가지로 오디오 작업이 일어나면 항상 Preemption이 일어나게 된다.

Q+에서는 기본적인 윈도우 관리 기능 및 윈도우 컨트롤을 지원하고 있다. 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(256색상) 기반으로 작성되었다. 따라서 디코딩된 비디오 데이터의 재생을 위해 디더링이 필요하며 이때 시스템 팔레트를 조정하게 된다. 또한 비디오 재생 화면은 재생 속도를 높이기 위하여 프레임 버퍼에 바로 쓰는 Direct Draw를 사용하였다.

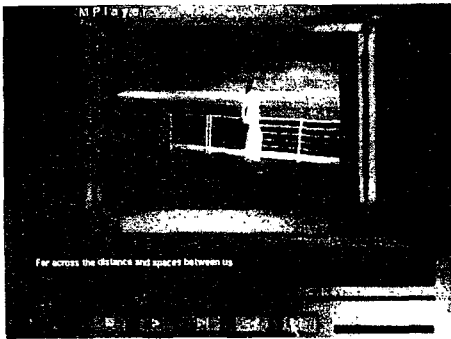
디코딩된 오디오 데이터는 오디오 드라이버로 재생된다. 실제 오디오 재생 루틴은 실시간성을 보장하기 위해 8KB의 데이터를 연속적으로 보내야 한다. 본 응용에서는 오디오 재생을 위해 8KB로 구성된 이중버퍼를 사용함으로써 디코딩과 오디오 재생의 병행성을 높였다. 또한 디코딩 루틴 중 오디오의 프레임 버퍼로 데이터를 전송하는 기간 중에 태스크 리스케줄링이 일어나지 않게 만들었다.

Q+에서는 IPC를 위해 Message Queue, Semaphore, Mutex, Condition Variable, Event을 지원한다. 미디어의 VCR 연산시 동기화 및 CS 보호를 위해 세마포어를 사용한다. 각 미디어 사이의 동기화를 위해서 SCR(System Clock Reference)에 기반하여 클럭을 맞추는 과정에 tick을 이용한 타이머를 사용하였다.

Q+는 네트워크 인터페이스로 표준 버클리 소켓을 사용한다. 본 응용에서는 Delivery Manager에서 서버와 클라이언트의 데이터 및 제어 전송을 위해 소켓을 사용하게 된다. 현재 서버와 클라이언트는 4KB씩 전송하고 있다. 실제 구현 결과 네트워크는 시스템에서 차지하는 비중은 크지 않은 것을 알 수 있었다.

3.2. 성능 평가 및 실행 화면

클라이언트 단말 장치의 CPU 는 SA110 으로, 일반적으로 2.1 MIPS 의 성능으로 알려지고 있다. 실제로 MP3, MPEG-1 스트리밍 서비스를 하기에는 프로세스의 성능이 많이 떨어진다. 본 단말장치가 낮은 CPU 에 기반하여 인터넷 서비스를 구현하는 것을 목적으로 하였기 때문에 제공하는 미디어 데이터에 대한 약간의 제약이 존재한다. 본 시스템에서는 범용의 미디어 파일을 사용하지 않고 특정 파일에 대한 재생을 실시간으로 수행하는 것에 중점을 맞추었다. (그림 4)는 실제 클라이언트에서 MPEG-1 화면 실행 화면을 나타낸다.



(그림 9) MPEG-1 실행 화면

구현결과 스트림 서비스의 기능상의 문제점은 없음을 확인하였다. 이는 재생의 속도가 문제가 될뿐 연산의 정확도는 문제가 없다는 것이 증명되었다. 성능 측정 결과로는 MP3 스트림의 경우 Layer III, 32KHz, 모노, 16 비트 샘플, 128kbps 전송률에 대하여 실시간 스트리밍 서비스가 가능하다. MPEG-1 의 경우 352*240 화질의 데이터를 초당 3 프레임 실시간으로 재생 가능하다. 셋톱박스 특징상 프로세스의 성능이 한정되어 있기 때문에 MPEG-1 의 경우 오디오 프레임을 전부 재생하는 것으로 하고, 비디오 데이터는 프로세스의 성능에 맞추어 선택적으로 프레임을 디코딩하였다.

본 구현에서 성능을 올리기 위해 각 태스크 별로 프로세싱 량을 계산하여 라운드 시간을 조정하였다. 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. 또한 기존에 존재하던 MP3 디코딩 알고리즘을 수정하여 실수형 연산으로 디코딩하는 것을 정수형으로 계산하였다. 이는 64 비트의 실수 표현을 32 비트의 정수 표현으로 고치고 연산 결과를 정수형으로 복원하는 형태로 사용하였다. 최종적으로 오디오 디코더에 사용되는 값은 정수이기 때문에 실제로는 알고리즘 상 실수의 정수 표현을 묵시적으로 프로그래밍 하였다.

4. 결 론

본 논문에서는 Q+에서 작동하는 미디어 재생기에 대하여 설명하였다. 본 시스템은 서버로 Linux 환경의 PC 서버를 사용하며, 클라이언트로는 Q+가 탑재한 내장형 시스템을 대상으로 개발되었다.

본 연구의 인터넷 셋탑박스는 저가의 인터넷 단말장치를 고려하였다. 사용자에게 판매되는 셋탑박스의 가격에 CPU 가 중요한 요소가 되기 때문에 상업성을 고려하여 가격이 낮은 CPU 를 우선적으로 고려하였다. 또한 소프트웨어로 미디어 데이터를 디코딩함으로써 하드웨어 디코딩에서 필요한 칩을 단말 장치에서 배제시킴으로써 전체 단말 장치의 가격을 줄일 수 있다. 이는 응용 프로그램의 개발시 성능에 대한 많은 고려를 필요로 하였다.

서버에서 실시간 전송을 수행한다. 이는 미디어 데이터를 자체 단말 장치에 저장하기 힘든 점을 고려하였다. 저가의 단말장치에서 오는 성능 상의 문제를 해결하기 위해 CPU 성능을 고려하여 프레임을 버림으로써 선택적으로 프레임을 디코딩하고, 디코딩 연산시 연산 속도를 빠르게 하기 위해 실수형 연산을 정수형으로 대체하여 작업하였다.

본 응용은 자체 개발한 커널, 라이브러리를 사용하였다. 본 응용 시스템의 작동으로 자체 개발한 커널 및 라이브러리의 안정성을 검증할 수 있다. 또한 응용의 개발 과정에서 발생한 여러 문제점을 Q+의 커널 및 라이브러리에 반영하였다. 응용의 컴파일, 로딩, 디버깅 과정을 도구를 사용함으로써 개발 기간을 단축시켰다. 또한 개발을 하면서 필요한 도구의 기능이 추가 보완되었고, 도구의 안정성을 높게 되었다.

앞으로는 실험 데이터를 더 많이 수집해서 정교한 라운드 주기 계산과 우선순위 조정, 스케줄링 방법 개선을 거쳐 재생되는 프레임 수를 늘리고자 한다. 또한 point-to-point 접속으로 미디어 데이터를 전송하는 것이 아니라, 서버에서 브로드캐스팅을 수행하고, 해당되는 비디오 스트림을 받는 방식으로 개발하고자 한다.

참고문헌

[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] Wind River, "VxWorks Programmer's Guide, 5.3.1 Edition 1," Wind River Systems, Inc., 1997
 [4] <http://www.mentor.com/index.html>, Embedded software development solutions Mentor Graphics Embedded Software Division
 [5] <http://www.microware.com/>, Microware Systems Corporation Homepage
 [6] <http://www.microsoft.com/windows/embedded/>, Windows Embedded home page
 [7] <http://luz.cs.nmt.edu/rtlinux.new/index.html>, RTLinux.org Home Page
 [8] <http://www.itc.ukans.edu/kurt/>, KURT The KU Real-Time Linux
 [9] ISO/IEC, Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s - Part 1,2,3, IS 11172-1,2,3, 1993
 [10] ISO/IEC, Information Technology-Coding of Audio-Visual Objects - Part 1,2,3,6, IS 14496-1,2,3,6, July 2000
 [11] ITU-T, Recommendation H.263: "Video coding for low bitrate communication, 3, 1996
 [12] ITU-T, Recommendation G.723.1: "Dual rate speech coder for multimedia communication transmitting at 5.3 & 6.3 kbit/s", 3, 1996
 [13] ISO/IEC, Information Technology-Coding of Moving pictures and audio-MPEG-4 Systems: Intermedia Format (MP4) VM text, N2612p4, Dec, 1998