논문 03-02-09

DTV 수신을 위한 소프트웨어 Demultiplexer 개발

Development of software demultiplexer for DTV Signal Reception

陳 鉉 埈*, 朴 魯 京*, 金 茂 漢**
Hyun-Joon Jin, Nho-Kyung Park, Moo-Han Kim**

요 약

본 논문에서는 지상파 디지털 방송에 적용 가능한 MPEG-2 트랜스포트 스트림(Transport Stream)용 demultiplexer를 개발하였다. Demultiplexer는 MPEG-2 형식의 멀티미디어 데이터에서 영상부분, 음성부분, 그리고 데이터 부분을 분리하여 각각의 디코더로 전송하는 기능을 담당한다. 현재 대부분의 MPEG-2 transport stream용 demultiplexer들은 하드웨어 시스템으로 구성되어졌으나, 고성능 컴퓨터의 빠른 발전으로 인하여 소프트웨어적으로도 실시간으로 동작 할 수 있게 되었다. 개발되어진 소프트웨어 demultiplexer는 윈도우즈 환경에서 동작할 수 있도록 하기 위하여 COM(Component Object Model)기반의 Microsoft의 Directshow를 이용하여 filter 형태의 소프트웨어 모듈로 개발하였다. Demultiplexer의 동작은 GraphEdit tool을 사용하여 확인하였으며, 이를 위해 MPEG-2 transport stream 형식의 테스트 파일을 랜더링 하였다.

Abstract

In this paper, a demultiplexer for MPEG-2 Transport Stream which can be applied to terrestrial digital broadcast is developed. The demultiplexer separates video, audio, and data from MPEG-2 multimedia stream and transports them to each decoders respectively. While most existing demultiplexers of MPEG-2 transport stream have been developed as hardware systems, but the fast increment of computer's performance enables a software demultiplexer to be worked in realtime. The developed demultiplexer is implemented as a software module called a filter using DirectShow of Microsoft which is based on COM(Component Object Model)and works on the Windows system. The operation of the demultiplexer is verified by using the GraphEdit tool and rendering a test file formatted as MPEG-2 transport stream.

Keywords: Digital TV, MPEG-2, Transport Stream, Demultiplexer, COM

* 湖西大學校 電氣情報通信工學部,
(Dept. of Info. & Comm., Hoseo Univ.)
(주)주홍정보통신*
(Joohong Infor. & Comm. Ltd.*)
接受日:2003年 8月 14日, 修正完了日:2003年 11月 19日

I. 서 론

40여년전 우리나라에 TV가 처음 소개된 이후 새로운 영상 매체에 대한 욕구와 전자산업의 발달에 힘입

어 TV의 보급은 빠른 속도로 확대되었다. 또한 1980년대에 컬러 TV가 우리의 앞에 등장했던 순간부터 TV의 메커니즘 자체에 대한 일대 혁신이 일어나기도 했다. 그 후 20년 후인 오늘날, 디지털 방송시대를 맞이하게 되었다.

2001년 10월부터 MPEG-2 형태의 지상파 디지털 본방송이 수도권을 중심을 시작되었으며, 2002년 3월부터는 위성 디지털 방송을 시작으로 우리나라의 방송도 디지털 방송시대로 접어들게 되었다[1].

MPEG-2의 표준화의 채택은 비디오나 오디오의 압축분야 뿐만이 아닌 전송 시스템 분야까지 포함한 것이며, 이를 구현하는 디지털 TV용 수상기의 구조에 많은 영향을 끼치게 되었다. 이처럼 디지털 TV의 방송의 시작으로 고화질의 영상에 대한 관심이 증가되고 있는 실정이다. 하지만 고화질의 디지털 TV용 수상기는 아직은 비싼 가격으로 쉽게 구입하고 있지 못하는 현실이다.

요즘 고급사양의 개인 컴퓨터와 넓은 화면, 높은 화질의 모니터의 보급으로 인하여 개인 컴퓨터를 이용한 디지털 TV의 시청에 관한 수요가 생기게 되었다. 또한 디지털 스트림을 이용하는 디지털 TV를 간소화된 하드웨어와 소프트웨어를 사용하여 디지털 TV용 수신장비의 가격을 조금 더 저렴하게 하고자 하는 움직임들이 있다[2].

본 논문의 구성은 다음과 같다. Ⅱ장에서는 DTV의 수신 시스템의 기본 구조와 demultiplexer의 역할에 대하여 기술한다. Ⅲ장에서는 MPEG-2 트랜스포트 스트림의 분석과 demultiplexer구조에 대하여 기술한다. Ⅳ장에서는 demultiplexer 개발에 사용된 Directshow의 기본 개념과 Directshow를 이용한 소프트웨어 개발 그리고 Windows에 적용 가능한 demultiplexer 소프트웨어 개발에 관한 내용을 기술한다. Ⅴ장에서는 소프트웨어 모듈의 실험과 실행 결과와 실험 고찰에 대하여 기술한다. 마지막으로 Ⅵ장에서 결론으로 본 논문의 끝을 맺는다.

Ⅱ. DTV 수신 시스템 개요

그림 2.1은 DTV의 수신 시스템의 블록도를 보여주고 있다. 하드웨어로 구성되어진 부분은 안테나와 DTV Tuner뿐이고 Demultiplexer와 비디오, 오디오, render 등은 모두 소프트웨어 모듈로 구성되어진 부분이다. 본 논문에서는 굵은 선으로 표시한 Demux 부분을 소프트웨어 모듈로 개발하였다.

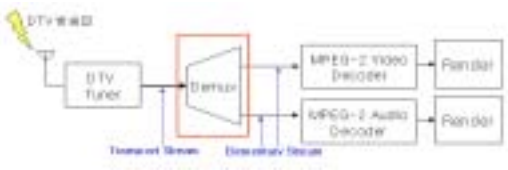

그림 2.1. DTV 수신 시스템 블록도
Fig. 2.1. Block Diagram for DTV reception

DTV Tuner를 통해 수신된 방송파 신호는 A/D 변환을 통해 디지털 형태의 데이터를 추출하여 Demultiplexer로 트랜스포트 스트림을 넘겨준다. Demultiplexer는 스트림을 받아 비디오, 오디오와 같은 elementary stream으로 변경하여 각각의 디코더로 넘겨준다. 각각의 디코더들은 입력되어진 elementary 스트림의 형태에 따라 다른 디코더를 사용하며, 각 디코더들은 입력되어진 데이터를 디코딩하고 화면 출력을 위해 랜더링을 하게 된다.

Ⅲ. MPEG-2 시스템

3.1 개요

MPEG-2는 데이터 처리 형태에 따라 크게 저장 시스템과 전송 시스템 두 가지로 나눌 수 있다. 저장 시스템은 영상을 저장 매체에 적용하기 위한 시스템으로 데이터의 형태는 프로그램 스트림이며 현재 MPEG-2의 저장 시스템을 사용하는 분야는 DVD 매체이다. 전송 시스템은 트랜스포트 스트림으로 대변되며, MPEG-2 및 기타 방법으로 압축된 영상 및 음성을 전송하기 위한 일정한 형식을 의미한다. 현재 전송 시스템은 DTV의 전송 시스템에서 사용되어지고 있다.[3,7,9]

3.2 트랜스포트 스트림

트랜스포트 스트림(이하 TS)은 MPEG-2의 전송 시스템에서 사용되어지는 비트 스트림이다. TS는 각각이 188byte인 패킷들로 구성되며, 이는 MPEG-1, MPEG-2 및 기타의 방법으로 압축된 elementary 스트림 등을 전송 에러가 존재하는 채널로 전송하기 위하여 정한 일정한 형식을 말한다.

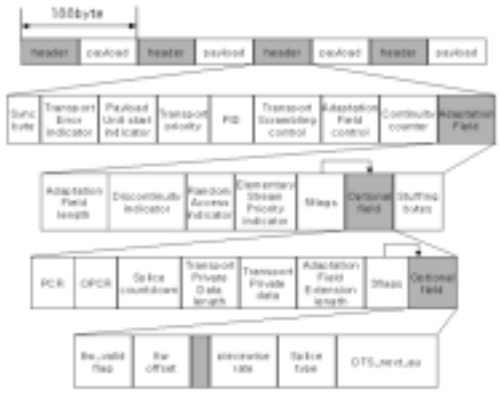

그림 3.1. 트랜스포트 스트림 데이터 패킷
Fig. 3.1. Data Packet of the Transport Stream

TS 패킷은 그림 3.1에 나타난 것과 같이 구성되며, 스트림 속에서 패킷을 찾아내기 위해 헤더에 Sync_byte 필드가 존재한다. Sync_byte는 "0100 0111"의 값을 가지고 있으며, 주기적으로 반복되는 다른 부분에서 이 값은 사용될 수 없다.

MPEG 시스템에서 언급하고 있는 프로그램이란 프로그램 element들의 집합인데, 여기서 말하는 프로그램element는 곧 elementary 스트림을 의미한다. 보편적으로 MPEG 시스템 파트에서 말하는 프로그램이란 동일한 시간기준 값을 갖는 elementary 스트림 들의 집합이라고 말할 수 있다. MPEG-2 TS demultiplexer는 프로그램 들을 구분하고 elementary 스트림을 추출하기 위하여 PSI 정보를 TS 패킷을 통하여 전달받는다.

PAT와 PMT에서는 어떤 패킷이 어떤 PID를 갖고 있는가 하는 정보를 보내주는데 PID는 TS 패킷 헤더에 있는 패킷 ID를 말한다. 이 PID는 TS 패킷의 소속을 나타내주고 있기 때문에 demultiplexing하는 과정에서는 이 PID만을 보고 패킷들을 구분하게 되는 것이다. MPEG에서는 PID가 0 인 TS 패킷은 PAT 정보를 갖고 있는 것으로 처음부터 규정을 하고 있다. PAT에는 현재 전송되고 있는 TS 스트림이 어떤 프로그램들로 구성되어 있는지를 나타내는 프로그램 번호와 이에 해당하는 PID를 갖고 있다.

PMT는 한 프로그램에 포함되어 있는 elementary 스트림들에 대한 내용 및 PID를 나타내는데 이 PMT 정보는 PAT에서 지정한 PMT_PID를 PID로 갖는 TS 패킷으로 전송된다. PAT와 PMT는 demultiplexing을 위해 일정 시간마다 TS에 삽입되어져 계속 적으로 전송되어진다. 만약 PAT 나 PMT 내의 데이터가 변경되면, 수신 시스템은 다시 초기화되어져 설정되게 된다.

3.3 Demultiplexing 동작

PSI Demultiplexing 동작은 처음 도착한 TS로부터 Sync_byte를 찾는 일부터 시작되어진다. Sync_byte를 찾은 demultiplexer는 패킷의 PID가 0인 패킷을 검색한다. PID 0인 패킷은 PAT를 가지고 있으며, PAT는 PMT를 가지고 있다. PMT는 elementary 스트림의 패킷 PID에 대한 정보를 가지고 있다. 이 과정을 수행하는 동안의 PSI를 포함하지 않은 패킷은 모두 버려지게 된다. 그림 3.2는 PSI Demultiplexing 동작을 본 논문에서 구현한 소프트웨어의 플로우를 개념적으로 나타낸 것이다.

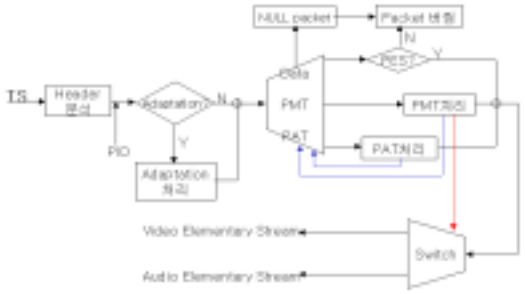

그림 3.2. Demultiplexing 플로우 개념도
Fig. 3.2. Block diagram of Demultiplexing Flow

TS가 demultiplexer로 들어오면 패킷을 구분하는 작업을 한다. 그리고 188byte씩 데이터를 입력받아 헤더를 분석한다. 헤더 내용을 분석하여 PID를 추출한 후 이를 이용하여 PAT와 PMT를 검색하게 된다. PAT와 PMT내의 정보를 이용하여 데이터 패킷을 분류하고 이를 다시 스위치를 통하여 unpacket을 한 후 각각 elementary 스트림으로 출력하여 준다. PAT와 PMT를 수신하기 이전의 모든 패킷은 널(NULL) 패킷으로 간주하여 모든 패킷을 폐기한다.

Ⅳ. Demultiplexer의 개발

4.1 DirectShow 와 필터

DirectShow는 처음에는 'Quartz' 라는 코드 명으로 시작하여 'ActiveMovie' 라는 이름으로 발표되었다. ActiveMovie 2.0부터 DirectShow 2.0이며, 이때부터

DirectX 기술에 편입되기 위한 작업이 추진되었다. DirectX 5.0부터 DirectShow란 이름이 알려졌으며, DirectX 8.0부터는 DirectShow도 DirectX의 배포 버전에 포함되어져 배포 되게 되었다. DirectShow는 필터(filter) 라는 구조의 컴포넌트를 도입하고 이들을 조합하여 다양한 멀티미디어 환경에 대응 할 수 있도록 설계되었다. 필터는 Microsoft사의 COM(Component Object Model) 기술을 기반으로 제작되며, 이에 따라 생성된 객체(object)는 독립된 COM 객체로 취급된다. 그림 4.1은 필터의 기본 구조를 나타낸 것이다.

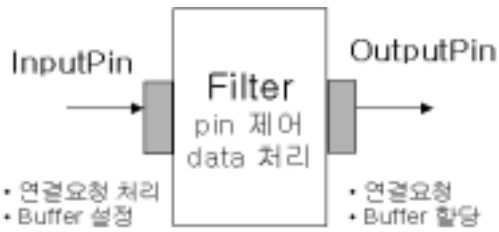

그림 4.1. 필터의 구조
Fig. 4.1. Structure of the Filter

필터는 소스 필터나 랜터 필터를 제외하고는 input pin과 output pin을 가지고 있다. 각각의 필터에는 한 개 또는 다수의 핀이 존재 할 수 있으며, 각각의 핀들은 복수의 연결은 지원하지 않는다.

4.2 Demultiplexer 필터

Input pin은 필터의 입력을 담당함과 동시에 up stream과의 연결을 담당한다. Input pin은 CBaseInputPin 클래스를 상속받으며, 모든 소스 필터는 pull 모드 동작을 한다는 점을 인식하여 internal 클래스로 CPullPin 클래스를 상속받아 pull 모드를 구현하였다.

CDemuxInputPin 클래스는 그림 4.2에 보이는 CBaseInputPin 클래스에서 상속받아 설계하였다.

그림 4.3은 CDemuxInputPin 클래스의 멤버들을 보여주고 있다. CDemuxInputPin 클래스는 CBasePin 클래스를 상속받아 설계한 클래스로 연결핀과의 media type을 검색하여주는 순수가상함수(pure virtual function)인 CheckMediaType()을 필수적으로 구현하여 media type 협상 부분을 구현하였다.

그림 4.2. Input Pin 기저 클래스
Fig. 4.2. Input Pin Base Class

CDemuxInputPin 클래스는 pull 모드에서의 동작을 위하여 CPullPin 클래스를 internal 클래스로 사용하였다.

그림 4.3. CDemuxInputPin 클래스 멤버
Fig. 4.3. CDemuxInputPin Class Member

Demultiplex 필터는 pull 모드로 동작하는 소스 필터와 연결되는 필터r이므로 pull 모드 동작을 위하여 CInPullPin 클래스를 internal 클래스로 사용한다. Input pin은 pull 모드의 소스 필터와 IAsyncReader 인터페이스를 통하여 데이터를 전달받는다. 반면 CBaseInputPin 클래스를 상속받은 CDemuxInputPin 클래스는 IMemInputPin 인터페이스를 사용하므로, NonDelegatingQueryInterface()를 이용하여 IMemInputPin 인터페이스에 대한 인터페이스반환을 하지 않도록 구현하였다. Memory allocator할당과 IAsyncReader 인터페이스를 위하여 CompleteConnect() 내에서 CInPullPin 클래스를 호출

하여 memory allocator 협상 과정을 구현하였다.

그림 4.4. CPullPin 클래스
Fig. 4.4. CPullPin Class

CInPullPin 클래스는 그림 4.4의 CPullPin 클래스를 상속받아 설계하였다. Pull 모드의 소스 필터로부터 media 샘플을 가져와 push 모드로 전송하기 위해 thread의 구현이 필요하다. 이를 위해 CAMThread를 상속받은 CPullPin 클래스를 구현함으로 input pin은 thread를 구현 할 수 있다.

그림 4.5. CInPullPin 클래스 멤버
Fig. 4.5 CInPullPin Class Member

그림 4.5는 CDemuxInputPin 클래스의 internal 클래스로 CInPullPin 클래스를 보여주고 있다. Memory allocator 협상에 관한 부분은 DecideAllocator()에 구현하였다. 또한 순수 가상 함수인 BeginFlush(), EndFlush(), EndOfStream(), OnError(), Receive()는 CDemuxInputPin 클래스를 member pointer로 선언한 m_pPin을 통하여 CDemuxInputPin 클래스의 BeginFlush()을 실행시키도록 설계하였다.

그림 4.6. CDemuxInputPin의 buffer 협상과정
Fig. 4.6. Buffer Negotiation of CDemuxInputPin

그림 4.6은 input pin과 소스 필터의 output pin과의 버퍼 협상 과정을 설명하고 있다. Media type 협상 과정에서 ReceiveConnection()함수가 호출된 후 demultiplex 필터의 input pin은 소스 필터의 output pin에 QueryInterface()를 이용하여 IAsyncReader 인터페이스를 요청한다. 그런 후 인터페이스의 RequestAllocator()를 통해 버퍼 협상을 진행한다. 이러한 모든 과정은 CInPullPin 클래스의 Connect()가 일괄적으로 처리한다. Connect() 함수는 ReceiveConnection()을 리턴하기 전인 CompleteConnect() 함수 내에서 호출되어진다.

Output pin은 필터의 출력을 담당함과 동시에 down 스트림과의 연결을 담당한다. Output pin은 CBaseOutputPin 클래스를 상속받으며, 소스 필터를 통하여 들어온 데이터를 demultiplexing하여 down stream 필터로 push 모드로 전달한다. Output pin을 구현하기 위한 클래스는 그림 4.7과 같다. CDemuxOutputPin 클래스는 CBaseOutputPin 클래스를 상속받아 설계하였다.

그림 4.7. Output pin 기저 클래스
Fig. 4.7. Output pin Base Class

CDemuxOutputPin 클래스는 그림 4.7에 보이는 CBaseOutputPin 클래스를 상속받아 설계하였다.

그림 4.8. CDemuxOutputPin 클래스 멤버
Fig. 4.8. CDemuxOutputPin Class Member

그림 4.8에 보이는 CDemuxOutputPin 클래스는 CDemuxInputPin 클래스의 pull 모드동작이 아닌 push 모드동작이므로 buffer 협상과정을 down stream의 IMemInputPin 인터페이스를 통하여 할 수 있다. Buffer협상 과정에서 순수 가상 함수인 DecideBufferSize() 함수를 구현하여 down stream과의 버퍼 협상에 관한 버퍼의 크기를 설정하였다.

CDemuxOutputPin 클래스는 다수의 output pin을 생성하여야 하므로 생성되어진 output pin counter 증가를 위해 NonDelegatingAddRef() 함수와 감소시키기 위한 NonDelegatingRelease() 함수를 추가적으로 overriding 하였다.

CheckMediaType() 함수는 down stream 필터의 input pin과의 media type을 협상하고, SetMediaType() 함수는 협상과정이 성공적으로 이루어지면 output pin의 media type을 설정하는 역할을 한다. Media type 협상과정에서 생성되어진 pin이 비디오를 위한 pin인지 오디오를 위한 pin인지를 확인하는 과정을 통하여 각각 media type을 협상하게 된다.

Demultiplex 필터는 필터의 instance를 생성하고 input pin을 통하여 전달되어진 데이터를 처리하여 output pin을 통하여 down stream으로 전달하는 역할을 한다. 그림 4.9는 CDemuxfilter 클래스의 기저 클래스 계층 도를 보여 주고 있다. 필터를 삽입하고 제어하기 위해서 IBaseFilter 인터페이스를 제공하고 있다. CBaseFilter 클래스는 GetPin()과 GetPinCount()를 순수 가상 함수로 남겨두어 사용자에 의하여 필히 구현되어야 한다.

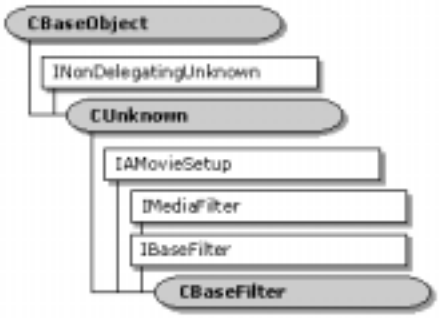

그림 4.9. 필터 기저 클래스
Fig. 4.9. Filter Base Class

그림 4.10은 CDemuxfilter 클래스 멤버들을 보여주고 있다. CDemuxfilter 클래스에서는 CreateInstace() 함수를 통하여 COM class factory를 구현하여 COM 객체를 생성하게 된다. COM 객체가 생성되어질 때 input pin과 output pin 역시 이때 생성된다.

그림 4.10. CDemuxfilter 클래스 멤버
Fig. 4.10. CDemuxfilter class member

순수 가상 함수인 GetPin()과 GetPinCount() 멤버는 CDemuxfilter 클래스를 구현하면서 필수적으로 구현하여야 한다. GetPin() 멤버는 parameter로 int n을 받아 n번째의 pin의 CBasePin 클래스의 pointer를 리턴 값으로 갖는다. GetPinCount() 멤버는 리턴 값으로 pin이 생성되어진 개수를 갖는다. CDemuxfilter 클래스는

output pin이 다수일 수 있으므로 CGenericList를 이용하여 output pin을 생성하고 삭제하는 관리를 하였다. Output pin 생성과 삭제를 위해 IniOutputPinList(), CreateOutputPin(), DeleteOutputPin() 멤버 함수를 선언하여 이곳에서 pin의 생성과 삭제 그리고 초기화를 할 수 있도록 하였다.

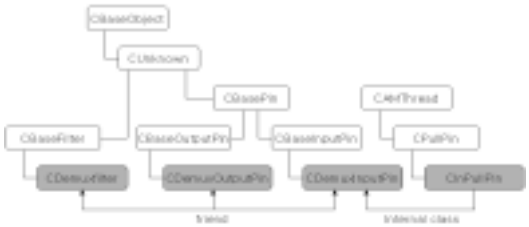

그림 4.11. Demultiplex 필터의 클래스 관계도
Fig. 4.11. Class Relationship of Demultiplexer Filter

그림 4.11에는 demultiplex 필터의 클래스 관계도가 나타나 있다. 각각의 기저 클래스와 기저 클래스의 상속 관계를 나타내고 있다. 본 논문에서 개발되어진 CDemuxfilter, CDemuxOutputPin, CDemuxInputPin 클래스들은 모두 friend관계로 서로의 클래스에서 접근이 가능하다. 또한 pull 모드 동작을 위해 input pin에 내부 클래스로 선언한 CInPullPin 클래스의 관계를 확인 할 수 있다. Demultiplex 필터역시 COM으로 제작되어졌으므로 registry에 등록되어져야 한다. 이를 위해 DirectShow는 그림 4.12과 같은 전역 함수와 변수들을 통하여 간단하게 필터를 등록하고 삭제 할 수 있도록 하고 있다.

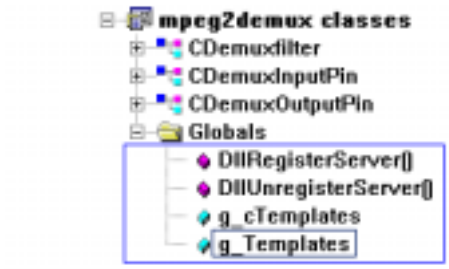

그림 4.12. COM 서버 등록과 삭제
Fig. 4.12. Registration and Delete for COM Server

DllRegisterServer()와 DllUnregisterServer()는 필터를 registry에 등록하고 해지하는 역할을 한다. Registry에 정보를 등록할 때 사용하는 필터의 이름과 CLSID 값은 CFactoryTemplate 값에서 얻기 때문에 그림 4.13와 같이 설정하여 주었다.

그림 4.13. CFactory 템플레이트
Fig. 4.13. CFactoryTemplate

Ⅴ. 실험 및 결과

5.1 MPEG-2 TS Demultiplex 필터

그림 5.1은 본 논문에서 개발한 demultiplex 필터의 구조를 보여주고 있다.

그림 5.1. MPEG-2 TS Demultiplex 필터
Fig. 5.1. MPEG-2 TS Demultiplex Filter

Demultiplex 필터의 input pin은 pull 모드로 동작하는 소스 필터에 memory allocator를 할당하기 위해 CPullPin 클래스의 IAsyncReader:: RequestAllocator()를 이용하고 있다. 필터 부분에서는 IAsyncReader 인터페이스로 전달받은 데이터를 처리하여 output pin을 통하여 디코더들로 데이터를 전송하게 된다. 그림 5.2는 소스 필터와 디코더 필터와의 사이에서 demultiplex 필터의 동작에 대하여 설명하고 있다.

그림 5.2. 필터의 동작
Fig. 5.2. Filter Operation

Demultiplex 필터는 CPullPin::Receive()를 이용하여 전달받은 데이터를 demultiplexing 하여 output pin을 통하여 디코더 필터로 전송하게 된다.

5.2 Filter graph를 이용한 실험

그림 5.3은 필터 그래프를 이용하여 개발되어진 Demultiplex 필터에서 TS 스트림 파일을 읽어 들여서 출력하는 그래프를 생성한 것이다. 비디오 디코더는 Ravisent의 디코더를 사용하였다. 그림 내에 굵은 선으로 보이는 부분이 본 논문에서 제작한 필터 부분이다.

그림 5.3. GraphEdit에 삽입되어진 필터 그래프
Fig. 5.3. Filter Graph on GraphEdit

그림 5.4는 그림 5.3의 필터 그래프를 Pentium 4 1.6G에서 실행하여 얻어진 화면이다. 실행 시 영상의 끊어짐이나 깨짐 등은 보이지 않았으며 고화질의 선명한 영상을 얻을 수 있었다.

그림 5.4. 필터 그래프 실행 결과
Fig. 5.4. Filter Graph Result

V. 결 론

본 논문에서는 MPEG-2 트랜스포트 스트림의 비디오 정보와 오디오 정보를 분리시키는 Demultiplexer를 소프트웨어 모듈로 구현하였다. 구현된 모듈은 마이크로소프트의 DirectShow 툴을 이용하여 필터 형태로 제작되었다. Demultiplexer 필터는 트랜스포트 스트림을 제공하는 소스 필터로부터 데이터를 전송받아 비디오 정보와 오디오 정보를 분리한 후 elementary 스트림으로 변환하여 디코딩을 위한 디코더 필터로 전송하게 된다. 필터들 간의 데이터 전송과 인터페이스를 위하여 input/output pin을 구현하였고 input pin은 pull 모드의 소스 필터로부터 입력을 받고 thread를 생성하기 위해 pull 모드로 그리고 output pin은 디코더 필터로 전송하기 위해 push 모드로 동작하게 된다.

구현된 필터의 동작을 확인하기 위하여 DirectX에서 제공하는 GraphEdit 툴을 사용하였다. 트랜스포트 스트림을 파일 형태로 저장한 소스 필터와 pull 모드로 연결하고 push 모드로 동작하는 디코더 간에 스트림 thread를 생성하여 demuxing하는 과정을 확인하고 결과는 랜더링 필터를 이용하여 화면으로 출력하게 하였다.

본 논문에서 구현한 Demultiplexer는 기존의 하드웨어로 구성된 기능을 소프트웨어로 대체 할 수 있는 가능성을 보여주었고 PC 성능의 지속적인 향상으로 실시간으로 동작하는 데 문제가 없으리라 사료된다. 향후 비디오와 오디오 간의 동기화 문제와 공중파 신호를 이용하여 실제적인 환경에서의 실험이 필요할 것이다.

참 고 문 헌

[1] 이호석, “알기 쉬운 MPEG-2" , 홍릉과학 출판사, 2001
[2] “영상신호처리 기술 교육”, 전자부품 연구소, 2000
[3] 유시룡, “MPEG system", 대영사, 1997
[4] “Microsoftⓒ DirectX 8.1 SDK Document", Microsoft, 2001
[6] 신화선, “DirectShow 멀티미디어 프로그래밍”, 한빛미디어, 2002
[7] "International Organization For Standardization Ogranisation Internationale De Normalisation ISO/IEC JTC1/SC92/WG11 Conding of Moving Pictures and Associated Audio", November. 1993, MPEG93
[8] 이상엽, “Visual C++ Programming Bible ver. 6.x", 영진출판사, 1998
[9] Grand Alliance, "HDTV system Specification Draft Document", February 1994
[10] 삼양출판사, “Visual C++ Object-Oriented Programming", 삼양출판사, 1998
[11] "HDTV 이론과 기술", 대한전자공학회
[12] "www.digital-tv.or.kr", Digital TV
[13] "www.mpeg.org", MPEG

저 자 소 개

陳 鉉 埈(Hyun-Jun Jin)

1984. 2 : 고려대학교 전자공학과 졸업
1986. 2 : 고려대학교 전자공학과 공학석사
1998. 1 : 미국 리하이 대학교 전산학 박사
1986. 3 - 1991. 2 : 삼성전자 시스템개발실
1998년 - 현재 : 호서대학교 전기정보통신공학부 조교수
<관심분야> 시스템프로그램, 멀티미디어 정보처리

朴 魯 京(Nho-Kyung Park)

1984. 2 : 고려대학교 전자공학과 졸업
1986. 2 : 고려대학교 전자공학과 공학석사
1990. 2 : 고려대학교 전자공학과 공학박사
1984. 3 - 1987. 2 삼성반도체통신(주) 반도체 연구소 연구원
1999. 3 - 2000. 2 미국 오레곤 주립 대학교 ECE 연구교수
1997. 3 - 1999. 2 호서대학교 공업기술연구소장
2000. 12 - 현재 IDEC 호서대 WG 책임교수
1988. 4 - 현재 : 호서대학교 정보통신공학전공 정교수
<관심 분야> 회로 및 시스템 설계(DAB, HDTV, Chip Design, Telematics)

金 茂 漢 (Moo-Han Kim)

2001. 2 : 호서대학교 정보통신공학과 졸업
2003. 2 : 호서대학교 정보통신공학과 공학석사
현재 : (주)주홍정보통신 기술연구소 연구원
<관심분야> 영상압축, 모뎀 소프트웨어