DirectShow 필터를 이용한 DSM-CC Object Carousel 인코더의 설계 및 구현

이은성*, 최성종*, 박민식**, 최진수**
* 서울시립대학교 전자전기컴퓨터공학부
** 한국전자통신연구원 방송미디어연구부

Design and Implementation of DSM-CC Object Carousel Encoder using DirectShow Filters

Eun Sung Lee*, Seong Jong Choi*, Min Sik Park**, Jin Soo Choi**
* Dept. of electrical and Computer Eng., University of Seoul
** Broadcasting Media Research Department, Electronics and Telecommunications Research Institute
E-mail : {twoss;choijs}@mmlab.net , {pms;jschoi}@etri.re.kr

요 약

본 논문에서는 Microsoft DirectShow기반의 DSM-CC Object Carousel과 Data Download 프로토콜을 위한 인코더의 설계 및 구현에 대해 기술한다. DirectShow는 Component Object Model (COM) 기술을 기반으로 하고 있다. 따라서, 작성된 코드의 재사용과 유지보수가 용이하고, 멀티미디어(Multimedia) 데이터를 편리하게 처리할 수 있는 기능을 제공한다. 또한, 개발자가 COM component를 쉽게 제작할 수 있는 방법을 제공한다. 하지만, 스트리밍 오디오/비디오 등의 디코딩을 위주로 개발된 DirectShow기술이 데이터 방송 서버에서의 실시간 인코딩 작업에 사용될 때에는 문제점이 발생한다. Data Carousel 방식에서는 인코딩 도중에 Update를 지원해야 한다. 즉, Carousel데이터의 Update가 있을 경우, 스트리밍(Streaming) 하는 동안 Update된 데이터를 인코딩 하여야 한다. 이러한 상황은 DirectShow의 기본 상태로는 표현하기 어려우므로 기본 상태를 확장하여 별도의 사용자 정의 상태를 추가 하였다. 또한, 두 작업을 동시에 수행하기에 적합한 스레드(thread) 모델과 Push/Pull 버퍼 모델을 설계하였다. DirectShow를 이용하여 인코더를 구현함으로써 개발자는 개발시간과 비용을 절약 할 수 있고, 사용자에게는 인코더를 등록하고 실행하기 쉬운 환경과 인코딩 상태의 실시간 모니터링 기능을 제공한다.

I. 서 론

디지털 TV방송에서 데이터 방송은 오디오/비디오와 더불어 데이터를 제공함으로써 시청자에게 더 많은 정보를 접하고 응용할 수 있는 환경을 제공한다. 데이터 방송을 하기 위해서는 표준화된 전송 프로토콜에 따라 데이터를 캡슐화 하는 데이터 서버가 필요하다[1].

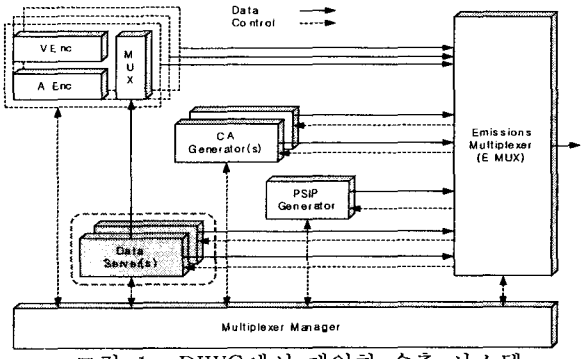

그림 1. DIWG에서 제안한 송출 시스템

그림 1은 ATSC의 Data Implementation Working Group (DIWG)에서 제안하는 송출 시스템의 구조이다[1]. 다양한 데이터방송 서비스를 제공하는 데이터 서버를 구현하기 위해서는 그림 2와 같은 표준화된 프로토콜 스택에 따라 데이터를 캡슐화 하는 기능을 가진 인코더들이 요구된다[2]. 상/하위 프로토콜의 인코더 간에는 인코딩 된 데이터를 전달하기 위한 버퍼가 존재하는 데, 데이터를 실시간으로 스트리밍 하면서 데이터의 손실을 방지하기 위해서는 버퍼의 설계와 안정적인 관리가 중요하다. 데이터 전달은 Push/Pull 방식 중 하나로 선택하거나 둘의 혼합된 형태를 사용한다. 버퍼의 사용과 관리에는 버퍼 크기 결정, 사용 후 해제, 데이터 전달 시 부가적인 설명의 첨부 등과 같은 기능이 요구된다. 그러므로 개발자는 신뢰할 수 있는 개발 툴이 제공된다면 그것을 사용하여 개발과정을 간략화 할 수 있다. 데이터 서버 운용자는 서버의 작업 진행 상황을 확인하고, 이상 유무를 확인하여 조치할 수 있는 기능을 필요로 한다. 개발자와 운용자는 모두 프로토콜간에 전달되는 데이터를 저장하고 분석하고자 하는 요구가 발생한다. 이와 같은 요구사항에 맞는 개발 툴이 Microsoft의 DirectShow이며[3], 이것을 데이터서버를 위한 인코더의 설계와 구현에 적용한다. 본 논문에서는 DSM-CC의 Object Carousel 프로토콜을 위한 인코더들을 다룬다.

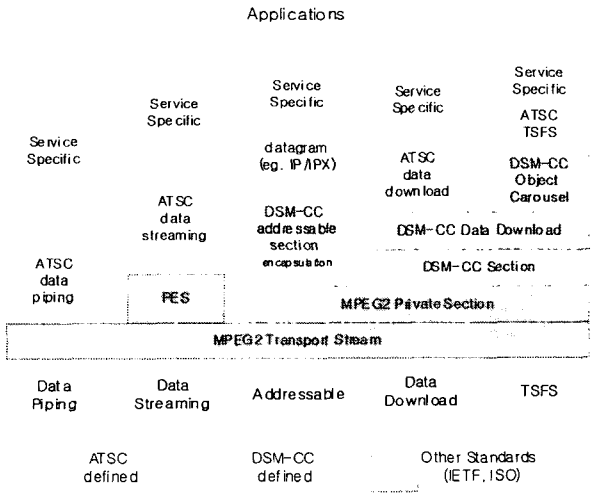

그림 2. ATSC의 표준 프로토콜 스택[5]

II. 배 경

1. DirectShow

DirectShow는 Microsoft Windows기반의 멀티미디어 데이터 처리를 위한 Framework이다[3]. DirectShow를 이용하여 응용프로그램을 개발하면 보다 쉬운 방법으로 멀티미디어 데이터의 재생, 인코딩/디코딩, 저장, 캡쳐 등이 가능하다. DirectShow는 Component Object Model (COM) 기술을 기반으로 하고 있어 작성된 코드의 재사용과 유지보수가 용이하다[3]. 또한, 개발자가 COM component를 제작하는 과정을 간략화 해주어 보다 쉬운 제작이 가능하다. DirectShow를 구성하는 것은 크게 네 가지로 필터(Filter), 핀(Pin), 미디어 타입(Media Type), 미디어 샘플(Media Sample)들이 있다[3]. DirectShow에서는 데이터를 처리하는데 필요한 기능들을 나누고 각 부분에 맞는 작업을 하는 모듈로 구성되는데, 이렇게 나누어진 기능블록을 필터라 한다. 필터는 COM object이며, DirectShow의 가장 기본적이며 중요한 구성 요소이다. 필터간의 연결과 데이터 이동은 핀이라 불리는 COM object에 의해 이루어진다. 핀은 데이터의 흐름에 따라 방향(Input/Output)을 가지며 특정한 필터에 속해있다. 필터가 연결되었을 때 앞쪽에 있는 필터를 Upstream 필터, 뒤쪽의 필터를 Downstream 필터라 한다. Upstream 필터의 Output 핀은 Downstream 필터의 Input 핀과 연결된다. 처리할 작업에 따라 필터들이 다양한 형태로 연결될 수 있는데, 이렇게 필터들이 모여 연결되어 있는 것을 필터 그래프(Filter graph)라 한다[3]. 그림 3는 AVI파일의 재생을 위한 필터 그래프이다.

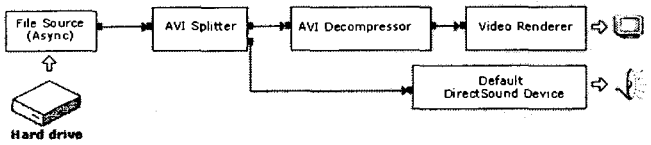

그림 3. AVI 파일을 재생하기 위한 필터 그래프

필터는 필터 그래프상에서의 위치와 역할에 따라 세 종류로 나뉜다. 첫째, Source 필터는 필터 그래프에서 처리할 데이터를 생성하는 역할을 한다. 보통 하드디스크, 네트워크, 카메라 등이 데이터의 근원이 된다. 둘째, Transform 필터는 앞에서 입력된 데이터를 처리하고 뒤쪽으로 출력하는 역할을 한다. 인코더/디코더 등이 Transform 필터의 예이다. 셋째, Renderer 필터는 필터 그래프의 맨 뒤에 있는 필터로 데이터를 사용자에게 표현한다. 비디오 데이터를 VGA card에, 오디오 데이터를 sound card로 보내는 등의 작업을 한다. DirectShow는 처리되는 디지털 미디어의 데이터 형식을 표시하기 위해 미디어 타입을 정의하여 사용한다. 연결된 두 필터 사이에 전달되는 데이터는 모두 미디어 타입을 가진다. 필터들은 미디어 타입의 협상이 성공했을 경우에만 연결된다[3]. 핀 사이의 데이터 전달을 위해 사용되는 버퍼는 미디어샘플에 의해 관리되는데, 이 역시 COM object이다. 미디어 샘플은 전달되는 데이터를 가리키는 버퍼 포인터(pointer), Time Stamp, 다양한 플래그들, 미디어 타입 등을 포함한다. 필터는 세 가지 상태를 가지는데, 각각 Stopped, Paused, Running 상태이다. Stopped 상태는 아무런 작업을 하지 않는 것을 말한다. Paused 상태는 데이터를 대기시켜 놓는 역할을 하여 Running 상태로의 전환이 일어나면 즉시 원활한 작업을 할 수 있게 해준다. Running 상태는 필터의 역할에 맞는 작업을 진행하는 것이다. 필터 그래프는 반드시 Stopped 상태와 Running 상태사이에 Paused 상태를 거친다. Paused 상태는 필터들의 종류에 따라 다른 동작을 한다. Source 필터와 transform 필터는 Running 상태와 같은 동작을 하는 반면 Renderer 필터는 데이터를 처리하지 않고 대기한다.

2. Object Carousel과 Data Download

Object Carousel과 Data Download는 데이터 방송에서 사용되는 프로토콜들로 Digital Media Storage Command and Control (DSM-CC)[4]에서 정의되었다. Object Carousel은 다수의 Object들이 모여 계층적인 형태를 이루고 있는 Service Domain을 인코딩 하여 전송하는 데 사용되는 프로토콜이다[4][5][6]. 컴퓨터에서 디렉터리와 파일들이 이루고 있는 구조가 Service Domain의 예가 될 수 있다. Data Download는 여러 개의 모듈(Module)이 모여 이루어진 그룹을 한 개 혹은 여러 개를 인코딩 하여 전송하는데, 한번 혹은 반복적으로 전송한다[4]. 여러 차례 반복적으로 전송하는 형태를 Data Carousel 방식이라 한다. Data Carousel 방식에서는 인코딩 도중에 Update가 발생한다. Update란 데이터를 Carousel 형태로 전송하고 있는 도중, 전송내용의 일부 혹은 전체가 변경되어 이것을 반영하는 것이다.

III. 문제제기

DirectShow가 제공하는 기능들을 사용하면 인코더 개발 시간과 노력을 절약 할 수 있지만, 스트리밍 오디오/비디오 등의 디코딩을 위주로 하는 DirectShow가 실시간 데이터 방송을 위한 데이터 인코딩 작업에 사용될 때에는 문제점이 발생한다. Carousel 데이터에 Update가 일어났을 경우, 현재의 Carousel 데이터를 스트리밍 하는 동안 Update된 데이터를 미리 인코딩 해 두어야 한다. 이렇게 해야 스트리밍의 중단 없이 전송 할 수 있기 때문이다. 본 논문에서는 스트리밍의 중단 없이 Update를 처리 하는 구조와 방식을 제안하고 이렇게 동작하는 것을 Dynamic Update라 정의한다. 이런 형태의 작업은 일반적인 디코딩 과정에서는 일어나지 않는다.

IV. 설계 및 구현

1. 필터 그래프 설계

설계의 첫번째 단계로 프로토콜 스택의 각 프로토콜들에 해당하는 기능들을 기능블록으로 나눈다. 그리고 각 블록들이 연결되어 인코딩 작업을 하는 형태 갖도록 한다. 각 기능 블록은 필터로 제작된다. 하나의 인코딩 작업에 필요한 필터들이 연결된 것을 필터 체인 (Filter Chain)이라 부른다. 필터 체인은 필터 그래프의 일부분이 된다. 그림 4는 본 논문에서 제안한 필터 체인이다.

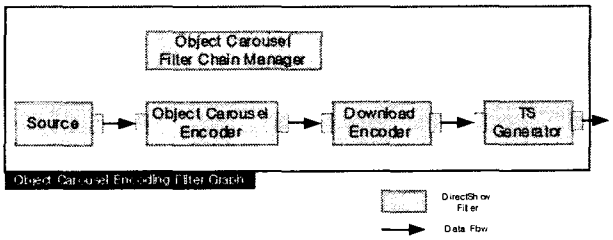

그림 4. 인코딩을 위해 제안된 필터 체인

그림 4에서 Source는 인코딩에 필요한 데이터를 출력하는데 이 데이터는 File BIOP 메시지의 유료부하(payload)가 된다. Object Carousel 인코더는 ServiceGateWay/ Directory/ File/ Stream/ Stream Event BIOP 메시지들을 생성하고 모듈로 구성하여 Download 인코더에 전달한다. 모듈은 Download 인코더에서 DDB(DSM-CC 섹션 (Section)) 메시지의 유료부하가 된다[5]. Download 인코

더는 DSI, DII, DDB, DC 메시지를 생성하는데, Carousel 방식으로 동작할 경우에는 DDB 메시지를 내부에 저장하여 Object Carousel 인코더로부터 데이터를 입력 받지 않고도 반복적으로 데이터를 전송 한다[4]. TS Generator는 섹션을 MPEG-2 TS 패킷(packet)으로 캡슐화 한다. 필터체인관리자(Filter Chain Manager)는 Source, Object Carousel 인코더, Download 인코더, TS Generator 필터들의 생성, 소멸, 메타데이터의 전달, 상태 변화 등을 담당하는 기능블록이며 역시 필터로 제작된다.

2. 데이터 전달 방식 (Data Delivery Mechanism)

DirectShow에서는 연결된 두 필터 사이에 데이터를 전달하는 방식으로 Push 방식과 Pull 방식을 사용한다[3]. Push 방식에서는 Upstream 필터가 생성한 데이터를 Downstream 필터로 전달한다. Pull 방식은 Downstream 필터의 데이터 처리 요구에 의해 Upstream 필터가 생성한 데이터를 전달한다. 필터의 데이터 전달 유형은 각각의 핀마다 다를 수 있다. 실제로 데이터 서버가 운용될 때에는 PAT/PMT/DST(ATSC)등을 생성하는 필터가 연결되어 동작하며, 각 필터 체인의 TS Generator 뒤에는 MUX가 연결된다. MUX는 여러 Elementary Stream(ES)들을 혼합해주는 역할을 한다. MUX 필터의 입력 핀은 Push 방식으로 동작하기 때문에 TS Generator의 출력 핀은 Push 방식을 지원해야 한다. 이런 동작의 특성에 따라, 그림 5와 같이 Source부터 TS Generator까지의 필터들은 모두 Push mode로 데이터를 전달하도록 설계하였다.

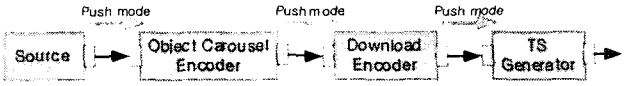

그림 5. 버퍼 동작 방식

3. Dynamic Update를 위한 상태 분석

본 절에서는 그림 4와 같은 필터 체인에서 Dynamic Update을 지원하는 Object Carousel/Download 인코더 필터의 상태와 상태 전이에 대해 분석한다. 그 첫 단계로, Object Carousel 인코딩 필터 체인의 동작 특성을 다음과 같이 그림으로 표현했다.

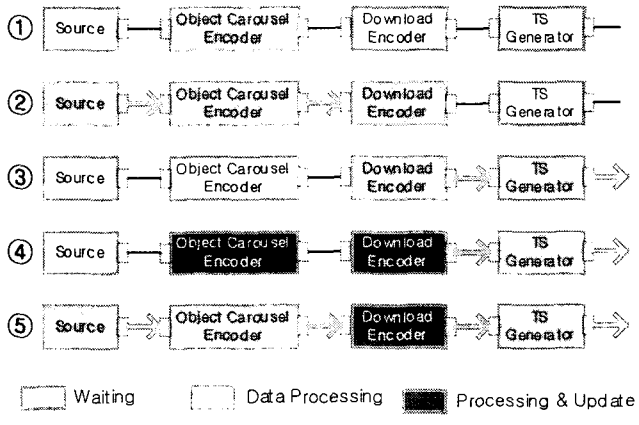

그림 6. 인코더의 동작

그림 6에서 원으로 둘러싸인 숫자는 시간의 흐름에 따른 상태를 나타낸다. ①상태는 인코딩을 진행하기 위한 첫 단계이다. 필터체인관리자가 인코딩에 필요한 메타데이터를 Object Carousel 인코더 필터에 전달하면, 이 필터는 BIOP 메시지들을 생성한다. BIOP 메시지가 생성되어 Download 인코더로 보내질 모듈의 크기가 결정되면, Download 인코더를 위한 메타데이터를 생성하여 전달한다. 메타데이터 전달이 완료되면 상태②로 전환된다. 여기서 Object Carousel 인코더는 Source에서 보낸 데이터를 BIOP 메시지의 유료부하로 캡슐화 하여 Download 인코더로 전송한다. Download 인코더는 이것을 DDB 섹션으로 만들어 파일로 저장한다. 저장이 완료되면 상태③으로 전환된다. 여기서 Source와 Object Carousel 인코더는 동작하지 않고, Download 인코더가 저장된 데이터를 이용해 반복적으로 스트리밍 한다. ④에서는 스트리밍 도중에 Dynamic Update가 발생한 경우로 Object Carousel 인코더와 Download 인코더가 인코딩 작업과 Update작업을 동시에 진행한다. Object Carousel 인코더는 인코딩 시간이 짧아 인코딩 작업과 Update 작업을 동시에 진행하는 것은 드물다. Download 인코더는 항상 인코딩과 스트리밍 작업을 하고 있으므로 Update작업이 병렬적으로 진행된다. Source와 TS Generator는 Dynamic Update의 특성과는 상관없이 대기/데이터처리와 같은 작업 형태를 가진다. Object Carousel/Download 인코더는 대기/데이터처리에 더불어 Update요청에 대한 처리를 진행하는 과정이 포함됨을 알 수 있다. 이것은 인코딩 상태에서도 데이터의 인코딩과 스트리밍 작업을 유지하며, 동시에 새로운 Update에 대한 처리를 진행하는 것을 의미한다. 이와 같은 특징에 의해 Object Carousel 인코더와 Download 인코더는 DirectShow에서 정의한 기본 상태에 새로운 상태를 추가하여 동작의 상태를 정의할 필요성이 있다. 즉, Dynamic Update를 지원하기 위해서는 필터 체인의 필터들 중 일부가 새로운 상태를 가져야 하는 것이다.

4. 스레드 모델과 버퍼 모델

필터 체인에 존재하는 스레드의 수에 따라 버퍼를 관리하는 방식이 달라진다. 다수의 스레드가 동시에 작업을 진행하는 상황에서는 스레드들 간에 작업 진행속도의 차이에 의해서 발생될 수 있는 race condition과 같은 문제 점들이 있으므로 동기화 방법(synchronization mechanism)이 필요하다. 또한 속도차이를 보상할 방법으로 큐(Queue)가 사용 되기도 한다. 이것을 그림 7에 나타냈다.

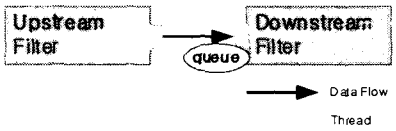

그림 7. 스레드 모델과 버퍼모델

Source는 데이터를 읽기 위한 I/O작업 때문에 처리지연 시간이 길고 불규칙 하여 별도의 스레드가 동작 하는 것이 좋다. 인코더들은 Dynamic Update를 지원하기 위해 스트리밍 스레드와는 별도의 스레드를 두어 동시 진행이 가능하게 한다. 그림 4의 필터 체인에서는 Download 인코더가 작업을 하는 시간이 가장 길기 때문에 Download 인코더는 스레드가 반드시 필요하다. 이러한 이유들로 Source, Object Carousel 인코더, Download 인코더에는 스레드가 존재 한다. TS Generator는 동작이 수동적이어서 Download 인코더에 존재하는 스트리밍 스레드에 의해 동작한다. 위에서 설명하고 제시한 스레드 모델과 버퍼 모델에 따라 구성한 필터 체인을 그림 8에 나타냈다.

그림 8. 필터체인의 스레드 모델과 버퍼모델

5. 새로운 상태

DirectShow에서 정의한 상태는 Running, Paused, Stopped이며 그림 9와 같이 표현할 수 있다. 이러한 상태 더불어 위에서 언급한 새로운 상태의 필요성에 따라 다음과 같이 새로운 상태를 정의하고 추가한다.

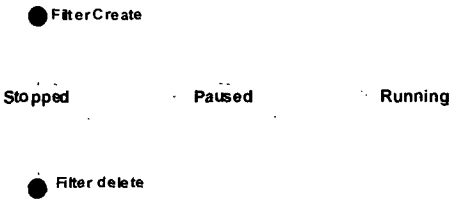

그림 9. DirectShow의 기본 상태

이 새로운 상태들은 Object Carousel 인코더와 download 인코더만이 가진다. 첫째로, NormalRunning 상태는 일반적인 DirectShow 필터의 Running 상태와 같은 동작을 의미한다. Object Carousel 인코더는 모듈을, Download 인코더는 섹션을 downstream 필터로 전달한다. 둘째로, WaitForUpdate 이다. Update 메타데이터가 수신된 경우에도 Carousel의 한 주기는 모두 진행되어야 하므로 작업을 곧바로 중단 할 수는 없다. 또한 기존 Carousel데이터를 전송하던 스레드가 데이터를 잃는 과정에서 오류를 발생시킬 수 있으므로 곧바로 새로운 정보를 반영하기 위한 인코딩을 시작 할 수도 없다. 새로운 내용을 반영할 수 있을 때까지 작업을 유지 하는데, 이와 같이 Update를 위해 대기하는 시간동안의 상태를 말한다. 필터 그래프가 구성된 초기에 첫 메타데이터를 받고 나서부터 데이터를 출력하기 시작할 때까지의 상태를 나타내기도 한다. 셋째로, UpdateProcess 상태는 새로운 내용을 반영하는 작업이 진행되는 과정이며, 여기에는 추가된 모듈들을 새로 받거나, 필요 없는 모듈을 삭제하는 과정들이 포함된다. Object Carousel 인코더와 Download 인코더는 모두 이 상태 동안 기존 Carousel 데이터의 인코딩과 스트리밍 작업을 유지한다. 추가로 정의된 세 가지 상태를 포함한 상태도를 그림 10에 나타냈다.

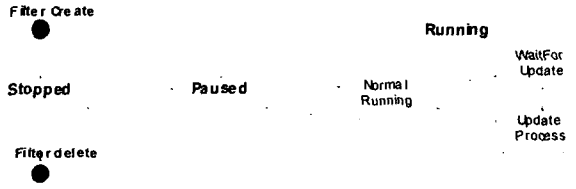

그림 10. Dynamic Update를 위한 필터의 상태도

V. 실험 및 검증

위에서 설계하고 구현한 필터들을 GraphEdit를 이용하여 동작시켜본 것이 그림 11이다. GraphEdit는 Microsoft DirectX Software Development Kit (SDK)에 포함된 유틸리티이다. 이것을 사용하여 필터의 생성, 소멸, 필터간의 연결, 필터 그래프의 동작을 시작, 중지 하는 등의 작업이 가능하다.

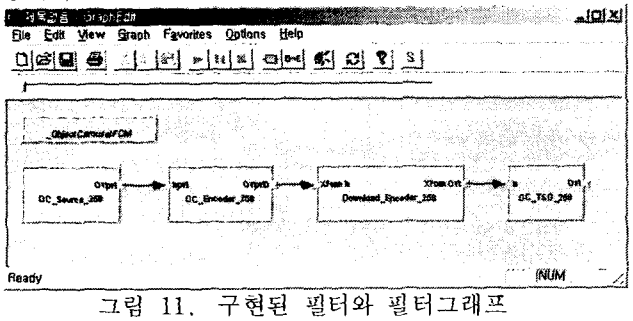

그림 11. 구현된 필터와 필터그래프

Object Carousel 인코더와 Download 인코더의 Property Page를 그림 12와 13에 나타냈다. Property Page는 필터의 상태정보를 시각적으로 표시해주어 인코딩이 진행되는 상태를 확인 할 수 있다. 이와 같은 확인 과정으로 필터 체인이 이상 없이 동작하는 것을 확인하였으며, Tektronix 사의 분석장비 MTS-300을 이용하여 실시간으로 동작하는 과정에 오류가 없음을 확인하였다.

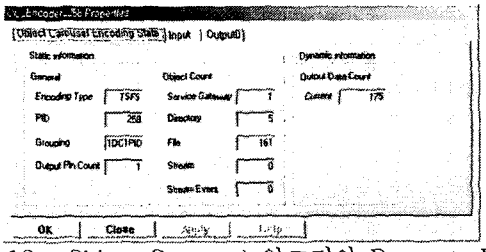

그림 12. Object Carousel 인코더의 Property Page

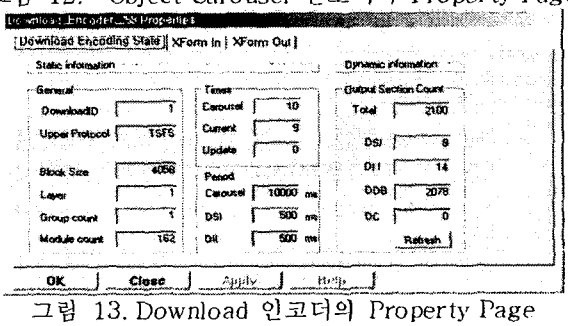

그림 13. Download 인코더의 Property Page

VI. 결 론

본 논문에서는 데이터 서버에서 사용되는 인코더를 DirectShow 필터로 구현했다. 1995년에 도입되어 현재에 이르기까지 지속적으로 개선된 DirectShow는 안정적인 동작환경을 제공한다. DirectShow가 제공하는 Base Class들을 이용함으로써 버퍼의 관리, 인코딩 상태의 표시, 스레드의 동기 맞춤 등을 쉽게 구현했다. 인코딩 상태를 세분화 하여 명확하게 구분하기 위해, DirectShow의 기본 상태를 확장하여 새로운 사용자 정의 상태를 추가 하였다. 인코딩에 알맞은 스레드 모델과 Push/Pull 버퍼 모델을 제시하고 설계하였다. DirectShow를 이용하여 구현함으로써 개발자와 운용자 모두에게 안정되고 편리한 기능들을 제공하는 것을 보여 주었다.

감사의 글

본 연구는 정보통신부의 "지능형 통합정보방송(SmarTV) 기술 개발" 과제의 지원을 받아 이루어졌으며, 논문작성에 도움을 주신 한국전자통신연구원 방송미디어연구부원들에게 감사드립니다.

참고문헌

[1] ATSC Document IS/151, "Implementation of Data Broadcasting in a DTV Station", ATSC, 1999
[2] ATSC Document A/90, "ATSC Data Broadcast Standard", ATSC, 2000
[3] Microsoft Document , "DirectX 9.0 Programmer's Reference", Microsoft, 2002
[4] ISO/IEC 13818-6, "Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for Digital Storage Media Command and Control, July 1998
[5] ATSC Document A/95, "Transport Stream File System Standard", ATSC, 2003
[6] DVB Document TS 102 812, "Digital Video Broadcasting (DVB) Multimedia Home Platform (MHP) Specification 1.1.1", DVB, 2003