분산환경을 위한 DirectShow의 확장 방법

강명헌*, 최성종*
* 서울시립대학교 전자 전기컴퓨터공학부
E-mail : frechuni@mmlab.net; choijs@mmlab.net

Expansion Method of DirectShow for Distributed Environment.

Myung-Hun Kang*, Seong-Jong Choi*
* Dept. of Electrical and Computer Engineering, University of Seoul

요 약

본 논문에서는 In-Process COM Component기반의 DirectShow를 분산환경에서 적용하기위해 해결해야 할 문제점을 분석하고, 해결방법을 제시한다. Microsoft사에서 제공하는 DirectShow는 Multimedia 데이터 처리용 라이브러리와 이 라이브러리를 사용하는 응용프로그램을 용이하게 제작하기 위한 Framework이다.[2] DirectShow의 라이브러리는 COM기술을 사용하여 제작되기 때문에 재사용 및 유지보수가 용이하다. 하지만, DirectShow는 주로 멀티미디어 데이터의 재생을 위한 기술로서 멀티미디어 서버와 같이 여러 Encoder와 다중화기와 같은 기능을 구현하는데 많은 문제점이 있다. 특히, Multi-Protocol Encoder와 같이 계산양이 많은 작업을 해야 할 경우 분산환경을 사용해야 할 필요가 있다. 본 논문에서는 네트워크로 분리되어 있는 두개의 필터를 연결하기 위해 필터간의 메시지 교환을 대리하는 Proxy 필터를 설계/구현 하였다. 이러한 Proxy 필터를 사용하면 기존의 필터를 수정하지 않고 사용할 수 있는 장점을 갖고 있다. 특히 Binary로 배포된 필터와 연동하여 사용할 수 있다. 구현된Proxy 필터를 데이터 방송 서버에 활용함으로써 그 기능을 검증하였다.

I 서 론

Microsoft사에서 제공하는 DirectShow는 Multimedia 데이터 처리용 라이브러리와 이 라이브러리를 사용하는 응용프로그램을 용이하게 제작하기 위한 Framework이다. DirectShow의 라이브러리는 COM기술을 사용하여 제작되기 때문에 재사용 및 유지보수가 용이하다. 또한, 멀티미디어 처리용 라이브러리는 여러 형태의 abstraction을 사용하여 사용자와 개발자가 편리하도록 한다. DirectShow에서 사용하는 주요 abstraction으로 filter와 pin이 있다. Filter는 멀티미디어 데이터 처리 과정을 추상화 한다.

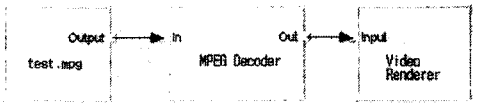

그림 1 Filter Graph

그림 1에서 MPEG Decoder Filter는 MPEG-2 표준으로 Encoding된 데이터를 RGB또는 YUV형태의 비디오 신호로 처리하기 위한 알고리즘과 데이터 구조의 집합이다. 하나의 필터가 멀티미디어 데이터를 처리하기 위해 데이터 입/출력 처리기가 필요하다. 데이터 전달을 위한 버퍼 관리, 데이터 형태 관리에 관련된 모든 기능들은 Pin으로 추상화 한다. Media Player와 같은 응용프로그램을 제작하기 위해 여러 기능의 필터를 연결(Connection)해야 하고, 이와 같이 연결된 필터의 집합을 Filter Graph라 한다. 두개의 필터는 Pin을 통해서 서로 연결할 수 있다.

1.

두 Pin의 연결은 Input pin과 Output pin사이의 메시지 교환의 절차를 거친다. 메시지 교환으로 크게 두 가지 과정을 처리한다. 하나는 Media Type 협상이고, 다른 하나는 Buffer 협상이다. 각 필터는 처리할 수 있는 Media Type을 갖고 있다. Upstream 필터는 여러 종류의 Media Type으로 출력할 수 있고, 그 중에서도 선호하는 Media Type이 있다. Downstream 필터도 여러 종류의 Media Type처리할 수 있고, 선호하는 Media Type이 존재한다. Upstream 필터가 선호하는 Media Type으로 연결할 것인지 Downstream 필터가 선호하는 Media Type으로 연결할 것인지 결정 할 수 있다. 한 쪽 필터에서 모든 Media Type을 지원하는 경우를 대비하여 Pin 연결 시에 Input pin의 Media Types으로 한번 연결 시도하고 output pin의 Media Types으로 연결 시도를 하게 된다.[3]

성공적으로 Media Type 협상이 끝나면, 필터간에 사용하게 될 Buffer(Allocator)협상을 하게 된다. 필터들간에 같은 Buffer를 사용할 수도 있고, 각각 다른 Buffer를 사용할 수도 있다. 같은 Buffer를 사용하는 경우는 이 Buffer를 사용하는 필터의 input pin과 output pin에서 사용하는 Buffer의 크기가 같은 경우이다. Buffer의 생성은 input pin이 해 줄 수도 있고, output pin이 해 줄 수 있다. 이 것은 각 필터마다 특성에 따라 다를 수 있다. 일반적으로, push mode에서는 output pin에서 생성하고, pull mode로 동작하는 경우에는 input pin이 buffer를 생성한다.

II 문제 제기

두 개의 필터간 연결은 크게 두 가지 과정으로 구분된다. 하나는 Media Type협상 과정이고 다른 하나는 Buffer 협상하는 과정이다. 일반적으로 DirectShow는 하나의 프로세스에서 동작한다. 아래 그림은 하나의 프로세스에서 Filter Graph를 구성한 예이다.

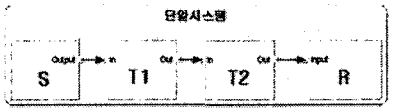

그림 2 단일 프로세스에서 Filter Graph

만약 T1과 T2가 CPU time을 많이 소모하는 작업을 한다면, 아래 그림처럼 두개의 System에서 Filter Graph를 구성하고 Network으로 Streaming Data를 전송하여야 한다. T1, T2 필터들은 Binary 형태로 제공되기 때문에 Network 담당하는 기능을 추가할 수 없다. 따라서, 아래

그림과 같이 Network 전송기능을 대리할 필터가 필요하게 된다.

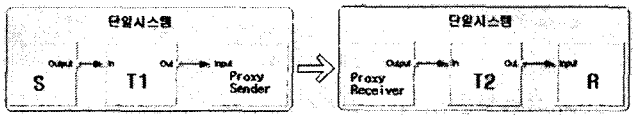

그림 3 Network으로 분리된 상태의 Filter Graph

앞으로, 전송하는 측의 Network 담당 필터를 Proxy Sender, 수신하는 측을 Proxy Receiver 필터라 부르고 두개를 통칭하여 Network Proxy 필터라 하겠다. 이러한 Network Proxy 필터는 T1, T2 필터가 단일 시스템에서 동작하듯이 연결 투명성과 Data Flow 투명성을 제공해야 한다.

이러한 투명성을 제공하기 위해 Pin에 정의된 Interface를 수정하여 Network 투명성을 제공하는 방법이 있을 수 있다. 그러나 T1, T2는 기존의 Binary형태로 제공되는 필터일 경우 이 방법을 사용할 수 없다.

DCOM으로 구현하는 경우에도 상황은 비슷하다. T1, T2가 기존에 구현된 Binary 필터이기 때문에 Network 통신을 담당하는 Proxy 필터가 필요하다. DCOM을 사용함으로써 Network관련부분은 직접적으로 신경을 안 써도 되는 장점이 있다. 그러나 Multimedia Software의 특성상 대용량의 Data를 고속으로 처리해야 한다. DCOM은 분산환경에서 Data를 주고 받을 때, Marshaling/Unmarshaling을 사용하여 처리한다[5]. 이는 Multimedia Streaming 데이터를 처리할 때, 성능을 저하시키는 Overhead가 될 수 있다.

T1, T2가 하나의 System에서 연결하고 Streaming 동작했던 기능을 Network Proxy 필터에게 위임하여, Network Proxy 필터는 T1, T2가 하였던 Connection, Streaming 동작을 대리 수행하게 한다.

III 설계 및 구현

그림 4는 Proxy Sender, Receiver 필터를 추상한 그림이다. 각 필터 내부에 있는 CSender, CReceiver class가 소켓[1] 통신을 담당하는 부분이다. 그림 4에서 점선의 화살표는 Control 메시지를 의미하며 실선의 화살표는 Streaming Data의 전달 과정을 표시한다.

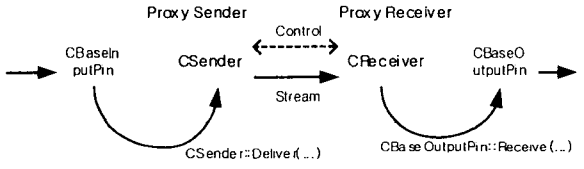

그림 4 Proxy Sender, Receiver 전체 그림

Proxy Sender, Receiver간에 교환되는 메시지는 크게 두 가지이다. 하나는 제어 메시지이고 다른 하나는 멀티미디어 데이터이다. 이 두 가지를 전달하기 위해서는 하나의 소켓으로 처리할 수도 있으나 하나의 소켓만을 사용하는 경우에는, 제어 메시지인지 데이터인지 항상 확인해야 하는 작업이 요구된다. 이럴 경우, 소켓 하나로 다 처리하면 성능에 커다란 저하가 있다.

제어 메시지 소켓은 Media Type협상, Buffer협상, Event 통지를 담당한다. 데이터 소켓은 Multimedia Streaming Data를 항상 Proxy Sender 필터에서 Proxy Receiver 필터로 전달한다. 이것의 전달과정은 CBaseInputPin::Receive(...)가 호출되어 Streaming Data를 받으면 CSender::Deliver(...)함수를 호출한다. 이 Deliver(...)함수 안에서 데이터 소켓을 통하여 Streaming Data를 전송하게 된다. CReceiver 클래스 내에 데이터 소켓을 통하여 Streaming Data를 받으면 Downstream 필터에게 수신한 Streaming Data를 전달하기 위하여 , CBaseOutputPin::Receive(...)함수를 호출한다.

이후 Streaming Data 전달 과정은 DirectShow에서 정의한 방법대로 Downstream 필터에게 전달하게 된다. 만약 데이터 소켓 연결이 끊기는 경우, End Of Stream Event로 처리한다.

일반적으로 필터간의 연결은 IPin::Connect(...)함수를 사용한다. 그러나 Network으로 분리되는 경우, Proxy Sender와 Proxy Receiver 필터가 Connection을 대리한다. Proxy Sender와 Receiver필터는 Socket을 통한 메시지 교환으로 Connection 과정을 수행한다.

필터들의 Pin 연결 시, Source 필터를 포함한 Upstream 필터들이 연결 되어 있어야 한다. 왜냐하면, Pin 연결 시 Media Type과 Buffer를 협상해야 하는데, Upstream 필터가 어떤 Media Type으로 연결할 지 결정되지 않은 상태에서 연결은 무의미하기 때문이다. Proxy Sender, Receiver 필터 연결 시에도 똑같이 적용된다. 적어도 Sender측 필터 Graph 구성이 완료되어 있어야 한다. 그러나 Receiver 측은 Filter Graph가 구성완료 되어 있을 수도 있고 Receiver 필터가 하단 필터와 연결이 안되어 있을 수도 있다.

Proxy Sender, Receiver 필터는 기본적으로 GUID_NULL형태의 모든 Media Type을 지원한다. 그래서 그림에서 Upstream 필터와 Sender 필터가 연결될 때는 Upstream 필터가 선호하는 Media Type와 Buffer 정보로 연결되고, Receiver 필터와 Downstream 필터가 연결 될 때는 Downstream 필터가 선호하는 Media Type와 Buffer 정보로 임시 연결된다. Receiver 필터에서 Sender 필터로 Control 소켓 접속 요청할 때와 Sender 필터에서 Receiver 필터로 Control 소켓 접속 요청할 때 선택하는 선호 Media Type이 다르다. Sender에서 Receiver로 소켓 연결 시에는 Sender 측 필터의 선호하는 Media Type으로 Media Type 협상하며, Receiver에서 Sender로 소켓 연결 시에는 Receiver가 선호하는 Media Type으로 Media Type협상하게 된다.

1) Receiver측 Filter Graph가 구성 완료 후

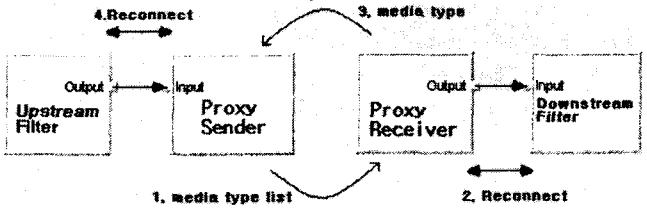

그림 5 Output pin이 선호하는 Media Type으로 협상

Proxy Sender 필터는 Upstream 필터에서 지원하는 Media Type List 와 Buffer 정보를 Proxy Receiver 필터에게 전달한다. Proxy Receiver 필터는 수신한 Media Type List와 Buffer정보로 Reconnect를 수행하게 된다.

성공적으로 Media Type을 수용하고 Buffer협상에 성공하면 그 Media Type과 Buffer 정보를 Proxy Sender 필터에게 전송한다. Proxy Sender 필터는 Upstream 필터와 Reconnect을 수행하여 Downstream 필터에서 선택한 Media Type과 Buffer정보로 Connection과정을 완료하게 된다.

input pin이 선호하는 Media type으로 협상

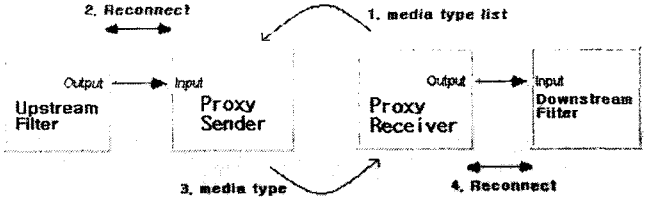

그림 6 input pin이 선호하는 Media type으로 협상

이번 경우도 위와 비슷하다. 단지, Proxy Receiver의 Media Type List를 Proxy Sender 필터로 전송한 후에, Receiver 필터가 선호하는 Media Type으로 연결을 시도한다.

2) Receiver측 Filter Graph가 구성 완료 전

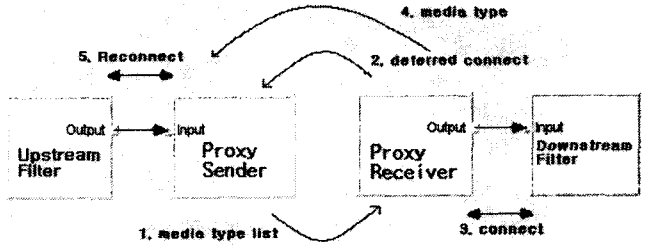

그림 7 Output Pin이 선호하는 Media Type으로 협상

Receiver 측 Filter Graph가 구성 전에는 Sender가 Receiver에게 Connection에 관련된 Media Type List와 Buffer 협상 정보를 Receiver까지 전달한다. 후에 Receiver측이 Connection이 되었을 경우, 여기서 선택된 Media Type과 Buffer 정보를 Sender측에 전달한다. Sender측 Filter Graph는 Reconnect 함으로써 Receiver측에서 선택한 Media Type과 Buffer 정보를 토대로 Connection을 재시도 한다.

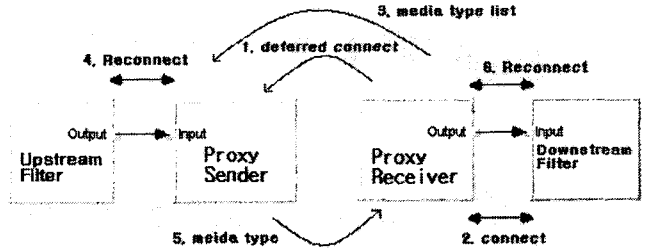

그림 8 Input pin이 선호하는 Media Type으로 협상

Receiver측이 Sender측에게 Connect이 나중에 이루어진다는 메시지를 날린다. 후에 Receiver가 Connect을 할 경우, Downstream 필터로부터 Media Type List정보와 Buffer 협상 정보를 Sender측에게 전달한다. Sender측은 Media Type List와 Buffer 정보를 갖고 Upstream 필터와 Reconnection을 한다. Sender는 여기서 선택된 Media Type을 Receiver측에 전달하여 Receiver필터가 전달 받은 Media Type과 Buffer정보 Reconnection 수행하게 된다.

Proxy Sender, Receiver가 연결이 완료되었으면, Streaming 동작과 관련 여러 Event가 있는데, Event에는 Run, Pause, Stop, EndOfStream 기타 등이 있다. Run, Pause, Stop Event는 Proxy Sender 또는 Proxy Receiver 둘 중 어느 곳 먼저 Event가 발생하더라도 상관없이 처리해야 한다.

1) State 변화

DirectShow에서 State 변화는 오른쪽 필터에서 왼쪽 필터로 상태가 차례로 변한다. Network으로 분리되는 경우도 비슷하게 처리한다. 아래 그림 9는 Sender측이 먼저 상태 변화가 발생하였을 경우를 설명한 그림이다.

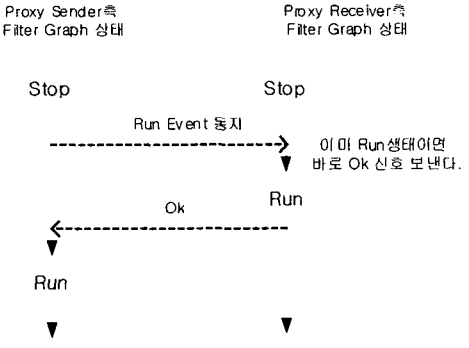

그림 9 Sender측이 먼저 상태 변화가 발생할 때

Sender측이 먼저 상태변화가 발생할 때, 자신이 상태를 먼저 바꾸면 안 된다. 자신을 먼저 상태 변화를 시킬 경우, 곧바로 Streaming이 시작하게 된다. Receiver측은 다른 상태일 수 있기 때문에 Deadlock 상태에 빠질 수 있다. 이를 피하기 위하여 Sender측은 요청한 상태 Event를 통지하고 통지결과를 받을 때까지 Blocking 상태에 빠뜨린다. Receiver측이 이 Event를 받고 현재 자신의 Filter Graph 상태가 통지된 상태가 아닐 경우, 자신의 Filter Graph 상태를 변화 시키고 Ok 메시지를 Sender측에 전송한다. Sender측이 Ok 메시지를 수신하면 Blocking 되었던 상태에서 빠져 나가고 Sender측의 Filter Graph상태를 요청했던 상태로 바꾼다.

아래그림 10은 Receiver측이 먼저 상태가 바뀔 경우이다. 이 경우는 그림 9와 약간 다르다.

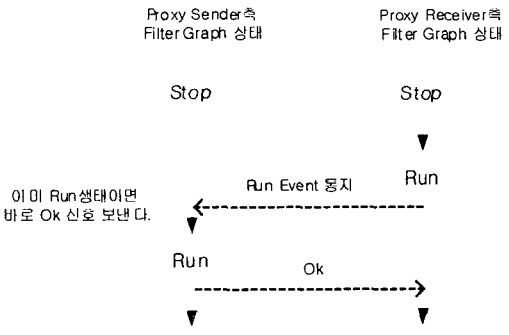

그림 10 Receiver측이 먼저 상태변화가 발생할 때

Receiver측이 먼저 상태변화가 발생할 때는, 먼저 자신의 Filter Graph상태를 변화 시킨다. 그리고 변화된 상태를 Sender측에게 전달한다. 이때 Receiver측 필터는 Blocking 상태에 빠질 필요가 없다. Sender측의 Filter Graph 상태가 변경되었으면, Receiver측에게 결과를 알려

준다.

2) End Of Stream Event 처리

Source 필터가 모든 Data를 읽었을 경우, Streaming이 끝났다는 메시지가 EOS (EndOfStream)이다..
Network으로 분리되는 경우, Source는 두개가 존재한다. 하나는 Proxy Sender측의 Source와, Receiver측의 Proxy Receiver 필터가 Source 필터에 해당된다. Sender 측에서 Source 필터에서 EOS이 발생하면, Proxy Sender가 IFilterGraph에게 바로 통지하지 않고 Receiver측에 EOS Event를 통지한다.

Receiver가 EOS을 통지 받으면 Renderer 필터쪽으로 차례로 EOS 통지한다.

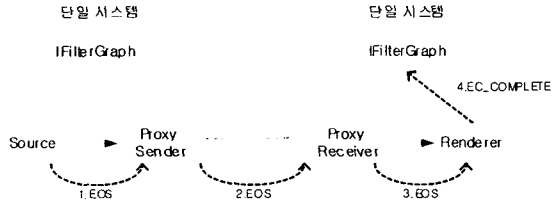

그림 11 EndOfStream 전달과정

EOS 통지 결과로 Receiver측 IFilterGraph는 EC_COMPLETE가 통지되고 Filter Graph 전체 상태가 Stop 된다[3]. Proxy Receiver가 Stop가 될 때 Sender 측에 Stop 상태 Event를 통지한다. Sender측 Filter Graph도 아래 그림처럼 Stop 상태로 차례로 변한다.

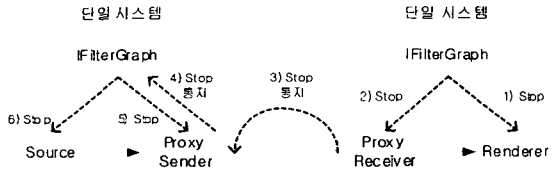

그림 12 EndOfStream후 상태변화 과정

IV 결과

아래 그림은 ATSC기반 데이터 방송 서버에서 Proxy Sender 필터와 Proxy Receiver 필터를 사용하여 Download라는 Encoder와 Trigger라는 Encoder를 Network으로 분리된 그림이다[4]. Network Proxy 필터를 사용하기전과 사용 후에 생성된 EMux.ts 파일

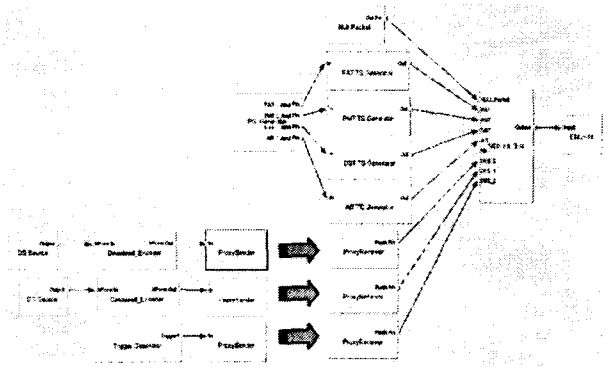

그림 13 Proxy Sender/Receiver를 Test화면

내용이 똑같았다.

V 결론

본 논문에서는 DirectShow를 분산환경 하에서 적용할 때 문제점 분석하고 해결방법을 기술하였다. DirectShow는 주로 Decoder용으로 하나의 시스템에서 사용하기 때문에 분산환경 하에서 앞서 제기한 문제점이 발생한다.

본 논문에서는 분산환경을 위한 투명성 제공하기 위한 방법을 제시하였고, Network Proxy 필터를 설계 및 구현하여 이를 검증하였다.

이 연구는 DirectShow를 사용하여 서버를 개발할 때 활용할 수 있을 것이다. 예를 들면, ATSC 기반의 데이터 방송 서버 제작 시, 그림 13처럼 여러 Encoder들이 Muxing되는 환경에서 각 Encoder를 분산환경 하에서 동작 시키는데 응용할 수 있다. 이로써 데이터 방송 서버는 성능 향상에 많은 기대를 할 수 있을 것이다.

VI Reference

[1] W.Richard Stevens, "TCP/IP Illustrated, Volum1,2", Addison Wesley, 1995

[2] Microsoft Document , "DirectX 9.0 Programmer's Reference", Microsoft, 2002

[3] 신화선, "DirectShow 멀티미디어 프로그래밍", 한빛미디어, 2002

[4] ATSC Document A/90, "ATSC Data Broadcast Standard", ATSC, 2000

[5] 전병선,"ATL COM", 삼양출판사,2000