스트리밍 프레임워크에서 미디어관리자의 설계 및 구현

이승재○ 이승룡 이재욱
경 희 대 학 교 전 자 계 산 공 학 과
{sjlee, sylee, jwlee}@oslab.kyunghee.ac.kr

The Design and Implementation of Media Manager in Multimedia Streaming Framework

Seung-Lee○, SungYoung Lee, Jae-Wook Lee
Dept. of Computer Engineering, Kyung Hee University

요 약

본 논문에서는 멀티미디어 스트리밍 프레임워크에서 미디어 관리자의 설계와 구현을 기술한다. 미디어 관리자는 스트리밍 프레임워크 내에서 미디어 스트림이 어떠한 타입의 소스로부터 얻어지며, 어떠한 종류의 스트림인가를 판별하고, 그 미디어를 가장 적절하게 처리할 수 있는 코덱을 선택한 후, 어떠한 미디어 디바이스를 통해 재생되어야 효과적인지를 식별하고 관리하기 위해서 필요하다. 제안된 미디어 관리자는 제안된 미디어 관리자는 미디어 소스와 싱크 모듈로 구성되어 있는데, 이는 멀티미디어 데이터베이스와 연동기능을 지원함으로써 높은 부가가치 서비스 제공을 가능케 하였고, RTP/RTSP 소스필터나 WinAmp 게이트웨이 기능도 지원함으로써 융통성을 제공한다. 더우기, 향후 새로운 형태의 미디어 소스가 출현하더라도 이를 용이하게 스트리밍 프레임워크에 추가시켜 서비스할 수 있는 유연성과 확장성을 지원한다.

1. 서론

본 논문에서는 다양한 네트워크 환경에서 동작할 수 있는 스트리밍 프레임워크에서 미디어 관리자의 설계와 구현에 대한 개발 경험을 소개한다.

기존의 미디어 관리자들은 코덱 구현 시 독립적인 전송 프로토콜의 송수신 모듈에 DirectShow 또는 ActiveMovie 기술의 트랜스폼 필터를 연결한 단순한 구조가 주류이지만 제안된 미디어 관리자는 파일, 데이터베이스 또는 라이브 비디오 카메라와 같이 여러 형태의 소스로부터 얻어낸 미디어를 일관성 있게 추상화시킬 뿐만 아니라, DirectShow[5] 기술을 적용하고, RTP/RTSP [2][3] 프로토콜을 처리할 수 있는 소스 필터와 Winamp[6] Gateway를 개발하여 확장성과 사용자의 편의성을 제공할 뿐만 아니라, 향후 새로운 형태의 미디어 소스가 출현하더라도 이를 용이하게 스트리밍 프레임워크에 수용하여 서비스할 수 있는 유연성과 확장성을 가지고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 미디어 관리자와 유사한 관련 연구를 소개하고, 3장에서는 미디어 관리자의 구성과 기능 그리고 각 세부모듈의 설계에 대하여 기술하고, 4장에서는 미디어 관리자의 구현 사례를 보여준 뒤, 5장에서 결론을 내린다.

2. 관련 연구

본 장에서는 스트리밍 시스템에서 미디어 관리자에 대한 기존의 연구에 대해 살펴보겠다.

JMF(Java Media Framework)[4]는 SUN 사에서 출시한 API로 자바에서 다양한 음악 파일과 동영상 파일을 재생할 수 있게 해주며, 컴포넌트화된 구조를 가지고 있다. 또한 네트워크나 로컬에 위치한 파일을 소스로 채택할 수 있으며, 데이터의 흐름을 서버 쪽에서 관리하며 클라이언트 쪽으로 데이터를 밀어 넣는 프로토콜인 RTP 방식과, 데이터의 흐름을 클라이언트 쪽에서 관리하며 서버 쪽에 미디어를 요청해서 가져오는 방식인 HTTP나 FILE 프로토콜의 두 가지 방식을 모두 지원한다. 그리고 JMF는 다양한 포맷의 미디어 파일을 플랫폼에 독립적으로 실행시킬 수 있는 장점을 가지고 있다. 그러나, 소스 채택 면에서는 멀티미디어 데이터베이스와 연동 기능을 제공하지 못한다는 점, 전송 부문에서는 UDP 기반의 RTP를 지원하지 못한다는 점이 본 논문에서 제시하는 미디어 관리자와 다르다.

[7]에서는 고속 전후진 기능을 제공하는 VOD 시스템을 개발하였는데, 이 시스템은 웹을 이용한 사용자 인터페이스, 실시간 데이터 전송을 위한 서버, 미디어를 재생하기 위한 클라이언트로 구성되어 있다. 그러나, 이 시스템은 미디어의 관리측면에서 볼 때 소스 채택 시 로컬 파일 시스템에서 읽어오는 방식 만 있다는 점, 전송 부문에서도

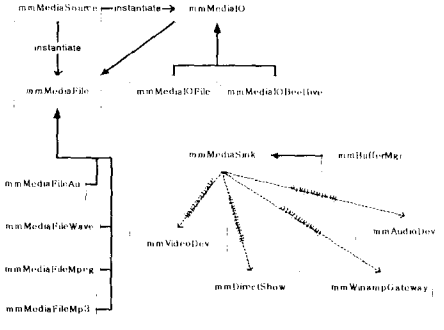
TCP/IP만을 이용하므로 멀티캐스팅이 어렵다는 점, 코덱 관점에서는 ActiveMovie를 지원하지만 DirectShow의 기준을 따르는 소스 필터를 제작하지 않아 그 확장성이 떨어진다는 점이 본 논문에서 제시하는 미디어 관리자와 다르다.

3. 미디어 관리자의 설계

미디어 관리자는 미디어 소스와 미디어 싱크의 두 가지 모듈로 구성되어 있다. 미디어 소스의 주요 기능은 입력된 미디어를 추상화시키는 것으로써, 멀티미디어 데이터베이스인 BeeHive[1]나, 실시간 동영상을 위한 디바이스와 같이 성격이 다른 여러 소스로부터 얻어지는 미디어를 일관된 방법으로 처리할 수 있는데 있다. 미디어 싱크는 클라이언트 측에서 받아온 미디어 데이터를 가장 적당한 방법으로 다룰 수 있는 관련된 코덱과 연결시키거나 디바이스에 연결시켜 사용하도록 배분해주는 분배기 역할을 해준다. 그리고, 클라이언트 측에서 전송 관리자로부터 수신된 미디어 데이터를 넘겨받아 적절한 오디오/비디오 장치로 재생한다.

오디오/비디오 코덱은 미디어 소스와 싱크에게 MPEG-1, MPEG-2, MPEG-4, MP3, WAVE, AU등의 다양한 미디어코덱을 제공한다. 제안된 미디어 관리자는 DirectShow 기술을 사용하여 대부분의 미디어를 처리하지만, 현재 DirectShow에서 기본적으로 제공하지 못하는 MPEG-2와 같은 형태의 미디어의 경우에는 그 형태의 포맷을 처리할 수 있는 다른 외부 모듈을 생성한다. 그런 다음 서비스 중에 그러한 포맷의 스트림을 요구받으면 미디어 싱크는 생성된 모듈과 요구된 스트림을 연결시켜주는 역할을 한다.

미디어 관리자는 라이브 비디오 카메라와 마이크를 사용하는 실시간 화상회의 미디어 스트림, 파일 시스템의 미디어 파일, 멀티미디어 데이터베이스의 관리를 받는 미디어까지 처리할 수 있을 뿐만 아니라, 향후 새로운 형태의 미디어 타입이 출현하더라도 이를 쉽게 수용할 수 있는 유연성과 확장성을 가지게 된다.



[그림 1] 미디어 매니져의 클래스 다이어그램

미디어 관리자의 클래스 다이어그램은 [그림 1]에서 보여주는데, mmMediaSource는 얻어온 미디어 스트림을 전송 관리자를 통해 mmMediaSink로 전송한다. 미디어 관리자에서 외부와의 연결은 mmMediaSource와 mmMediaSink 클래스를 통해 이루어지는데, mmMediaSource 클래스는 전송에 필요한 미디어 데이터를 제공하는 역할을하며, mmMediaSink는 전송된 미디어 데이터를 수신받아 클라이언트측의 버퍼를 통하여 실제 미디어가 재생될 디바이스로 전달되어 미디어가 재생되도록 한다.

3.1 DirectShow RTP 소스 필터

제안한 미디어 관리자에서는 RTP 수신단을 포함하는 DirectShow의 소스 필터를 개발하였는데, 이 필터를 이용하는 것으로 DirectShow를 이용해 개발된 다른 미디어 플레이어를 본 소스 필터로 대체하는 것으로서 RTP 프로토콜을 이용할 수 있는 성과를 얻었다.

```
If there are user's requirement {
        Switch requirement
        Case PLAY:
                send a requirement to the client session;
                send a requirement to the client I/O Object;
                initialize client's I/O operation;
                If connection is established to the server session
          and success to initialization  {
                    initialize the transport manager;
                    the server sends a media stream to the client;
                    after doing buffering, the client sends
                  a stream to the transform-filter;
                }
                Else {
                        the server stops a stream sending;
                        disconnect the client/server session;
                        initialize buffer;
                }
        Case STOP:
                the server stops a stream sending;
                disconnect the client/server session;
                initialize buffer;
        Case PAUSE:
                the server is pending to send a stream;
                keep the server/client's session connection;
                standby for the next user's requirement;
}
```

[그림 2] RTP 소스 필터의 동작 알고리즘

[그림 2]는 RTP 소스 필터의 동작 알고리즘을 보여준다. 현재 RTP 소스 필터에서 미디어 스트림에 관련되어 사용자가 선택할 수 있는 요구사항은 PLAY, STOP, PAUSE의 3가지 종류이다. PLAY는 미디어 재생을 위한 요구이고, STOP은 미디어 재생을 끝마치고 미디어 전송에 관련된 정보를 가진 서버 세션이 더 이상 필요하지 않을 때의 요구이며, PAUSE는 미디어 전송에 관련된 정보를 가진 서버 세션과의 연결은 유지한 채 잠시 미디어 재생을 멈춘 상태이다. PAUSE 다음 요구가 PLAY인 경우에는 멈춘 지점부터 다시 재생이 되며, 다음 요구가 STOP인 경우에는 재생을 마치고 세션과 연결을 끊게 된다.

3.2 버퍼관리자

다양한 미디어 재생 장치에 데이터 공급을 위해 단일 메모리에서 Push 방식과 Pull 방식을 모두 지원할 수 있

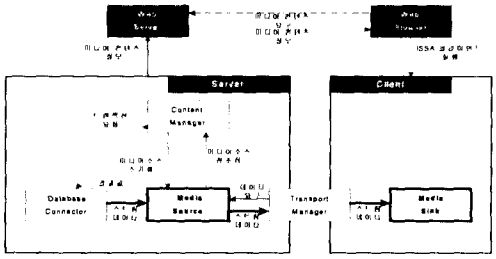
고, 네트워크 지터의 적응을 위해 버퍼링 기능을 제공할 수 있는 유연한 클라이언트에서의 Push/Pull 버퍼 관리 기법을 제안하였다[8]. 제안된 기법은 다양한 미디 재생 장치의 각기 다른 버퍼관리 요구를 단일 메모리에서 일관되게 처리함으로써 메모리의 낭비를 방지할 수 있으며, 융통성있고 단순한 버퍼관리가 가능하다. [표 1]은 버퍼관리 기법에 사용되는 주요 알고리즘들 이다.

[표 1] 버퍼 관리 기법을 위한 알고리즘

InitBuffer	버퍼를 초기화하는 알고리즘
DeinitBuffer	버퍼를 소멸시키는 알고리즘
StartBuffering	버퍼링을 시작하는 알고리즘
StopBuffering	버퍼링을 중단하는 알고리즘
AddData	버퍼에 데이터를 추가하는 알고리즘
PushData	데이터를 Push 방식으로 전달하는 알고리즘
PullData	데이터를 Pull 방식으로 전달하는 알고리즘

4. 미디어 관리자의 구현

본 논문에서 설계한 미디어 관리자를 테스트하기 위하여 간단한 VOD/AOD 서버를 구축하였다. [그림 3]는 미디어 관리자의 구현관점에서 본 서버의 논리적인구조이다.



[그림 3] VOD/AOD 서비스의 논리적 구조

[그림 4]과 [그림 5]는 각각 VOD와 AOD서비스의 작동 화면을 보여준다.

[그림 4] RTP 소스필터를 이용한 VOD 서비스 작동 구현화면

미디어 관리자의 미디어 싱크와 미디어 소스는 전송 관리자, 컨텐츠 관리자, 데이터베이스 커넥터와 직접적인 연동을 이루며, 웹 브라우저를 통해 접수되는 사용자의 미디어 컨텐츠에 대한 요구는 웹서버 를 통해 스트리밍 프레임워크로 전달된다. 서버에서는 사용자의 요구에 따른 미디어를 미디어 소스를 통해 추상화 한 후에 전송 관리자로 넘겨주며, 이 미디어를 전송 관리자로부터 얻은 미디어 싱크는 DirectShow, 하드웨어 보드, 또는 Winamp 게이트웨이 중 가장 적절한 처리 방법을 선택하여 미디어를 재생한다.

[그림 5] Winamp의 작동 구현 화면

5. 결론

본 논문에서는 스트리밍 프레임워크에서 미디어 관리자의 설계와 구현에 대한 경험을 기술하였다. 미디어 관리자는 로컬 파일이나, 데이터베이스에서 관리되는 미디어 스트림, 실시간 동영상 시스템에서 얻어온 미디어 스트림들의 추상화를 통하여 일관성 있는 처리를 가능케 하였다. 또한, DirectShow의 기술을 이용함으로써 편리하고 강력한 디코딩을 지원하는 코덱을 제공하고, 범용성 및 유연성과 확장성을 갖추고 있어 향후에 나타날 새로운 형태의 미디어 소스나 미디어 포맷을 쉽게 수용할 수 있음을 보여주었다.

6. 참고문헌

[1] J. Stankovic and S. H. Son, "An Architecture and Object Model for Distributed Object-Oriented Real-Time Databases," *Journal on Computer Systems Science and Engineering*, Special Issue on Object-Oriented Real-Time Distributed Systems, vol. 14, no. 4, pp 251-259, July 1999.

[2] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control", IETF RFC 1890, Jan. 1996

[3] H. Schulzrinne, "RTSP: Real-Time Streaming Protocol", IETF RFC 1890, Apr. 1998

[4] http://java.sun.com/marketing/collateral/jmf.html

[5] http://www.microsoft.com/DirectX/dxm/help/ds/

[6] http://www.winamp.com/customize/

[7] 이종민, 차호정, "웹 기반의 VOD 시스템 구현", 1997년 한국정보처리학회 추계 학술발표논문집 제 4권 제2호, pp.50-53, 1997년 11월.

[8] 정찬균, 이승룡, "멀티미디어 통신시스템을 위한 클라이언트에서의 Push/Pull 버퍼관리기법", 정보처리학회 멀티미디어 특집 논문집, 2000년 3월 게재 예정.