Java Media Framework을 이용한 웹 기반 실시간 멀티미디어 스트림 전송의 설계 및 구현

* 김원현, 김윤호
안동대학교 컴퓨터공학과

A Design and Implementation of Web Based Real-Time Multimedia Streamming using Java Media Framework

* Won-Hyun Kim, Yun-Ho Kim
Dept. of Computer Engineering, Andong National University

요 약

본 논문에서는 Sun의 JMF(Java Media Framework)를 이용하여 웹 기반 실시간 스트리밍 시스템의 설계 및 구현을 제시한다. 웹 기반 실시간 스트리밍 시스템 구현은 미디어 소스의 채택 부분에서 볼 때 크게 두 부분으로 나누어질 수 있는데, 첫 번째는 서버 측의 로컬 파일 시스템에서 미디어를 읽어오는 On-Demand 형식의 PULL 방식과 두 번째는 캡쳐된 영상 또는 음성을 RTP를 이용하여 실시간 처리할 수 있는 PUSH 방식을 들 수 있다. PULL 방식의 경우, 미디어파일들을 DB화 시켜 JDBC를 이용하여 On-Demand 재생을 구성할 수 있으며, PUSH 방식의 경우, 각종 캡쳐 장비를 이용하여 얻은 미디어 소스를 RTP 송신하여 실시간 미디어 재생을 구성할 수 있다.

1. 서론

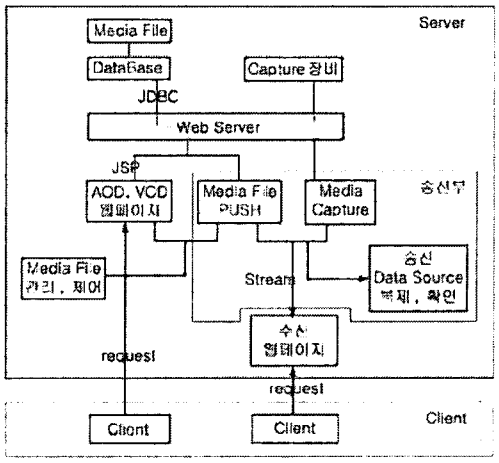
본 논문은 다양한 포맷의 미디어 파일을 플랫폼에 독립적으로 실행시킬 수 있다는 Java[1]의 장점을 이용한 JMF(Java Media Framework) API[2]로 웹 기반의 실시간 스트리밍 시스템을 구현하였다.

JMF는 SUN과 IBM의 공동연구에 의해 만들어진 자바기반 멀티미디어 프로그래밍을 위한 API이다. 즉, Microsoft사의 Video for windows, 또는 DirectX와 같은 멀티미디어 재생, 녹화, 검색, 전송 등의 기술을 자바 기반으로 만들어 이용하기 위한 도구이다. JMF는 다양한 Media file format을 지원하며, component화 된 구조를 가지고 있다. 임익진[3], 이승재[4]는 멀티미디어 스트림을 수신하는데 있어, 클라이언트 측에 Windows Media Player나 Winamp와 같은 어플리케이션이 필요하다. 제시하는 시스템에서의 멀티미디어 스트림 수신은 웹페이지에 임베드된 애플릿 형태로 구현하였다.

멀티미디어 스트림 전송은 둘로 나눌 수 있는데 첫 번째로 데이터 흐름을 클라이언트 쪽에서 관리하여 서버 쪽의 미디어 데이터를 요청해서 HTTP나 FILE 프로토콜 방식을 이용하여 서비스를 제공하는 PULL 방식의 On-Demand 재생과 두 번째로 미디어 파일 또는 각종 캡쳐 장비 등을 이용하여 얻은 미디어 소스를 PUSH 방식으로 RTP(Real-Time Transport Protocol)[5] 송신하는 방법이다.

2. 실시간 스트리밍 전송시스템

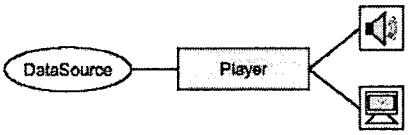
본 논문에서 소개된 On-Demand 방식과 RTP 송수신을 이용하여 실시간 스트리밍 전송시스템을 구현할 수 있는데, 모든 구현은 Java 기술을 이용하였다.



[그림 1] JMF 웹기반 실시간 전송시스템 구성도

2.1. On-Demand 방식 구현

먼저 On-Demand 방식의 구현을 보자면, 미디어 관리자는 스트리밍 해줄 DataSource를 Database화 한 후, JDBC[6] 및 JSP[7]를 이용하여 웹페이지 형식으로 클라이언트 측에 제공해 줄 수 있다. Parameter 값으로 얻어온 DataSource를 애플릿 형태로 된 Player가 Play하는 형태로 구성되어 있는데, DataSource 부분을 데이터베이스화 하여 Play 뿐만 아니라 다양한 정보 등을 제공할 수 있으며, 검색 또한 가능하게끔 설계되었다.

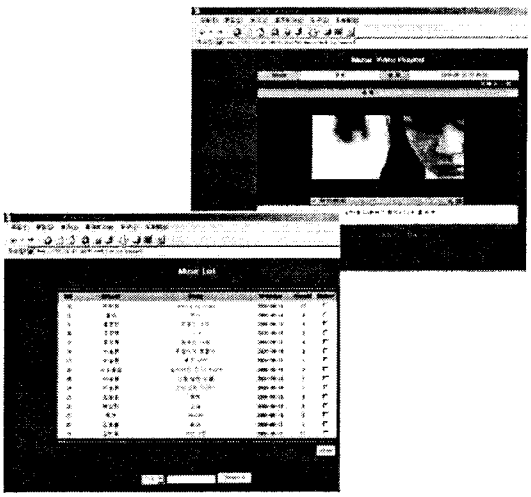


[그림 2] JMF Player 모델

· JMF Player 애플릿 동작 알고리즘

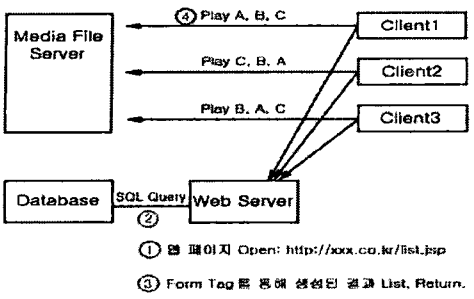
Step 1: Parameter를 읽고, 재생할 미디어의 위치를 알게 되면 Manager.createPlayer로 Player object을 생성.
Step 2: controllerListener에 등록하여 Player와 관계되는 이벤트가 발생하면 애플릿에 통지.
Step 3: player.realize()는 unrealized상태를 realized상태로 바꾼다.
Step 4: Player가 realization상태가 되었다면, ControllerUpdate는 RealizeCompleteEvent를 받는다. 이때 애플릿은 Player의 visualComponent와 controlComponent를 구성할 수 있다.
Step 5: Player가 끝에 이르면 EndofMediaEvent를 발생.
Step 6: setMediaTime()을 이용하여 Player를 초기화 시키거나, Loop를 이용하여 반복 재생, 또는 DataSource가 Array에 List 형태로 입력이 되어 있을 시에는 새로운 DataSource 값을 받아 Step 1부터 다시금 처리한다.

화면 구성 등은 Java의 서버측 스크립트 언어인 JSP를 사용하여 구현하였으며, 이를 위해 Web Server는 Java Web Server를 사용하였다. Database와의 연동을 위하여 Java의 JDBC 기술을 사용하였다.



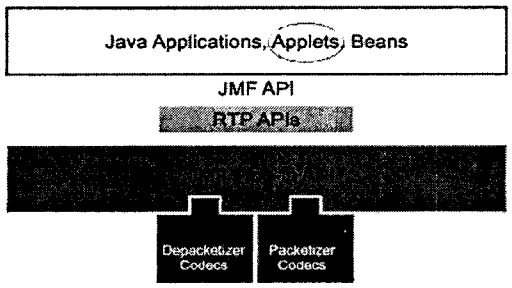
[그림 3] 웹페이지로 구성된 AOD/VOD 서비스 구현화면

Client의 사용자 편의를 위해 One Play Mode와 List Play Mode를 제공하고 있는데, One Play Mode인 경우는 Database에 저장해 놓았던 정보들을 이용하여 - 예를 들어, 음악 Data를 Play 할 때, 가사 또는 작사, 작곡자 등에 관한 데이터를 - Client 측에 제공할 수 있다. List Play Mode인 경우는 클라이언트가 Playlist를 요구 시, Form Tag 및 JSP를 응용하여 동적으로 Playlist가 작성되어 곡의 이어듣기가 가능하도록 구성하였다.



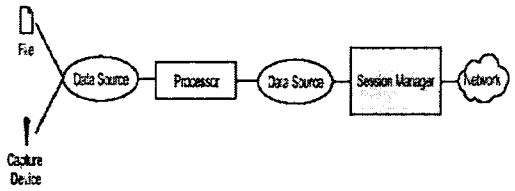
[그림 4] Dynamic Playlist 구현

2.2. 실시간 방식 구현



[그림 5] JMF RTP 구조

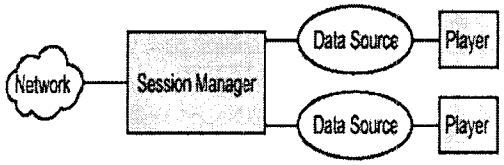
PULL 방식을 포함하여 RTP 송수신을 통한 실시간 스트림 전송의 구현에서는 관리 및 사용의 편이성을 위해 웹브라우저 내에서 모든 작업을 할 수 있게끔 하였다. 위에서 언급 한대로 RTP 송수신을 통한 스트림 전송은 미디어 관리자를 위한 송신부분과 클라이언트를 위한 수신부분으로 나누었으며, 송신부분은 다시 미디어 파일의 PUSH 방법이냐, 실시간 캡쳐 스트림 전송이냐에 따라 나누었다.



[그림 6] RTP 송신

· DataSource의 RTP 송신 알고리즘

Step 1: 파일송신일 경우는 Parameter를 읽고, 재생할 미디어의 위치를 찾고 캡처 장치일 경우는 장치 드라이버를 찾고 세팅.
Step 2: Manager.createDataSource로 DataSource를 생성.
Step 3: 전송한 미디어 데이터를 자신이 보기 위해서 DataSource를 복제.
Step 4: 프로세서 생성과 RTP 전송을 위한 포맷지정.
Step 5: 전송을 위한 SessionManager 생성과 이를 이용한 Session 생성.
Step 6: RTP 전송처리.



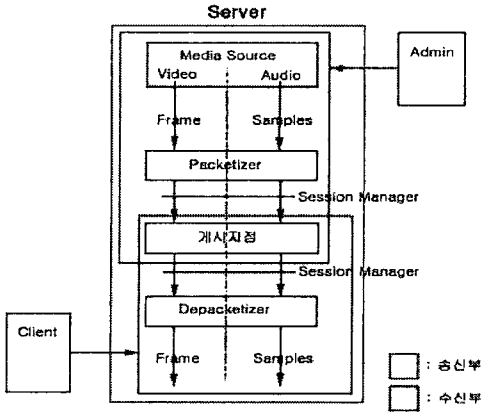
[그림 7] RTP 수신

· DataSource의 RTP 수신 알고리즘

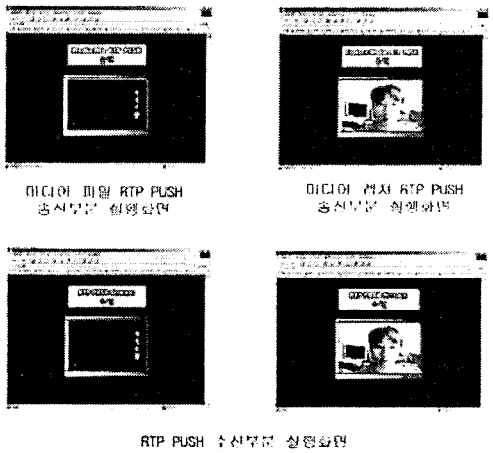
Step 1: RTP Session 부분의 Setup. 즉, 송신부에 대응하는 수신측 SessionManager 생성, 초기화.
Step 2: startSession을 호출하여 RTP Session을 시작.
Step 3: ReceiveStreamListener의 update()에서 NewReceiveStreamEvent에 대한 이벤트 처리함수에 의해 새로운 데이터 스트림이 검출되면 알려준다.
Step 4: getReceiveStream을 호출하여 ReceiveStream을 얻어온다.
Step 5: getDataSource를 호출하여 ReceiveStream으로부터 RTP DataSource를 얻어온다.
Step 6: 위에서 얻은 DataSource를 Manager.createPlayer로 넘겨서 player를 생성.

미디어 파일의 PUSH 방법의 경우, 3개의 입력 인자가 필요한데, 송신할 DataSource의 위치 정보와 전송될 미디어의 게시지점, 즉 IP address 부분, 마지막으로 사용이 가능한 유효한 포트번호가 필요하다. 여기서 송신할 DataSource의 위치 정보는 On-Demand 방식과 비슷하게 애플릿의 Parameter 값을 이용하여

HTTP 또는 FILE 프로토콜로 나타낼 수 있다. 실시간 캡쳐 송신의 경우는, DataSource의 위치 정보를 찾는 것이 아니라, 캡쳐 장치를 찾아서 DataSource를 생성해주는 부분이 필요하다. 본 시스템에서는 송신부에서도 전송될 미디어를 확인할 수 있게끔 하였는데, 이를 위해서 전송될 미디어 데이터의 복제부분이 필요하다. 수신부분에서는 PUSH된 미디어 파일과 실시간 캡처 전송된 것을 수신하는 것 둘 다 동일하다. 전송부분에서 Packetizer가 DataSource를 패킷 형태로 변환하여 송신한 것을 Depacketizer가 패킷 형태의 DataSource를 다시 Play 가능한 미디어의 형태로 변환하는 일을 한다.



[그림 8] RTP 송수신 구조



[그림 9] RTP 송수신을 통한 방송 실행화면

3. 결론

논문에서 제시한 멀티미디어 스트림 전송시스템은 웹 기반에서 PULL 방식의 On-Demand 재생과 미디어 파일 또는 각종 캡쳐 장비 등을 이용하여 얻은 미디어 소스를 PUSH 방식으로 RTP 송신하는 것이 모두 가능하다. 그러므로 기존의 미디어 서버가 행할 수 있는 모든 서비스를 대신할 수 있다. 더욱이 기존의 미디어 서버들이 특정 플랫폼에서만 운영될 수 있는 것에 비하여 본 시스템은 Java를 이용하였기에 Java의 장점인 플랫폼에 독립적으로 구축할 수 있다는 것이 장점이다. 또한 기존의 미디어 서버가 특정 미디어 데이터 형식만을 스트림 전송의 소스로 쓰고 있는 것에 반하여, 한 종류의 미디어 데이터만이 아닌, 다양한 종류의 미디어 데이터를 전송 소스로 쓰일 수 있다는 장점을 가지고 있다. 향후 과제로서는 인터페이스의 개선과 전송에서의 초기 지연시간, 많은 시스템 자원 요구 등의 문제점을 해결해야 한다.

[참고문헌]

[1] http://java.sun.com

[2] http://java.sun.com/products/java-media/jmf/

[3] 임익진, 이승룡, "멀티미디어 스트리밍 프레임워크에서 세션 관리자의 설계 및 구현", 한국 정보처리학회 2000년 춘계 학술발표 논문집, pp. 373-375, 2000년 4월.

[4] 이승재, 이승룡, 이재욱, "스트리밍 프레임워크에서 미디어관리자의 설계 및 구현", 한국 정보처리학회 2000년 춘계 학술발표 논문집, pp. 253-255, 2000년 4월.

[5] H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 1889, Jan. 1996

[6] http://java.sun.com/products/jdbc/

[7] http://java.sun.com/products/jsp/