Journal of Korea Multimedia Society Vol. 9, No. 4, April 2006(pp. 490-504)

분산 환경에서 멀티미디어 실시간 스트리밍을 위한 패키지 설계

서봉근†, 김윤호††

요 약

분산 환경에서 멀티미디어 라이브 스트리밍을 지원하는 멀티미디어 서비스를 구현하기 위해서는 다양한 플랫폼에 분산되어 있는 객체들을 제어하고, 멀티캐스트를 지원하지 않는 네크워크 환경에서 다수의 수신자에게 멀티미디어를 전송하는 것이 필요하다. 본 논문에서는 분산 객체 기술인 RMI와 멀티미디어 전송/처리에 사용하는 JMF를 확장하여 분산 환경에서 1:N의 멀티미디어 라이브 스트리밍을 지원하는 DLS (Distributed Live Streaming) 패키지를 제안한다. 효율적인 분산 처리를 위해 전송과 제어를 분리하여 설계한 자바 기반의 DLS 패키지는 1:N의 전송을 지원하고 플랫폼 독립적인 멀티미디어 서비스의 개발에 적용이 가능하다.

A Package Design for Multimedia Live Streaming in Distributed Environment

Bong-Kun Seo†, Yun-Ho Kim††

ABSTRACT

It needs to control each objects on various platform and transmit multimedia data to multiple receivers which are for developing a multimedia service with multimedia live streaming in a distributed environment. In this paper, we present a DLS (Distributed Live Streaming) package which support 1:N multimedia live streaming in a distributed environment. Also, it has extended RMI which is a distributed object technology and JMF using multimedia transmission/processing. A Java-based DLS package has been designed to separate a transmission and a control for more efficient distributed processing. It is possible to apply in development of multimedia service supported 1:N transmission and runned independently to any platform.

Key words: RMI(원격 메소드 호출), JMF(자바 미디어 프레임워크), Distributed Object Technology(분산 객체 기술), Live Streaming(라이브 스트리밍)

1. 서 론

네트워크상에 분산되어 있는 대용량의 멀티미디어 데이터를 처리하여 실시간으로 전송하기 위해서는 컴퓨팅 기술과 고속 통신 기술의 발전은 필수이다. 최근 고성능의 컴퓨팅 파워를 가진 개인 PC의 보급과 고속 통신 기술의 발달로 화상회의, NOD (News on Demand), VOD (Video on Demand)와 같은 멀티미디어 서비스들이 개발, 제공되고 있다.

그러나 이와 같은 멀티미디어 서비스는 네트워크상의 이질적인 플랫폼에 분산되어 있는 멀티미디어 객체들 간의 통신, 상호 운용 등의 복잡함으로 인해

※ 교신저자(Corresponding Author) : 서봉근, 주소 : 경상북도 안동시 송천동 388(760-749), 전화 : 054)820-5664, FAX : 054)820-6164, E-mail : eagle@comeng.andong.ac.kr
접수일 : 2005년 8월 9일, 완료일 : 2005년 11월 21일

† 준회원, 안동대학교 대학원 컴퓨터공학과 석사과정
†† 정회원, 안동대학교 전자정보산업학부 부교수
(E-mail : unokim@andong.ac.kr)

개발에 어려움이 따른다. 또한 동일한 멀티미디어 데이터를 다수의 수신자에게 동시에 전달 (1:N 전송)하기 위한 IP 멀티캐스트 (Multicast)[1]가 여러 가지 기술적 문제점들로 인하여 네트워크 장비 (라우터)에서 지원하지 않고 있으므로 이를 해결하기 위한 대안이 필요하다.

최근의 분산 객체 기술의 발달은 분산 환경에서 간단한 객체 통신 방법을 제공함으로써 분산 객체들 간의 상호 운용성 문제를 해결하는데 적합하다. 그 중 Sun의 분산 객체 기술인 RMI (Remote Method Invocation)[2]는 자바 기반으로써 플랫폼에 독립성을 제공하며 객체 지향 언어의 다양한 장점을 가지고 있다. 하지만 멀티미디어 객체의 라이브 스트리밍 기능을 지원하지 않으므로 분산 환경에서 멀티미디어를 라이브 스트리밍 하는 멀티미디어 서비스를 개발하는데 적용이 어렵다.

따라서 분산 환경에서 멀티미디어 라이브 스트리밍을 가능케 하기 위해서는 분산 객체 기술을 사용하여 객체를 제어하고 멀티미디어를 1:N로 전송하고 수신, 처리하는 부분을 분리하여 개발하는 것이 필요하다. 본 논문에서는 분산 환경에서 객체간의 멀티미디어 라이브 스트리밍을 지원하기 위한 자바 기반의 DLS (Distributed Live Streaming) 패키지를 제안한다. DLS는 자바의 분산 객체 기술인 RMI를 사용하여 분산 객체들의 상호 운용성 문제를 해결하기 위해 설계한 DOC (Distributed Object Communication) 패키지와 JMF (Java Media Framework)[3]를 확장하여 멀티미디어 데이터를 복제하고 다수의 수신자에게 유니캐스트 (Unicast)로 라이브 스트리밍을 수행할 수 있게 설계한 LS (Live Streaming) 패키지로 구성된다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 분산 객체 라이브 스트리밍의 소개와 문제점 및 이를 해결하기 위한 DLS 패키지를 제안하고 3장에서는 2장에서 제안한 DLS 패키지와 하부 단을 구성하는 DOC, LS 패키지를 차례로 설계하고 설계한 DLS 패키지를 검증한 내용을 기술한다. 마지막으로 4장에서는 결론과 향후 연구 과제에 관하여 기술한다.

2. 분산 객체 기술과 라이브 스트리밍

분산 환경에서 미디어 캡쳐 장치 (CAM, 마이크)를 이용하여 얻은 멀티미디어 데이터를 네트워크를 통해 송/수신하기 위해서는 먼저 송/수신측에서 필요한 준비를 하여야 한다. 수신 준비를 하지 않은 수신측에게 송신측에서 멀티미디어 데이터를 생성하여 송신을 한다면 수신측은 이것을 수신할 수 없다. 반대로 송신측에서 언제 데이터를 전송할지 알 수 없는 상황에서 수신측에서 멀티미디어 수신과 재생 준비를 모두 마치고 대기를 하게 된다면 시스템 자원의 낭비가 된다.

이것은 송신에 앞서 수신측에게 메시지를 보내어 수신 준비를 마치게 한 후 멀티미디어 데이터를 송신하여 해결이 가능하다. 전통적인 TCP 또는 UDP기반의 소켓을 사용하여 메시지를 전송하는 것은 분산 환경을 구성하는 객체의 수가 증가할 경우 상당히 복잡해 질 수 있다[4].

분산 환경에서 복잡한 객체의 상호 운용을 효율적으로 처리하기 위해 발표된 Microsoft의 DCOM (Distributed Component Object Model)[5], OMG (Object Management Group)의 CORBA (Common Object Request Broker Architecture)[6] 그리고 Sun의 RMI 등과 같은 분산 객체 기술은 분산 객체들의 복잡한 통신, 제어, 상호 운용성 등을 해결하는데 적합하다.

Windows 플랫폼에서 확고한 위치를 차지하고 있는 DCOM은 Microsoft의 독자적인 분산 객체 구조를 가지고 있지만 그 외의 플랫폼에 적용이 어려운 실정이다. 국제 표준으로 제시된 CORBA는 특정 언어에 구애받지 않고 다양한 언어로 구현된 객체들 간의 통신을 지원하여 플랫폼에 종속성은 제거하였으나 이를 위해 IDL (Interface Definition Language)[6]을 별도로 정의하여야 하며 분산 처리 미들웨어를 설치해야 한다.

반면 Sun의 자바 기반 RMI는 JRE (Java Runtime Environment)만 설치되어 있으면 별도의 미들웨어를 구매할 필요 없이 플랫폼에 독립성을 제공하는 분산 객체 환경을 구축할 수 있다. 그러나 앞서 기술한 분산 객체 기술들은 멀티미디어 라이브 스트리밍 기능을 지원하지 않으므로 분산 객체 환경에서 멀티미디어 라이브 스트리밍을 위한 다양한 대안들이 연구되고 있다[4,7,8].

동일한 멀티미디어 데이터를 원하는 다수의 수신자에게 동시에 전송해야 하는 멀티미디어 서비스에

서 최적의 데이터 전송 기술은 IP 멀티캐스트이다. 그러나 멀티캐스트는 다양한 기술적 문제점들로 인하여 현재 네트워크 장비 (라우터) 중 이를 지원하지 않는 장비가 존재하므로 보편적으로 사용되지 못하고 있다. 이의 대안으로 현재 안정적으로 제공되는 유니캐스트를 사용하여 모든 수신자에게 동일한 데이터를 반복적으로 전송하거나 라우터 대신 일반 PC와 같은 종단 단말기로 이루어진 멀티캐스트 망을 구축하여 유니캐스트 환경에서도 IP 멀티캐스트와 유사한 기능을 갖는 오버레이 멀티캐스트 기술 등이 연구/제안되고 있다[9-11].

본 논문에서는 제시하는 DLS 패키지는 RMI를 사용하여 분산 환경 구축하고 분산된 객체들의 제어에 사용하는 DOC 패키지와 RMI의 라이브 스트리밍 기능 부재를 해결하기 위해 JMF를 확장하여 멀티미디어 라이브 스트리밍에 적합하게 설계한 LS 패키지로 구성된다. LS 패키지는 자바의 빈약한 멀티미디어 지원을 보강하기 위해 SUN에서 발표한 자바의 확장 패키지인 JMF를 사용하여 멀티미디어 데이터의 생성, 전송, 재생에 사용하며 멀티캐스트를 지원하지 않는 네크워크 환경에서도 유니캐스트를 사용하여 동일한 데이터를 수신자의 수만큼 복제하여 전송함으로써 멀티캐스트와 유사한 1:N의 전송을 지원할 수 있다.

이와 같이 자바 기반의 RMI와 JMF를 사용하여 설계한 DLS 패키지는 플랫폼에 독립적인 멀티미디어 서비스 개발에 사용이 가능하며 분산 객체의 제어와 실제 멀티미디어 데이터의 전송을 분리하여 효율적이고 유연한 구조를 가진다. 또한 RMI와 JMF의 복잡한 API를 분산 멀티미디어 라이브 스트리밍에 적합하게 통합하고 단순화한 API를 제공함으로써 이를 적용하여 개발된 소프트웨어의 유지, 보수에 드는 비용을 줄일 수 있을 것이다.

3. DLS (Distributed Live Streaming) 패키지 설계

분산 객체들 간의 라이브 스트리밍을 지원하기 위한 DLS 패키지는 운영 플랫폼에 독립성을 제공하기 위해 자바를 사용하여 설계한다. DLS 패키지의 하부를 구성하는 두 개의 패키지 중 DOC 패키지는 분산 객체들 간의 통신과 제어에 사용하기 위하여 RMI를 사용하며, 또 하나의 구성 요소인 LS 패키지는 다수의 수신자에게 전송을 할 때 필요한 멀티미디어 데이터를 복제하고 이를 실제로 전송/수신하여 처리하는 기능을 가진다. 이 LS 패키지는 JMF에서 지원하는 RTP(Real-time Transfer Protocol)[12,13]를 사용하여 멀티미디어 데이터의 전송을 담당하게 된다. 따라서 DLS 패키지의 사용자는 분산 객체들의 제어나 미디어 스트리밍에 관한 지식 없이도 DLS 패키지만을 이용하여 애플리케이션을 개발할 수 있다 (그림1 참조).

제안하는 DLS 패키지의 하부를 구성하는 DOC 패키지와 LS 패키지는 서로의 기능이 분리되어 있으므로 DLS 패키지를 통하지 않고 독자적으로 이들 패키지를 사용하는 애플리케이션의 개발에도 이용할 수 있다. 또한 이들 패키지는 보다 나은 성능을 제공하는 패키지를 설계하여 교체할 수도 있다.

3.1 DLS (Distributed Live Streaming) 패키지

DLS 패키지는 DOC 패키지와 상호 작용을 통하여 분산 객체들의 제어를 하며, LS 패키지와의 상호 작용을 통해서 분산 객체들 간의 멀티미디어 라이브 스트리밍을 수행한다. 이런 특징은 DLS 패키지가 제공하는 API를 사용하여 분산 객체 라이브.스트리밍을 응용하는 애플리케이션을 개발할 때 분산 객체들 사이에 제어와 라이브 스트리밍에 대한 내용을 파악하지 않아도 간단하고 신속하게 애플리케이션 개발을 가능하게 하므로 개발 시간과 비용을 줄일 수 있게 한다.

(1) DLS 패키지의 구성

DLS 패키지는 하부의 DOC 패키지와 LS 패키지

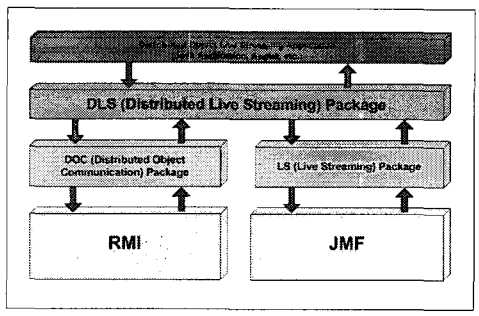


그림 1. 분산 객체 라이브 스트리밍을 사용한 애플리케이션 내부 구조

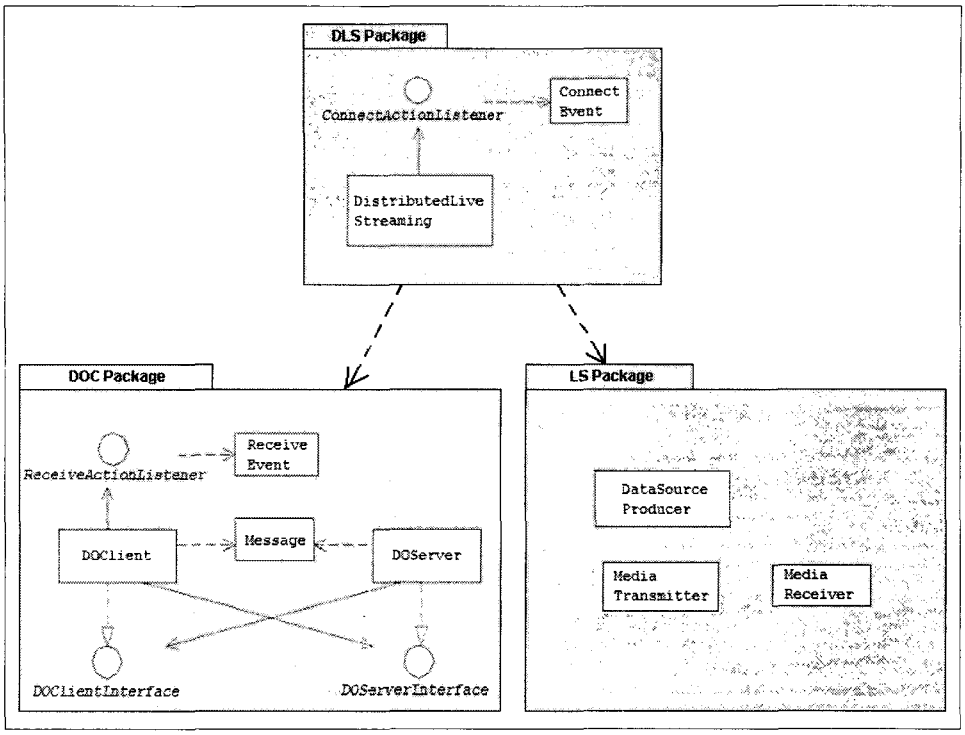


그림 2. 전체 패키지 구조

에서 분산 객체 라이브 스트리밍에 필요한 기능만을 수용하여 하나의 인터페이스와 두개의 클래스로 구성한다.

그림 2의 전체 패키지 구조에서 아래의 DOC 패키지를 구성하는 클래스들은 분산 객체를 제어하는 기능을 담당하고 LS 패키지를 구성하는 클래스들이 라이브 스트리밍에 대한 기능을 처리하게 되므로 DLS 패키지는 분산 객체들의 연결 (접속, 접속 해제)에 필요한 기능과 라이브 스트리밍의 시작과 중지를 위한 간단한 클래스로 이루어진다.

표 1은 DLS 패키지를 구성하는 클래스와 이들이 제공하는 메소드들의 목록이다. DLS 패키지에서 접

표 1-1. DistributedLiveStreaming 클래스

생성자 & 메소드	설 명
DistributedLiveStreaming(JPanel[] panel)	멀티미디어 재생을 위한 JPanel[]을 받아서 객체를 생성
void connect(String ip)	주어진 서버 IP로 접속
void disconnect()	접속 해제
void start(String ip, int port, int option, String path)	수신자 IP, port, option, path를 인자로 멀티미디어 데이터를 생성하고 전송 시작
void stop(String ip)	멀티미디어 데이터 전송 중지
String[] getClientList()	접속한 클라이언트들의 IP 획득
void addConnectListener(ConnectActionListener listener)	클라이언트가 접속할 때 발생하는 이벤트를 처리하기 위한 Listener 등록

표 1-2. *ConnectActionListener* 인터페이스

메소드	설 명
void clientConnected(ConnectEvent event)	클라이언트가 접속할 때 호출됨

속에 관련된 이벤트의 정보를 가지고 있는 ConnectEvent 클래스는 ConnectActionListener 인터페이스에서 사용하고 있다. 이 클래스는 사용자에게 제공하지 않는 DLS 패키지의 내부 클래스이다. DLS 패키지에서 1:N의 멀티캐스트 지원은 DistributedLiveStreaming 클래스의 start() 메소드가 지원한다. 이 메소드를 N의 수만큼의 수신측 정보를 입력하여 중복으로 호출함으로써 1:N의 전송을 수행할 수 있다.

(2) DLS 패키지의 통신 시나리오

DLS 패키지를 사용하여 분산 객체 미디어 스트리밍을 하기 위해서는 먼저 미디어를 수신 받을 대상을 네트워크에 분산되어 있는 객체들에서 검색하여 가져와야 한다. DLS 패키지에서는 이 과정을 DOC 패키지에게 위임한다. DOC 패키지는 클라이언트/서버 분산 환경을 구축하고 모든 클라이언트들을 서버에 접속하게 하여 객체들의 연결한다. 연결된 객체들은 서버를 통하여 서로 자신의 존재를 알릴 수 있게 된다. 그 다음 대상을 선택한 후 DOC 패키지를 이용하여 미디어의 수신 준비를 하라는 메시지를 보낸다. 이 메시지를 받은 대상은 수신을 준비하고 대기를 하게 된다. 실제적인 미디어 스트리밍은 수신 준비가 끝나면 LS 패키지를 사용하여 수행한다. 미디어 스트리밍의 중지 역시 같은 방법으로 먼저 수신을 중지하라는 메시지를 DOC 패키지로 보낸 후 LS 패키지를 사용하여 미디어 스트리밍을 중지한다. 마지막으로 미디어 스트리밍을 마치면 DOC 패키지를 이용하여 접속을 해제한다. 이와 같이 DLS 패키지를 사용하여 미디어 스트리밍을 하기 위해서는 크게 접속, 스트리밍 시작/중지, 그리고 접속 해제의 단계를 거친다.

① 접속

접속은 미디어 스트리밍을 하기에 앞서 서버에 접속하고 수신할 대상을 찾기 위한 단계이다. DLS 패키지는 DOC 패키지를 이용하여 그림 3과 같은 과정으로 수행한다. 이 그림은 UML[14]의 시퀀스 다이어그램으로 표현한 것이다.

아래 ***step***에서 그림 3의 dls1:DistributedLiveStreaming과 doc1:DOClient의 생성은 초기화 과정에서 이루어지는 것을 전제로 한다. dls1은 미디어 재생을 위한 javax.swing.JPanel의 객체를 인자로 받아 생성한다. 그리고 설명의 편의상 Receiver는

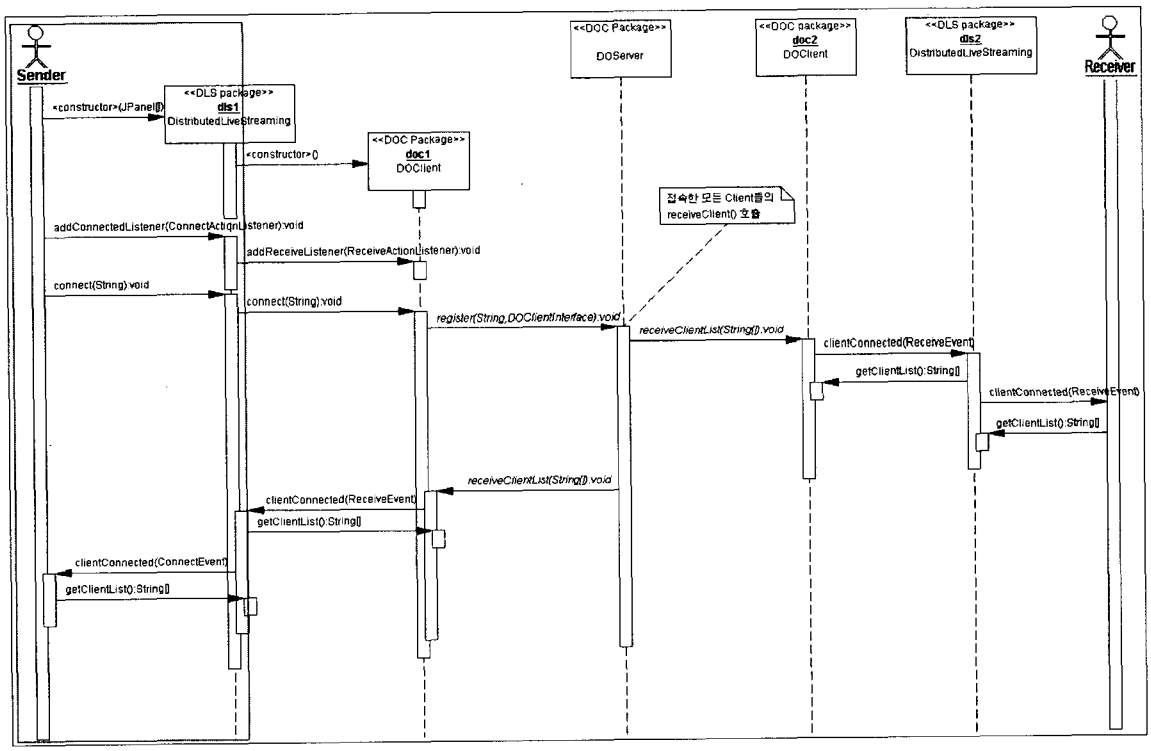


그림 3. DLS 패키지의 접속 과정

dls2:DistributedLiveStreaming과 doc2:DOClient를 생성하고 먼저 아래와 동일한 과정으로 DOServer에 접속한 것으로 가정하고 설명한다. 그림 3의 점선으로 감싸여진 사각형 부분에는 접속을 위해 DLS 패키지가 사용자에게 제공하는 메소드들이 나타난다.

이벤트 리스너 등록 : addConnectedListener() 메소드 호출 - Sender가 DOServer에 접속하기 이전에 호출하는 메소드로써 DOServer에 다른 DOClient가 접속하였을 때 dls1이 발생시키는 이벤트 (ConnectEvent)를 처리하기 위해 *ConnectActionListener* 인터페이스를 상속한 Sender를 등록한다.

step 1 : addConnectonListener() 메소드는 곧바로 다른 DOClient가 DOServer에 접속할 때 doc1이 발생시키는 이벤트 (ReceiveEvent)를 처리하기 위해 addMessageListener() 메소드를 호출

접속 : connect() 메소드 호출 - DOServer에 접속을 위해 DOServer의 IP를 인자로 받는다.

step 1 : DOClient의 connect() 메소드 호출 - 접속을 위해 DOServer의 IP를 인자로 받음

step 2 : DOServer의 register() 원격 메소드 호출 - 인자로 doc1의 IP와 doc1의 원격 참조자 전달

step 3 : DOServer는 java.util.Hashtable에 IP와 참조자를 저장하고 이미 접속한 doc2의 receiveClientList() 원격 메소드 호출 - doc2에게 접속한 doc1의 정보를 전달

step 4 : doc2는 ReceiveEvent를 발생 시키고 dls2의 clientConnected() 메소드 호출

step 5 : dls2는 접속한 doc1의 정보를 가져오기 위해 doc2의 getClientList() 메소드 호출

step 6 : dls2는 ReceiveEvent를 발생시키고 Receiver의 clientConnected()메소드 호출

step 7 : Receiver는 접속한 doc1의 정보를 가져오기 위해 dls2의 getClientList() 메소드 호출

※*step 3*~*step 7*까지와 동일한 방법으로 Sender에게 이미 접속해 있는 Receiver의 정보를 전달

② 미디어 스트리밍 시작/중지

DLS 패키지를 사용한 미디어 스트리밍의 시작은 미디어를 수신 받을 대상을 선택하고 전송측의 정보, 수신측의 정보 및 전송을 시작한다는 메시지를 DOC 패키지를 이용해서 먼저 수신측에 보내어 시작한다. 수신측은 메시지를 받아 전송측의 정보를 분석하고 이에 적합한 LS 패키지의 수신자 (MediaReceiver)를 생성하고 수신 준비를 한다. 실질적인 미디어 스트리밍은 수신 준비가 끝나면 LS 패키지의 전송자 (MediaTransmitter)를 이용하여 시작된다.

그림 4의 점선으로 감싸여진 사각형 부분이 DLS 패키지에서 미디어 스트리밍을 위해 사용자에게 제공하는 메소드들을 포함한 부분이다. 가는 실선의 윗

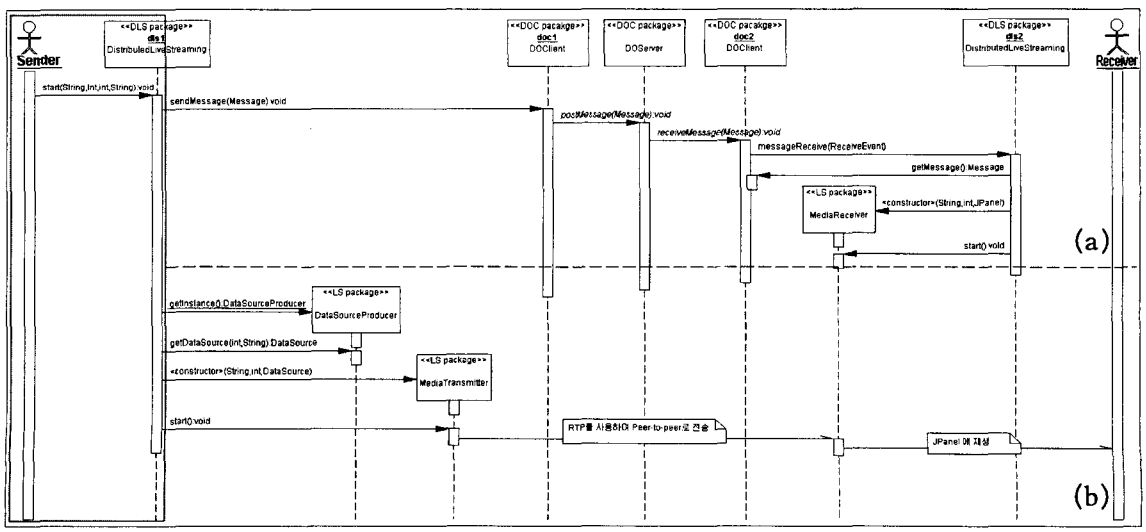


그림 4. DLS 패키지의 미디어 스트리밍 과정

부분 (a)는 미디어 스트리밍을 시작하기 전에 먼저 DOC 패키지를 이용하여 메시지를 전송하여 수신을 준비 시키는 과정이며, 아랫부분 (b)는 미디어 스트리밍을 위한 미디어 데이터를 생성하고 실질적인 미디어 스트리밍을 수행하는 부분이다. 미디어 스트리밍의 중지는 시작과 동일한 과정을 거치게 되므로 생략 하였다. 즉 스트리밍의 중지도 시작과 같이 먼저 DOC 패키지를 이용하여 중지 메시지를 전송하여 수신을 중지 시킨 다음 실제 미디어 스트리밍을 중지한다.

미디어 스트리밍 시작 : start() 메소드 호출 - 수신측 IP와 전송 Port, 전송할 미디어 데이터의 정보를 인자로 받음

step 1 : doc1의 sendMessage() 메소드 호출 - 인자는 전송측의 IP, 전송 Port, 수신측의 IP 및 전송 시작의 정보를 가지는 Message 오브젝트

step 2 : DOServer의 postMessage() 원격 메소드 호출 - DOServer는 받은 Message에서 수신측 (doc2)의 IP를 추출

step 3 : doc2의 receiveMessage() 원격 메소드 호출 - doc2에게 Message 전달

step 4 : doc2는 ReceiveEvent를 발생시키고 dls2의 messageReceive() 메소드 호출

step 5 : doc2의 getMessage() 메소드 호출 - doc2가 DOServer에서 받은 Message를 가져옴

step 6 : Message에서 전송 시작 메시지를 확인한 후, 전송측의 IP와 전송 Port를 추출해서 미디어를 재생할 javax.swing.JPanel과 함께 인자로 MediaReceiver를 생성

step 7 : MediaReceiver의 start() 메소드 호출 - 미디어 수신을 위한 준비를 하고 수신 대기

step 8 : DataSourceProducer의 getInstance() 메소드 호출 - 스트리밍에 필요한 미디어 데이터 (DataSource)의 생성을 담당하는 DataSourceProducer의 오브젝트를 반환

step 9 : DataSourceProducer의 getDataSource() 메소드 호출 - 인자로 받은 생성할 미디어 데이터의 정보 (비디오, 오디오, 미디어 파일)에 적합한 DataSource를 생성해서 반환

step 10 : MediaTrnasmitter 생성 - 실제적인 미디어를 전송하는 MediaTrnasmitter를 수신측의 IP와 전송 Port, 전송에 사용하는 DataSource를 인자로 생성

step 11 : MediaTrnasmitter의 start() 메소드 호출 - 수신측의 MediaReceiver로 미디어 스트리밍 시작, RTP를 사용하여 Peer-to-peer로 스트리밍 시작

step 12 : 미디어를 수신하는 MediaReceiver는 생성될 때 받은 javax.swing.JPanel에 미디어를 재생

③ 접속 해제

접속 해제는 접속과 유사한 과정을 따르며 DOC 패키지를 이용하여 그림 5와 같은 과정으로 수행한다.

그림 5의 왼쪽 점선으로 감싸여진 사각형 부분은 DLS 패키지에서 접속 해제를 위해 사용자에게 제공하는 메소드를 포함하는 부분이다.

접속 해제 : disconnect() 메소드 호출 - DOServer로의 접속 해제를 위해 호출

step 1 : DOClient의 disconnect() 메소드 호출

step 2 : DOServer의 unregister() 원격 메소드 호출 - 인자로 doc1의 IP를 전달

step 3 : DOServer는 doc1의 IP를 사용하여 java.util.Hashtable에서 doc1의 원격 참조자를 제거하고 접속을 유지하고 있는 doc2의 receiveClientList() 원격 메소드 호출 - doc2에게 접속 해제한 doc1의 정보를 전달

step 4 : doc2는 ReceiveEvent를 발생 시키고 dls2의 clientConnected() 메소드 호출

step 5 : dls2는 접속 해제한 doc1의 정보를 가져오기 위해 doc2의 getClientList() 메소드 호출

step 6 : dls2는 ReceiveEvent를 발생시키고 Receiver의 clientConnected() 메소드 호출

step 7 : Receiver는 접속 해제한 doc1의 정보를 가져오기 위해 dls2의 getClientList() 메소드 호출

3.2 DOC(Distributed Object Communication) 패키지

DOC 패키지는 분산 객체 환경을 구축하여 객체들

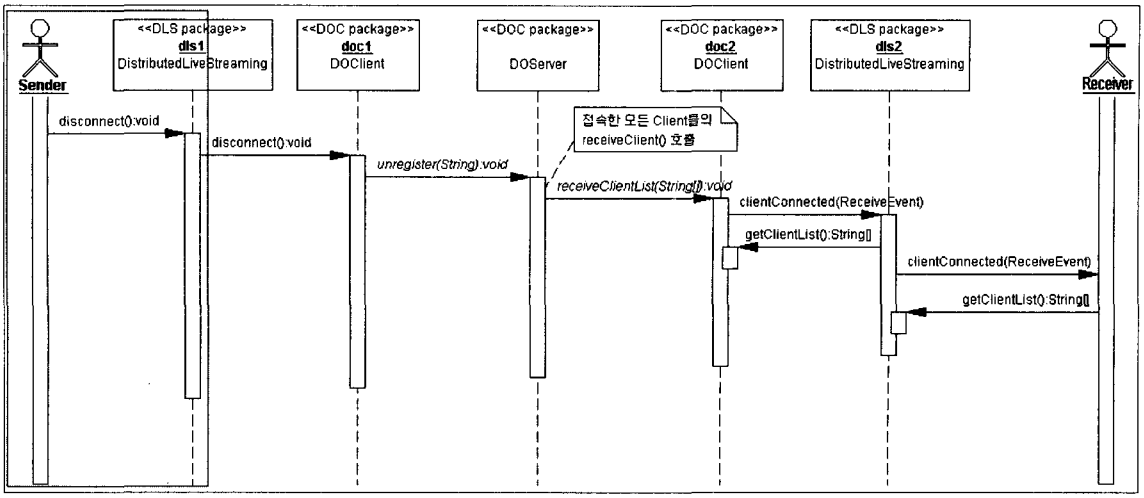


그림 5. DLS 패키지의 접속 해제 과정

을 연결하고 이들 사이의 제어와 통신을 담당하기 위한 패키지이다. RMI 기반의 DOC 패키지에서 분산 객체 환경을 구성하는 객체는 네트워크에 분산되어 있는 객체 (이하 클라이언트)와 이들을 제어하기 위한 객체 (이하 서버)이다. 모든 분산 객체들은 메시지를 서버를 통하여 전송하게 된다. 이는 일반적인 클라이언트/서버 환경에서 이루어지는 통신 방법과 동일하지만 이들 사이의 메시지 전송과 접속은 모두 원격 메소드 호출을 통하여 이루어진다.

분산 객체들의 제어와 통신에 사용하는 DOC 패키지는 메시지 전송에 Message 클래스를 정의 하여 사용한다. 이 Message 클래스는 상속을 통한 확장을 가능하게 한다. 이는 DLS 패키지에서 라이브 스트리밍을 위하여 제어에 사용하는 용도 이외에 상속을 통하여 채팅 메시지나 화이트보드 등에 사용하는 다양한 메시지를 추가할 수 있도록 한다. 이러한 DOC 패키지의 특징은 분산 객체 라이브 스트리밍 분야뿐만 아니라 분산 객체 환경에서 다양한 애플리케이션의 개발에 이용할 수 있게 할 것이다. RMI 기반으로 설계하여 전통적인 소켓 기반으로 구축된 분산 환경에서 객체들을 제어하는 것보다 간편한 분산 객체 환경을 구축할 수 있지만 RMI를 사용한 원격 메소드 호출을 위해서는 원격으로 호출될 메소드들을 원격 인터페이스로 정의해야 하고, RMI Registry에 원격 인터페이스를 상속, 구현한 원격 객체를 생성, 등록하고 검색하는 등의 번거로운 과정을 거쳐야만 한다. DOC 패키지는 이러한 모든 과정을 미리 구현하여 사용자에게는 더욱 간편하게 분산 객체 환경을 구축할 수 있게 한다.

(1) 분산 객체의 원격 호출 절차

그림 6은 클라이언트가 서버에 접속하고 메시지를 다른 클라이언트에게 전송하는 과정을 간략화한 것이다. 클라이언트는 먼저 서버에 접속하고 먼저 접속한 클라이언트의 목록을 받아와서 목적지의 클라이언트를 선택, 서버를 통하여 제어 메시지를 전달한다. 이 모든 절차는 원격 메소드 호출을 통하여 수행한다.

DOC 패키지에서 그림 6의 클라이언트는 DOClient 클래스로 구현이 되며, 서버는 DOServer 클래스로 구현이 된다. 두 클래스에서 생성한 객체들이 서로 원격 호출을 하기위해서는 RMI Registry에 DOClient와 DOServer 클래스에서 생성된 원격 객

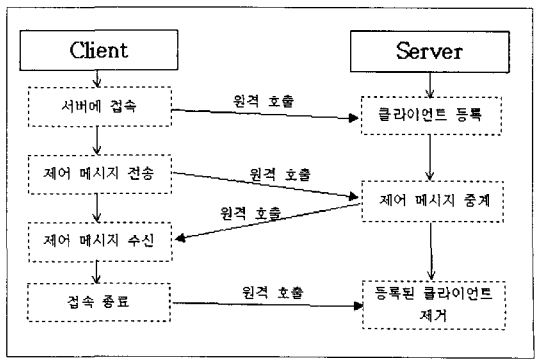


그림 6. 분산 객체의 원격 호출 절차

체를 저장하고 검색, 반환하는 설정 과정이 그림 7과 같이 필요하다.

DOC 패키지를 구성하는 주요 클래스인 DOClient와 DOServer는 쌍방 간의 원격 메소드 호출을 하기 위해 원격 객체를 생성하고 RMI Registry에 원격 객체의 참조자를 저장한다. 원격 메소드 호출을 하기 위한 상대는 RMI Registry에 저장된 참조자를 검색하여 반환되는 참조자를 통하여 원격 객체의 메소드를 호출할 수 있게 된다. 이 같은 과정은 크게 접속, 메시지 전송/수신, 접속 해제로 나누어 동작하는 DOC 패키지에서 DOClient가 DOServer에 접속할 때 수행된다. 접속 절차는 그림 3의 절차 설명 중 **이벤트 리스너 등록** *step 1*과 **접속**의 *step 1*~*step 5*에 나타난다. 메시지 전송/수신의 절차는 그림 4의 **미디어 스트리밍 시작**에서 *step 1*~*step 5* 사이에 나타나며, 그림 5 **접속 해제**의 *step 1*~*step 5*에 걸쳐 표시한 절차는 DOC 패키지의 접속 해제 과정을 나타낸다.

(2) DOServer

DOServer는 DOClient 객체들의 접속 (연결)과 이들 사이의 메시지 중계에 사용된다. 따라서 DOClient들이 접속과 접속해지를 위해 호출하는 register(), unregister() 원격 메소드와 메시지를 중계할 때 호출하는 postMessage() 원격 메소드가 필요하다. DOClient의 접속 정보는 java.util.Hashtable에 IP와 함께 원격 객체의 참조자를 같이 저장하여 유지한다.

```
public interface DOServerInterface extends Remote {
  public void register(String ip,
                       DOClientInterface client)
                       throws RemoteException;
  public void unregister(String ip)
                       throws RemoteException;
  public void postMessage(Message message)
                       throws RemoteException;
}
```

DOServer는 DOClient에서 원격으로 호출하는 메소드들을 위와 같이 DOServerInterface 원격 인터페이스에 정의하고 이를 상속하여 구현한다.

(3) DOClient

DOClient 클래스에는 DOServer에 접속한 후 이미 접속한 다른 DOClient 객체들의 목록을 받기 위한 receiveClientList() 원격 메소드와 메시지를 받기 위한 receiveMessage() 원격 메소드를 정의한다. DOClient에 정의된 원격 메소드는 모두 DOServer에서 호출한다.

```
public interface DOClientInterface extends Remote {
  public void receiveClientList(
                       String[] clientList)
                       throws RemoteException;
  public void receiveMessage(Message message)
                       throws RemoteException;
}
```

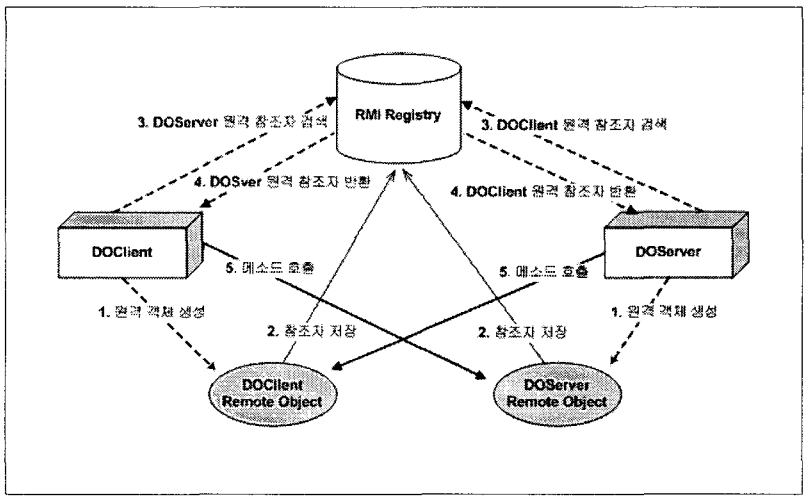


그림 7. DOC 패키지의 원격 호출을 위한 설정

표 2-1. DOClient 클래스

메소드	설 명
void connect(String ip)	주어진 서버 IP로 접속
void disconnect()	접속 해제
void sendMessage(Message message)	메시지 전송
Message getMessage()	수신된 메시지 획득
String[] getClientList()	접속한 클라이언트들의 IP 획득
void addReceiveListener(ReceiveActionListener listener)	클라이언트가 접속 & 메시지를 수신시 발생하는 이벤트를 처리하기 위한 Listener 등록

표 2-2. *ReceiveActionListener* 인터페이스

메소드	설 명
void clientConnected(ReceiveEvent event)	클라이언트가 접속할 때 호출됨
void messageReceived(ReceiveEvent event)	메시지가 수신될 때 호출됨

표 2-3. Message 클래스

메소드	설 명
void setState(boolean state)	전송 시작, 중지 설정
boolean getState()	전송 시작, 중지 설정 획득
void setSenderIP(String ip)	전송측 IP 설정
String getSenderIP()	전송측 IP 획득
void setReceiverIP(String ip)	수신측 IP 설정
String getReceiverIP()	수신측 IP 획득
void setPort(int port)	Port 설정
int getPort()	Port 획득

DOClient는 DOServer에서 원격으로 호출될 원격 메소드들을 위와 같이 DOClientInterface 원격 인터페이스에 정의하고 이를 상속하여 구현한다.

(4) DOC 패키지의 구성

DOC 패키지의 클래스들은 서버 쪽의 원격 인터페이스인 *DOServerInterface*와 이를 구현한 DOServer 클래스, 클라이언트 쪽의 원격 인터페이스인 *DOClientInterface*와 이를 구현한 DOClient 클래스, DOClient가 DOServer에 접속할 때와 메시지를 받을 때 발생하는 이벤트를 처리하기 위한 *ReceiveActionListener* 인터페이스, 그리고 서버와 클라이언트 사이에 전달되는 메시지를 구현한 Message 클래스로 구성된다. 그 외 접속과 메시지 수신시 발생하는 이벤트를 구현한 ReceiveEvent 클래스는 DOC 패키지에서 내부적으로 사용하는 클래스로써 애플리케이션 개발을 위한 메소드를 제공하지 않는다. 따라서 DOC 패키지에서 이 패키지의 사용자에게 제공하는 클래스와 메소드는 아래 표 2와 같다.

DOC 패키지의 사용자는 이 패키지를 이용한 애플리케이션의 구동을 위해 접속 관리와 메시지 중계에 사용하는 DOServer를 시작할 필요가 있으며 DOServer의 메소드는 애플리케이션의 개발에 사용할 필요가 없는 DOC 패키지의 내부 클래스이다. *ReceiveActionListener* 인터페이스는 DOClient가 DOServer에 접속할 때와 메시지를 받을 때 발생하는 이벤트를 처리하기 위한 이벤트 리스너이다. 이 인터페이스의 *clientConntcted()*와 *messageReceived()* 메소드는 구현이 되어 있지 않으므로 이를 사용하기 위해서 사용자는 상속을 받아 구현하여야 한다.

3.3 LS (Live Streaming) 패키지

라이브 스트리밍을 하기 위해 설계한 LS 패키지는 JMF를 사용하여 전송에 필요한 멀티미디어의 생

성과 전송, 수신 및 재생을 담당한다. LS 패키지의 주요 구성 요소는 전송자와 수신자 두 개의 객체로 구분되며 이들 사이의 데이터 전송은 RTP를 통하여 Peer-to-peer로 이루어진다. LS 패키지는 JMF의 복잡한 API를 확장하여 라이브 스트리밍을 위해 간략화 시킨 클래스와 메소드를 제공한다. 실제적인 미디어 스트리밍을 담당하는 LS 패키지는 본 논문에서 제안하는 DLS 패키지에서 사용하는 것 이외에 라이브 스트리밍을 위한 애플리케이션 개발에 적용이 가능하다.

(1) 라이브 스트리밍 절차

라이브 스트리밍은 멀티미디어를 데이터를 생성하고 이를 전송하는 전송자와 데이터를 수신하여 재생하는 수신자가 필요하다. 전송자는 먼저 전송할 멀티미디어 데이터를 캡쳐 장치 또는 로컬 파일로부터 생성하여 데이터 전송 준비를 하고 RTP를 사용하여 실제 라이브 스트리밍을 수행한다. 수신자는 멀티미디어 데이터를 수신하기 이전에 수신 준비를 마쳐야 한다. 수신 준비를 마친 수신자는 대기를 하게 되고, 데이터가 수신되면 이를 받아 재생 한다 (그림 8 참조).

이 같은 LS 패키지의 라이브 스트리밍 절차는 그림 4 아래의 **미디어 스트리밍 시작** 절차 설명에서 *step 6*~*step 12*에 걸쳐 보여주고 있다.

(2) 멀티캐스트 전송 지원

멀티캐스트를 지원하지 않는 네트워크 환경에서 멀티미디어 데이터 1:N 라이브 스트리밍을 하기 위해 안정적으로 지원되고 있는 유니캐스트를 이용하여 동일한 데이터를 여러 번 전송함으로써 멀티캐스트 효과를 가지도록 할 수 있다. LS 패키지는 JMF의 DataSource 복제 기능을 사용하여 DataSource를 수신자의 수만큼 복제하고 유니캐스트를 사용하여 각각의 수신자에게 전송하여 1:N 전송을 수행하게 된다.

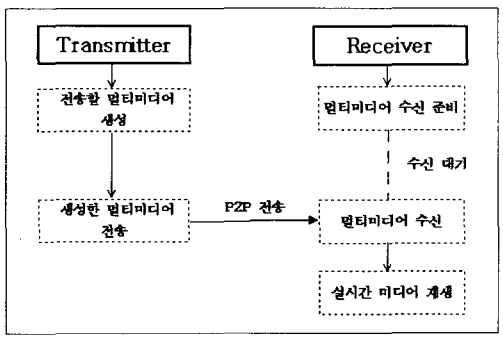


그림 8. 라이브 스트리밍 절차

서버를 이용한 채팅과 유사하게 동일한 데이터를 복사하여 다수의 수신자에게 전송하지만 JMF에서 복제한 DataSource는 원본이 사용중이여만 복제된 DataSource도 전송되어 재생에 사용 가능하다.

```
if(ds == null) {
    ds = createDataSource(ds);
    ds = Manager.createCloneableDataSource(ds);
    return ds;
}
ds = Manager.createCloneableDataSource(ds);
clonedDS = ((SourceCloneable)ds).createClone();
return cloneDS;
```

위와 같이 가장 처음 생성된 DataSource는 복제에 사용될 수 있는 DataSource로 변환되어 반환되고 그 후부터 복제된 DataSource가 반환된다. 최초 반환된 DataSource를 먼저 사용하고 그 후 반환되는 DataSource들은 그 수만큼의 라이브 스트리밍에 사용하는 MediaTransmitter 클래스의 객체를 생성하여 각각의 수신자들에게 전송한다.

그림 9의 (a)는 LS 패키지를 구성하는 클래스 중 DataSource의 생성을 담당하고 있는 DataSource-Producer에서 DataSource를 복제하고 각각의 복제된 DataSource의 전송을 위한 MediaTransmitter을 생성, 수신측에게 전송하는 것을 보여준다. 그림 9의 (b)는 이와 반대로 DataSource의 수신과 재생에 사용하는 MediaReceiver를 전송되는 DataSource의 수만큼 객체를 생성하여 수신과 재생에 사용하는 것을 보여준다. LS 패키지의 1:N 전송과 N:1 수신을 조합하여 그림 10과 같이 N:N 라이브 스트리밍을 할 수 있다.

LS 패키지의 MediaTransmitter 클래스와 MediaReceiver 클래스를 사용하여 각각 1:N 전송과 N:1 수신을 지원하기 위해서는 전송되는 수만큼의 DataSource와 MediaTransmitter 객체를 생성해서 사용해야 하고, MediaReceiver 역시 수신되는 수만큼의 객체를 생성하여야 한다. 본 논문에서 LS 패키지를 사용하는 DLS 패키지는 멀티캐스트를 지원하

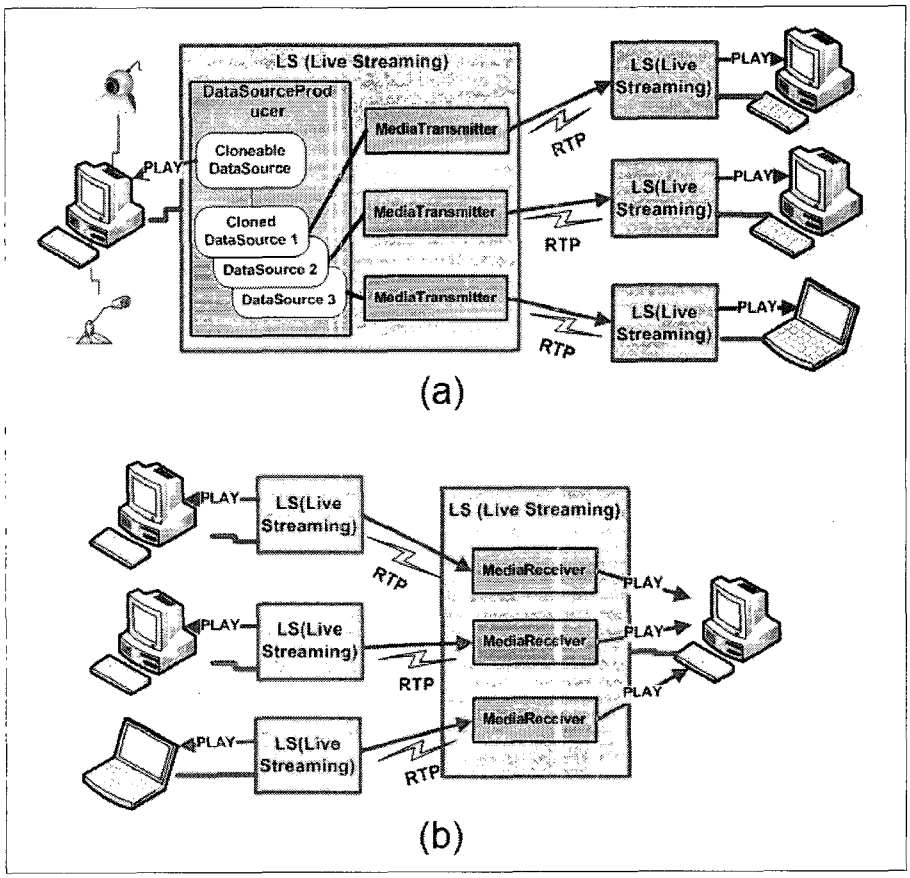


그림 9. 1:N/N:1 라이브 스트리밍

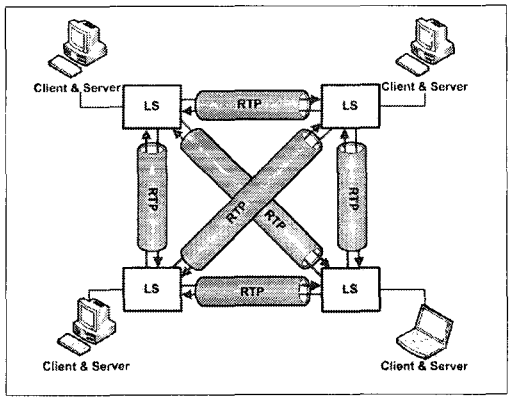


그림 10. N:N 라이브 스트리밍

기 위해 DataSource의 복제와 MediaTransmitter를 생성하고 미디어 스트리밍을 시작하는 표 1-1의 start() 메소드를 제공한다. DLS 패키지의 사용자는 start() 메소드를 N의 수만큼 호출함으로써 1:N의 전송을 수행할 수 있다.

(3) LS 패키지의 구성

LS 패키지의 클래스들은 표 3과 같이 3개의 클래스로 구성된다. LS는 캡처 장치 또는 로컬 파일로부터 멀티미디어 데이터를 획득하여 DataSource를 생성하고 수신자의 수만큼 DataSource를 복제하여 반환하는 메소드를 가진 DataSourceProducer 클래스, 생성된 DataSource에서 SendStream을 만들어 전송하기 위한 MediaTransmitter 클래스, 그리고 전송되는 스트림을 수신 받아 DataSource를 생성하고 이를 재생하는 MediaReceiver 클래스로 구성된다. 각 클래스에서 제공하는 메소드는 아래 표 3과 같다.

DataSourceProducer 클래스의 getDataSource() 메소드는 인자로 받는 option과 path를 사용하여 전송과 재생에 필요한 DataSource를 생성, 복제하여 반환한다. 첫 번째 인자 option에는 DataSource-Producer 클래스에 final static 변수로 선언된 AUDIO, VIDEO, OTHER가 사용된다. 두 번째 인자

표 3-1. DataSourceProducer 클래스

필드 & 메소드	설 명
final static int VIDEO	비디오 캡쳐 장치
final static int AUDIO	오디오 캡쳐 장치
final static int OTHER	파일
static DataSourceProducer getInstance()	DataSourceProducer 클래스의 객체를 생성해서 반환
DataSource getDataSource(int option, String path)	option에 적합한 DataSource를 생성하고 복제시켜 반환, path는 파일의 경로

표 3-2. MediaTransmitter 클래스

생성자 & 메소드	설 명
MediaTransmitter(String ip, int port, DataSource source)	수신자의 IP, port 그리고 전송할 멀티미디어의 DataSource를 받아 객체 생성
void start()	전송 데이터를 생성, 전송 준비를 하고 전송 시작
void stop()	전송 중지

표 3-3. MediaReceiver 클래스

생성자 & 메소드	설 명
MediaReceiver(String ip, int port, JPanel panel)	전송자의 IP, port 그리고 멀티미디어 재생을 위한 JPanel을 받아 객체 생성
void start()	수신 준비를 하고 수신대기
void stop()	수신 중지

인 path는 option이 OTHER일 때에만 사용하며 로컬 파일에서 DataSource를 생성할 때 필요한 파일의 경로를 지정할 때 쓰인다. 그 외의 option일 때는 null 값을 넘겨준다. MediaTransmitter 클래스의 start() 메소드는 전송에 필요한 모든 준비를 하고 RTP를 이용하여 수신자에게 전송을 한다. MeidaReceiver 클래스의 start() 메소드는 전송되는 멀티미디어 데이터 스트림의 수신을 준비하며 준비가 끝나면 수신 대기 상태로 들어가게 한다. 이 수신 대기 상태에서 데이터를 받으면 이벤트가 발생하여 실제 수신을 하며 이 멀티미디어 데이터를 JPanel에 재생한다.

사용자가 LS 패키지만을 이용하여 1:N의 멀티캐스트를 지원하는 애플리케이션을 개발하기 위해서는 DataSource를 복제하는 DataSourceProducer를 생성하여 N의 수만큼 dataSource를 복제하고, N의 수만큼 MediaTransmitter를 생성해서 사용해야 한다. 수신을 담당하는 MediaReceiver도 전송되는 데이터의 수만큼 생성하여 사용해야 한다.

3.4 DLS 패키지 검증

분산 환경에서 멀티미디어 라이브 스트리밍을 위해 제안한 DLS 패키지를 검증하기 위해 이 절에서는 설계한 DLS 패키지의 API를 사용하여 프로토타입의 애플리케이션을 구현하여 실험한 결과를 설명한다. 실험을 위한 환경은 분산 객체들을 제어와 메시지 중계를 위한 서버 (DOServer)와 DLS 패키지에서 제공하는 API를 사용하여 구현한 애플리케이션을 설치한 여러 대의 클라이언트로 구성된다 (그림 11 참조).

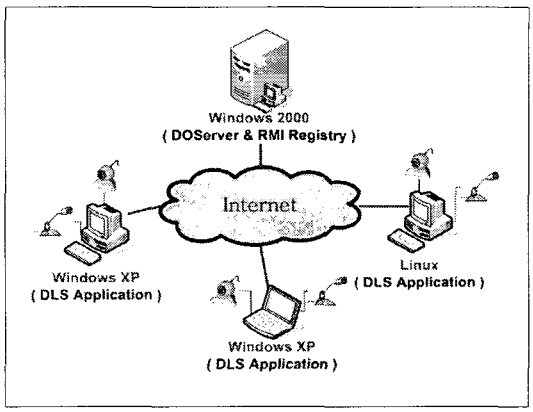


그림 11. 실험 환경

DLS 패키지를 사용한 애플리케이션의 플랫폼 독립성을 실험하기 위해 Windows 2000/XP와 Linux (한컴 리눅스 3.1)를 운영체제로 사용하는 펜티엄(Pentium)급 PC를 대상으로 분산 환경을 구축하였다. 서버에는 분산 객체 환경 구축에 필요한 JDK 1.4.2를 설치하여 분산 객체의 제어와 메시지 중계에 사용되는 DOC 패키지의 DOServer를 구동하였으며, 클라이언트에도 역시 JDK 1.4.2를 설치하고 멀티미디어 재생과 전송에 필요한 JMF 2.1.1e를 추가로 설치하여 DLS 패키지의 API를 사용하여 구현한 프로토타입 애플리케이션을 실행시켰다. 원격 객체의 참조자를 저장하는 RMI Registry는 사용자의 편의를 위해 DOServer에 포함되어 구현되어 있다.

클라이언트는 서버에 접속하여 이미 접속한 다른 클라이언트의 IP를 받아와 연결 설정을 하고 다른 클라이언트의 IP 정보를 입력받아 Peer-to-peer로 상대에게 멀티미디어 라이브 스트리밍을 한다. 1:N 전송의 실험을 위해 연결된 또 다른 클라이언트의 IP를 입력한 결과 위 그림 12와 같이 멀티미디어 데이터가 복제되어 다른 클라이언트에게도 라이브 스트리밍이 정상적으로 이루어져 재생되는 것을 볼 수 있었다.

결과를 통하여 DLS 패키지를 사용하여 구현한 애플리케이션이 분산 객체의 제어와 실제 멀티미디어 데이터의 전송을 분리하여 원활한 동작을 수행하고 플랫폼에 관계없이 동작함을 확인할 수 있었다. 이와 같이 DLS 패키지에서 제공하는 간단한 API를 사용하면 프로그램의 코드 작성이 간단해지고 분산 환경의 구축과 라이브 스트리밍에 관한 기반 지식이 없더라도 분산 환경에서 1:N의 멀티미디어 라이브 스트리밍을 지원하는 애플리케이션을 빠른 시간에 개발할 수 있다.

그림 12. DLS 패키지를 사용한 애플리케이션의 화면

다만 DLS 패키지를 사용하여 구현된 애플리케이션은 수신측의 클라이언트가 증가함에 따라 그 수만큼의 데이터를 복제하여 여러 번 전송하기 때문에 처리 속도가 느려지는 것을 볼 수 있었다. 이는 현재 고정된 데이터 포맷을 네트워크의 대역폭에 적절하게 조정하고 높은 사양의 시스템을 사용하여 어느 정도 해소할 수 있을 것이다. 그러나 여기에도 한계가 있으므로 멀티미디어 라이브 스트리밍에 적합한 멀티캐스트 응용 기술을 적용하여 최적화하는 연구가 필요하다.

4. 결 론

컴퓨팅 기술과 초고속 통신 기술의 발달은 분산 객체 기술의 발전을 가져왔다. 그러나 분산 객체 기술은 네트워크에 분산되어 있는 객체들 간의 간단한 객체 통신 방법을 제공하여 분산 객체들의 관리, 상호 운용성 등의 문제를 해결하는데 적합하지만 멀티미디어 객체의 라이브 스트리밍 기능이 결여되어 있으며 플랫폼이 다른 기종간의 통신에 적용하는데 어려움을 가지고 있다.

본 논문에서는 분산 환경에서 상이한 플랫폼에 존재하는 객체들 간의 라이브 스트리밍을 지원하기 위해서 플랫폼에 독립성을 제공하는 자바 기반의 RMI와 JMF를 확장하고 이들이 제공하는 복잡한 API를 분산 객체 라이브 스트리밍을 응용한 애플리케이션의 개발에 적합하게 단순화한 DLS 패키지를 제안했다. DLS 패키지는 하부단의 DOC 패키지와 LS 패키지를 이용하여 설계된다. 분산 객체의 관리, 상호 운용성 등을 해결하기 위해 RMI를 확장하여 설계한 DOC 패키지는 원격 호출을 통하여 분산 객체들을 제어하고, JMF를 라이브 스트리밍에 사용하기 위해 설계한 LS 패키지를 이용하여 멀티미디어 데이터를 복제, 유니캐스트로 다수의 수신자에게 라이브 스트리밍을 하여 멀티캐스트를 지원하지 않는 네트워크 환경에서도 1:N의 라이브 스트리밍을 지원할 수 있게 되었다. 그리고 두 패키지에서 제공하는 클래스와 메소드들을 통합시켜 단순화시킨 DLS 패키지의 클래스와 메소드를 사용하여 개발된 애플리케이션은 자바의 장점인 플랫폼의 독립성을 제공하며 개발에 들어가는 시간과 비용을 절감할 수 있다. 또한 서버

를 통한 분산 객체의 제어와 Peer-to-peer로 이루어지는 라이브 스트리밍이 분리되어 서버로의 미디어 서비스 집중을 해소할 수 있다.

제안한 DLS 패키지는 라이브 스트리밍에 사용될 멀티미디어 데이터의 포맷 고정되어 있어 DLS 패키지가 제공하는 클래스와 메소드를 사용하여 개발된 애플리케이션이 네트워크의 대역폭과 시스템 사양에 따라 원활한 동작을 보장하지 못 할 수도 있다. 따라서 네트워크 대역폭에 적합한 멀티미디어 데이터의 포맷을 변경할 수 있도록 해야 하며, 멀티미디어 1:N 전송에 적합한 멀티캐스트 응용 기술을 적용하여 최적화하는 연구가 필요하다. 향후 이 논문의 연구 결과는 화상 회의에 응용하거나 화이트보드 또는 채팅 등의 다양한 기능을 추가하여 화상 교육과 같은 라이브 스트리밍 분야에 적용할 수 있을 것이다.

참 고 문 헌

[1] S. Deering, "Host Extensions for IP Multicasting," Internet RFC 1112, 1989.

[2] Sun Microsystems Inc., *Java Remote Method Invocation*, http://java.sun.com/products/jdk/rmi.

[3] Sun Microsystems Inc., *Java Media Framework API*, http://java.sun.com/products/java-media/jmf.

[4] 김만수, 정목동, "CORBA/JMF 기반 오디오/비디오 스트림 시스템의 설계 및 구현," *멀티미디어학회논문지*, 제4권, 제4호, pp. 297-305, 2001.

[5] R. C. Morin, *COM/DCOM Unleashed*, SAMS, 1996.

[6] J. Pritchard, *COM and CORBA Side by Side*, Addison Wesley, 1997.

[7] G. Lu, *Communication and Computing for Distributed Multimedia Systems*, Artech House, Boston. London, 1996.

[8] T. H. Yun, J. Y. Kong, and J. W. Hong, "A CORBA-based Distributed Multimedia System," *Proc. of 1997 Pacific Workshop on Distributed Multimedia Systems*, pp. 1-8, 1997.

[9] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," *ACM SIGMETRICS '00*, pp. 1-12, 2000.

[10] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," *USITS 2001*, pp. 49-60, 2001.

[11] J. Jannotti, D. K. Gifford, K. L. Johnson, M. Frans Kaashoek, and J. W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network," *USENIX Symposium on Operating Systems Design and Implementation*, pp. 197-212, 2000.

[12] RFC 1889 RTP: A Transport Protocol for Real-Time Applications, 1996.

[13] RFC 1890 RTCP: RTP Profile for Audio and Video conferences with Minimal Control, 1996.

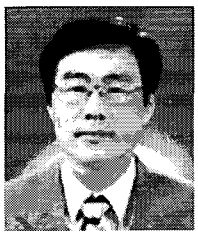
[14] M. Fowler and K. Scott, *UML Distilled 3rd*, Addison Wesley, 2003.

서 봉 근

2004년 2월 안동대학교 컴퓨터공학과 공학사

2004년 3월~현재 안동대학교 대학원 컴퓨터공학과 석사과정

관심분야 : Media Framework, 분산 컴퓨팅, 객체지향 분석/설계/프로그래밍



김 윤 호

1983년 2월 경북대학교 전자공학과 공학사

1993년 2월 경북대학교 대학원 컴퓨터공학과 공학석사

1997년 2월 경북대학교 대학원 컴퓨터공학과 공학박사

1997년 8월~현재 안동대학교 전자정보산업학부 부교수

관심분야 : 인터넷 컴퓨팅, 객체지향 분석/설계/프로그래밍, 분산객체시스템, 병렬처리