분산멀티미디어 원격제어 시스템 설계

이민경*, 조동섭
*이화여자대학교 컴퓨터공학과
e-mail : mg0426@gmail.com

A Design of Distributed Multimedia Remote control System

Min-Kyung Lee*, Dong-Sub Cho
*Dept. of Computer Engineering, Ewha Womans University

요 약

컴퓨터와 통신 기술의 발전은 네트워크에 분산 되어 있는 시스템 사이의 멀티미디어 서비스에 대한 관심이 높아지게 하고 있다. 비디오와 오디오, 영상이 합쳐진 멀티미디어데이터는 데이터는 그 특성상 용량이 크다. 멀티미디어데이터에 대한 효율적인 관리를 위해서는 중앙의 관리자가 분산된 환경의 다수의 사용자를 쉽고 빠르게 제어할 수 있어야 한다. 한정된 대역폭에서 보다 빠르고 효율적인 제어를 위해 소켓을 통한 명령의 전달을 이용한다. 본 논문에서는 분산된 환경의 멀티미디어 시스템을 중앙의 사용자가 좀 더 효율적으로 관리하기 위해 소켓통신을 이용한 분산멀티미디어 원격제어 시스템을 설계한다.

1. 서론

인터넷의 빠른 발달은 분산 환경에서 멀티미디어 서비스의 활성화를 촉진함에 따라 다양한 멀티미디어 통신기술의 개발이 필요 시 되고 있다. 물리적으로 분리되어 있는 서버에 저장되어 있는 멀티미디어 데이터를 전송하고 재생하기 위해 '저장 후 재생(Store-and-display)'방식과 '스트리밍(Streaming)'방식이 대표적이다. 멀티미디어 데이터는 데이터의 프레임을 블록 단위로 나누어 전송, 재생되며 전송 후 실행 시 프레임으로 다시 조합되며 이용할 수 있게 한다. '저장 후 재생'방식은 서버로부터 클라이언트 측으로 멀티미디어 데이터의 모든 블록이 완벽히 다운로드 된 후에 데이터를 이용할 수 있다.'스트리밍'방식은 완벽한 데이터의 전송을 기다리지 않고 블록이 도착하는 순서대로 바로 이용할 수 있다. 기존의 방법은 공통적으로 데이터 블록이 직접 전송되었다.

인터넷은 보장형(Guarantee)이 아닌 최선형(Best of Quality)서비스를 제공한다. 전송경로상의 라우터와 링크의 혼잡상태에 따라 달라지는 서비스를 제공함으로써 지연시간을 예측할 수 없고 지터나 데이터의 손실도 유발한다.

분산환경에서의 멀티미디어를 좀 더 빠르게 이용하고 효율적으로 관리하기 위해서는 네트워크를 통해 오고 가는 데이터의 양을 최소화 하고 중앙의 사용자가 각 분산지의 클라이언트들을 쉽게 제어할 수 있어야 한다. 그러기 위해 필요한 기술이 원격제어이다.

인터넷의 발달은 먼 거리에 있는 컴퓨터를 자신의 컴퓨터를 이용해 제어할 수 있는 원격제어 기능을 가능하게 하였다. 원격제어 기술의 발달로 제어자는 원격지까지 이동하는 수고를 하지 않고도 자신의 편의에 따른 장소에서 쉽게 원격지의 컴퓨터에 접근할 수 있다.

본 논문의 2 절에서는 원격제어를 위한 통신 방법을 알아보고 3 절에서는 분산멀티미디어 원격제어 시스템을 설계, 4 절에서는 본 논문의 결론 및 앞으로의 연구계획을 기술한다.

2. 관련연구

• CORBA(Common Object Request Broker Architecture)

CORBA 는 객체지향, 분산 애플리케이션들을 개발하기 위한 프레임워크를 정의하는 것으로 이 구조는 분산 애플리케이션이 마치 하나의 언어로 구현되어, 하나의 컴퓨터에서 상호 작용하는 것처럼 네트워크 프로그래밍을 훨씬 쉽게 이용할 수 있도록 해준다. 서로 다른 운영체제와 네트워크, 다양한 언어로 작성된 객체일 때에도 객체 수준의 통신을 가능하게 해준다. CORBA 는 여러 회사와 조직들로 구성된 OMG(Object Management Group) 컨소시엄에서 개발하고 발전시켜나가고 있는 개방형 표준으로 객체들 간의 통신을 가능하게 해주는 객체지향 기반의 미들웨어가 필요하다.

• RMI(Remote Method Invocation)

원격자바객체의 메소드를 네트워크를 경유하여 로컬자바객체의 메소드처럼 호출할 수 있는 기법이다. 자바프로그램을 실행하는 환경, 자바가상기계(Java Virtual Machine)에서 쉽게 이용할 수 있다. JDK1.1 에서부터 포함되었고 자바로 이루어진 개발환경에서는 자바의 특징들 이식성, 재사용성 등의 다양한 이점을 이용할 수 있으나 대규모시스템에서의 인프라스트럭처(infrastructure)에 필요한 요소를 만족시키는 데는 어려움이 따른다.

• 소켓(socket)

소켓은 애플리케이션이 네트워크에 플러그인(plug-in)하여 동일한 네트워크에 플러그인 된 다른 애플리케이션과 통신할 수 있도록 한다.

Socket()함수의 인자로 TCP, UDP 프로토콜이 대표적이다. TCP 는 데이터의 유실이 없는 신뢰적 전송 프로토콜이고 UDP 는 데이터의 유실을 책임질 수 없는 비신뢰적 전송 프로토콜이다.

• 차이점

CORBA 와 RMI 는 분산환경에서의 원격호출에 대한 ORB(Object Request Broker)로 많은 유사점이 있다. CORBA 는 객체지향, 분산애플리케이션 개발을 하기 위한 프레임워크를 정의하는 것으로 마치 하나의 언어로 구현되어, 하나의 컴퓨터에서 상호작용하는 것처럼 네트워크 프로그래밍을 쉽게 하고 분산환경 트랜잭션 서비스와 같은 인프라구조에 필수요소를 만족시키기 위한 다양한 서비스를 제공한다.
자바의 RMI 는 객체직렬화, 다운로드 가능한 객체실행, 자바인터페이스 등의 자바언어의 특성에 의존하는 프로그램으로, 자바언어가 가지고 있는 상속이나 재사용성 등의 다양한 이점을 이용할 수 있다. 또한 미들웨어의 구매 없이 JVM 에 RMI 구현 환경이 구현되어 있어 누구나 쉽게 사용할 수 있다. 그렇지만 대규모시스템에서의 인프라구조에 필요한 요소를 만족하는 데는 한계가 있다. 또한 객체단위 직렬화를 통해 데이터를 넘겨주고, 받는 쪽에서도 직렬화 된 스트림을 복구해서 다시 객체로 변환해야 하므로 오버헤드가 크고 전송속도도 느리다.
소켓의 경우 사용자가 직접 프로그래밍 할 수 있기 때문에 필요한 데이터에 대해서만 넘겨줄 수 있고, 다양한 환경에서의 전송이 가능하지만 프로그래머가 직접 구현해야 한다는 어려움이 있다.

3. 분산멀티미디어 원격제어 시스템

서울에 있는 보호자는 자신의 서울 집에 있는 컴퓨터를 통해 부산에 있는 환자의 병실 컴퓨터를 제어할 수 있다. 보호자는 부산에 있는 환자의 병실에 가지 않고도 환자의 컴퓨터를 제어해 환자의 필요에 따라 실시간으로 영화나 음악파일 등을 바꾸어 실행시켜 줄 수 있게 한다.

본 논문에서 제안하는 분산멀티미디어 원격제어 시스템을 크게 3 가지 일을 수행한다.

• Play / Stop

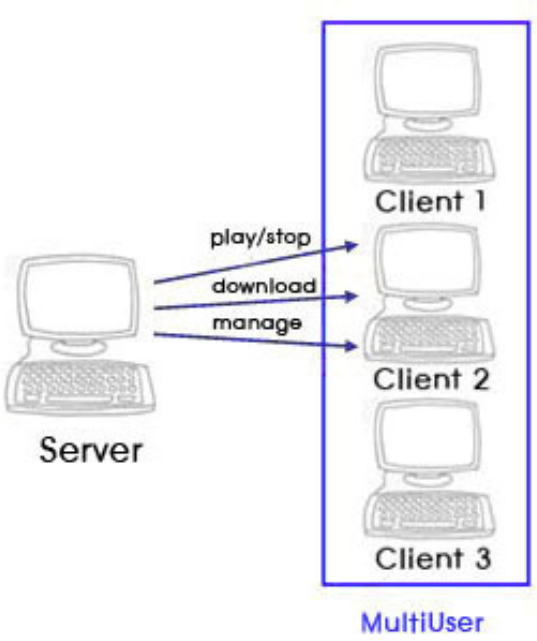
클라이언트 측으로부터 요청된 데이터가 클라이언트 측 DB 에 있으면 서버와의 연결을 형성하고 서버의 play()나 stop()을 실행할 수 있다.

• Download

클라이언트에 의해 요청된 데이터가 클라이언트 자신에 없을 때에는 서버는 자신의 DB 에 있는 데이터를 클라이언트에서 이용할 수 있게 한다.

• File manage

클라이언트가 서버로부터 데이터를 다운로드 받아 이용하고 사용이 끝나면 서버는 클라이언트 측에 전달된 데이터를 회수해야 한다. 뿐만 아니라, 사용이 끝난 자원들을 다음 요청에 사용할 수 있도록 회수해야 한다.



(그림 1) 분산멀티미디어 원격제어시스템 기능

본 논문에서 제안하는 시스템은 데이터의 직접적인 전송이 이루어 지지 않고, 소켓통신을 이용해 명령을 전송함으로써 실행할 수 있다. 서버에서 보내는 명령은 소켓을 통한 한번의 전송으로 같은 네트워크상의 모든 클라이언트들에 전달된다.

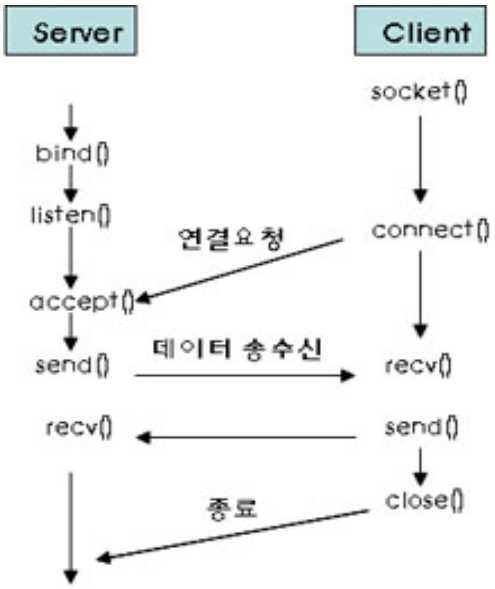
소켓은 패킷과 대비되는 개념으로 데이터를 보내는 클라이언트와 받는 서버가 패킷에 비하여 명확하게 구성되고 데이터의 서버가 패킷에 비하여 명확하게 구성된다. 또한 데이터의 유실도 작고 전송속도도 빠르다. socket()을 호출하면 소켓 기술자를 얻게 되고 send(), recv()등의 소켓에 관련된 함수를 이용할 수 있다. 소켓의 사용은 인터넷뿐만 아니라 UNIX 등의 서로 다른 환경에서 이용할 수 있다.

기존의 분산멀티미디어 시스템은 신뢰성을 보장을 보장하지는 않지만 속도가 빠른 UDP 프로토콜을 이용해왔다. 멀티미디어 데이터는 스트리밍 서비스와 같이 소리나 화면이 잠시 끊어져도 문제가 되지 않기 때문에 실시간 데이터 전송 서비스에 사용되었으나 인터넷의 발달과 사용자의 요구 증가로 QoS(Quality of Service) 관점에서 문제점이 야기되고 있다.
TCP 기반의 네트워크 환경에서는 서버와 클라이언트가 연결상태를 계속 유지하고 있기 때문에 클라이언트의 수 만큼 소켓을 생성해야 한다. 하지만 UDP 서버에서는 연결상태가 아니기 때문에 한 개의

소켓을 생성, 소켓 하나로 여러 클라이언트들과 데이터를 송, 수신 할 수 있다. UDP 가 TCP 보다 빠른 이유도 데이터의 송, 수신 전후에 거치는 과정과 전송과정에서 여러 단계가 생략되기 때문이다.

본 논문에서 제안된 시스템은 데이터의 직접적 전송이 아닌 명령의 전송으로 TCP 프로토콜을 사용한 제어가 가능하다. Data 의 유실을 막을 수 있지만 기존 시스템의 데이터 블록의 직접 전송이 아닌 소켓을 통한 명령의 전달이기 때문에 보다 빠른 실행이 가능하게 한다.

클라이언트와 서버가 TCP 소켓을 만들고 서로 연결한 다음 데이터를 송수신하고 소켓을 종료하는 절차는 다음과 같다.



(그림 2) TCP 소켓프로그래밍의 절차

```
server
svrsock = socket(AF_INET, SOCK_STREAM)
svrsock.bind(HOST, PORT)
svrsock.listen(1)
conn, addr = svrsock.accept()
svrsock.recv(bufsize) svrsock.send(string)
svrsock.close()
```

<표 1> Server 측 함수

서버는 소켓을 생성하고 HOST 컴퓨터의 PORT 에 연결한 후, 한 번에 처리 할 수 있는 연결 수를 설정한다. 클라이언트로부터 소켓의 연결이 올 때까지 기다린 후 connect 로부터 소켓이 연결되면 보내고 받는 동작을 반복하면 데이터를 전송한다. 데이터의 전송이 끝나면 소켓을 종료한다.

```
client
clientsock = socket(AF_INET, SOCK_STREAM)
clientsock.connect(HOST, PORT)
clientsock.recv()
clientsock.send()
clientsock.close()
```

<표 2> Client 측 함수

클라이언트는 소켓을 생성하고 서버의 특정포트에 연결한다. 연결이 되면 connect()를 통해 연결이 되었음을 알리고 데이터를 수신하고 전송한다. 모든 데이터가 전송이 되면 소켓을 종료한다. TCP 프로토콜의 사용은 네트워크 상태를 점검하며 통신을 할 수 있다는 장점을 가진다. 중앙의 관리자는 분산지의 사용자에게 소켓을 이용한 메세지의 전송을 통해 효율적인 제어를 할 수가 있다. 메시지의 전달 상황을 파악하며 다양한 질의를 할 수 있고 통신을 위한 함수를 쉽게 보낼 수 있다. Server 측 제어자가 다양한 클라이언트에게 좀 더 정확하고 손쉬운 제어를 할 수 있다.

본 논문에서 제안된 시스템은 많은 장점을 가진다. 소켓통신을 이용하기 때문에 서로 다른 네트워크 환경과 OS 에서의 구현과 사용이 쉽다. 소켓을 프로그래머에 의해 최적화 되어 넘겨 줄 수 있고, 소켓의 특성상 CORBA 나 RMI 등보다 명령의 전송 속도가 빠르다. 클라이언트의 DB 음원을 사용할 수 있어 멀티미디어 데이터의 품질도 우수하다. 확인 절차를 통한 전송이 이루어지기 때문에 불필요한 자원의 낭비도 막을 수 있다. 또한, 사용 후 클라이언트 측으로 보내진 데이터에 대한 회수가 이루어지기 때문에 멀티미디어 데이터에 대한 무단 복제나 수정 배포 등의 저작권(copyright) 문제가 발생할 가능성이 적다.

4. 결론 및 향후 연구과제

비디오(Video)와 오디오(audio), 영상이 합쳐진 멀티미디어 데이터는 그 용량이 클 수 밖에 없다.

분산된 환경에서의 멀티미디어 데이터의 전송은 많은 오버헤드가 발생한다. 본 논문에서 제안하는 시스템은 소켓통신을 이용한 원격제어 방법으로 기존의 시스템이 가지는 문제점을 해결하고자 했다.

물리적으로 분리되어있는 시스템 간의 데이터 전송 시 데이터 블록의 직접적인 전송이 아닌 TCP 소켓을 통한 명령만을 전달함으로써, 중앙의 관리자에게 보다 간편하고 쉽게 원격지의 클라이언트를 제어할 수 있게 한다. 미들웨어가 필요한 CORBA 나 자바기반의 RMI 가 아닌 소켓을 이용함은 데이터의 전송속도가 빠르고 여러 환경에서의 사용이 가능하기 때문이다.

향후 구현을 통해 중앙의 사용자가 다수의

클라이언트를 소켓통신을 이용해 원격으로 직접 쉽고 빠르게 제어할 수 있도록 검증할 계획이다.

참고문헌

[1] [1] Satoshi Itaya, Naohiro Hayashibara, Tomoya Enokido. Makoto Takiwaqa, “Distributed Coordination for Scalable Multimedia Streaming Model,” IEEE 2006

[2] Richard M.Adler, “Distributed Coordination Models for Client/Server Computing”, IEEE Computer, pp14-22, April, 1995.

[3] Robert Orfali, Dan Harkey, “Clinet/Server Programming with Java and CORBA”-2nd, 1998

[4] 웹을 통한 원격제어 시스템 이창희, 이광재, 원영진, 류희삼, 전자공학회논문지 제 39 권 TE 편 제 3 호, 2002. 9

[5] 정기훈, “원리로 이해하는 네트워크 입문”, 정보문화사, 2005

[6 김명호, 이윤준, 정연돈, ”멀티미디어 시스템 개론” 홍릉과학출판사, 2006

[7] 한윤기, 구용완, ”CORBA-ORB, JAVA-RMI, 소켓을 이용한 그룹 통신의 구현 및 성능분석”, 2002 한국인터넷정보학회(3 권 1 호)

[8] Microsoft PRESS, “Inside Distributed COM”, 1998