분산 미디어 객체 실시간 스트리밍 기법

서봉근* · 김윤호**

A Technique for Distributed Media Object Live Streaming

Bong-Kun Seo* · Yun-Ho Kim**

요 약

본 논문에서는 RMI와 JMF를 이용하여 분산 미디어 객체 실시간 스트리밍을 하기 위한 기법을 제시한다. 즉, 분산 객체의 복잡한 제어를 RMI를 사용하여 간소화 하고 멀티캐스트를 지원하지 않는 네트워크 환경에서 JMF를 사용하여 일대다 실시간 스트리밍을 하기 위한 기법을 다룬다. 자바 기반의 RMI와 JMF를 사용하여 전송과 제어를 분리할 수 있으며 이를 응용하여 플랫폼에 독립적인 멀티미디어 서비스 애플리케이션의 개발에 사용 가능하다.

ABSTRACT

In this paper, we present a technique for distributed media object live streaming using RMI and JMF. Then, we handle technique that simplify complex control of distributed objects using RMI and do one-to-many live streaming in network environment that do not support multicast using JMF. It can separate transmission and control to use RMI and JMF based on Java. Consequently the technique is available in development of multimedia service application that is independent to platform.

키워드

분산 미디어 객체 실시간 스트리밍, RMI, JMF

Ⅰ. 서 론

네트워크에 분산되어 있는 대용량의 멀티미디어 데이터를 처리하여 실시간으로 스트리밍하기 위해서는 컴퓨팅 기술과 고속 통신 기술의 발전은 필수이다. 최근 고성능의 컴퓨팅 파워를 가진 개인 단말기의 보급과 초고속 통신 기술의 발달로 화상회의, NOD (News on Demand), VOD (Video on Demand)와 같은 멀티미디어 서비스들이 개발, 제공되고 있다. 이와 같은 멀티미디어 서비스를 개발하기 위해서는 네트워크상의 이질적인 플랫폼에 분산되어 있는 멀티미디어 객체들 간의 복잡한 제어와 상호 운용성 문제를 처리해야 하며, 멀티미디어 데이터를 IP 멀티캐스트 (Multicast) [1]로 전송할 경우 이를 전달하는 네트워크상의 장비 (라우터)중 하나라도 멀티캐스트를 지원하지 않으면 수신자는 서비스를 받을 수 없으므로 멀티캐스트를 대체하는 기술을 적용하는 것이 필요하다.

분산 객체 컴퓨팅 기술인 DCOM (Distributed Component Object Model) [2], CORBA (Common Object Request Broker Architecture) [3], RMI (Remote Method Invocation) [4] 등은 분산 환경에서 기존에 개발된 소프트웨어의 재사용, 통합 그리고 상호 운용성 문제를 해결하는데 매우 효과적이지만 멀티미디어 객체의 교환 기능 결여로 인해 멀티미디어 실시간 스트리밍에 적용이 어려워 이를 해결하기위한 다양한 대안이 연구되고 있다 [5~8].

* 안동대학교 컴퓨터공학과 석사과정
** 안동대학교 전자정보산업학부 부교수

접수일자 : 2005. 8. 4

본 논문에서는 자바 기반의 RMI를 사용하여 네트워크에 분산되어 있는 멀티미디어 객체들 간의 상호 운용성 문제를 해결하기 위한 Distributed Object Communication (이후 DOC라 부름) 패키지 (Package)를 설계한다. 그리고 자바의 빈약한 멀티미디어 지원을 보강하기 위한 자바의 확장 패키지인 JMF (Java Media Framework) [9]를 사용하여 실시간 스트리밍에 적합하게 단순화하고 하나의 멀티미디어 데이터를 복제하여 다수의 수신자에게 유니캐스트 (Unicast)로 전송할 수 있게 하는 Live Streaming (이후 LS라 부름) 패키지를 설계한 후 두 개의 패키지를 통합하여 분산 객체 실시간 스트리밍에 적합한 Distributed Live Streaming (이후 DLS라 부름) 패키지를 설계/제안한다. 자바의 RMI와 JMF를 사용하여 설계한 DLS 패키지는 플랫폼에 구애받지 않는 실시간 스트리밍 애플리케이션 개발에 응용이 가능하다.

Ⅱ. 분산 객체 실시간 스트리밍 설계

분산 환경에서 캡쳐 장치 (CAM, 마이크)를 이용하여 얻은 실시간 미디어 데이터를 네트워크를 통해 전송하기 위해서는 먼저 송신측에서 보내는 데이터를 수신측에서 언제 받아야 하는지를 알아야 한다. 만약 수신측에서 미디어 데이터의 수신을 준비하지 않은 상태에서 송신측이 데이터를 송신하거나, 송신측에서 데이터를 송신하지 않는데 수신측에서 수신 준비를 마치고 대기하고 있다면 시스템 자원의 낭비가 될 것이다. 이러한 이유로 분산 객체들의 미디어 데이터 송수신을 제어하는 것이 필요하다.

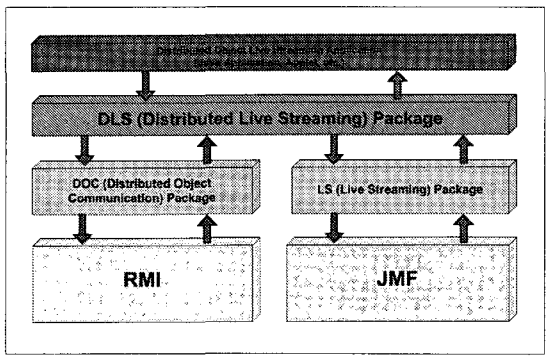

그림 1. DLS 프레임워크
Fig. 1 DLS framework

하지만 분산 객체들의 수가 증가할 경우 수많은 객체를 일반적인 통신으로 제어하는 것은 상당히 복잡해 질 수 있다. 또한 현재의 네트워크 장비들 중 멀티캐스트를 지원하지 않는 장비가 존재하므로 멀티캐스트를 사용하여 멀티미디어 데이터를 전송할 경우 수신을 하지 못하는 문제가 발생할 수 있다. 이와 같은 문제는 동일한 데이터를 수신자의 수만큼 복제하여 유니캐스트를 사용, 전송하여 해결할 수 있다.

본 논문에서 제시하는 DLS 패키지는 RMI를 사용하여 복잡한 분산 객체들의 제어를 단순화시키기 위해 설계한 DOC 패키지와 JMF의 미디어 실시간 스트리밍 기능과 멀티미디어 데이터를 수신자의 수만큼 복제하여 유니캐스트로 반복 전송하도록 설계한 LS 패키지를 통합하여 분산 객체 환경에서 미디어 실시간 스트리밍을 수행하기에 적합하고 간소화한 API를 제공한다.

(1) DOC 패키지 설계

네트워크에 분산되어 있는 객체 (이후 클라이언트로 부름)들을 제어하기 위해서는 분산 객체들의 연결 (접속)과 메시지를 제어를 담당할 객체 (이후 서버로 부름)가 필요하다. 그림 2는 클라이언트가 서버에 접속하고 제어 메시지를 다른 클라이언트에게 전송하는 과정을 간략히 표현한 것이다. 클라이언트는 먼저 서버에 접속하여 이미 접속한 클라이언트의 목록을 받아오고 목적지의 클라이언트를 선택, 서버를 통하여 제어 메시지를 전달한다. 이 모든 절차는 원격 호출을 통하여 실행한다.

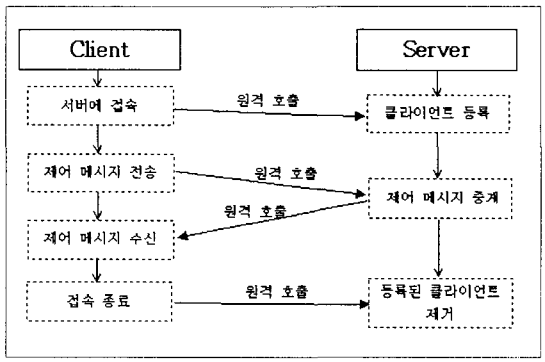

그림 2. 분산 객체의 원격 호출 절차
Fig. 2 Remote Invocation procedure of distributed objects

DOC 패키지는 그림 2와 같이 동작하는 클라이언트를 구현한 DOClient 클래스, 서버를 구현한 DOServer 클래스, 수신된 메시지와 접속한 클라이언트의 IP들을 처리하기 위한 ReceiveActionListener 인터페이스, 이벤트 클래스인 ReceiveEvent 클래스 그리고 서버와 클라이언트 사이의 메시지를 구현한 Message 클래스로 구성된다. 그 외에 DOClient 클래스와 DOServer 클래스가 상속 받는 DOClientInterface와 DOServerInterface 원격 인터페이스가 있다.

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

```
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;
}
```

*DOClientInterface*와 *DOServerInterface* 인터페이스에는 이들을 상속하는 DOClient와 DOServer 클래스에서 구현해야 할 원격 메소드들을 정의한다.

DOC 패키지에서 사용하는 DOServer 클래스는 DOClient 객체들 간의 연결 설정과 메시지 중계를 위한 메소드를 가지고 있지만 이는 응용 프로그램 개발에 사용할 필요가 없으므로 API를 제공하지 않는다.

DOClient 클래스는 접속:connect(), 접속 해제:disconnect(), 메시지 송신:sendMessage(), 수신된 메시지 획득:getMessage(), 접속한 클라이언트들의 IP 획득:getClientList(), 그리고 클라이언트가 접속할 때 또는 메시지를 수신할 때 발생하는 이벤트를 *ReceiveActionListener* 인터페이스를 구현한 클래스에게 처리를 넘기기 위한 addReceiveListener() 메소드가 정의된다.

ReceiveActionListener 인터페이스는 클라이언트의 접속과 메시지를 받을 때 발생하는 이벤트를 처리하기 위해 *clientConntcted()*와 *messageReceived()* 메소드를 가진다. 이 두 개의 메소드는 구현이 되어 있지 않으므로 이를 사용하는 응용 프로그램에서는 반드시 구현하여야만 한다.

Message 클래스는 메시지의 전송 시작과 중지 상태를 지정하는 boolean 타입의 필드인 action, 송/수신측의 IP를 저장하는 String 타입의 senderIP/receiverIP, 전송 포트(Port)를 저장하는 int 타입의 port 필드가 정의되며 이 정보들을 설정, 획득하기 위한 setter/getter 메소드들이 정의된다.

분산 객체들의 연결과 메시지 중계에 사용하기 위한 목적으로 설계한 DOC 패키지는 원격 호출을 통하여 복잡한 객체간의 통신을 간소화 했으며 실시간 스트리밍을 위한 메시지를 전송하는 것 이외에 Message 클래스를 상속하여 채팅 또는 화이트보드 등과 같은 응용 프로그램에 사용하는 메시지로 확장이 가능하므로 이를 사용하여 다양한 분산 응용 프로그램에 적용이 가능하다.

(2) LS 패키지 설계

JMF의 RTP (Real-time Transfer Protocol) [10][11]를 사용하여 실시간 스트리밍을 하고자 한다면 Transmitter (이후 전송자로 부름)는 먼저 전송할 멀티미디어 데이터를 마이크로 CAM을 통하여 실시간으로 생성하여야 한다. 그 후 전송 준비를 하고 RTP를 사용하여 실제 실시간 스트리밍을 수행한다. Receiver (이후 수신자로 부름)는 멀티미디어 데이터를 수신하기 이전에 수신 준비를 마쳐야 한다. 수신 준비를 마친 수신자는 대기를 하고 있다가 스트림이 수신되면 이를 받아 재생을 한다. RTP를 사용하여 전송자와 수신자 사이에 전송되는 데이터는 중간에 서버를 통하지 않고 Peer-to-peer로 바로 전송이 이루어진다.

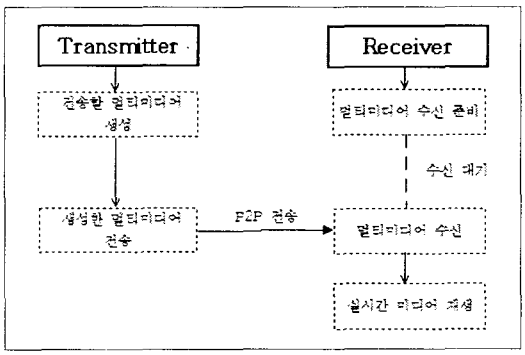

그림 3. 실시간 스트리밍 절차
Fig. 3 Procedure of Live streaming

위 그림 3과 같이 크게 전송자와 수신자로 이루어지는 LS 패키지는 캡쳐 장치로부터 멀티미디어 데이터를 획득하고 DataSource 객체를 생성하여 수신자의 수만큼 DataSource를 복제하고 반환하는 기능을 하는 DataSource Producer 클래스와 DataSource를 RTP로 실시간 전송하기 위한 MediaTransmitter 클래스, DataSource를 스트림으로 수신 받아 재생하는 MediaReceiver 클래스로 구성된다.

DataSourceProducer 클래스는 static으로 선언된 getDataSource() 메소드만을 가지고 있으며 인자로 int 타입의 option을 받아 RTP로 전송하기 위해 필요한 DataSource 객체를 생성하여 반환한다. option에는 AUDIO, VIDEO가 정의되어 있으며 모두 상수 (static final)로 DataSourceProducer 클래스 내에 선언된다. AUDIO와 VIDEO는 각각 마이크와 CAM에서 DataSource를 생성할 때 쓰인다.

getDataSource() 메소드가 반환하는 DataSource 객체는 다수의 수신자에게 전송하기 위해 필요한 복제된 DataSource이다. 가장 처음 생성된 DataSource는 CloneableDataSource로 변환되어 반환되고 그 다음부터 복제가 되어 반환된다. CloneableDataSource를 복제한 DataSource들은 CloneableDataSource가 먼저 재생을 하거나 전송되어야만 복제된 DataSource들도 재생 또는 전송되어 재생하는데 사용 가능하다. 전송에 사용하는 수만큼 복제되어 반환된 DataSource들은 그 수만큼의 MediaTransmitter 객체를 생성하여 각각의 수신자들에게 그림 4와 같이 전송한다.

그와 반대로 데이터를 수신하는 MediaReceiver 클래스는 수신할 수만큼의 객체를 생성하여 수신과 재생에 사용하여야 한다.

LS 패키지는 이와 같은 방법을 사용하여 멀티캐스트를 지원하지 않는 네트워크 환경에서도 일대다 전송을 지원하게 된다.

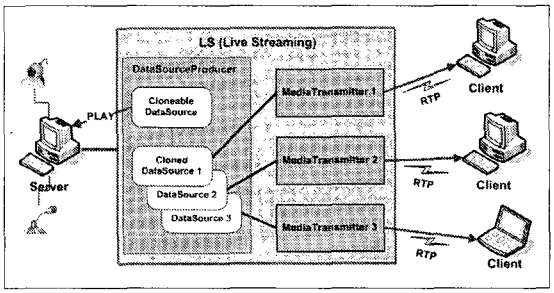

그림 4. 일대다 실시간 스트리밍
Fig. 4 One-to-many live streaming

MediaTransmitter 클래스는 전송에 사용하는 DataSource와 수신자의 IP, 전송 Port를 인자로 가지는 생성자와 전송을 시작하는 start(), 전송 중지를 위한 stop() 메소드가 정의된다. MediaTransmitter 클래스의 start() 메소드는 전송에 필요한 모든 준비를 하고 RTP를 이용하여 수신자에게 실시간으로 전송을 한다. 이 전송은 서버를 통하지 않고 수신자에게로 바로 Peer-to-peer로 전송된다.

MediaReceiver 클래스는 미디어 데이터를 전송하는 전송자의 IP, 전송 Port 그리고 수신 받은 데이터를 재생하는데 필요한 JPanel의 인스턴스를 가진 생성자를 정의한다. 그리고 수신 준비와 수신대기를 위한 start(), 수신 중지를 위한 stop() 메소드를 정의한다. MeidaReceiver 클래스의 start() 메소드는 전송되는 멀티미디어 데이터의 수신을 준비하며 준비가 끝나면 곧바로 수신 대기 상태로 들어가게 한다. 이 수신 대기 상태에서 멀티미디어 데이터의 스트림을 받으면 이벤트가 발생하여 수신을 하며 이를 재생에 필요한 DataSource 객체로 변환 후 MediaReceiver 객체를 생성할 때 생성자의 인자로 받은 JPanel 에 자동으로 미디어를 재생한다.

(3) DLS 패키지 설계

DLS 패키지는 DOC 패키지와 LS 패키지를 통합하고 단순화시켜 DistributedLiveStreaming 클래스와 *ConnectActionListener* 인터페이스, 이벤트 클래스인 ConnectEvent로 구성된다. 통합된 DLS 패키지는 분산 객체를 제어하는 부분과 실시간 스트리밍에 관련된 기능은 모두 DOC와 LS 패키지에

표 1. DistributedLiveStreaming 클래스
Table. 1 DistributedLiveStreaming class

생성자 & 메소드	설명
DistributedLiveStreaming (JPanel[] panel)	멀티미디어 재생을 위한 JPanle[]을 받아서 객체를 생성
void connect(String ip)	주어진 서버 IP로 접속
void disconnect()	접속 해제
void start(String ip, int port, int option)	수신자 IP, Port, option을 사용하여 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 2. *ConnectActionListener* 인터페이스
Table. 2 *ConnectActionListener* interface

메소드	설명
void clientConnected(ConnectEvent event)	클라이언트가 접속할 때 호출됨

서 처리하게 되므로 분산 객체들의 연결에 필요한 기능과 실시간 스트리밍의 시작과 중지를 위한 기능을 정의한 메소드만을 정의한다.

DistributedLiveStreaming 클래스의 connect() 와 disconnect() 메소드는 하부 DOC 패키지의 DOClient 클래스가 가진 동명 메소드를 호출하여 접속과 접속해제를 수행한다. start() 메소드는 멀티미디어 실시간 전송에 앞서 DOClient 클래스의 sendMessage() 메소드를 호출하여 수신자에게 메시지를 보내 수신 준비를 시킨다. 그 후 인자로 받은 option 값으로 LS 패키지의 DataSource Producer.getDataSource() 메소드를 호출하여 전송에 필요한 DataSource를 생성하고 MediaTransmitter 클래스의 start() 메소드를 호출하여 실시간 전송을 시작한다. stop() 메소드도 이와 유사한 방법으로 하부의 DOC 패키지와 LS 패키지의 메소드 호출을 통해 전송 중지 메시지를 수신측에 보내고 전송을 중지한다.

ConnectActionListener 인터페이스의 *clientConnected()* 메소드는 클라이언트의 접속 이벤트를 처리한다. 이는 다른 클라이언트의 접속과 접속 해제를 할 때 곧바로 현재 전송에 필요한 클라이언트의 리스트를 업데이트하기 위해 필요하다.

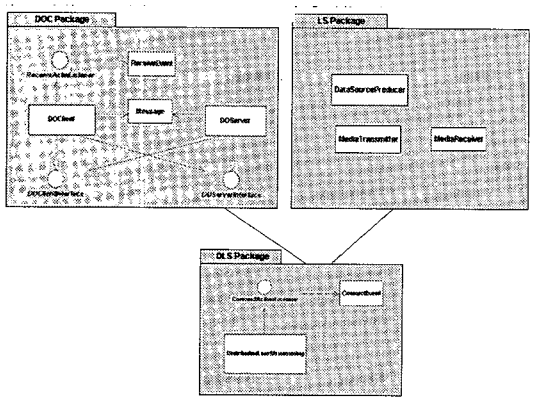

그림 5. 전체 패키지 구조
Fig. 5 Structure of entire package

DLS 패키지는 하부에서 분산 객체 실시간 스트리밍에 필요한 기능을 대부분 수행하게 하여 분산 객체 실시간 스트리밍을 응용한 애플리케이션을 개발할 때 분산 객체의 제어와 실시간 스트리밍에 대하여 고려하지 않아도 된다. 이러한 장점은 이를 응용한 애플리케이션 개발에 드는 비용과 시간을 절감할 수 있게 한다. 앞서 설계한 DOC와 LS, 그리고 DLS의 전체 패키지 구조는 그림 5와 같은 구조를 가진다.

Ⅲ. 결 론

본 논문에서는 분산 환경에서 실시간 스트리밍을 지원하기 위해 DLS 패키지를 제안 하였다. 제안한 DLS 패키지는 분산 객체 실시간 스트리밍을 위한 애플리케이션의 개발에 적합하도록 간소화한 API를 제시한다.

분산 객체의 제어와 상호 운용을 위해 RMI를 확장하여 설계한 DOC 패키지는 원격 호출을 통하여 분산 객체들을 제어하고, JMF를 실시간 스트리밍에 사용하기 위해 설계한 LS 패키지를 이용하여 멀티미디어 데이터를 복제, 유니캐스트로 다수의 수신자에게 실시간 스트리밍을 하여 멀티캐스트를 지원하지 않는 네트워크 환경에서도 일대다의 실시간 스트리밍을 지원할 수 있게 되었다.

DOC와 LS 패키지를 통합한 DLS 패키지는 간단한 API를 제공하므로 쉽고 빠른 응용 애플리케이션의 개발에 유용하게 이용할 수 있을 것이다. 또한 자바의 장점인 플랫폼의 독립성을 제공하며 서버를 통한 분산 객체의 제어와 Peer-to-peer로 이루어지는 실시간 스트리밍이 분리되어 서버로의 미디어 서비스 집중을 해소할 수 있다.

설계된 DOC, LS, DLS에는 실시간 스트리밍에 사용될 멀티미디어 데이터의 포맷을 고정하여 네트워크의 대역폭에 따라 원활한 동작을 보장하지 못 할 수도 있다. 이를 극복하기 위해서는 네트워크의 대역폭에 따라 전송되는 멀티미디어 데이터의 포맷을 적절히 변경할 수 있게 하여야 하며, 현재 LS 패키지에서 제공하는 일대다 전송 기능은 수신자의 수가 많아질수록 처리 속도가 느려지므로 최대 수신자의 수를 제한하거나 더 나은 일대다 전송 기술의 적용이 필요하다.

향후 이 논문의 연구 결과는 화상 회의에 응용하거나 화이트보드 또는 채팅 등의 다양한 기능을 추가하여 화상 교육과 같은 실시간 스트리밍 분야에 응용할 수 있을 것이다.

참고문헌

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

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

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

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

[5] 김만수, 정목동, "CORBA/JMF 기반 오디오/비디오 스트림 시스템의 설계 및 구현," 멀티미디어학회 논문지, 4권, 4호, 2001.

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

[7] 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.

[8] OMG, Audio/Video Streams v1.0, OMG Document formal, 2000.

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

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

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

저자소개

서 봉 근(Bong-kun Seo)

2004. 2 안동대학교 컴퓨터공학과 공학사

2004. 3 - 현재 안동대학교 대학원 컴퓨터공학과 석사과정

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

김 윤 호(Yun-Ho Kim)

1983. 2. 경북대학교 전자공학과 공학사

1993. 2. 경북대학교 대학원 컴퓨터공학과 공학석사

1997. 2. 경북대학교 대학원 컴퓨터공학과 공학박사

1997. 8. - 현재 안동대학교 전자정보산업학부 부교수

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