제 8 강좌 : JMF 응용하기와 확장 패키지의 결합

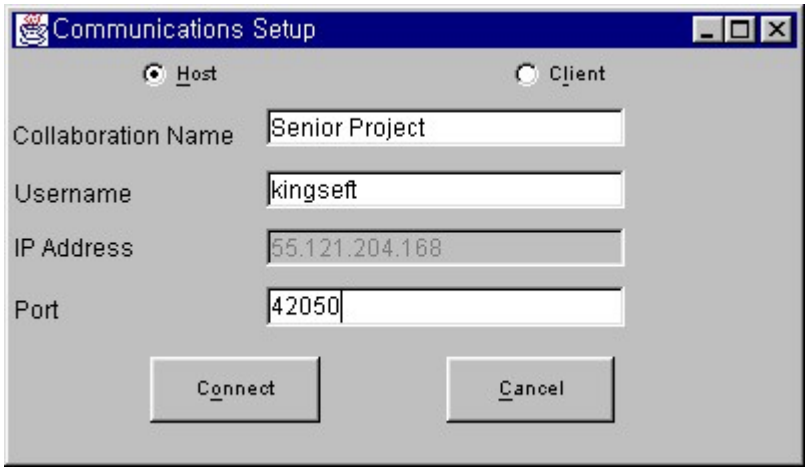
지난 강좌에서는 RTP 미디어 데이터의 수신을 위한 직접적인 구현 방법들과 이를 클라이언트 측에서 재생하고 데이터를 표현하는 부분에 관하여 알아보았으며, 수신중의 데이터 포맷의 변환에 대응하는 방법, 전송 및 수신시의 데이터 발생량을 제어하고 수신 버퍼의 버퍼링을 제어하는 방법과 비디오 데이터의 화질을 제어하는 방법등을 구현하였다.
이번 강좌에서는 지난 강좌까지의 내용들을 토대로 JMF를 이용한 공개용 응용 애플리케이션들의 구현을 살펴보고, 이러한 프로그램들의 적용을 검토하여 RTP 데이터의 모니터링 시스템을 구현해본다. 또한 JSDT(Java Shared Data Toolkit)를 통한 JMF와의 결합 기능에 대해서도 그 구현에 관하여 언급하고 여러기능이 결합된 애플리케이션의 구축을 소개한다. 마지막으로 멀티 캐스팅을 위한 Mbone망 및 이에 관한 주요 개념들과 관련 애플리케이션을 살펴보며 전체적인 JMF 강좌를 마무리 하도록하겠다.

삼성테크윈 정밀기기연구소
이재훈 전임연구원
kingseft@samsung.co.kr

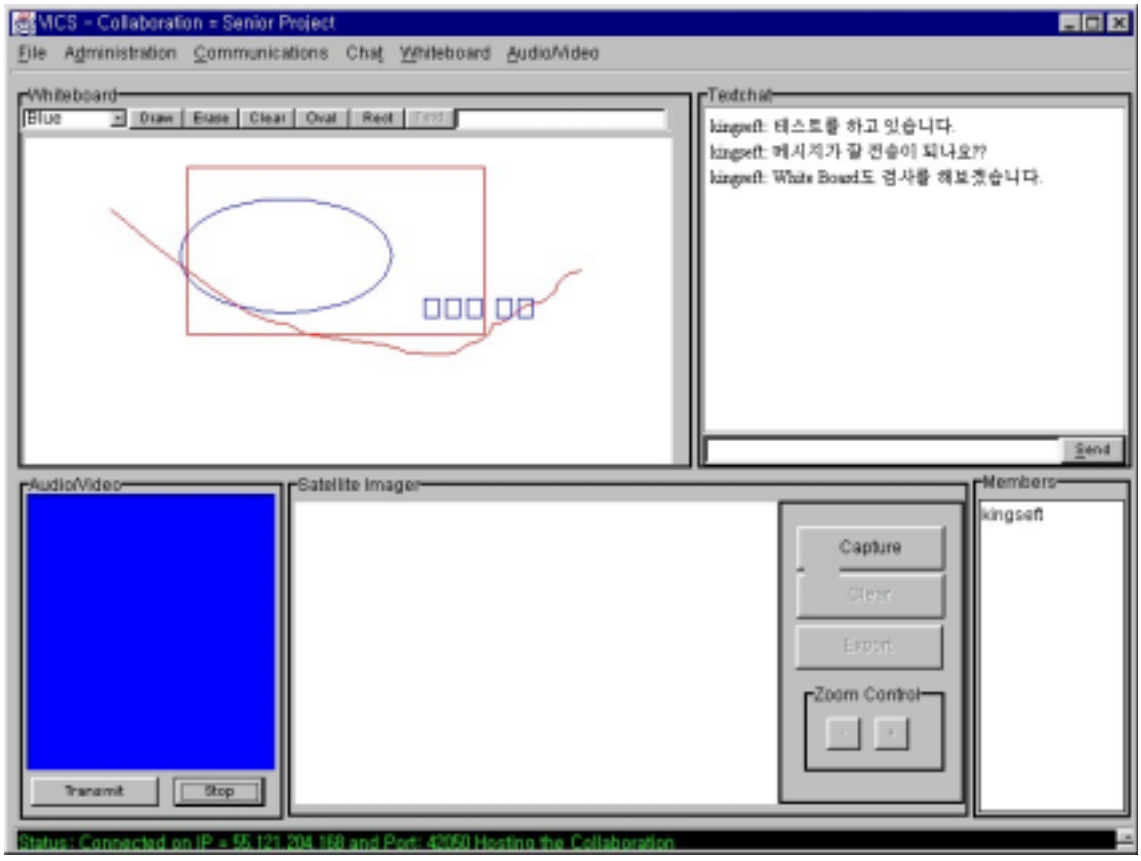
JSDT 살펴보기

이제 독자들은 JMF를 이용하여 영상 및 음성 데이터에 대한 송수신 기능을 구현하고, 기본적인 화상통신 프로그램의 골격을 구축했을 것이다. 이번에는 이렇게 작성된 프로그램에 몇가지 추가적인 기능을 더해보자. JMF 프로그래밍을 하면서 호기심 많은 독자들은 이미 JSDT(Java Shared Data Toolkit)에 대하여 들어보았을 것이다. JSDT는 인터넷 및 네트워크상에서 다수의 사용자들이 데이터의 공유와 협동 작업을 수행하기 위해 만들어진 Java의 확장 패키지이다. 먼저 JSDT에 대한 자세한 내용을 살펴보기 이전에 독자들의 이해를 돕기위하여 JMF와 결합된 애플리케이션의 한 예를 들어보도록 하겠다. [그림 1]은 애플리케이션을 구동한 초기 화면이다.

[그림 1]의 첫 선택버튼에서 처럼 이 애플리케이션은 호스트와 클라이언트의 기능을 동시에 가지고 있기 때문에 초기의 프로그램을 구동시에 호스트 또는 클라이언트로서의 구동 방식을 지정해 주어야 한다. 다음으로 사용자의 이름을 지정하고 IP 어드레스와 사용할 포트번호를 지정해준다. 이제 접속을 시도하면 [그림 2]와 같은 실제 구동되는 애플리케이션이 나타나게된다. JSDT에서는 이러한 애플리케이션의 구동을 위해서 먼저 환경 설정과 더불어 Registry의 세팅과 서버 및 클라이언트등의 지정을 추가로 요구하게 된다. 이에 관한 자세한 설정 사항은 다시 언급하도록 하겠다.



[그림 1] JSDT 애플리케이션의 초기 구동



[그림 2] JMF와 JSDT를 결합한 애플리케이션

[그림 2]에서는 독자들이 이미 확인한 것과 같이 본 애플리케이션에서는 전자 칠판기능을 구현한 화이트 보드와 채팅기능 및 오디오/ 비디오 데이터의 송수신 기능, 세션에 참여한 참여자들의 정보를 나타내는 멤버창등을 결합한 일종의 통신 메신져의 역할을 수행하는 애플리케이션이다. 물론 이 모든 기능들을 구현함에 있어서 반드시 JSDT를 이용해야하는 것은

아니지만, JSDT 확장 패키지와 JMF를 접목 시킴으로서 보다 유용한 애플리케이션 개발에 도움이 될것으로 생각된다. 먼저 JSDT 패키지를 http://java.sun.com/javamedia/jsdt 에서 다운로드 받고 설치하기 바란다.

JSDT의 설치 및 환경 설정

이 부분에서는 JSDT의 설치 및 환경설정 부분과 구동에 관하여 알아보도록 하자. JSDT 역시 JMF와 마찬가지로 기본적인 JDK에 포함되지 않는 Java의 확장 패키지이다. 그러므로 독자들은 Sun의 Java 싸이트에 접속하여 직접 다운로드를 받아 설치해야 한다. 먼저 JSDT 설치를 위한 시스템 사양을 살펴보자.

소프트웨어 요구사항과 하드웨어 요구사항
Windows 98 또는 Windows NT 4.0
JDK 1.3
Optional: HotJavaTM Browser 1.0 or 1.1
JSDT classes
Example applets and applications
LRMP classes (JSDT의 LRMP 구현시에 이용됨)
Video Capture 장치 및 Sound Card

JSDT를 시스템에 설치한후에 독자들은 몇가지 추가적인 작업을 통해 세팅을 완료할 수 있다. 먼저 JSDT를 특정 디렉토리 (예를들어 C:₩JSDT-2.0)등에 압축을 풀고 설치를 시작한다. 설치 과정이 끝나면 독자들의 시스템 환경 설정이나 autoexec.bat 파일등에서 CLASSPATH 가 올바르게 설정이 되었나를 확인해야 한다. 실제 JSDT의 설치는 InstallShield를 통해 자동으로 수행되지만, 만약 설치 과정중에서의 오류나 혹은 시스템의 환경설정이 바뀌는 경우에는 독자들이 직접 환경설정을 해주어야 한다. 예를들어 JSDT를 C:₩JSDT-2.0에 설정하였다면 독자들의 CLASSPATH 설정은 아래와 같을 것이다.

```
set JSDTHOME=C:₩JSDT-2.0/TT>
set CLASSPATH=%JSDTHOME%₩lib₩jsdt.jar;%JSDTHOME%₩examples₩classes;.;%CLASSPATH%
```

JSDT Registry 구동하기

이제껏 JMF에 익숙해진 독자들에게는 다소 이상할 수 있지만, JSDT에서는 어떠한 애플리케이션을 구동하기에 앞서서 적절한 타입의 Registry를 구동할 필요가 있다. 물론 이를 프로

그램 내부에서 구동시키는 것도 가능하며 프로그램과는 별도로 구동을 시키는 것도 가능하다. JSDT에서 Registry를 구동하기 위한 타입은 3개가 지원된다.

socket - a TCP/IP sockets 기반의 구현
lrmp - an LRMP (light-weight reliable multicast protocol) 구현
http - an HTTP 구현

Registry를 구동시키기 위해서는 커맨드 모드에서 다음과 같이 실행한다.
java com.sun.media.jsdt.type.Registry -port 4561

위의 구동 부분에서 타입정보는 socket, lrmp, rmi 또는 http의 한종류가 될 수 있으며 위와 같은 실행이외에도 com.sun.media.jsdt.RegistryFactory.startRegistry(type) 메쏘드를 직접 프로그램상에서 호출하여 Registry를 구동하는것 역시 가능하다. 기본적인 JSDT의 설치 과정을 마친후에, 이 부분에서는 JSDT를 이용하기 위한 기본 개념을 습득하도록 한다. 구체적인 프로그램의 설명보다 먼저 간단한 실행 방법과 결과 화면들을 확인하여 보자.

JSDT 프로그램의 구조

먼저 JSDT에 대한 완전한 설정을 했다는 가정하에서 JSDT를 설치하고 난후의 Sample Program 중에서 채팅서버와 클라이언트의 구현에 대하여 생각하여 보자.먼저 Chat 예제의 실행 방법은 다음과 같다.

1. 프로그램 구동
→ Java ChatServer - server 55.121.204.168 - port 4400 - type socket

JSDT의 실행을 위해 지정한 옵션들은 아래와 같다.
- server ← 서버의 이름 혹은 실제 서버 프로그램이 구동되는 컴퓨터의 IP 주소.
- port ← 통신에 사용할 포트의 번호를 지정.
- type ← 어떤 타입으로 통신을 진행할것인가를 결정.

기본적으로 JSDT 프로그램은 - server 의 경우 JSDT를 인스톨할 때 지정해준 서버 컴퓨터의 이름을 사용하고(설치시 IP를 지정했다면 IP 주소를 사용) 포트번호는 디폴트 포트번호를 이용하며(프로그램 내부적으로) 타입은 소켓 타입을 이용하게 된다. 예를 하나 들어보자. 컴퓨터가 3대 있다고 가정하고 3대의 컴중에서 한 컴퓨터는 서버 프로그램을 구동하고, 나머지 두대는 각각 클라이언트 프로그램을 구동한다고 가정한다면 앞에서 설명한 것 처럼 서

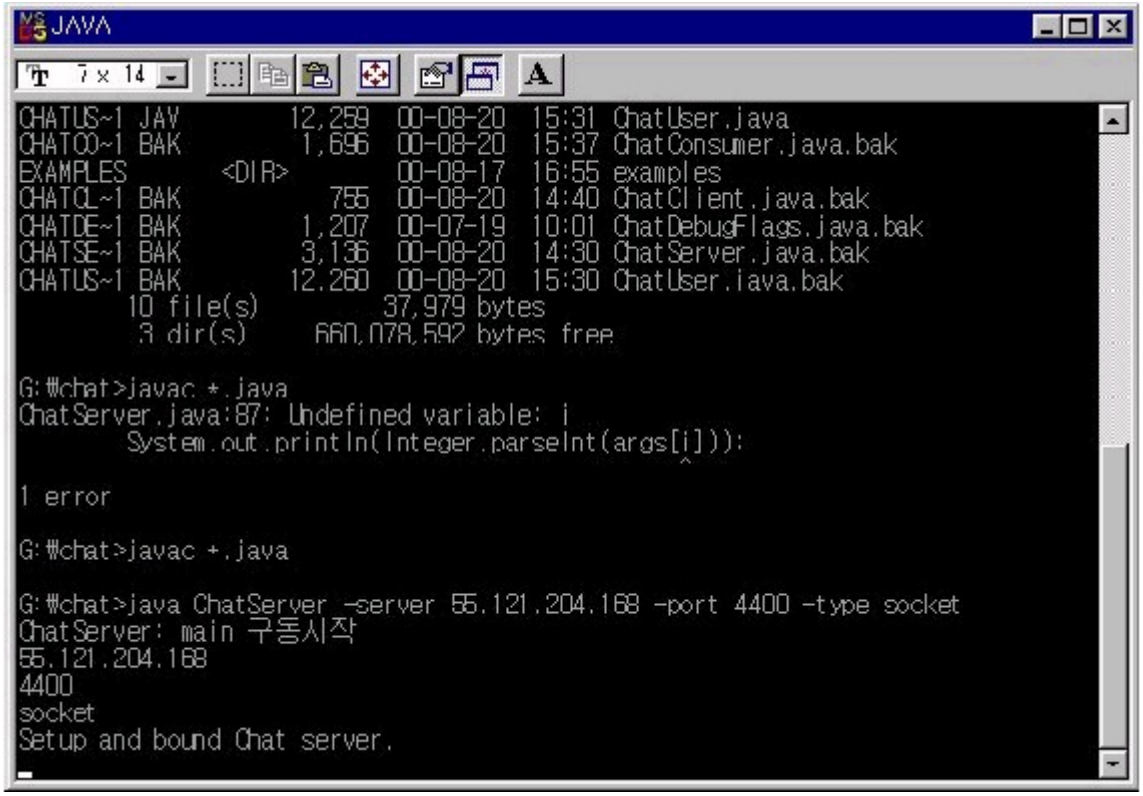
버 프로그램은

Java ChatServer - server 55.121.204.168 - port 4400 - type socket

라고 지정한다. 여기서 55.121.204.168 에 주의해야한다. 이것은 서버 프로그램이 구동되는 컴퓨터의 IP주소이다. JMF에서처럼 수신되는 컴퓨터의 IP 주소를 적는 것이 아니라 실제 서버가 구동되는 즉, ChatServer가 구동되고 있는 컴의 IP를 적어주어야 한다는 것이다. 이 방법 또는 서버 컴퓨터에 이름을 지정해준 경우 예를들어 MyServer 라고 JSDT 설치시에 지정했다면 서버 이름을 적는 부분에 MyServer 라고 적어주어도 된다. 만약 네트워크에는 연결이 되어있지만, 실험을 할 수 있는 컴퓨터가 한 개일경우 이럴때는 바로 서버 이름에 localhost 라고 적어주면 된다. 즉, 자신의 컴퓨터에 접속한다는 의미이다.

그럼 실제로 서버를 구동해 보자. [그림 3] 에서 보는바와 같이 먼저 서버를 구동하면 다음과 같은 메시지가 출력된다.

ChatServer : main 구동시작
55.121.204.168 ← 서버의 IP 주소
4400 ← 포트번호
socket ← 이용하는 타입종류
Setup and bound Chat server ← 서버가 설치되었고, 해당주소와 포트번호로 바인드 성공



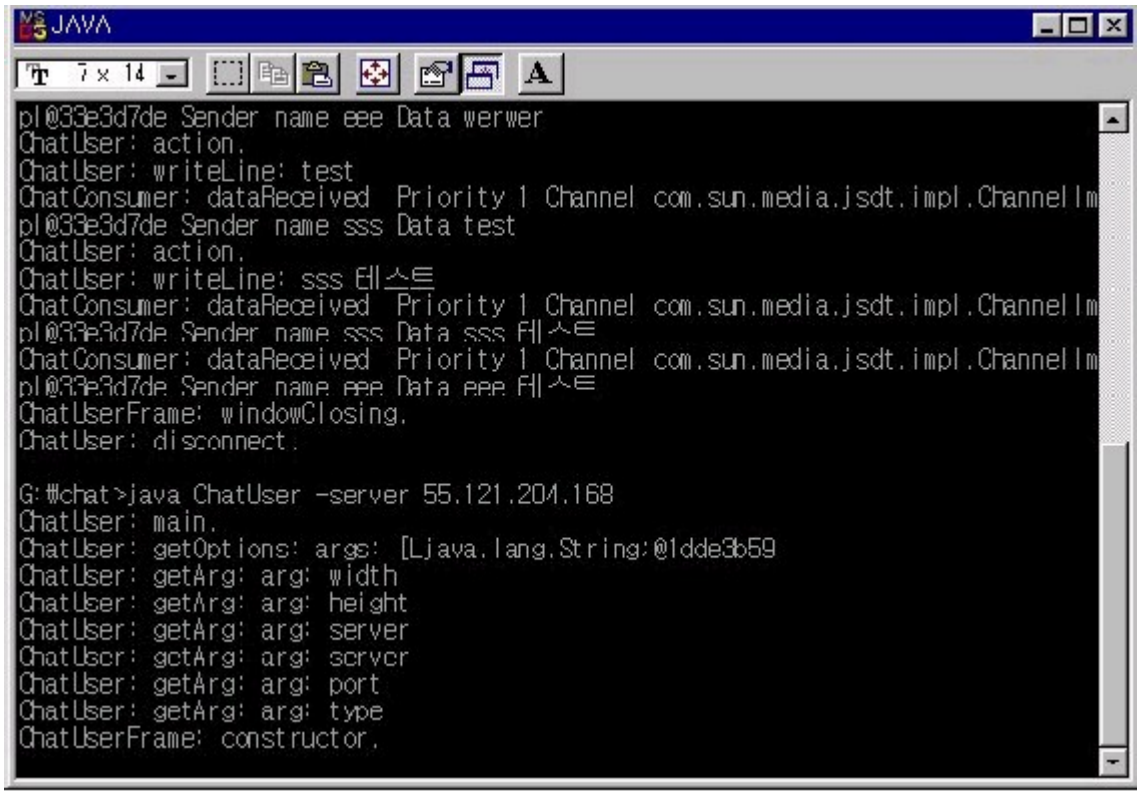
[그림 3] JSDT 서버의 구동

위에서와 같이 서버를 구동한후에 이제는 클라이언트를 구동할 차례이다. 클라이언트에서 서버로 접속하기 위해서도 동일한 옵션을 사용한다. 클라이언트 구동방법은 다음과 같다.

java ChatUser - server 55.121.204.168 - port 4400 - type socket

물론 여기서 서버의 주소 및 포트번호를 지정해 주어야 하며, 서버 구동시에 적어주었던 타입을 그대로 적어주어야 한다. 그러면 다음과 같은 메시지를 확인할 수 있다.

Chatuser : main
Chatuser: getOptions args: [Ljava.lang.String@mode… … …]
Chatuser: getArg : arg width
Chatuser: getArg : server
Chatuser: getArg: type
ChatUserFrame constructor



[그림 4] JSDT 클라이언트의 구동

이와 같은 메시지가 모두 나온후에 클라이언트 프로그램을 구동한다. 구동되는 그림은 [그림 5]를 확인하기 바란다. 보는바와 같이 본 프로그램은 애플릿 및 애플리케이션으로 모두 구동 가능하며 또한, 다음과 같이 3 개의 파트로 구성되어 있다.

- 유저의 이름을 넣는 부분 .
- 주고 받는 데이터를 표시하는 부분 .
- 전송할 데이터를 적는 부분 .

프로그램 구동후에 맨 처음으로 할 일은 채팅에 참여할 유저의 이름을 지정하는 것이다. 채팅 서버가 이미 구동하고 있고, 세션이 알맞게 열렸다면, SignOff 버튼은 DeActive 상태가 되어 실제 채팅에 참여하게 된다. 이렇게 세션에 참여하게 되면 Say 라고 적힌 텍스트 영역에 데이터를 적고 엔터를 침으로서 데이터를 보낼수 있다. 보낸 데이터 및 상대방으로부터 수신된 메시지는 Message : 라고 적힌 리스트박스에 나타나게 된다.

[그림 5] JSDT Chatting 프로그램의 구동

JSDT 프로그래밍

이제 직접적인 프로그래밍 부분을 살펴보도록 하자. JSDT 의 모든 예제들은 거의 비슷한 구조를 가지고 있으며 실제로 하나의 예제만 잘 분석해두면 나머지 다른 구현들은 70% 이상은 전부 분석이 된것이나 마찬가지이다. 위에서 예제로 든 프로그램은 다음과 같은 5 개의 클래스로 구성으로 이루어져 있다.

ChatServer.java
ChatUser.java
ChatClient.java
ChatConsumer.java
ChatDebugFlags.java

먼저 간략히 살펴보면 서버측은 ChatServer.java 파일과 ChatClient.java 파일이 필요하며 클라이언트측은 ChatUser.java ChatClient.java ChatConsumer.java ChatDebugFlags.java 파일등이 필요하다. 먼저 서버측부터 살펴보도록 하자. 일단은 ChatServer.java 소스 파일을 직접 확인하기 이전에, 개념적으로 한번 생각을 해보도록 하자. 서버측 프로그램은 일단 세션을 만들어야 한다. 즉, 통신을 주관하는 객체를 만들어야 한다는 것이며 이것을 세션이라고 한다. 세션을 만들때는 3가지의 요소가 필요하다 . 호스트컴퓨터의 이름, 포트번호, 그리고 세션의 타입이 바로 그 필요 파라미터들이다. 두번째로는 클라이언트가 접속할때를 대

비해서 클라이언트의 정보를 관리할 무엇인가가 필요하며 그 클라이언트를 통해서 데이터가 전달되니까 데이터가 전달될 통로, 즉 채널이 필요하게 된다. 한번 정리를 해보자.

서버측에서 필요한 사항
1. 세션을 만들기 위한 url 생성 호스트컴퓨터의 이름, 포트, 세션타입과 세션이름
2. 클라이언트를 관리할 클래스
3. 세션을 만들고 등록하는 부분
4. 세션을 만든후에 데이터 전송을 위한 채널을 생성하는 부분과 채널에 클라이언트를 연결

그럼 1번부터 해결해 보자. 먼저 세션을 만들 때 필요한 url 정보 생성을 위하여 호스트 컴퓨터의 이름을 구해오는 getHost, 포트번호를 얻어오는 getPort, 접속 타입을 정의해주는 getType 에 대한 구현을 해야한다. 이렇게 얻어진 접속 파라미터들은 URLString 의 createSessionURL의 인자로 넘겨져 우리가 필요로하는 url 정보를 생성하게 된다.

[리스트 1] 호스트 파라미터의 설정
```
hostname    = getHost(args);
hostport    = getPort(args);
sessionType = getType(args);
url  = URLString.createSessionURL(hostname, hostport, sessionType, sessionName);
```

JSDT에서는 URL 생성을 위해서 URLString이라는 클래스를 제공해주고, createSessionURL이라는 메쏘드를 제공한다. 여기서 하나 추가된부분, sessionName 에 주의해야 한다. 클라이언트와 서버가 서로 통신을 하기위해 세션들이 존재하는데, 각 세션간의 구별되는 사항이 없다면 데이터는 어느세션으로 가야할지 알 수가 없다. 그래서 세션이름을 부여해줌으로서 서로 엉뚱한 세션으로 가서 남에게 피해를 끼치는 사항을 없에버리자는 것이다. 예를들어 보자. 1번 채팅방에서 채팅하는 사람들이 있었고.. 2번 채팅방에서 채팅하는사람들이 있는데.. 대기실에 있던 사람이 1번 채팅방에 들어가고 싶은데… 세션이름을 2번 채팅방으로 적어주었다면 어떻게 될까?. 그래서 세션이름은 서버측에서도, 클라이언트측에서도 동일하게 지정해주어야 하는 것이다. 다음으로는 클라이언트 정보를 처리할 부분이다. JSDT에서는 클라이언트의 정보를 관리하고 처리하기 위해서 Client 라는 인터페이스를 제공한다. 그래서 우리가 클라이언트를 처리할 클래스를 만들고자 할때에는 Client 인터페이스를 구현해야 한다. 이렇게 구현할때에는 반드시 구현해주어야하는 메쏘드들이 따라오게 된다. Client 인터페이스에서는 authenticate와 getName 함수를 반드시 구현해야 한다.

[리스트 2] Client 인터페이스의 구현
```
public class ChatClient implements Client {
```

```
    protected String name;

    public ChatClient(String name) {
        this.name = name;
    }

    public Object authenticate(AuthenticationInfo info) {
        System.err.println("ChatClient: authenticate.");
               return(null);
    }

    public String getName() {
        return(name);
    }
}
```

위의 프로그램에서 실제 인증을 처리하는 authenticate 부분은 설명할 것 이 많기는 하지만, 일단은 아무 처리도 하지 않게하려고 null을 리턴했다. 그리고 getName을 보기 바란다. 이 함수는 현재 클라이언트가 누구인지.. 즉, 클라이언트의 이름을 얻어주는 부분이다. 세번째는 위에서 만든 클라이언트 정보와 서버의 url 정보를 가지고 세션을 실제로 만드는 부분이다. [리스트 3]을 확인하여 보자.

[리스트 3] 클라이언트 정보를 이용한 세션의 생성

```
if (RegistryFactory.registryExists(sessionType) == false) {
    RegistryFactory.startRegistry(sessionType);
}
chatSession = SessionFactory.createSession(client, url, false);
chatSession.createChannel(client, "ChatChannel", true, true, false);
```

[리스트 3]에서는 주의를 할 점이 있다. 서버가 구동되었는지 안되었는지는 바로 레지스트리라는 것을 이용한다는 것이다. 다시 말해서 등록정보를 모아 놓은것으로서 이 등록정보를 통하여 현재 존재하는가를 물어보고, 없으면 새롭게 등록을 해야한다는 의미이다. 이렇게 레지스트리를 시작한후에 세션을 구동하게 된다. 마지막으로, 세션을 만든후에는 실제 그 세션을 통해서 데이터가 지나갈 통로를 만들어야 한다. 그것이 바로 채널이다. 하나의 세션 안에서 여러 개의 채널을 통해 서로다른 종류의 데이터들이 전달될수 있다. 위의 프로그램에서는 일단 채널 하나를 만들어서 세션에 붙이는 작업을 하였다. 이렇게 해서 서버측의 모

든 구현이 모두 완료되었다. 그러나 많은 독자들이 의문을 제시할것이다. 기존의 채팅 프로그램에서 생각해오던 각 클라이언트의 접속, 접속자 관리며 메시지의 전달부분등등, 또한 비동기적 메시지 전달을 위한 쓰레드 클래스의 설계등등… 그런 작업은 어떻게 해야 하는가. 실제 이러한 부분들은 전부 JSDT의 내부에 숨겨져 있다. 그것이 바로 JSDT의 장점으로서 많은 부분을 직접적인 코딩을 하지 않고서도 필요한 구현을 할 수 있게 해준다. 이제 남은 부분은 클라이언트 측의 구현이다.

JSDT 클라이언측 구현

클라이언트도 일단 소스코드는 보지말고, 개념적으로 코딩을 해보도록 하자. 앞서서 서버에서 보았듯이 클라이언트에서도 통신을 하기 위해서 Connect() 부분, 연결을 끊는 disconnect() , 그리고 데이터를 쓰는 부분과 데이터를 얻어오는 부분이 있어야 한다는 것을 확인했을 것이다. 그리고 화면 GUI를 간략하게나마 꾸미는 부분이 있어야 할것이다.
필요한 첫번째 사항은 먼저 서버측과 동일하게 세션을 만들어야 한다는 것이다. 세션을 만들때에는 Server 이름과 Port 그리고 세션타입과 세션의 이름이 필요하다는 것을 독자들은 이미 알고있을 것이다.자. 그럼 먼저 세션을 만들고 서버에 연결되는 부분을 다시 한번 언급해 보자.

[리스트 4] JSDT 클라이언트 접속관계

```
url = URLString.createSessionURL(hostname, hostport, sessionType, sessionName);
SessionFactory.sessionExists(url);
client = new ChatClient(name);
session = SessionFactory.createSession(client, url, true);
channel = session.createChannel(client, "ChatChannel",true, true, true);
chatConsumer = new ChatConsumer(client.getName(), messageArea);
channel.addConsumer(client, chatConsumer);
```

먼저 서버측의 이름, 포트번호, 세션타입, 세션이름을 설정해서 url 을 생성한다. 그리고 난후에 세션의 존재유무를 판단하는 부분을 필요로 한다. 여기까지 한후에 클라이언트의 모든 정보를 관할하게 되는 클라이언트 클래스를 만들어야 한다. 이 부분은 아까 서버측에서 구현할때도 했었다. [리스트 4]를 살펴보면 먼저 세션을 만들고 , 세션에서 이용되는 채널을 만든후에 채널을 이용해서 데이터를 수신할 ChannelConsumer 생성하고 , 채널에 ChannelConsumer를 첨가한다. 자. 위에서 만든 url과 client 정보를 이용해서 세션을 만들었다. 여기까지는 아까 서버측 꾸밀 때도 해보았으니 그리 어렵지는 않을 것이다. 다음은

세션에서 실제 통신을 담당하는 채널을 만드는 일이다. 앞에서도 설명했지만 세션은 모든 통신을 담당하고 실제 데이터의 전송은 채널을 통한다고 했는데구 실제로 데이터가 통과하는 채널만 만들면 무엇인가 빠진 것 같은 느낌이 든다. 실제로 그 채널을 통해 전달되는 데이터를 이용하겠다고 말해주고 알려주어야 한다. 그것이 바로 Consumer 이다. 즉, 채널을 소비하는 즉, 사용하는 객체를 알려주어야 한다는 것이다. 그래서 채널을 만들어준 후에 이렇게 현재의 클라이언트가 이 채널의 소비자 즉, 이용하겠다는 것을 표현해 주어야 한다. 그 부분이 바로 addConsumer 부분이다. 자. 그러면 실제 구현 예를 살펴보자.

[리스트 5] Channel Consumer의 구현 부분

```
class ChatConsumer implements ChannelConsumer, ChatDebugFlags {
    protected String name;
    TextArea messageArea;

    public ChatConsumer(String name, TextArea messageArea) {
        if (ChatConsumer_Debug) {
            System.err.println("ChatConsumer: constructor.");
        }

        this.name = name;
        this.messageArea = messageArea;
    }

    public synchronized void dataReceived(Data data) {
        String  message;
        int     position   = 0;
        int     priority   = data.getPriority();
        String  senderName = data.getSenderName();
        Channel channel    = data.getChannel();
        String  theData    = data.getDataAsString();

        if (ChatConsumer_Debug) {
            System.err.println("ChatConsumer: dataReceived " +
                               " Priority "    + priority +
                               " Channel "     + channel +
                               " Sender name " + senderName +
                               " Data "        + theData);
```

```
        }

        message = senderName + ": " + theData + "₩n";
        position = messageArea.getText().length();
        messageArea.insert(message, position);
    }
}
```

[리스트 5]는 ChannelComsumer 를 구현하였다. 바로 채널의 데이터를 받기 위한 부분으로서 채널을 소비하겠다는 의미이다. 이때 반드시 구현해주어야 하는 함수가 바로 dataReceive 이다. 이 부분을 통해서 내가 지금 받은 데이터는 누구에게서 온것인지, 어떤 우선순위를 지니는지, 어떤 채널을 통해 받았는지를 확인할 수 있다. 자. 그럼 데이터를 받는 부분까지의 구현은 완성이 된것이다. 그럼 과연 데이터는 어떻게 전달해 줄까?. 그 부분이 바로 아래와 같다.

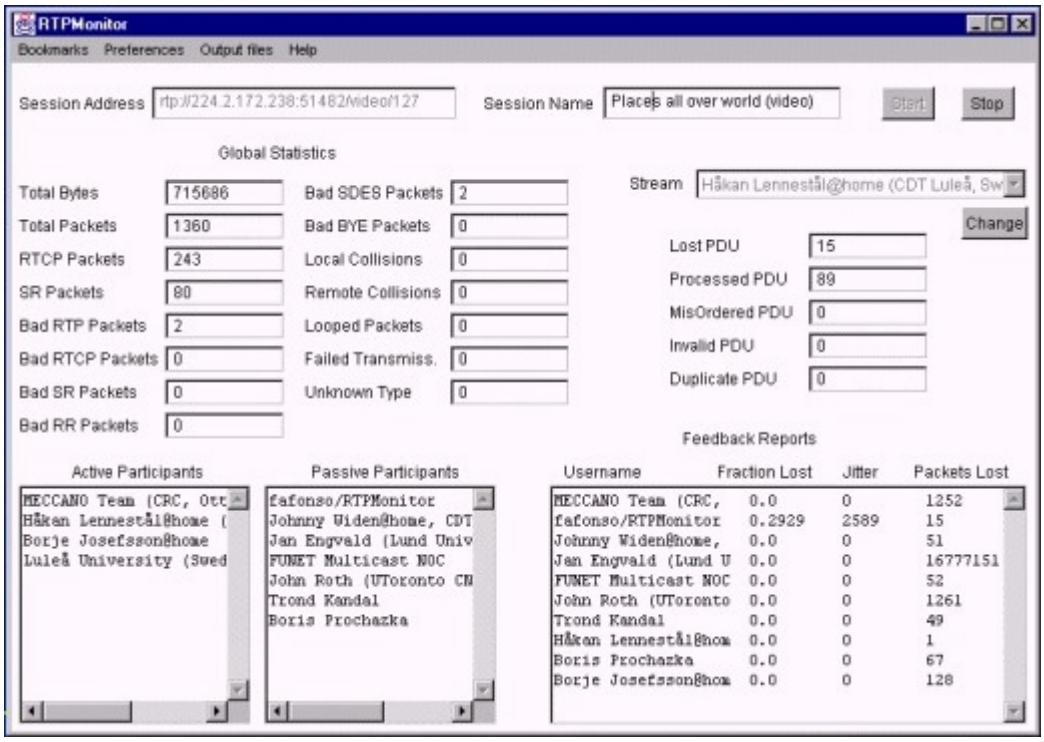
[리스트 6] JSDT의 데이터의 전달

```
data = new Data(message);
data.setPriority(Channel.HIGH_PRIORITY);
channel.sendToAll(client, data);
```

JSDT에서는 채널을 통해 전달되는 데이터는 Data라는 클래스를 통해서 변환을 시켜준다. 그리고 실제 데이터는 채널 클래스를 통해서 전달된다. 위의 예제에서는 채팅에 참여한 모든 사람에게 데이터를 전달하기 때문에 sendToAll 이라는 함수를 썼지만, 일부분의 사람이나 한사람만 데이터를 받게 할수도 있다. 자. 기본적인 JSDT의 개념과 코딩 방법을 소개하였지만 실제적인 구현이나 보다 자세한 API의 설명은 관련 문서들을 참조하기 바란다.

RTP Monitor 프로그램

지난 강좌들에서 이미 언급한 RTP를 통한 오디오/ 비디오 데이터의 송수신 과정을 이해한 독자라면 RTP 상에서의 패킷 데이터들의 흐름과 이를 모니터링 하는 과정을 쉽게 구현할 수 있을 것이다. 이번에는 하나의 구현 사례로서 JMF에서 지원하는 RTP 패키지 클래스들을 이용한 RTP 모니터 프로그램에 대하여 살펴보도록 하자. 본 프로그램은 Francisco Afonso 가 공개한 vrtp 프로그램이다. 프로그램의 구조에 앞서 먼저 실행 화면을 [그림 6]에 나타내었다.

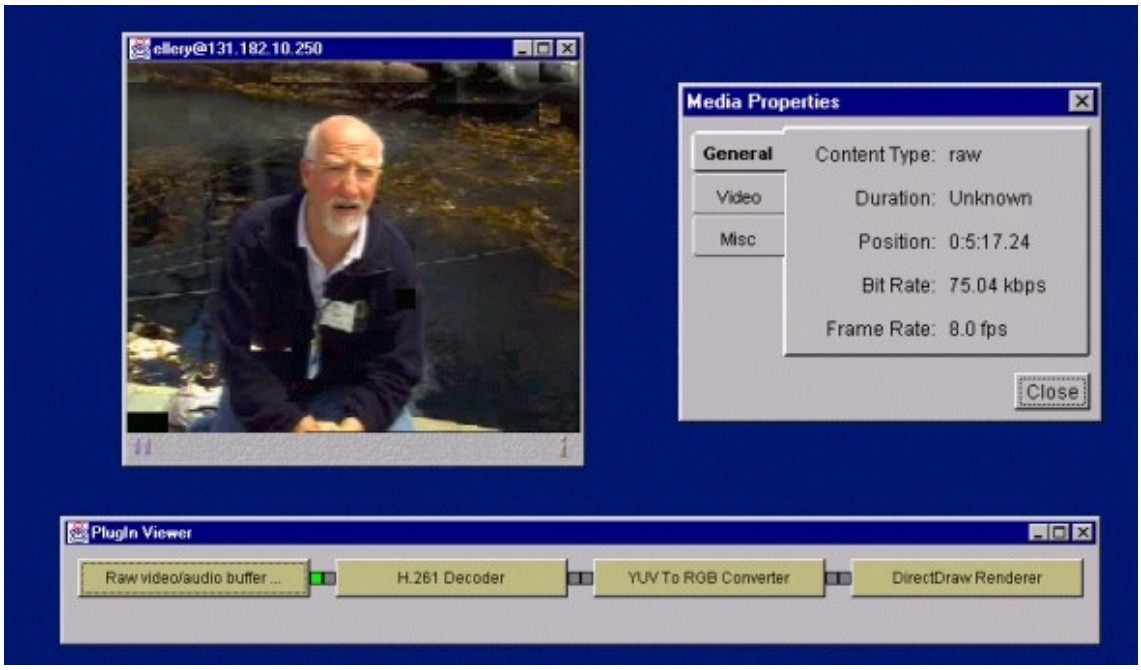


[그림 6] RTP Monitoring 프로그램

[그림 6]에서 보는 바와 같이 본 프로그램은 JMF의 RTP 패키지를 기반으로 작성된 애플리케이션으로서 세션에 참여한 모든 참여자들에 의해 전달된 패킷들을 수신하고, 분산 모티터링을위한 서비스의 Quality를 추정하며 오류 검출과 통계정보를 나타내기 위한 프로그램이다. 수신을 위하여 Session Address의 지정부와 SessionName을 지정하는 판넬, Global RTP Packet 통계 정보를 나타내어주는 부분, 세션 참여자들의 Active / Passive 상태 및 세션 참여자별 패킷의 손실 정보를 나타내어준다. 또한 오디오 비디오 스트림에 대하여 패킷 모니터링을 함과 동시에 로컬에서의 재생기능을 수행한다. 모니터링의 결과로서 나타나는 재생화면은 [그림 7]을 통해 나타내었다.

본 프로그램에서는 몇 단계의 부분적인 구현으로 프로그램 구조를 나누었다. 먼저 GUI 구현부에서는 rtpmonitor의 메인 윈도우 화면을 구성하고 Bookmark 메뉴를 구성하여 세션이름과 세션주소 정보를 생성/삭제할 수 있도록 구성하였다. Preference 메뉴에서는 현재 모니터링을 수행하는 세션 데이터의 통계 정보를 획득할 것이지 또는 로컬에서의 재생을 할 것인지를 선택하도록 구성하였다. 그림에서 보듯 메인윈도우의 상단부에서는 rtp MediaLocator의 포맷과 세션이름에 따라서 세션 정보를 포함하고 있으며 나머지 부분은 rtp 헤더 정보 및 실제 수신되는 데이터의 통계 정보이다. 통계 정보의 구성을 위하여 Global 통계정보에서는 모든 관계된 세션에 대한 정보를 수집하여 보여주며, Stream 통계정보에서는 단일 스트림에 대하여 수신된 통계정보를 생성하여 준다. 실제 본 프로그램의 구현을 위한 클래스 디자인

부분에 대하여 알아보도록 하자. 전체적인 구현소스는 지면관계상 생략하고 핵심적인 디자인 개념과 전반적인 구현부분에 대하여 언급하도록 하겠다. 실제적인 rtp monitor 클래스의 디자인은 rtp 모니터링을 수행하는 타스크의 생성을 위한 기본적인 클래스들의 생성을 목적으로 하고 있으며 아래의 [그림 8]은 수신된 패킷데이터와 rtp monitor의 구현 클래스간의 데이터 플로우 관계를 설명한것이다.



[그림 7] 수신된 비디오 데이터의 재생 및 데이터 플로우관계

먼저 RtpMonitorManager와 RtpUtil 클래스에 대하여 살펴보자.

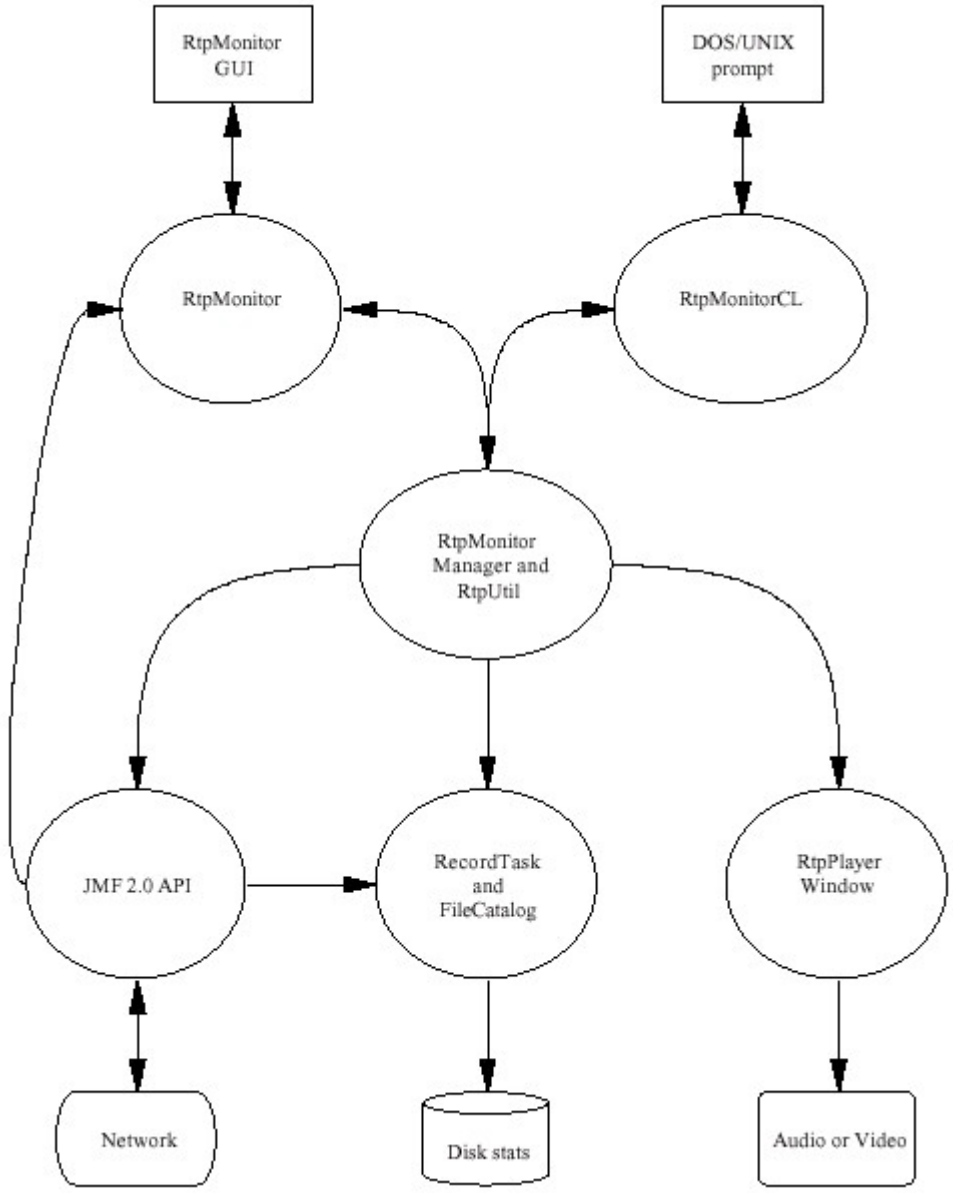
[리스트 7] RtpMonitorManager 클래스의 구현

```
public class RtpMonitorManager implements ReceiveStreamListener {
   private RtpMediaLocator rtpml = null;
   private SessionManager mgr = null;
   private SessionAddress sessaddr = null;
   private boolean flgPart;
   private boolean flgPlay;
   private boolean flgRecord;
   private Hashtable windowlist;
   private RecordTask recTask;

   public RtpMonitorManager( String locatorString, boolean willParticipate,
```

```
                        boolean willPlayStreams, boolean willRecord ,
                        double recordInterval)
    throws MalformedURLException, UnknownHostException,SessionManagerException,
           IOException
{
   flgPart = willParticipate;
   flgPlay = willPlayStreams;
   flgRecord = willRecord;
   rtpml = new RtpMediaLocator(locatorString);
   mgr = new RTPSessionMgr();
   if(flgPlay){
      mgr.addReceiveStreamListener(this);
      windowlist = new Hashtable();
   }

   InetAddress destaddr = InetAddress.getByName(rtpml.getSessionAddress());
     int port = rtpml.getSessionPort();

   sessaddr = new SessionAddress ( destaddr, port, destaddr, port+1 );
   String cname = mgr.generateCNAME();
   String username = null;
   try{
       username = System.getProperty("user.name")+ "/rtpMonitor";
   }
   catch(SecurityException e) {
       username = "RTPMonitor-user";
   }
   SourceDescription [] userdesclist = new SourceDescription[3];
   userdesclist[0] = new
       SourceDescription( SourceDescription.SOURCE_DESC_NAME,
                          username, 1, false);
   userdesclist[1] = new
       SourceDescription( SourceDescription.SOURCE_DESC_CNAME,
                          cname, 1, false);
   userdesclist[2] = new
       SourceDescription( SourceDescription.SOURCE_DESC_TOOL,
```

```
                                "RTPMonitor v1.0" , 1, false);

    SessionAddress localaddr = new SessionAddress();

    double rtcpFraction = 0.05;
    if( ! flgPart )
       rtcpFraction = 0.0;

    mgr.initSession( localaddr, userdesclist, rtcpFraction , 0.25 );

      int ttl = rtpml.getTTL();
    mgr.startSession( sessaddr, ttl, null);

    if(flgRecord){
       recTask = new RecordTask( this, recordInterval );
    }

  }
```

[리스트 7]에서 RtpMonitorManager 클래스는 JMF와 실제 유저 애플리케이션간의 주된 인터페이스 역할을 위해 생성된 클래스로서 JMF의 RTPSessionManager 객체로서 표현되는 세션에 대하여 세션 생성과 구동을 시키는 기능을 수행하며 세션 정보의 저장과 입력되는 세션데이터에 대한 재생기능을 수행한다. RtpMonitorManger 클래스는 하나의 RTP 세션내에서 각각의 단일 통계정보를 수신하는 메쏘드를 갖지는 않는다. 만약 애플리케이션이 각각의 개별적인 통계 정보에 대한 억세스를 할 필요가 있다면, 즉, 로컬에서의 재생목적등이 요구되는 경우에는 RtpSessionManager의 객체를 수신하고 이 객체의 메쏘드를 이용하여 원하는 통계정보의 집합을 획득할 수 있다. 본 구현에서는 새로운 RtpMonitorManager 객체의 초기화에 필요한 파라미터 정보를 다음과 같이 정의하였다. 먼저 세션을 생성하기 위한 세션 어드레스와 세션의 통계정보를 기록할지의 여부를 나타내는 값, 수신되는 패킷 데이터를 로컬에서 재생할지의 여부와 세션내에서 현재 모니터가 active 참여를 할지의 여부를 나타내는 값등으로 구성된다. 초기 구동기간에 RtpMonitorManager 클래스는 JMF내에서 RTPSessionManager 객체를 생성하게된다. [리스트 8]의 RtpUtil 클래스는 JMF에 대해 특정 기능을 수행하기 위해 필요한 통계 정보를 얻는 메쏘드들을 구현하였다.



[그림 8] rtp monitor의 구현 클래스간의 데이터 플로우

[리스트 8] RTPUtil 클래스의 구현

```
public class RtpUtil
{
  public static String getUsername(Participant part){
     Vector sdeslist = part.getSourceDescription();
      if(sdeslist == null){
         return null;
      }
```

```
        SourceDescription des;
        for( int ii=0; ii < sdeslist.size(); ++ii){
            des = (SourceDescription) sdeslist.elementAt(ii);
            if( des.getType() == SourceDescription.SOURCE_DESC_NAME ){
                return des.getDescription();
            }
        }
        return null;
    }

    public static String getUsernameOrCNAME(Participant part){
        String username = getUsername(part);
        if(username == null){
            return part.getCNAME();
        }
        else{
            return username;
        }
    }

    public static long correctSSRC( long ssrc ){
        if(ssrc < 0){
            return (4294967296L + ssrc);
        }
        return ssrc;
    }
}
```

다음으로 RecordTask와 FileCatalog에 대하여 설명하겠다. RecordTask는 RtpMonitorManager 객체에 의해서 이용되며 세션 데이터의 통계정보를 기록하는 기능을 수행한다. 이 타스크는 독립적인 쓰레드로서 생성이 되어 일정기간동안 통계정보를 기록하게된다. FileCatalog 클래스는 실제로 데이터의 전달을 담당하는 역할을 수행한다. [리스트 9]의 RtpPlayerWindow 클래스는 패킷 데이터의 오디오 /비디오 재생을 위한 윈도우의 생성을 수행하는 클래스이다. 이 클래스는 PlayerWindow의 서브 클래스로 구션이 되었으며 각 클래스는 JMF에서 제공하는 클래스이다.

[리스트 9] RtpPlayerWindow 클래스의 구현

```
import javax.media.Player;
import java.awt.*;
import com.sun.media.ui.*;

public class RtpPlayerWindow extends PlayerWindow {

    public RtpPlayerWindow(Player player, String title) {
        super(player);
        setTitle(title);
    }
    public void Name(String title){
        setTitle(title);
    }
}
```

다음으로 [리스트 10]을 통하여 RtpMediaLocator의 구현을 살펴보자. RtpMonitor 클래스는 MediaLocator 클래스를 확장시킨 클래스로서 세션내의 미디어의 위치정보를 표현하는 기능을 구성한다. 본 클래스에서는 InetAddress 의 정보를 획득하며, 세션 주소 및 포트의 값과 TTL 정보를 획득하는 메쏘드들로 구성이 된다.

[리스트 10] RtpMediaLocator 클래스의 구현

```
public class RtpMediaLocator extends MediaLocator{
    public static final int TTL_UNDEFINED = 1;
    private String address = "";
    private int port;
    private int ttl = TTL_UNDEFINED;
public RtpMediaLocator(String locatorString) throws MalformedURLException
    {
        super( locatorString);
        parseLocator( locatorString);
    }
    private  void parseLocator(String locatorString) throws MalformedURLException{
        String remainder = getRemainder();
        int colonIndex = remainder.indexOf(":");
```

```
        int slashIndex = remainder.indexOf("/",2);
        if (colonIndex != -1)
            address = remainder.substring(2, colonIndex);
        else {
            throw new MalformedURLException(
              "RTP MediaLocator is Invalid. Must be of form rtp://addr:port/ttl");
        }
        try{
            InetAddress iaddr = InetAddress.getByName(address);
        }
        catch (UnknownHostException e){
            throw new MalformedURLException("Valid RTP Session Address must be
given");
        }
        String portstr = "";
        if (slashIndex == -1)
            portstr = remainder.substring(colonIndex +1,
                                          remainder.length());
        else
            portstr = remainder.substring(colonIndex +1,
                                          slashIndex);

        try{
            Integer Iport = Integer.valueOf(portstr);
            port = Iport.intValue();
        }catch (NumberFormatException e){
            throw new MalformedURLException("RTP MediaLocator Port must be a valid
integer");
        }

        if (slashIndex != -1){

            String ttlstr = remainder.substring(slashIndex+1,
                                                remainder.length());
            try{
                Integer Ittl = Integer.valueOf(ttlstr);
```

```
                ttl = Ittl.intValue();
            }catch (NumberFormatException e){}

        }
    }
public String getSessionAddress(){
        return address;
    }

public int getSessionPort(){
        return port;
    }

public int getTTL(){
        return ttl;
    }

}
```

RtpMonitor 클래스는 본 프로그램의 메인 클래스로서 Frame을 확장하고 rtpMonitor의 GUI부분을 구현한 클래스이다. 본 구현에서는 RtpMonitor와 RtpMonitorCommandLine 두개의 구현 클래스가 user level의 기능을 수행하게된다. RtpMonitor에서는 사용자의 선택정보를 취득하여 ModifyPreference 클래스를 구동하는 역할을 하며, 북마크에 대한 기능을 제공한다. 또한 주기적인 시간 간격을 통하여 표시되는 통계정보의 갱신기능을 수행한다. 이 클래스는 DisplayTask 객체를 생성하며, 스크린상의 세션정보를 갱신하는 역할을 한다. DisplayTask의 구조를 [리스트 11]에 나타내었다.

[리스트 11] DisplayTask 클래스의 구현

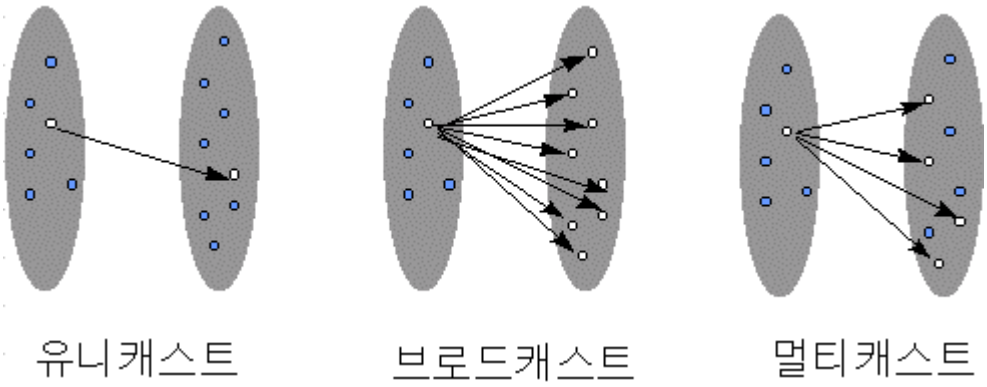
```
 public class DisplayTask implements Runnable {
    RtpMonitor myMon;
    SessionManager mymgr;
    Thread thread;
    int intervalParam;

public DisplayTask(RtpMonitor mon, double interval){
        myMon = mon;
```

```
        intervalParam = (int) (interval*1000);
        thread = new Thread(this,"DisplayTask thread");
        thread.setDaemon(true);
        thread.start();
    }

public void run(){
        while(myMon.isMonitoring()){
              myMon.showGlobalStats();
              myMon.showParticipants();
              myMon.showStreamStats();
              myMon.showFeedbacks();

            try{
                Thread.sleep(intervalParam);
            }
            catch (InterruptedException e){}

        }
    }

}
```

Mbone 환경과 응용 애플리케이션 프로그램의 비교

이번에는 JMF 와 비교될만한 다른 응용 애플리케이션들을 살펴보고, 이러한 애플리케이션의 구동을 위한 Mbone 환경에 대하여 알아보도록 하자. 기존의 인터넷 서비스들이 TCP/IP 란 인터넷 프로토콜을 이용하여 사용자와 정보 제공자간의 비동기적인(asynchronous)인 형태로 텍스트 기반의 정보들을 주고받는 기능을 수행하지만 TCP/IP 프로토콜은 메시지의 송수신 대상이 동시에 여럿인 경우를 고려하지 않았다. 최근 인터넷상의 데이터들은 실시간 정보들을 포함하는 다양한 멀티미디어(audio, video 등) 기반 정보들이 많아졌으며, 또한 여러 사용자간의 인터렉션을 포함한 상호작용 환경을 필요로 하게 되었다. 이를 해결하기 위해 일련의 노력의 결과가 바로 인터넷 미래망으로 불리우는 Mbone 이다. MBone 이란 Multicast Backbone 의 약자로 멀티캐스팅을 지원하는 테스트 성격을 가진 네트워크 백본을 가리킨다. 이 말은 1992 년 MBone 을 탄생시킨 주역들인 스티브 캐스너(Steve Casner),

스티브 디어링(Steve Deering), 한스 에릭슨(Hans Eriksson)이 IETF 7월 회의에서 정한 이름으로 유럽의 네트워크 백본인 EBone(Euroupean Backbone)을 본따 지었다. MBone은 그 이름에서 보는 바와 같이 멀티캐스트를 지원하는 네트워크이다. 인터넷의 전송 방식은 전송에 참여하는 발신지-수신지(source-destination) 관점에서 나누어 유니캐스트, 멀티캐스트, 브로드캐스트로 구분할 수 있다.



[그림 9] 멀티캐스팅과 유니캐스트/브로드캐스트의 비교

유니캐스트 전송 방식이란 하나의 송신자가 다른 하나의 수신자로 데이터를 전송하는 것으로 telnet, ftp 등 일반적인 인터넷의 응용 프로그램이 모두 이와 같은 전송 방식을 사용하고 있다. 브로드캐스트 전송 방식은 하나의 송신자가 같은 서브 네트워크상의 모든 수신자에게 데이터를 전송하는 방식으로, 서브 네트워크상의 주소 해결(address resolution)을 위한 ARP, 역 ARP 등이 이를 이용하여 데이터를 주고 받는다. 멀티캐스트 전송 방식은 하나 이상의 송신자들이 특정한 하나 이상의 수신자들에게 데이터를 전송하는 방식으로 인터넷 화상 회의 등의 응용 분야를 생각해 볼 수 있다. 인터넷상에서 동일한 데이터(예를 들면, 같은 내용의 전자 메일이나 화상 회의를 위한 화상/음성 데이터)를 동시에 둘 이상의 다른 수신자들에게 전송하고자 한다고 가정해 보자. 이러한 멀티캐스트 전송의 응용 분야에서 기존의 유니캐스트 전송 방식을 이용하여, 전송하고자 하는 데이터 패킷을 다수의 수신자에게 각각 여러 번 전송한다면, 동일한 패킷의 중복 전송으로 인한 네트워크 효율 저하를 가져온다. 또한 전송 대상자의 수가 늘어남에 따라 송신자의 전송 부담은 더욱 커질 것이다. 특히 전송 데이터의 특성상 실시간 전송이 필요한 경우에는(화상이나 음성 등) 이로 인한 효율 저하가 전송 자체의 성능에 큰 영향을 끼치게 될 것이다. 이와 같이 멀티캐스트 전송을 할 때, 데이터의 중복 전송으로 인한 네트워크 Resource(Bandwidth) 낭비를 최소화하고, 실시간 공동작업을 효율적으로 보장하기 위해서 인터넷 관련기술을 연구하고 시험할 수 있는 네트워크가 바로 MBone이다

IP 멀티캐스팅 방식

이번에는 인터넷상의 멀티캐스트 전송을 지원하는 MBone 을 이해하기 위해 그 기술적 기반이 되는 IP Multicasting 에 대해 살펴보기로 하자. 멀티캐스트 전송이 일반적인 유니캐스트 인터넷 응용 분야와 다른 점은 우선 그 전송 패킷(packet)에 있다. 일반적으로 TCP/IP 상의 유니캐스트 인터넷 응용 프로그램은 데이터의 송신자가 이를 수신할 수신자의 인터넷 주소(IP Address)를 알고, 수신자 주소를 전송 패킷의 헤더(header)에 표시해 패킷을 전송한다. 이러한 패킷이 올바른 수신자에게 전달되기 위해서는 인터넷상의 많은 라우터(router)가 패킷의 헤더를 보고 전송경로를 결정하게 된다. 이와 같이 패킷의 송신자가 수신자의 주소를 표시해 패킷을 전송하는 방식을 source-oriented 전송 방식이라 한다. 그러나 멀티캐스트 전송을 위한 패킷은 그 구성이 조금 다르다. 패킷의 송신자는 그 헤더에 수신자의 주소 대신, 수신자들이 참여하고 있는 그룹 주소(Group Address)를 표시하여 패킷을 전송한다. 멀티캐스트 전송을 위한 그룹 주소는 D-class IP address(224.0.0.0~239.255.255.255)로, 전세계 개개의 인터넷 호스트(host)를 나타내는 A, B, C-class IP address 와는 달리 실제의 호스트를 나타내는 주소가 아니다. 따라서 이와 같은 멀티캐스트 패킷을 전송받은 수신자는 자신이 패킷의 그룹에 속해 있는가를 판단해 패킷의 수용 여부를 결정하게 된다. 이러한 전송 방식을 receiver-oriented 전송 방식이라 한다. 멀티캐스트를 위한 MBone 은 이와 같은 receiver-oriented 전송 방식의 IP Multicast 원리를 그 기반으로 하고 있다. [표 1]은 IETF 에서 IP 멀티캐스트 주소를 그 사용 범위에 따라 할당하여 권고한 것이다. 일반 사용자들은 표에서 할당이 안된 어드레스를 임의로 사용하면 된다.

[표 1] 멀티캐스트 주소에 따른 할당 내용

IP 멀티캐스트 주소	할당된 내용
224.0.0.0	Reserved
224.0.0.1	All Systems on this subnet
224.0.0.2	All Routers on this Subnet
224.0.0.3	Unassigned
224.0.0.4	DVMRP Routers
224.0.0.5	OSPFIGP OSPFIGP All Routers
224.0.0.6	OSPFIGP OSPFIGP Designed Routers

224.0.0.7	ST Routers
224.0.0.8	ST Hosts
224.0.0.9	RIP2 Routers
224.0.0.10 ~ 224.0.0.255	Unassigned
224.0.1.0	VMTP Manage Group
224.0.1.2	SGI-Dogfight
224.0.1.3	Rwhod
224.0.1.4	VNP
224.0.1.5	Artificial Horizons-Aviator
224.0.1.6	NSS-Name Service Server
224.0.1.7	AUDIONEWS - Audio News Multicast
224.0.1.8	SUN NIS Information Service
224.0.1.9	MTP Multicast Transport Service
224.0.1.10 ~ 224.0.1.255	Unassigned
224.0.2.1	rwho Group(BSD) (unofficial)
224.0.2.2	SUN RPC RMAPPROC_CALLIT
224.0.3.0 ~ 224.0.3.255	RFE Generic Service
224.0.4.0 ~ 224.0.4.255	RFE Individual Conferences
224.1.0.0 ~ 224.1.255.255	ST Multicast Group
224.2.0.0 ~ 224.2.255.255	Multicast Conference Calls
232.x.x.x	VMTP transient groups

Mbone 구성요소

이번에는 이러한 Mbone 네트워크를 구현하기 위한 구성 요소들에 관하여 알아보도록 하자. 먼저 가상 네트워크이다. MBone은 인터넷상에서 화상 회의와 같이 여러 참가자가 있고 이들간에 멀티미디어 데이터 즉 오디오 데이터와 비디오 데이터를 전송하는 응용 프로그램을 가동할 수 있도록 하기 위해 만들어진 '가상 네트워크'다. 여기서 '가상 네트워크'라고 하는 것은 실제 컴퓨터와 컴퓨터들간에 전송 라인을 연결하여 만들어진 네트워크가 아니라, 기존의 인터넷에 연결되어 있는 컴퓨터들을 소프트웨어적으로 처리하여 마치 또 다른 네트워크가 인터넷 위에 있는 것처럼 보이도록 했기 때문이다. 두번째가 터널링과 mrouter 이다. TCP/IP는 기본적으로 유니캐스트만을 고려해서 라우팅(routing)이나 어드레싱(addressing)을 해왔다. 그런 상황에서 멀티캐스트에 대한 요구를 만족시키기 위해 '터널링(tunneling)'이라는 개념을 사용한다. IP 프로토콜에서 네트워크상의 노드들을 구별하기 위해 주소를 부여하는데, 이 주소는 네트워크 주소와 호스트 이름으로 나뉘어진다. 현재 인터넷에서 호스트들을 구별하기 위해 부여하는 주소는 'IP 유니캐스트 주소(IP unicast address)'이다. 즉, 하나의 유니캐스트 주소에는 명백하게 하나의 호스트가 대응된다. 그런데 멀티캐스트를 지원하기 위해 'IP 멀티캐스트 주소(IP multicast address)'가 있는데, 이 주소 하나에 여러 개의 호스트가 대응된다. 이 때, 이렇게 하나의 IP 멀티캐스트 주소에 대응되는 일단의 호스트들을 '호스트 그룹'이라 부른다. 하나의 네트워크 주소마다 멀티캐스트 라우팅을 제공할 호스트, 즉 mrouter가 하나만 있으면 된다. 왜냐하면 네트워크내 나머지 호스트들에게는 '브로드캐스트'하면 되기 때문이다. 두 mrouter 간에 '가상의 연결'을 만드는데 이 연결이 tunnel이고 이 tunnel들을 한데 모으면 바로 MBone이다. 하나의 mrouter는 둘 이상의 tunnel에 속할 수 있다. 여기서 '속한다'는 것은 tunnel의 끝점이 된다는 뜻이다. 다시 잘 생각해보면 MBone상에서 tunnel은 하나의 간단한 연결로 보이지만 실제 인터넷상에서는 중간에 여러 호스트들이 존재해서 메시지 전송을 위해서는 이 호스트들을 거쳐서 이루어져야 한다. 이 호스트들을 거치는 데는 유니캐스트를 사용하여 전송이 이루어진다. 멀티캐스트 메시지들은 어떤 다른 형식을 갖는 것이 아니고 유니캐스트 메시지 패킷에 포함되어 전달된다. 이 멀티캐스트 메시지가 mrouter에게 도착하면 이 mrouter는 메시지의 헤더(header)를 분리해 내서 수신자의 멀티캐스트 주소를 알아낸다. 해당 멀티캐스트 주소에 속한 호스트가 자신의 네트워크에 있으면 해당 호스트에게 메시지를 전달하고, 아니면 자신의 터널의 반대편 끝 호스트에 전달한다. 한 호스트가 처음 멀티캐스트 주소에 속하고자 한다면, 자신의 네트워크에 있는 mrouter에게 알린다. 어디 다른 조직이나 기관에 알리는 그런 종류의 오버헤드는 전혀 없다. 이후로 mrouter는 주기적으로 아직도 그 멀티캐스트 주소에 속한 호스트가 자신의 네트워크에 있는지 물어보고, 속한 호스트는 자신이 아직 속해 있음을 mrouter에게 알려준다. 세번째가 threshold와 TTL이다. 이제 멀티캐스트 패킷 전송 원리와 전송 범위 조절 방법에 대해 설명한다. 터널이라는 가상의 링크는 실제 그 경로상에 여러 개의 멀티캐스트를 지원하지 않는 일반 라우터를 경유하게 된다. 따라서 이들 라우터가 멀티캐스트 패

킷의 최종 수신자 서브넷까지 패킷을 전송하기 위해서는 라우팅을 위한 Encaps ulation이 필요하다. Encapsulation이란 송신자 주소와 그룹 주소로 이루어진 멀티캐스트 패킷 앞에 mrouter간에 설정된 터널의 양끝단 주소를 덧붙여 전송하는 방법이다. 이렇게 Encapsulation된 멀티캐스트 패킷은 터널을 따라 일반 라우터를 지날 때, 기존의 일반 유니캐스트 인터넷 패킷과 같은 방법으로 라우팅되어 최종적으로 터널의 종착지인 mrouter로 전송될 수 있다. 이를 수신한 mrouter는 Encap sulation 헤더를 분리해 다시 자신의 서브넷상의 그룹 멤버에게 전송하여 멀티캐스트 전송을 이루게 된다. 그러나 이렇게 Encapsulation된 멀티캐스트 패킷이 전세계의 MBone을 통해 모두 전송이 되는 것은 아니다. 특히 화상, 음성 등을 포함하는 광대역 멀티미디어(multimedia) 데이터의 경우에는 실제 전송될 때 일반 인터넷 데이터와 같은 네트워크를 이용함으로 TCP/IP 네트워크의 특성상 심각한 영향을 줄 수도 있으므로 그 전송 범위를 조절할 필요가 있다. 예를 들면, 학교 내에서 교내 화상 회의를 열고자 할 때, 그 참석자가 모두 교내의 네트워크상에만 있다면 이것을 전체 MBone에 전송할 필요는 없을 것이다. 이와 같이 멀티캐스트 패킷의 전송 범위를 조절하기 위해서 MBone에서는 터널의 Threshold와 패킷의 TTL을 이용한다. MBone상의 각 터널은 자체의 Threshold를 가진다. Threshold는 터널의 허용 레벨을 나타내는 값으로, 전체 MBone 구조에서 일종의 계층적 구조를 가질 수 있다. 또한 모든 멀티캐스트 패킷은 자신의 TTL(Time-To-Live)을 가진다. 패킷의 TTL은 하나의 mrouter를 지날 때마다 1씩 감소하여 패킷의 수명을 나타내는 값이다. 송신자가 멀티캐스트 패킷을 MBone상에 전송할 때, 각 터널은 자신의 Threshold보다 큰 TTL을 가진 패킷만 통과시킴으로써 패킷의 전송 범위를 조절할 수 있다. 즉, 앞의 교내 화상 회의의 경우처럼 교내의 참여자에게만 패킷이 전송되기를 원하는 경우에는 송신자는 자신의 패킷 TTL을 외부와 연결된 터널의 Threshold보다 작게 전송하면 된다. 이러한 MBone의 특성을 고려하여 미리 터널의 계층적 구조를 잘 이용하면, 다른 인터넷 데이터 전송에 영향을 최소화하는 MBone 구조를 설계할 수 있다. 예를 들어 두 서브 네트워크 간에 터널이 threshold 16으로 다음과 같이 연결되어 있다면

tunnel 129.254.201.13 143.248.172.41 metric threshold 16

TTL 16 이상의 값을 갖는 패킷들만 두 서브 네트워크 간을 오고 갈 수 있다. 예를 들면 127~255 TTL 은 MBone 상의 모든 서브 네트워크에 멀티캐스트 스트림을 보내는 데 사용되며, 학교 내와 같은 소규모 지역망의 경우로 멀티캐스팅을 제한하기 위해서는 16 TTL 을 사용한다. 이러한 문제는 망 전체의 성능에 커다란 영향을 미칠 수 있는데, 예를 들어 일반적으로 정해진 비디오 스트림은 대역폭의 약 128Kbps, 혹은 T1 라인의 10% 정도를 소모한다. 고대역폭 세션의 동시 사용은 네트워크 전체를 포화시킬 수 있다. IETF 에서는 다음과 같이 threshold 와 TTL 가이드 라인을 정하였다.

[표 2] 터널 threshold 가이드라인

	TTL	threshold
IETF chan 1 low-rate GSM audio	255	224
IETF chan 2 low-rate GSM audio	223	192
IETF chan 1 PCM audio	191	160
IETF chan 2 PCM audio	159	128
IETF chan 1 video	127	96
IETF chan 2 video	95	64
local event audio	63	32

Internet-KIG MBone WG에서는 1996년 10월 회의에서 국내 MBone 사용의 일관성과 네트워크 자원의 효율적인 사용을 위하여 터널의 threshold와 세션 중계시 TTL 값에 대한 가이드라인을 수립하였다. 터널을 설정하거나, 새로운 세션을 생성하고자 할 때는 이 가이드라인을 따르기를 권장한다. 세션 중계시 지정하는 TTL 값에 대한 가이드라인은 [표 3]과 같다.

[표 3] 세션 중계시 TTL 가이드라인

	TTL
해외 중계	65 이상(127)
국내 중계	63 이하(63)
o 오디오	47 이하(필요시 63)
o 비디오	15 이하(15)
개별 기관 내 중계	

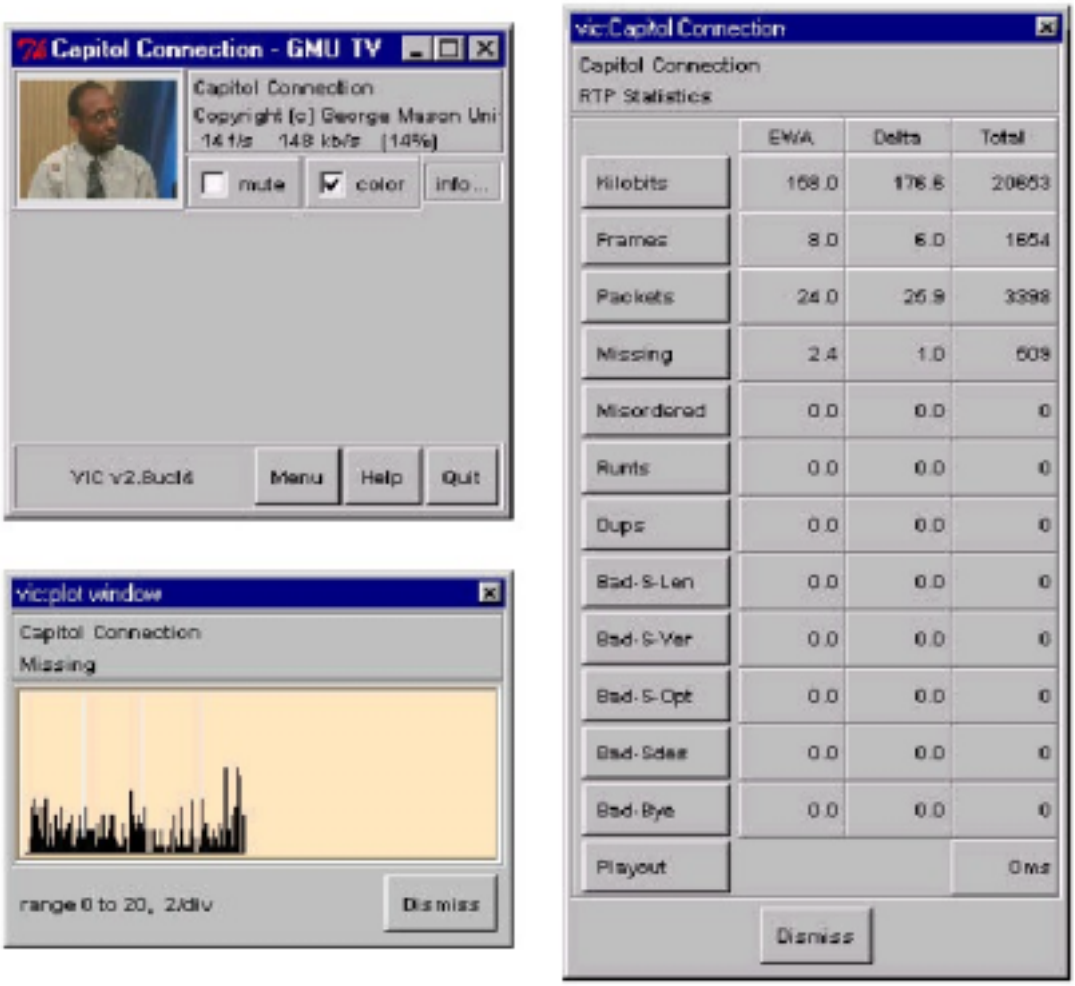
MBone의 주요 제약은 대역폭에 있다. 멀티캐스트 스트림이 대역폭에 영향을 받는 이유는 하나의 패킷이 MBone상의 모든 호스트로 전송될 수 있기 때문이다. 이러한 문제는 망 전체를 제어하는 데 있어 커다란 문제를 야기시킬 수 있다. 따라서 128Kbps 비디오 스트림(1~4 프레임/초)은 약 1~20 호스트 정도가 수신할 수 있는 정도의 대역폭을 사용하는 반면, 같

은 멀티캐스트 패킷은 라우터에 의해 일반적으로 제한을 받게 된다. 이러한 제어는 망제어 관점에서 타당하다. MBone은 다음과 같은 두 가지 방법으로 인터넷상에서 멀티캐스트 패킷 분배를 제어한다.

- 멀티캐스트 패킷의 생존 시간을 제한하는 방법 : TTL
- 멀티캐스트 트리 prunning 알고리즘을 사용하는 방법

멀티캐스트 서브 트리에 대한 자동적인 pruning/grafting 개발에 대한 연구가 활발히 진행중이나, 현재 사용중인 MBone 의 대부분은 threshold 에 따른 TTL 을 사용하고 있다.

다음으로 Mbone 에서 구동되는 응용 애플리케이션에 대하여 알아보자. 먼저 VIC 이다. [그림 10]은 Mbone 상에서 구동되는 VIC 애플리케이션의 구동 모습이다. VIC 는 Video Conferencing Tool 의 약자로서 Lawrence Berkeley National Lab.에서 UC Berkeley 대학과의 공동연구로 개발된 비디오 화상회의용 소프트웨어이다.



[그림 10] VIC 프로그램의 구동

강좌를 마치면서

이제까지 8회에 걸쳐서 자바의 멀티미디어 확장 패키지인 JMF에 관하여 살펴보고, 이를 통하여 화상회의 시스템 구축을 위한 캡쳐 장치의 인식부분, 영상 음성 데이터의 획득부분, 획득 데이터의 포맷 설정과 변경방법, RTP 세션 생성과 세션 컨트롤 방법, 세션 데이터의 수신 방법과 원격지에서의 데이터 재생에 관한 방법등에 관하여 알아보았으며, 애플릿 구동시의 애플릿 인증부분에 관해서도 살펴보았다. 또한 JMF와 더불어 또하나의 확장 패키지인 JSDT의 결합부분과 멀티미디어 화상회의 시스템 구축을 위한 Mbone 환경 및 응용 애플리케이션을 살펴보았다. 실제 JMF 확장 패키지만을 이용하여 화상회의 시스템을 구축하기에는 아직까지 많은 제약점이 있기는 하지만, 많은 개발자들의 참여와 논의를 통하여 더욱 우수하고 안정적인 JMF의 신버젼이 발표되기를 바란다. 본 원고에서 언급한 모든 프로그램들의 소스코드와 컴파일된 코드들은 프로그램 세계의 자료실이나 필자의 홈페이지(http://myhome.naver.com/kingseft)를 참조하기 바란다.