자바 ATM API의 설계 및 구현

성종진 · 이근구 · 김장경
한국전자통신연구원 표준연구센터

Design and Implementation of Java ATM API

Jongjin Sung · Keun-Ku Lee · Jangkyung Kim
ETRI Protocol Engineering Center
E-mail : jsung@pec.etri.re.kr

요 약

이 논문에서는 자바 환경에서 사용될 수 있는 ATM API를 정의하고, 이렇게 정의된 자바 ATM API를 WinSock 2 환경 상에서 구축할 경우 요구되는 소프트웨어의 구조와 구현 방법을 제시한다. 제안된 자바 ATM API는 기존의 자바 프로그래밍 환경에서 제공되는 JDK 중에서 인터넷 통신 기능을 정의하고 있는 java.net 패키지의 확장된 형태로 정의되었다. 동시에 순수 ATM 서비스의 표준인 ATM 포럼의 "*Native ATM Services: Semantic Description, Version 1.0*" 규격에 따른 표준화된 ATM 서비스 기능들을 제공할 수 있도록 정의되었다. 표준화된 ATM 서비스 제공을 위해 java.net에 추가적으로 정의된 자바-ATM API 용 클래스로는, ATM 어드레싱을 위한 AtmAddress, BLLI/BHLI 정보의 이용을 위한 AtmBLLI와 AtmBHLI, 소켓 개념의 통신 프로그래밍을 위한 AtmSocket, AtmServerSocket, AtmMulticastSocket, AtmSocketImpl 그리고 ATM 통신의 장점인 연결의 특성 표현을 위한 AtmConnAttr 등이다.

I. 서 론

이 논문에서는 ATM (Asynchronous Transfer Mode) 서비스와 자바 (Java) 운영 환경을 결합하는 한가지 방법을 제시한다. 즉 ATM 통신 서비스의 인터페이스인 ATM API를 자바 언어로 정의하여, 자바 프로그래밍 환경에서 순수 ATM 서비스를 이용할 수 있도록 하는 자바 ATM API를 설계하고 구현한 내용을 제시한다.

ATM 통신 환경에서 ATM 계층과 ATM 적응계층 (AAL: ATM Adaptation Layer)의 서비스를 직접 응용에게 제공하여 순수한 ATM 서비스를 이용하도록 하는 ATM API는 ATM 서비스의 기능과 성능을 최적으로 제공할 수 있는 응용 프로그래밍 인터페이스라 할 수 있다[1]. 마이크로소프트 윈도우 운영체제 환경에서는 WinSock 2 API가 ATM API의 기능을 수행할 수 있도록 확장 정의[2]되면서 사실상 업계의 표준 ATM API로 인정받고 있으며, 유닉스와 애플 운영체제 환경에서는 XTI와 Socket이 ATM API 기능을 수행할 수 있도록 확장 정의[3]되면서 역시 업계의 표준 ATM API로 인식되고 있는 실정이다.

근래에는 마이크로소프트 윈도우나 유닉스와 같은 운영체제 상에서 자바 가상 운영체제를 이용하는 응용 프로그램의 개발이 늘고 있다. 통신을 기본으로 하여 운영된다고 할 수 있는 자바 운영체제는 현재 인터넷 통신 서비스를 주로 이용하고 있다. 그러나 자바 환경에서도 ATM과 같은 우수한 통신 서비스의 이용이 곧 확산될 것이고 순수 ATM 서비스를 활용하려는 필요성이 대두될 것으로 보인다.

이 논문에서 제안하는 자바 ATM API는 순수 ATM 서비스를 제공함에 있어서, 기능상의 의미적 내용(semantic) 면에서는 ATM 포럼의 표준인 "Native ATM Services: Semantic Description, Version 1.0"[4]을 따르도록 하였고, API의 외형적 형태(syntax) 면에서는 기존의 Java Core API (JDK)[5]의 통신용 패키지인 java.net 패키지에 정의된 클래스들의 형태를 따르도록 하였다.

II. 자바 ATM API의 정의

2.1 표준화된 형태의 자바 ATM API의 제공

자바 환경에서의 통신 응용 제작시 ATM 통신 서비스를 이용할 수 있도록 하기 위한 자바 API는 여러 형태로 정의될 수 있다. 그러나 하나의

표준화된 형태의 API를 정의하여 제공하는 것이 바람직하므로 ATM 포럼에서는 이 자바 ATM API를 위한 표준을 제정하기 위한 움직임을 시작하고 있다.

이 논문에서는 Java Core API 패키지들 중의 하나인 java.net 패키지를 확장 정의하여 표준화된 자바 ATM API의 형태로 사용하는 방안을 제안한다. 기존의 TCP/UDP 통신 서비스에 대한 API를 정의하고 있는 java.net 패키지에 ATM 통신 서비스를 위한 클래스를 추가하는 것이다. 이 때 이 추가된 클래스들은 순수 ATM 서비스의 표준인 ATM 포럼의 "Native ATM Services: Semantic Description, Version 1.0" 규격에 따른 표준화된 ATM 서비스 기능들을 제공할 수 있도록 고려되었다.

기존 java.net을 바탕으로 순수 ATM 서비스 기능을 추가하여 자바 ATM API를 정의하는 방법을 선택한 이유는 대부분의 네트웍 프로그래머들이 기존의 java.net에 정의되어 있는 클래스들을 이용하는 것에 익숙해져 있기 때문에 기존 API의 모습과 사용방법을 최대한 수용하기 위해서 이다. ATM 서비스를 위해 추가된 클래스들의 형태와 사용방법은 TCP/UDP 서비스를 위한 클래스들과 유사하게 정의하였다. 또한 java.net에 포함되어 있는 소켓(socket) 관련 클래스들은 BSD 소켓을 수용하여 정의된 것이므로 기존의 소켓 프로그래밍에 익숙해져 있는 네트웍 응용 프로그램 제작자들에게는 더욱 쉽게 받아들여질 수 있을 것이다.

2.2 Java.net의 확장 정의

일반적으로 자바 API라고 하면 JDK (Java Development Kit) API를 의미하며 이 논문에서 참조한 버전은 "Java Platform 1.1.6 Core API Specification"[5]이다. 이 Java Core API는 여러 종류의 API 패키지(package)들로 구성되어 있다. 이 패키지들 가운데 통신 서비스와 관련된 것으로는 java.net 패키지가 존재한다. Java.net 패키지는 URL, TCP 소켓, UDP 소켓, IP 주소, binary-to-text 변환 등과 관련된 기능들을 제공한다.

이러한 기존의 java.net 패키지에 다음과 같은 새로운 클래스와 인터페이스를 추가하여 순수 ATM 서비스를 지원할 수 있도록 한다.

- Class java.net.AtmAddress
- Class java.net.AtmBLLI
- Class java.net.AtmBHLI
- Class java.net.AtmMulticastSocket
- Class java.net.AtmServerSocket
- Class java.net.AtmSocket
- Class java.net.AtmSocketImpl
- Class java.net.AtmConnAttr
- Interface java.net.AtmSocketImplFactory

순수 ATM 서비스의 지원을 위해서는 먼저 ATM 어드레싱에 필요한 ATM 주소와 BLLI (Broadband Low Layer Information) 정보, BHLI (Broadband High Layer Information) 정보를 표현할 수 있어야 한다. 이를 위해서 AtmAddress, AtmBLLI, AtmBHLI 클래스들을 정의하였다. java.net이 기본적으로 소켓 개념을 바탕으로한 통신 API로 정의되어 있으므로 ATM 통신을 위한 확장된 java.net에서도 역시 소켓 개념의 통신에서 필요한 서버/클라이언트 소켓 클래스들이 정의되어야 한다. ATM 소켓 클래스들로는 AtmServerSocket, AtmSocket, AtmSocketImpl을 정의하였다. 그리고 ATM 통신의 장점인 연결에 대한 특성을 표현하기 위한 클래스로 AtmConnAttr를 정의하였다. 인터페이스 AtmSocketImplFactory는 AtmSocket과 AtmServerSocket의 실제 동작 기능을 구현하는 AtmSocketImpl을 생성하고자 할 때 필요한 인터페이스로 정의하였다.

확장된 java.net 패키지의 새로운 클래스들은 그림 1과 같은 계층구조를 갖는다.

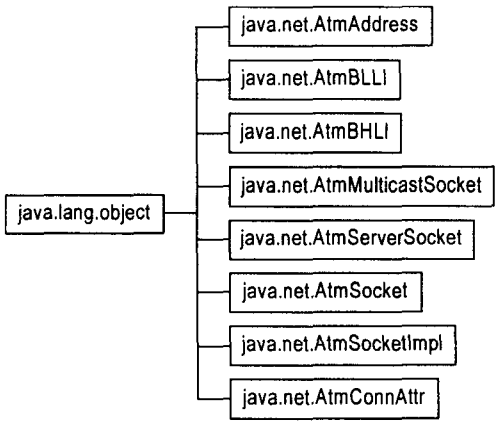

그림 1. 순수 ATM 서비스를 위해 확장된 java.net 패키지 클래스 계층구조

2.3 ATM 서비스를 위한 java.net 클래스 정의

이 절에서는 앞에서 설명한 바와 같이 순수 ATM 서비스를 제공하기 위하여 java.net 패키지에 추가되어야 하는 클래스들의 정의 내용을 기술한다.

2.3.1 Class java.net.AtmAddress

AtmAddress 클래스는 다음과 같이 정의하였다.

```
public final class AtmAddress extends Object
        implements Serializable {
    // Variables
    int addressType;
    Int numOfDigits;
    byte addr[20];

    // Constants used for AddressType
    Private static final int ATM_E164;
    Private static final int ATM_AESA;

    // Methods
    public boolean isMulticastAddress();
    public String getHostName();
    public byte[] getAddress();
    public String getHostAddress();
    public boolean equals(Object obj);
    public String toString();
    public static AtmAddress getByName(String host)
        throws UnknownHostException;
    public static AtmAddress[] getAllByName(String
        host) throws UnknownHostException;
    public static AtmAddress getLocalHost() throws
        UnknownHostException;
}
```

AtmAddress는 ATM 주소체계를 이용하기 위한 클래스이다. 이 클래스는 IP 어드레스 제공을 위한 기존의 InetAddress 클래스와 유사하게 정의한 것으로 InetAddress에서 사용되는 메소드들과 유사한 기능을 수행하는 메소드들을 정의하였다.

새로운 AtmAddress 객체를 생성하려면 getLocalHost(), getByName() 또는 getAllByName() 메소드를 이용하도록 한다.

2.3.2 Class java.net.AtmBLLI

ATM 시스템은 AtmAddress 정보로 식별이 가능하다. 그러나 그 시스템 내부에 존재하는 여러 응용 프로그램들 중에서 특정한 것을 식별해내기 위해서는 BLLI 정보와 BHLI 정보가 필요하다. 이 정보를 Java ATM API에서는 각각 AtmBLLI와 AtmBHLI로 나타내도록 하였다.

```
public final class AtmBLLI extends Object
        implements Serializable {
    // Variables
    Int layer2Protocol;
    Int layer2UserSpecifiedProtocol;
    Int layer3Protocol;
    Int layer3UserSpecifiedProtocol;
    Int layer3IPI;
    byte snapID[5];
}
```

AtmBLLI 클래스는 하위 계층인 2 계층과 3계층의 정보를 표현하고 있다.

2.3.3 Class java.net.AtmBHLI

```
public final class AtmBLLI extends Object
        implements Serializable {
    // Variables
    Int layer2Protocol;
    Int layer2UserSpecifiedProtocol;
    Int layer3Protocol;
    Int layer3UserSpecifiedProtocol;
    Int layer3IPI;
    byte snapID[5];
}
```

AtmBHLI 클래스는 상위계층의 정보를 표현한다.

2.3.4 Class java.net.AtmMulticastSocket

AtmMulticastSocket 클래스의 내부 상세 정의는 아직 미완성이다. ATM 포럼에서 작업이 진행중에 있는 "Native ATM Service: Semantic Description, Version 2.0"이 완성되면 작성 될 수 있을 것이다.

```
public class AtmMulticastSocket extends Object
{
    // To be defined later;
}
```

2.3.5 Class java.net.AtmServerSocket

```
public class AtmServerSocket extends Object {
    // Constructors
    public AtmServerSocket(AtmBLLI blli, AtmBHLI
        bhli) throws IOException;
    public AtmServerSocket(AtmBLLI blli, AtmBHLI
        bhli, int backlog) throws IOException;
    public AtmServerSocket(AtmBLLI blli, AtmBHLI
        bhli, int backlog, AtmAddress bindAddr)
        throws IOException;

    // Methods
    public AtmAddress getAtmAddress();
```

```
    public AtmBLLI getLocalAtmBlli();
    public AtmBHLI getLocalAtmBhli();
    public AtmConnAttr getConnectionAttribute();
    public AtmSocket accept() throws IOException;
    protected final void implAccept(AtmSocket s)
        throws IOException;
    public void close() throws IOException;
    public synchronized void setSoTimeout(int
        timeout) throws SocketException;
    public synchronized int getSoTimeout() throws
        IOException;
    public String toString();
    public static synchronized void
        setAtmSocketFactory(AtmSocketImplFactory
        fac) throws IOException;
}
```

AtmServerSocket 클래스는 순수 ATM 서비스를 위한 서버 소켓을 표현한다. 이 서버 소켓은 ATM 네트웍에서의 어떠한 요구자로부터의 특정 요구를 받아들여 그 요구에 따른 동작을 수행한 후, 그 결과를 그 요구자에게 돌려준다.

ATM 서버 소켓의 실제 동작은 AtmSocketImpl 객체에 의해 수행된다. accept(), close() 등과 같은 기본적인 행위를 위한 메소드들 외에도 ATM 주소와 연결의 특성을 확인할 수 있도록 하기 위한 getAtmAddress(), getAtmBlli(), getAtmBhli(), getConnectionAttribute() 등과 같은 메스드들을 정의하였다.

2.3.6 Class java.net.AtmSocket

```
public class AtmSocket extends Object {
    // Constructors
    protected AtmSocket();
    protected AtmSocket(AtmSocketImpl impl) throws
        SocketException;
    public AtmSocket(String host, AtmBLLI blli,
        AtmBHLI bhli, AtmConnAttr connattr) throws
        UnknownHostException, IOException;
    public AtmSocket(AtmAddress address, AtmBLLI
        blli, AtmBHLI bhli, AtmConnAttr connattr)
        throws IOException;
    public AtmSocket(String host, AtmBLLI blli,
        AtmBHLI bhli, AtmAddress localAddr,
        AtmBLLI localBlli, AtmBHLI localBhli,
        AtmConnAttr connattr) throws IOException;
    public AtmSocket(AtmAddress address, AtmBLLI
        blli, AtmBHLI bhli, AtmAddress localAddr,
        AtmBLLI localBlli, AtmBHLI localBhli,
        AtmConnAttr connattr) throws IOException;

    // Methods
    public AtmAddress getAtmAddress();
    public AtmBLLI getAtmBlli();
    public AtmBHLI getAtmBhli();
    public AtmConnAttr getConnectionAttribute();
    public AtmAddress getLocalAddress();
    public AtmBLLI getLocalAtmBlli();
    public AtmBHLI getLocalAtmBhli();
    public InputStream getInputStream() throws
        IOException;
    public OutputStream getOutputStream() throws
        IOException;
    public void setSoLinger(boolean on, int val)
        throws SocketException;
    public int getSoLinger() throws SocketException;
    public synchronized void setSoTimeout(int
        timeout) throws SocketException;
    public synchronized int getSoTimeout() throws
        SocketException;
    public synchronized void close() throws
        IOException;
    public String toString();
    public static synchronized void
        setAtmSocketImplFactory(AtmSocketImplFact
        ory fac) throws IOException;
}
```

AtmSocket 클래스는 순수 ATM 서비스를 위한 클라이언트 소켓을 표현한다. 소켓은 두 시스템간의 통신을 수행할 때 하나의 끝점에 해당한다. ATM 소켓의 실제 동작은 AtmSocketImpl 객체에 의해 수행된다.

AtmSocket의 생성자(constructor)는 연결이 되어 있지 않은 소켓의 생성만 수행하는 것으로부터, 소켓 생성 후 특정 상대방과의 연결 설정과 자신의 ATM 주소와의 바인딩까지의 기능들을 일괄적으로 수행하는 것까지 여섯 가지를 정의하였다. getAtmAddress(), getAtmBlli(), getAtmBhli()를 통해 연결 상대방의 주소와 BLLI/BHLI 정보를 알아내고, getLocalAddress(),getLocalAtmBlli(), getLoaclAtmBhli()를 통해 자신의 주소와 BLLI/BHLI 정보를 알아볼 수 있게 한다. getConnectionAttribute()를 통해서는 연결의 특성에 대한 정보를 파악할 수 있다.

AtmSocket 클래스의 주요 기능인 데이터 송수신 기능을 위한 메소드로는 getInputStream()과 getOutputStream()을 정의하였다. 이들은 각각 데이터 수신과 송신 기능을 수행하게 된다.

2.3.7 Class java.net.AtmSocketImpl

```
public abstract class AtmSocketImpl extends
        Object {
    // Variables
    protected FileDescriptor fd;
    protected AtmAddress address;
    protected AtmBLLI blli;
    protected AtmBHLI bhli;
    protected AtmBLLI localBlli;
    protected AtmBHLI localBhli;

    // Constructors
    public AtmSocketImpl();

    // Methods
    protected abstract void create() throws
```

```
        IOException;
    protected abstract void connect(String host,
        AtmBLLI blli, AtmBHLI bhli, AtmConnAttr
        connattr) throws IOException;
    protected abstract void connect(AtmAddress
        address, AtmBLLI blli, AtmBHLI bhli,
        AtmConnAttr connattr) throws IOException;
    protected abstract void bind(AtmAddress host,
        AtmBLLI blli, AtmBHLI bhli) throws
        IOException;
    protected abstract void listen(int backlog) throws
        IOException;
    protected abstract void accept(AtmSocketImpl s)
        throws IOException;
    protected abstract InputStream getInputStream()
        throws IOException;
    protected abstract OutputStream
        getOutputStream() throws IOException;
    protected abstract int available() throws
        IOException;
    protected abstract void close() throws
        IOException;
    protected FileDescriptor getFileDescriptor();
    protected AtmAddress getAtmAddress();
    protected AtmBLLI getAtmBlli();
    protected AtmBHLI getAtmBhli();
    protected AtmBLLI getLocalAtmBlli();
    protected AtmBHLI getLocalAtmBhli();
    protected AtmConnAttr getConnectionAttribute();
    public String toString();
}
```

AtmSocketImpl 클래스는 ATM 클라이언트/서버 소켓들의 동작들을 실제로 구현하게 되는 곳이다. 이 클래스 자체는 추상클래스 (abstract class)로써 실제 동작 구현 내용을 담게될 클래스들의 공통 수퍼클래스 (super class)가 된다. 이 클래스에서 정의된 내용 그대로를 구현하게 되는 클래스를 "PlainAtmSocketImpl" 이라는 이름으로 만들어 두어 이용하도록 한다. 이 클래스에 구현되어 있는 메소드들은 AtmServerSocket 클래스와 AtmSocket 클래스에 의해 사용된다. 이러한 방식의 사용 예를 그림 5에서 찾아볼 수 있다.

2.3.8 Class java.net.AtmConnAttr

```
public final class AtmConnAttr extends Object {
    // Variables
    int aalType;                int aal5FwdMaxSdu;
    int aal5BakMaxSdu;          int aal5SscsType;
    int userDefinedAalInfo;     int fwdPcrClp0;
    int fwdPcrClp1;             int bakPcrClp0;
    int bakPcrClp1;             int fwdScrClp0;
    int fwdScrClp1;             int bakScrClp0;
    int bakScrClp1;             int fwdMbsClp0;
    int fwdMbsClp1;             int bakMbsClp0;
    int bakMbsClp1;             int bestEffort;
    int fwdTagging;             int bakTagging;
    int bearerClass;            int trafficType;
    int timeReq;                int clippingInd;
    int connectConfig;          int appIdType;
    int appId;                  int layer2Id;
    int layer2Mode;             int layer2WindowSize;
    int layer2UserId;           int layer3Id;
    int layer3Mode;             int layer3PacketSize;
    int layer3WindowSize;       int layer3UserId;
    int layer3IpiId;            int layer3OuiId;
    int layer3PidId;            int calledAddrFormat;
    int calledAddr;             int calledSubaddrType;
    int calledSubaddr;          int callingAddrFormat;
    int callingAddr;            int presentationInd;
    int ScreeningInd;           int callingSubaddrType;
    int callingSubaddr;         int causeCoding;
    int causeLocation;          int causeValue;
    int causeDiagnostics;       int fwdQosClass;
    int bakQosClass;            int networkId;
}
```

AtmConnAttr 클래스는 ATM 연결의 특성을 표현한다. 이 클래스에서는 AAL 정보, ATM 트래픽 설명, broadband bearer capabilities, BLLI 정보, BHLI 정보, 어드레싱 정보, QoS 정보 등의 내용을 변수로 정의하였다. 이 내용들은 "Native ATM Services: Semantic Description, Version 1.0"의 "ANNEX A: Connection Attributes" 부분을 충실히 반영한 것이다. 이들 변수에 의해 표현되는 연결의 특성을 확인하고자 할때는 주로 AtmServerSocket과 AtmSocket 클래스에 정의되어 있는 getConnectionAttribute() 메소드를 이용하도록 한다.

2.3.9 Interface java.net.AtmSocketImplFactory

```
public interface AtmSocketImplFactory {
    public abstract AtmSocketImpl
    createAtmSocketImpl();
}
```

이 인터페이스는 AtmServerSocket과 AtmSocket에 의해 사용된다.

AtmServerSocket과 AtmSocket 클래스의 기능들을 실제로 구현해둔 AtmSocketImpl 객체를 생성할 때 createAtmSocketImpl() 메소드가 이용된다.

III. 자바 ATM API의 구조 및 구현

java.net을 확장하여 정의한 자바 ATM API는 그림 2의 (a)와 같은 구조를 통하여 제공될 수 있다. 이 구조는 마이크로소프트 윈도우 운영체제 상에서 자바 가상운영체제 환경을 구축할 경우에 해당되는 구조이다. 유닉스나 기타 운영체제 상에서 자바운영체제 환경을 구축했을 경우는 그 구조가 다를 수 있다. 그림에서는 기존의 TCP/UDP 지원을 위한 java.net의 이용(그림 2의 (b))과 ATM 서비스 지원을 위해 확장된 자바 ATM API의 이용(그림 4의(a))시의 소프트웨어 구조를 비교하여 나타내었다. 우리가 정의한 자바 ATM API는 java.net을 확장한 것이므로 TCP/UDP 지원을 위한 기존의 java.net의 이용

구조와 유사하다.

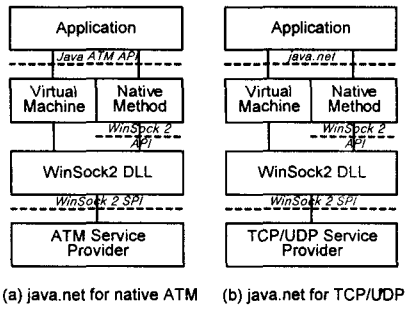

그림 2 WinSock 상에서의 java.net 이용구조

java.net은 TCP/UDP 지원을 위한 대부분의 기능을 native 메소드를 통해 제공하고 있다. java.net 클래스들이 정의하고 있는 소켓 기능들은 결국은 C/C++ 언어로 구현되어 있으며 이 C/C++로 구현된 함수들을 java.net 내부 클래스들이 native 메소드 방식으로 불러 쓰도록 되어 있다. 이 논문에서 제안하는 확장된 java.net의 ATM 지원 클래스 내부의 소켓 기능들도 native 메소드 방식으로 구현하도록 하였다. 그림 2의 (a)에서 보면 WinSock 상에서 제공되는 확장된 java.net 형태인 자바 ATM API가 native 메소드 방식으로 WinSock 2 API를 이용하도록 되어 있다.

그림 3에서는 응용 프로그램에서 자바 ATM API를 이용할때 실제로 어떠한 방식으로 ATM 서비스를 제공받게 되는지를 프로그램 소스 차원에서 상세하게 나타내었다. 그림 3의 윗쪽에서부터 보면, 응용 프로그램인 application.java에서 AtmSocket.java 파일에 정의되어 있는 AtmSocket() 생성자를 이용하여 ATM 소켓을 생성하고 연결을 설정하는 부분을 보여주고 있다. 여기에서 사용된 AtmSocket() 생성자의 기능은 ATM 소켓 생성 (create()) 후 상대방 시스템 'hostname'과 연결의 특성 'connattr'를 가지고 연결을 설정 (connect()) 하는 것 까지를 일괄적으로 수행한다.

수행되는 과정 중에서 연결 설정 과정을 보다 자세하게 들여다보면, AtmSocket() 생성자는 연결을 설정하기 위해 AtmSocketImpl 클래스의 connect() 메소드를 이용하고 있음을 알 수 있다. 이 메소드는 PlainAtmSocketImpl.java 파일에 구현되어 있으며 그 메소드 내부에서는 다시 C/C++ 함수인 atmSocketConnect()를 native 메소드로 불러 사용하고 있다. C/C++ 함수 atmSocketConnect()는 결국 WinSock 2 API를 사용해서 ATM 연결을 생성하게 된다.

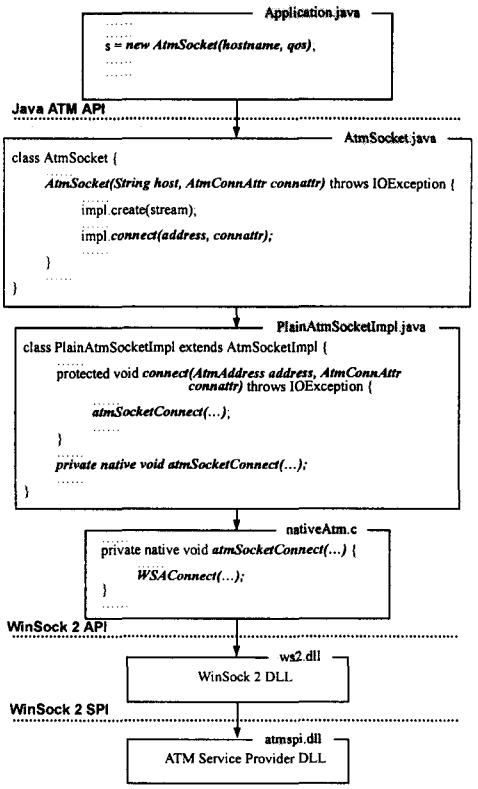

그림 3. 연결 설정 기능 부분을 나타내는 WinSock 환경에서의 자바 ATM API 구현 소스 코드 구조

그림 3에서는 연결 설정 부분만을 예로 들어 도식적으로 설명하였으나 앞에서 정의한 자바 ATM API의 나머지 부분들도 이러한 방법으로 기능들을 구현하게 된다.

IV. 결 론

이 논문에서는 자바 환경에서 순수한 ATM 서비스를 이용할 수 있도록 하는 자바 ATM API를 정의하여 제안하였으며, 그 구현 방법을 제시하였다. 제안된 자바 ATM API는 기존의 Java Core API (또는 JDK API) 패키지들 중에서 인터넷 통신 기능을 정의하고 있는 java.net 패키지를 확장한 형태로 정의되었다.

이 자바 ATM API의 기능상 내용면에서는 순수 ATM 서비스의 표준인 ATM 포럼의 "Native ATM Services: Semantic Description, Version 1.0" 규격에 따른 표준화된 ATM 서비스 기능들을 제공할 수 있도록 정의되었다.

ATM 어드레싱을 위한 AtmAddress, BLLI/BHLI 정보의 이용을 위한 AtmBLLI와 AtmBHLI, 소켓 개념의 통신 프로그래밍을 위한 AtmSocket, AtmServerSocket, AtmMulticastSocket, AtmSocketImpl 그리고 ATM 통신의 장점인 연결의 특성 표현을 위한 AtmConnAttr 등이 자바 ATM API의 클래스로 정의되어 제안되었으며, 순수 ATM 서비스 기능 구현을 위한 프리미티브들은 이 클래스들의 내부 함수들로 정의되었다.

이 논문에서는 이렇게 정의된 자바 ATM API를 WinSock 2 환경 상에서 구축할 경우 요구되는 소프트웨어의 구조와 구현 방법을 또한 제시하였다.

참고문헌

[1] Ross, T., "ATM APIs: The Missing Links," Data Communications, pp. 119-128, Sep., 1995.

[2] ATM Forum, 'WinSock 2 ATM Annex,' ATM_Forum/96-0190, Feb., 1996.

[3] ATM Forum, 'XNET ATM API Specifications, Appendix X and Y,' ATM_Forum/96-1169, Oct., 1996.

[4] ATM Forum, 'Native ATM Services: Semantic Description, Version 1.0,' af-saa-0048.000, Feb., 1996.

[5] Sun Microsystems, Inc., 'Java Platform 1.1.6 Core API Specification,' http://java.sun.com/products/jdk/1.1/docs/index.html

[6] Lauback, M., "Classical IP and ARP over ATM," IETF RFC 1577, Jan., 1994.

[7] ATM Forum, 'LAN Emulation over ATM, Version 1.0,' af-lane-0021.000, Jan., 1995.

[8] ATM Forum, 'Multiprotocol over ATM, Version 1.0,' Just Passed in the Final Ballot, Jul., 1997.

[9] ATM Forum, 'ATM User-Network Interface Specification, Version 3.1,' af-uni-0010.002.