

네트워크 프로그래밍

목차

1. 네트워크 기본개념
2. 네트워크 프로그래밍 기초
3. TCP 프로그래밍
4. UDP 프로그래밍
5. 객체지컬화를 이용한 네트워크 프로그래밍

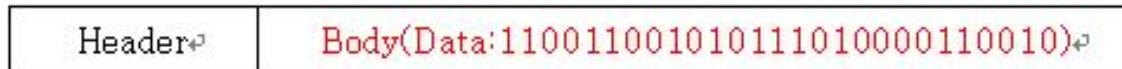
1. 네트워크 기본개념

[1] 네트워크(Network)와 네트워킹(Networking)

- 정보 기술에서 네트워크는 통신 경로들에 의해 상호 연결된 일련의 지점(point)들이나 노드(node)들을 의미하며 네트워킹이란 네트워크에 연결된 디바이스들 간의 데이터 교환을 의미한다.
- 데이터 통신에서는 통신 장치간의 데이터 교환에 필요한 모든 규약의 집합체를 프로토콜(protocol) 이라 말하며 물리적인 부분과 논리적인 부분으로 구성된다.
- 물리적 측면의 프로토콜은 데이터 전송에 사용되는 전송매체, 접속용 커넥터 및 전송 신호와 같은 물리적 요소를 의미한다.
- 논리적 측면에서의 프로토콜은 데이터의 표현, 의미와 기능, 데이터 전송절차 등을 의미한다

1. 네트워크 기본개념

네트워크를 통해 데이터를 전송할 때 비트로 전달하며 패킷을 사용한다. 패킷은 송신자와 수신자의 주소와 패킷이 손상되지 않았음을 보장하기 위한 checksum과 네트워크로 전송할 때 필요한 기타 유용한 정보들을 하는 헤더와 전송할 데이터를 바이트 그룹으로 포함하는 바디로 구분된다.



프로토콜은 데이터를 전송할 때 데이터를 주로 1024 비트씩 여러 개의 묶음으로 만들어 그 묶음을 보내는 방식으로 각종 에러검사용 정보를 담아 보내기 때문에 송 도중 에러가 발생하면 전체 데이터를 다시 전송하지 않고 해당 묶음만 다시 전송하기 때문에 속도가 빠르다.

1. 네트워크 기본개념

[2] OSI7계층

- 네트워크 구조는 1970년대 말 IOS(International Organization for Standardization)에 의해서 만들어진 OSI7계층(Open Systems Interconnection 7 Layer)을 기초로 한다. OSI7 계층은 아래와 같다.
- 7 계층 : 애플리케이션(NFS, FTP, HTTP)
- 6 계층 : 프리젠테이션(XDE, XML, ASCII, Java Serialization)
- 5 계층 : 세션(Sun RPC, DCE RPC, IIOP, RMI)
- 4 계층 : 트랜스포트(TCP, UDP)
- 3 계층 : 네트워크(IP)
- 2 계층 : 데이터 링크(wire formats for messages)
- 1 계층 : 물리(wires, signaling)

1. 네트워크 기본개념

물리계층 : 노드(Node) 간 네트워크 통신을 하기 위한 가장 저수준의 계층으로서 상위 계층인 데이터 링크 계층에서 형성된 데이터 패킷을 전기 신호나 광신호로 바꾸어 송수신하는 역할을 담당한다

데이터 링크 계층(Data Link Layer)

네트워크 계층으로부터의 메시지를 비트로 변환해서, 물리 계층이 전송할 수 있게 한다. 또한 메시지를 데이터 프레임의 포맷으로 만들고, 수신지와 발신지 하드웨어 주소를 포함하는 헤더를 추가한다.

네트워크 계층(Network Layer)

네트워크 계층은 다른 장소에 위치한 두 시스템 간의 연결성과 경로 선택을 제공한다. 라우팅 프로토콜을 사용하여 서로 연결된 네트워크를 통한 최적의 경로를 선택하며, 선택된 경로를 따라 정보를 보낸다.

트랜스포트 계층(Transport Layer)

애플리케이션 계층, 프리젠테이션 계층 그리고 세션 계층이 애플리케이션에 관련되어 있다면 하위 계층 네 개는 데이터 전송과 관련되어 있다. 따라서 트랜스포트 계층은 데이터 전송 서비스를 제공한다.

1. 네트워크 기본개념

세션 계층(Session Layer)

세션 계층은 애플리케이션 간 세션을 구축하고 관리하며 종료시키는 역할을 한다

프리젠테이션 계층(Presentation Layer)

프리젠테이션 계층은 한 시스템의 애플리케이션에서 보낸 정보를 다른 시스템의 애플리케이션 계층에서 읽을 수 있게 하는 곳이다.

애플리케이션 계층(Application Layer)

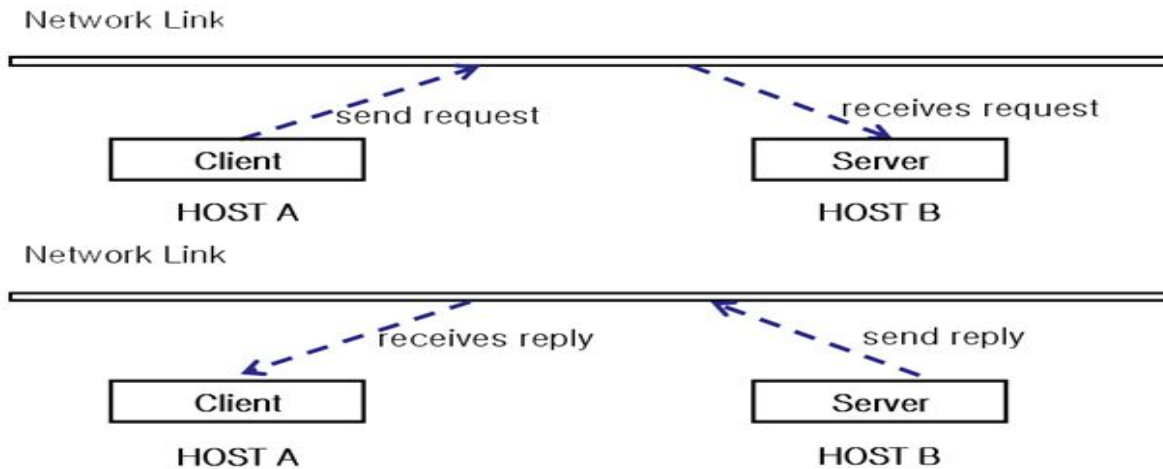
사용자와 컴퓨터가 통신하는 곳으로, 통신하고자 하는 상대를 식별하고 그 상대와의 통신을 확보하는 역할을 한다.

1. 네트워크 기본개념

[3] 프로그램 관계모델

▣ 클라이언트/서버 모델

- 클라이언트/서버는 두 프로그램 사이의 관계를 나타내는 용어로, 하나는 Client 프로그램으로 서비스를 요청하고 다른 하나는 Server 프로그램으로서 그 요청에 대해서 서비스를 제공하는 방식에 의하여 상호 통신하는 관계를 나타내는 용어이다.



클라이언트/서버 모델은 현재 네트워크 컴퓨팅의 중심 개념으로 되어 있다. 인터넷의 기본 프로그램인 TCP/IP도 클라이언트/서버 모델을 사용하여 작성된다.

1. 네트워크 기본개념

□ peer-to-peer 모델

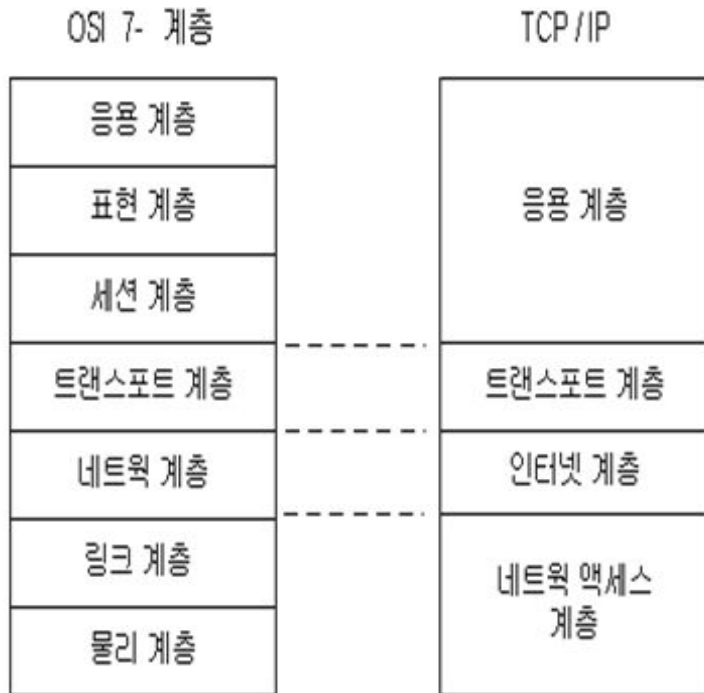
- Peer-to-peer 모델은 클라이언트/서버 모델과 대비되는 모델이며 일반적으로 P2P라는 용어로 사용된다. 클라이언트/서버 환경에서는 서비스 제공자와 서비스 수요자가 명확하게 구분된다. 서비스 제공자인 서버가 서비스 수요자인 클라이언트에게 일방적으로 서비스를 제공하는 것이다.
- P2P에서는 하나의 사용자가 서버이자 클라이언트 역할을 수행한다. 이는 네트워크에 연결되어있는 모든 컴퓨터들이 서로 대등한 동료의 입장에서 데이터나 주변장치 등을 공유할 수 있다는 것을 의미한다. 따라서 하나의 서버로 집중되는 요청으로 인해 발생하는 부하를 줄일 수 있다.

1. 네트워크 기본개념

□TCP/IP 프로토콜

- ▶ ARPANET에서 임의의 서브네트워크를 통해 접속된 장비들의 종점간 연결과 라우팅을 제공하기 위하여 미국 국방성에서 제정한 프로토콜이 TCP/IP 프로토콜이다.
- ▶ TCP/IP 프로토콜을 구성하는 주요 두 프로토콜은 IP와 TCP이다.
- ▶ TCP/IP 프로토콜이라고 하면 TCP와 IP 두 프로토콜만을 지칭하는 것이 아니라 UDP(User Datagram Protocol), ICMP(Internet Control Message Protocol), ARP(Address Resolution Protocol), RARP(Reverse ARP) 등 관련된 프로토콜을 통칭하는 것이다.

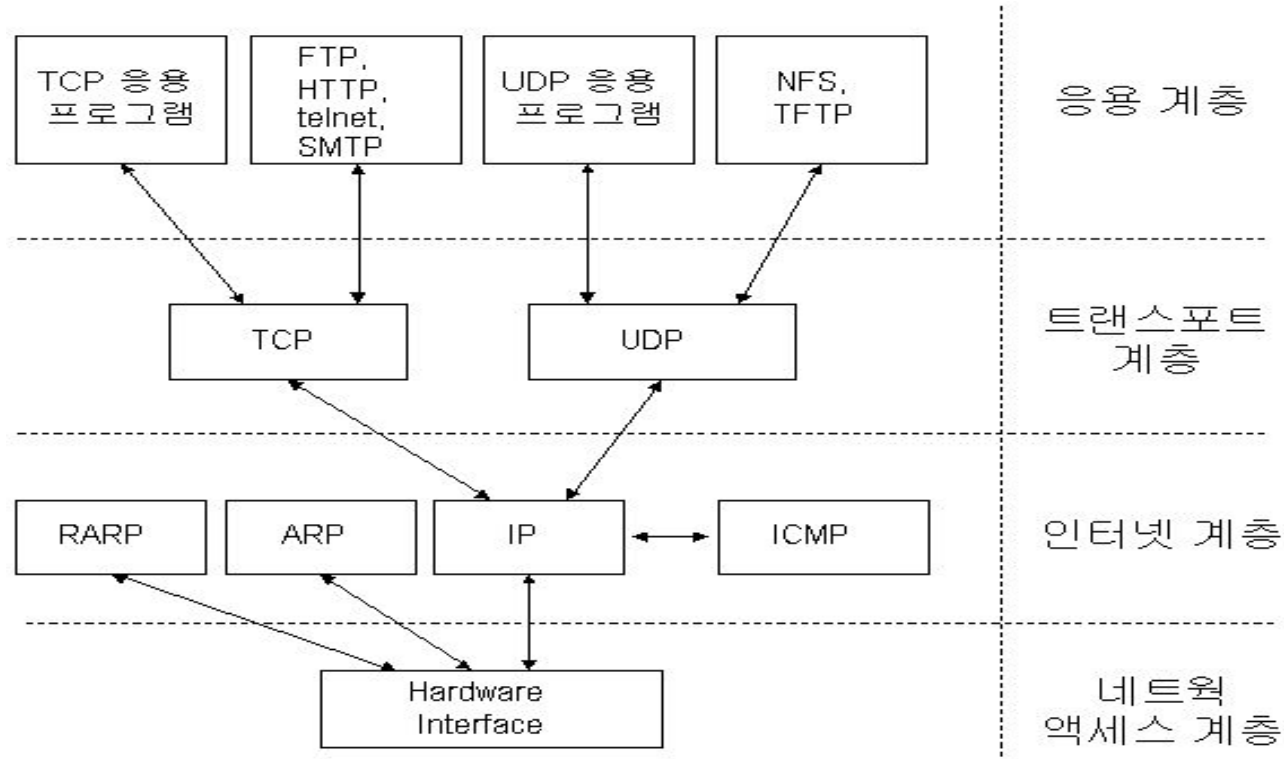
1. 네트워크 기본개념



- ▶ 네트워크 액세스 계층은 IP 패킷의 물리적인 전달을 담당하는 서브네트워크 기능을 제공하며 dial-up 회선, LAN, X.25 패킷망 등이 여기에 해당된다.
- ▶ 인터넷 계층은 비연결형 서비스 즉, 데이터그램 방식으로 호스트 사이에 IP 패킷을 전달하는 기능과 라우팅 등을 수행한다.
- ▶ 트랜스포트 계층은 호스트 사이의 종점간 연결을 제공하고 종점간의 데이터 전달을 처리한다.
- ▶ 트랜스포트 프로토콜에는 TCP와 UDP 두 개의 프로토콜이 있다.

1. 네트워크 기본개념

- ▶ TCP는 신뢰성 있는, 즉 재전송에 의한 오류제어와 흐름제어를 하는 스트림(stream) 형태의 연결형 서비스를 제공한다.
- ▶ UDP는 재전송이나 흐름제어가 없는 비연결형 서비스를 제공한다.
- ▶ 응용 계층은 TCP/IP 프로토콜을 이용하는 응용 서비스로서 TCP 또는 UDP가 지원하는 응용으로 각각 구분할 수 있다.



1. 네트워크 기본개념

트랜스포트프로토콜	응용 계층 서비스
TCP	<ul style="list-style-type: none">- File Transfer Protocol(FTP)- TELNET- Simple Mail Transfer Protocol (SMTP)- Hyper Text Transport Protocol (HTTP)
UDP	<ul style="list-style-type: none">- Network File System(NFS)- Trivial FTP(TFTP)
TCP, UDP (모두지원)	<ul style="list-style-type: none">- Echo- Daytime- Time

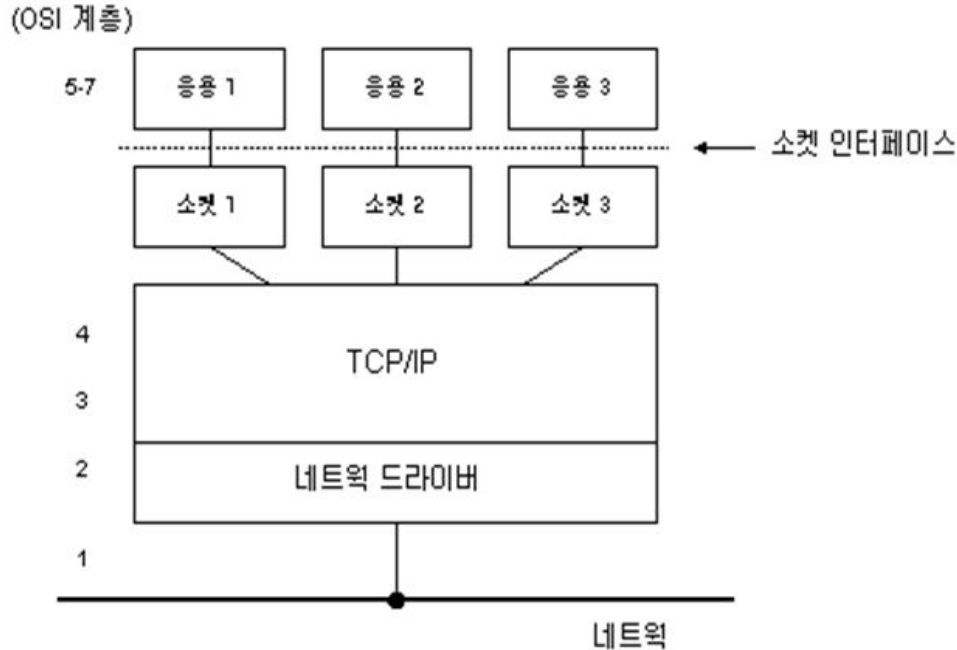
2. 네트워크 프로그래밍 기초

[1] 소켓이란 ?

- ▶ 소켓(socket)은 1982년 BSD(Berkeley Software Distribution) UNIX 4.1에서 처음 소개되었으며 현재 널리 사용되는 것은 1986년의 BSD UNIX 4.3에서 개정된 것이다.
- ▶ 소켓은 소프트웨어로 작성된 통신 접속점이라고 할 수 있는데 네트워크 응용 프로그램은 소켓을 통하여 통신망으로 데이터를 송수신하게 된다.
- ▶ **소켓은 응용 프로그램에서 TCP/IP를 이용하는 창구 역할** 을 하며 응용 프로그램과 소켓 사이의 인터페이스를 소켓 인터페이스라고 한다.
- ▶ 한 컴퓨터내에는 보통 한 세트의 TCP/IP가 수행되고 있으며, 네트워크 드라이버는 LAN 카드와 같은 네트워크 접속 장치(NIU: Network Interface Unit)를 구동하는 소프트웨어를 말한다.

2. 네트워크 프로그래밍 기초

- ▶ 세 개의 응용 프로그램이 각각 소켓을 통하여 TCP/IP를 공유하고 있는 것을 나타냈다.



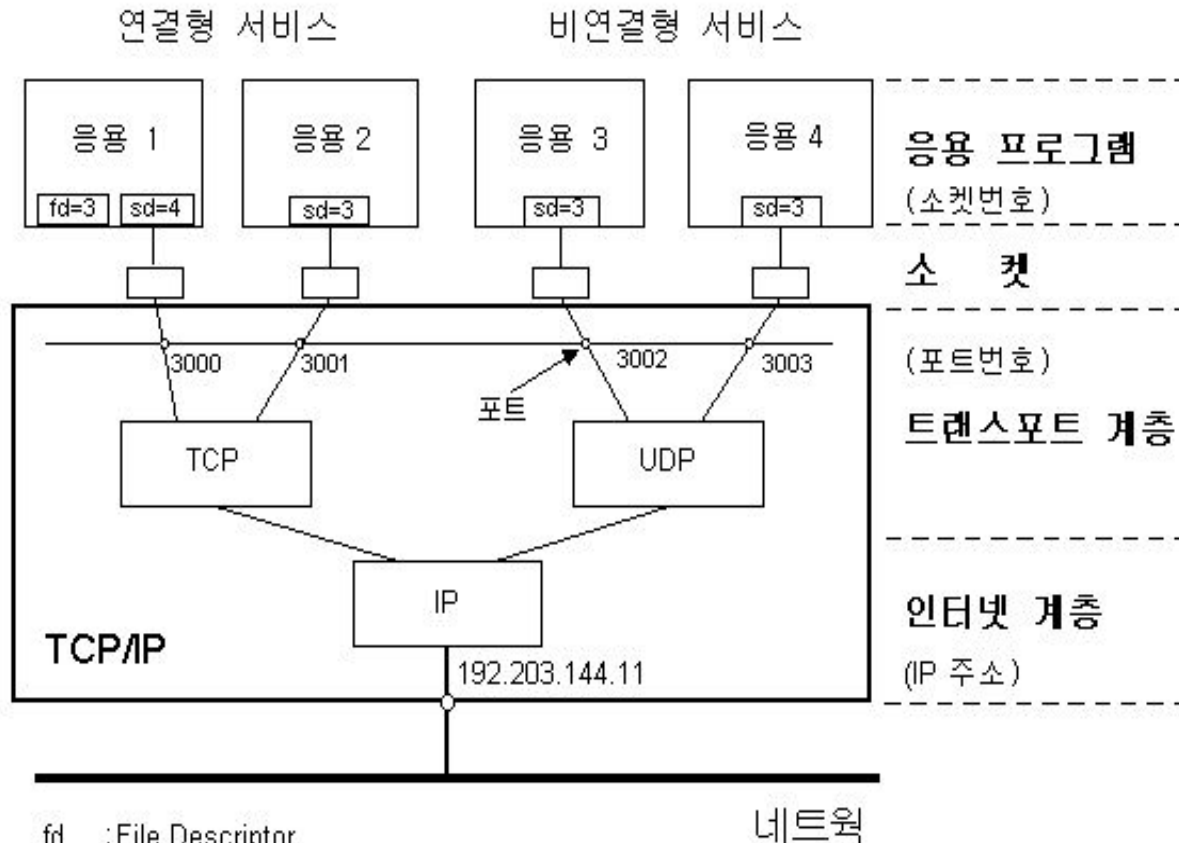
- SOCK_STREAM(TCP) : 바이트를 주고 받을 수 있는 스트림(Stream) 통신을 구현할 수 있게 해주는 소켓으로 양방향 통신이 가능하다.
- SOCK_DGRAM(UDP) : 데이터그램 통신용 소켓으로 SOCK_STREAM과 마찬가지로 양방향 통신이 가능하다.
- SOCK_RAW : 자바에서 지원하지 않는다.

2. 네트워크 프로그래밍 기초

[2] 소켓번호

- ▶ 응용 프로그램에서 이 소켓을 통하여 목적지 호스트와 연결을 요구하거나 패킷을 송수신할 때 해당 소켓을 사용하게 된다
- ▶ 한편 소켓번호는 응용 프로그램내에서 순서대로 배정되며 그 프로그램내에서만 유일하게 구분되면 되는 것이므로 서로 다른 응용 프로그램에서 같은 소켓번호를 사용하는 것은 문제가 되지 않는다.
- ▶ 포트번호는 TCP/IP가 지원하는 상위 계층의 프로그램을 구분하기 위한 번호이므로 하나의 컴퓨터 내에 있는 응용 프로그램들은 반드시 서로 다른 포트번호를 사용하여야 한다.

2. 네트워크 프로그래밍 기초



▶ IP 주소 192.203.144.11은 이 컴퓨터에 배정된 IP 주소인데 예를들어 목적지 IP 주소가 192.203.144.11인 IP 패킷은 모두 컴퓨터로 전달된다.

▶ 이 패킷을 수신할 응용 프로그램은 TCP(또는 UDP) 헤더에 있는 16비트의 포트번호를 참조하여 구분된다

fd : File Descriptor
IP : Internet Protocol
sd : Socket Descriptor
TCP : Transmission Control Protocol
UDP : User Datagram Protocol

2. 네트워크 프로그래밍 기초

[3] InetAddress 클래스를 활용한 도메인과 IP변환

- Java.net 패키지에 있는 InetAddress 클래스를 이용하면 도메인 주소를 IP 주소로 변환하거나 반대로 IP 주소를 도메인 주소로 변경할 수 있으며, 문자열이나 바이트 배열 형태로 IP 주소에 대한 정보를 얻을 수도 있다

자신이 사용중인 시스템의 IP 주소를 InetAddress 타입으로 얻으려면 `getLocalHost()` 메서드를 사용한다.

InetAddress 타입의 주소로부터 도메인 네임, dotted decimal 주소 또는 4바이트 IP 주소를 얻으려면 각각 `getHostName()`, `getHostAddress()` 또는 `getAddress()` 메서드를 사용한다.

```
InetAddress address = InetAddress.getLocalHost();
```

```
String domain = address.getHostName(); //도메인 네임을 얻음
```

```
String ip = address.getHostAddress(); // dotted decimal주소를 얻음
```

```
byte[] ipByte = address.getAddress(); // 4바이트 IP 주소를 얻음
```

2. 네트워크 프로그래밍 기초

- InetAddress 클래스가 제공하는 주요 메서드

메서드명	설명
static InetAddress[] getAllByName(String host)	호스트의 모든 IP 주소에 대한 정보를 InetAddress 배열 형태로 반환한다.
static InetAddress getByAddress(byte[] addr)	바이트로 표현된 addr 에 해당하는 IP 정보를 InetAddress 객체 형태로 반환한다.
static InetAddress getLocalHost()	로컬 호스트의 IP주소에 대한 정보를 InetAddress객체 형태로 반환한다.
byte[] getAddress()	IP 주소를 바이트 형태로 반환한다.
String getHostAddress()	호스트의 IP 주소를 점으로 구분되는 10진수 형태로 반환한다.
String getHostName()	호스트의 도메인명을 문자열로 반환한다.
boolean isMulticastAddress()	주소가 멀티캐스트 주소인지 확인한다. 멀티캐스트 주소일 경우 true를 리턴 한다.
String toString()	IP 주소를 문자열 형태로 반환한다.

3. Tcp 프로그래밍

[1] TCP(Transfer Control Protocol)

- ▶ 소켓을 이용한 네트워크 응용 프로그램에서 상대방과 IP 패킷을 주고받기 위하여 다음의 다섯 가지 정보가 정해져야 한다.
 - ① 통신에 사용할 프로토콜(TCP 또는 UDP)
 - ② 자신의 IP 주소
 - ③ 자신의 포트번호
 - ④ 상대방의 IP 주소
 - ⑤ 상대방의 포트번호
- ▶ 통신에 사용할 프로토콜은 연결형 또는 비연결형을 말하는데 인터넷 프로그램에서는 연결형 서비스를 TCP 또는 스트림(stream) 서비스라고도 부르고, 비연결형 서비스를 UDP 또는 데이터그램 서비스라고도 부른다.
- ▶ 자신의 IP 주소는 응용 프로그램이 수행되는 컴퓨터의 IP 주소를 말하며, 자신의 포트번호는 이 컴퓨터에서 수행되고 있는 응용 프로그램들 중 본 응용 프로그램을 구분하는 고유번호이다.
- ▶ 상대방의 IP 주소는 통신하고자 하는 상대방(목적지) 컴퓨터의 IP 주소를 말하며, 상대방의 포트번호는 목적지 컴퓨터 내에서 수행중인 여러 응용 프로그램 중 나와 통신할 프로그램을 지정하는 번호이다.
- ▶ 소켓 프로그래밍에서 첫번째로 해야 할 일은 통신 창구 역할을 하는 소켓을 만드는 것으로 이것은 서버와 클라이언트에서 모두 필요하다

3. Tcp 프로그래밍

소켓 관련 클래스는 java.net 패키지가 제공하는 데 연결형 서비스(TCP)를 위하여 Socket과 ServerSocket 클래스가 있으며, 비연결형 서비스(UDP)를 위하여 DatagramSocket과 DatagramPacket이 정의되어 있다.

TCP 프로그래밍에서 가장 중요한 클래스는 java.net.ServerSocket과 java.net.Socket클래스이다. ServerSocket은 서버 쪽에서 클라이언트의 접속을 대기하기 위해서 반드시 필요한 클래스며, Socket은 서버와 클라이언트가 통신하기 위해서 반드시 필요한 클래스다.

ServerSocket 클래스가 제공하는 주요 메서드는 다음과 같다.

메서드	설명
Socket accept()	클라이언트의 접속요청을 받아 새로운 Socket 객체를 리턴한다.
void close()	서버소켓을 닫는다.
InetAddress getInetAddress()	서버 자신의 인터넷 주소를 리턴한다.
int getLocalPort()	자신의 포트 번호를 리턴한다.

3. Tcp 프로그래밍

Socket 클래스가 제공하는 주요 메서드는 다음과 같다.

메서드	설명
<code>void close()</code>	소켓을 닫는다.
<code>InetAddress getInetAddress()</code>	상대방의 <code>InetAddress</code> 를 리턴한다.
<code>InputStream getInputStream()</code>	이 소켓과 연결된 <code>InputStram</code> 을 얻는다.
<code>InetAddress getLocalAddress()</code>	자신의 <code>InetAddress</code> 를 리턴한다.
<code>int getLocalPort()</code>	자신의 포트 번호를 리턴한다.
<code>OutputStream getOutputStream()</code>	이 소켓과 연결된 <code>OutputStream</code> 을 얻는다.
<code>int getPort()</code>	상대방의 포트 번호를 리턴한다.
<code>boolean getTcpNoDelay()</code>	<code>TCP_NODELAY</code> 옵션 상태를 확인한다.
<code>void getSoTimeout(int timeout)</code>	<code>Read()</code> 가 기다리는 최대 값을 지정한다.(milli second 단위)
<code>void setTcpNoDelay(Booleam on)</code>	<code>TCP_NODELAY</code> 옵션을 지정한다.

3. Tcp 프로그래밍

[2] TCP기반의 에코서버/클라이언트를 만들어 보자

□ TCP 에코 서버 작성순서

1. `ServerSocket(7777)`을 생성하여 특정 포트에서 클라이언트의 접속을 대기한다.
2. `ServerSocket`의 `accept()` 메서드를 이용하여 클라이언트의 접속을 기다린다.
3. 클라이언트의 접속 요청이 들어오면 `accept()` 메서드가 실행되어 클라이언트와의 통신을 위한 `Socket` 객체를 생성한다.
4. 생성된 `Socket` 객체로부터 통신을 위한 `InputStream`, `OutputStream`을 얻는다.
5. `InputStream`, `OutputStream`을 이용하여 클라이언트와 통신한다.
6. 통신에 사용된 `IO스트림`과 `Socket` 객체를 `close()`한다.

3. Tcp 프로그래밍

□ TCP 에코 클라이언트 작성 순서

1. 서버와 통신을 위한 Socket 객체를 생성한다. 이때 접속 요청할 서버의 IP주소와 Port 번호를 매개변수로 지정한다.
2. Socket 객체로부터 서버와의 통신을 위한 InputStream, OutputStream을 얻는다.
3. 생성된 InputStream, OutputStream을 이용하여 서버와 통신한다.
4. 통신이 완료되면 통신에 사용된 IO스트림과 Socket 객체를 close() 한다.

4. UDP 프로그래밍

[1] UDP(User Datagram Protocol)

- UDP(User Datagram Protocol)는 데이터그램 통신 프로토콜이라고도 한다.
- UDP는 TCP와 다르게 비연결성(Connectionless) 프로토콜이다.
- UDP는 패킷을 보낼 때마다 수신 측의 주소와 로컬 파일 설명자를 함께 전송해야 한다
- UDP를 이용한 프로그래밍을 하려면 클라이언트와 서버 모두 java.net 패키지 안에 있는 DatagramSocket객체를 생성해야 한다. 또한 클라이언트와 서버는 데이터를 주고받기 위해서 DatagramPacket 객체를 이용해야 한다.
- DatagramSocket은 DatagramPacket을 보내거나 받을 때 모두 필요하다.

4. UDP 프로그래밍

- DatagramPacket 클래스가 제공하는 생성자 메서드

메서드	설명
DatagramPacket(byte[], buf, int length)	Buf 배열에 length 길이만큼 데이터를 전송받기 위한 생성자
DatagramPacket(byte[], buf, int length, InetAddress addr, int port)	Buf 배열의 length 길이만큼의 데이터를 addr 주소의 port 번 포트로 전송하기 위한 생성자
DatagramPacket(byte[], buf, int offset, int length, InetAddress addr, int port)	Buf 배열에서 offset 길이만큼 띄운 위치에서부터 length 길이만큼의 데이터를 전송하기 위한 생성자

4. UDP 프로그래밍

DatagramPacket 클래스가 제공하는 메서드

메서드	설명
<code>InetAddress getAddress()</code>	이 객체를 보낸 곳의 IP 주소를 리턴한다.
<code>byte[] getData()</code>	이 객체에 담긴 데이터의 내용을 byte배열로 리턴한다.
<code>int getLength()</code>	데이터의 길이를 리턴한다.
<code>int getOffset()</code>	보낸 곳의 데이터 offset 값을 리턴한다.
<code>int getPort()</code>	보낸 곳의 포트 번호를 리턴한다.
<code>void setAddress(InetAddress addr)</code>	IP주소를 설정한다.
<code>void setData(byte[] buf)</code>	데이터를 설정한다.
<code>void setData(byte[] buf, int offset, int length)</code>	데이터를 특정 offset 위치에서부터 length 개만큼 설정한다.
<code>void setLength(int length)</code>	데이터의 길이를 설정한다.
<code>void setPort(int port)</code>	포트번호를 설정한다.

4. UDP 프로그래밍

DatagramSocket 클래스의 주요 생성자 메서드

메서드	설명
DatagramSocket()	UDP로 데이터를 전송하기 위한 기본 소켓 생성
DatagramSocket(int port)	특정 포트를 통해 데이터를 전송하는 Socket 객체 생성
DatagramSocket(int port, InetAddress)	특정 InetAddress의 특정 포트를 통해 통신하는 소켓 객체 생성

4. UDP 프로그래밍

DatagramSocket 클래스의 주요 메서드

메서드	설명
void close()	소켓 통신을 종료한다.
void connect(InetAddress addr, int port)	원격 지역 addr의 포트에 접속한다.
void disconnect()	연결을 종료한다.
InetAddress getInetAddress()	현재 소켓이 바인딩된 주소를 리턴한다.
InetAddress getLocalAddress()	이 소켓이 바인딩된 주소를 리턴한다.
int getLocalPort()	이 소켓이 바인딩된 포트번호를 리턴한다.
int getPort()	현재 소켓의 포트번호를 리턴한다.
void receive(DatagramPacket p)	p를 통해 전달된 데이터를 수신한다.
void send(DatagramPacket p)	p를 소켓 통로로 전송한다.
void setSendBufferSize(int size)	이 소켓의 전송 버퍼 크기를 설정한다.
void setSoTimeout(int timeout)	timeout값을 설정한다.

4. UDP 프로그래밍

[2] UDP 에코 클라이언트 서버를 다음과 같은 순서로 작성한다.

□ 에코 서버 작성순서

1. 특정 포트에서 동작하는 DatagramSocket 객체를 생성한다.
2. 클라이언트가 전송한 DatagramPacket을 받기 위해 바이트 배열과 , 길이를 가진 DatagramPacket 객체를 생성한다.
3. 생성한 DatagramPacket을 매개변수로 DatagramSocket이 제공하는 receive() 메서드를 호출한다. 여기까지 실행되면 클라이언트가 DatagramPacket을 전송할 때까지 서버는 계속 대기한다.
4. 클라이언트가 전송한 데이터를 서버 콘솔에 출력한다.
5. DatagramSocket의 close()를 호출하여 연결을 해제한다.

4. UDP 프로그래밍

□ UDP 에코 클라이언트 작성 순서

1. DatagramSocket 객체를 생성한다.
2. 전송할 데이터, 데이터 길이, 서버 IP, 서버 포트번호를 매개변수로 하여 DatagramPacket 객체를 생성한다.
3. 생성한 DatagramPacket을 매개변수로 하여 DatagramSocket이 제공하는 send() 메서드를 호출하여 서버쪽에 DatagramPacket을 전송한다.
4. DatagramSocket의 close()를 호출하여 연결을 해제한다.

5 .객체 직렬화를 이용한 네트워크 프로그래밍

- 객체 직렬화란 말 그대로 객체를 일렬로 늘어선 바이트의 흐름으로 만드는 기술을 말한다. 이때 사용하는 IO 객체가 `ObjectInputStream`과 `ObjectOutputStream`이다.
- 네트워크 프로그래밍에서도 `ObjectInputStream`과 `ObjectOutputStream`을 이용해서 소켓을 통해서 객체를 주고받을 수 있다.

직렬화 가능 객체

사용자가 작성한 클래스로부터 생성된 객체가 직렬화 가능한 객체가 되기 위해서는 `java.io.Serializable` 인터페이스를 implements해야 한다. `Serializable` 인터페이스는 마크 인터페이스로서 implements할 메서드가 아무것도 없다.

단지 이 클래스로부터 생성된 객체가 직렬화 가능한 객체임을 표현하기 위한 인터페이스인 것이다. 그리고 이렇게 작성된 클래스의 인스턴스 변수들 역시 기본 데이터형이거나 직렬화 가능한 클래스형이어야 한다.

5 .객체 직렬화를 이용한 네트워크 프로그래밍

계산기 서버/클라이언트 프로그래밍을 만들어 보자.

□ 오브젝트 SendData 생성 : 두수와 연산자를 관리하는 클래스를 생성한다

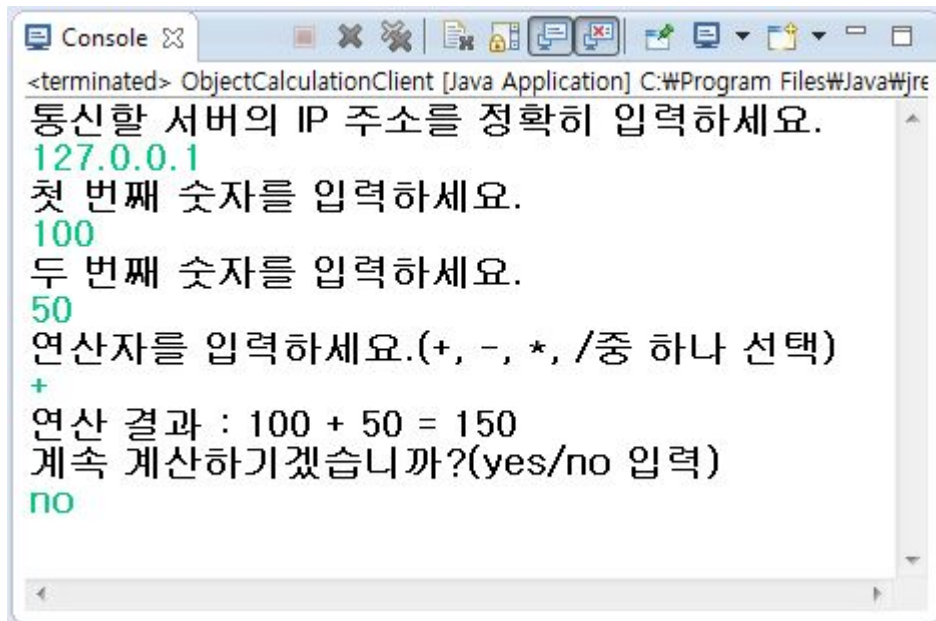
□ 계산기 서버

- 계산기 서버는 먼저 클라이언트로부터 객체를 통해 입력받은 두수와 연산자를 전송받아야 한다.
- 클라이언트로부터 전송 받은 값을 이용해서 연산자에 의한 연산 후 결과를 클라이언트에게 다시 전송한다.
- 따라서 클라이언트 객체를 통해 `socket.getInputStream()` 를 통한 `ObjectInputStream` 을 생성하고 `socket.getOutputStream()` 을 통한 `ObjectOutputStream` 를 생성해야 한다

5 .객체 직렬화를 이용한 네트워크 프로그래밍

□ 계산기 클라이언트 작성

클라이언트는 서버의 IP를 입력받아 소켓 객체를 생성해서 두수와 연산자를 SendData 클래스를 통해 서버로 스트림을 통해 전달 한다음 결과를 다시 리턴받는다.



```
<terminated> ObjectCalculationClient [Java Application] C:\Program Files\Java\jre
통신할 서버의 IP 주소를 정확히 입력하세요.
127.0.0.1
첫 번째 숫자를 입력하세요.
100
두 번째 숫자를 입력하세요.
50
연산자를 입력하세요.(+, -, *, /중 하나 선택)
+
연산 결과 : 100 + 50 = 150
계속 계산하기겠습니까?(yes/no 입력)
no
```