2019학년도 2학기

JAVA 프로그래밍 II

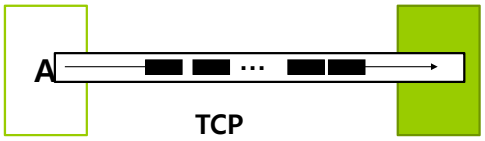
514770
2019년 가을학기
11/27/2019
박경신

Lab #8 (Networking)

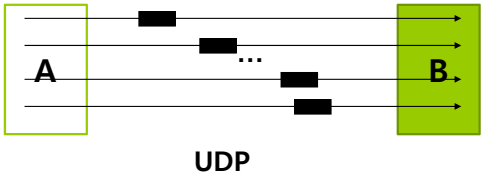
- 기존 요구사항 분석
  - Lab #7는 Thread, Runnable과 Worker Thread를 이용한 다양한 멀티스레드 기능을 사용
  - **Lab #8는 TCP/UDP 등 다양한 네트워크 프로그래밍 기능을 사용**
- TCP, UDP, HTTP, File Transfer

TCP vs UDP

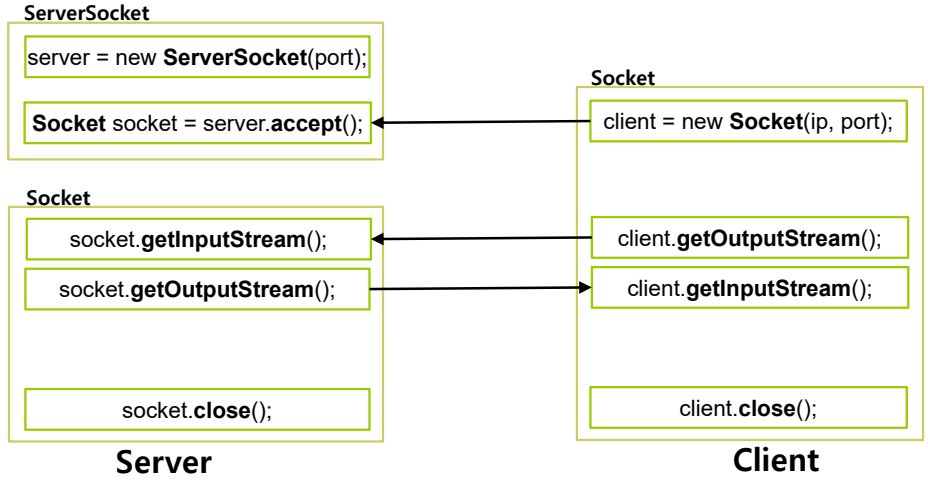
- **TCP** is a **connection-oriented reliable stream** transport protocol



- **UDP** is a **connectionless unreliable datagram** transport protocol



TCP Java Socket



Lab #8_1 TCP

- Lab#7_1에서는 TCPServer/TCPClient 클래스를 구현한다.
  - public class TCPServer {

```
    private ServerSocket serverSocket = null;
// 중간생략..
    public int init(int port) {
        serverSocket = new ServerSocket(port);
        if (serverSocket != null) return OK;
    }
    public TCPClient checkForNewConnections() {
        Socket s = serverSocket.accept();
        if (s != null) return new TCPClient(s);
    }
}
```

Lab #8_1 TCP

- public class TCPClient {

```
    private Socket socket = null;
    public int connectToServer(String ip, int port) {
        socket = new Socket(ip, port);
        if (socket != null) return OK;
    }
    public void close() {       socket.close();       }
    public int read(byte[] buffer, int nbytes, boolean blocking) {
        InputStream is = socket.getInputStream();
        is.read(buffer, 0, nbytes);
    }
    public int write(byte[] buffer, int nbytes, boolean blocking) {
        OutputStream os = socket.getOutputStream();
        os.write(buffer, 0, nbytes);
    }
}
```

Lab #8_1 TCP

- public class TCPServerTest {

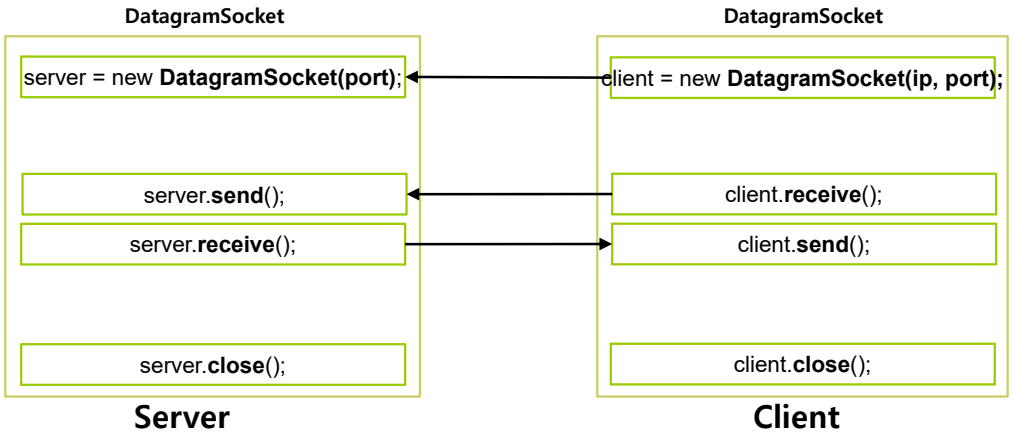
```
    TCPServer server = null; TCPClient client = null;
    byte[] sendData = new byte[512]; bytes[] receiveData = new byte[512];
    public TCPServerTest(int port) {    server = new TCPServer(port);    }
    public void listen() {
        client = server.checkForNewConnections();
    }
    public String receive() {
        client.read(receiveData, 512, true);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        client.write(sendData, 512, true);
    }
}
```

Lab #8_1 TCP

- public class TCPClientTest {

```
    TCPClient client = null;
    byte[] sendData = new byte[512]; bytes[] receiveData = new byte[512];
    public TCPClientTest(String ip, int port) {    client = new TCPClient(ip, port);    }
    public String receive() {
        client.read(receiveData, 512, true);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        client.write(sendData, 512, true);
    }
}
```

UDP Java Socket



Lab #8_2 UDP

- Lab#8_2에서는 UDPSocket 클래스를 구현한다.
  - *public class UDPSocket {*

```
private DatagramSocket socket = null;
public UDPSocket(int port) {
    socket = new DatagramSocket(port);
}
public UDPSocket(String ip, int port) {
    socket = new DatagramSocket(); setSendAddress(ip, port);
}
public void setSendAddress(String ip, int port) {
    destAddr = InetAddress.getByName(ip);       destPort = port;
}
public void copyReceiveAddressToSendAddress() {
    setSendAddress(receivedAddr, receivedPort);
}
```

Lab #8_2 UDP

```
public int send(byte[] buffer, int size) {
    DatagramPacket dp = new DatagramPacket(buffer, size, destAddr, destPort);
    socket.send(dp);
}
public byte[] receive(int size) {
    byte[] buffer = new byte[size];
    DatagramPacket dp = new DatagramPacket(buffer, size);
    socket.receive(dp);
    receivedAddr = dp.getAddress();
    receivedPort = dp.getPort();
}
}
```

Lab #8_2 UDP

- *public class UDPServerTest {*

```
UDPSocket server = null;
byte[] sendData = new byte[512];
public UDPServerTest(int port) {    server = new UDPSocket(port);  }
public void copyReceiveAddressToSendAddress() {
    server.copyReceiveAddressToSendAddress();
}
public String receive() {
    byte[] receiveData = server.receive(512);
    return Utility.getReceivedString(receiveData);
}
public void send(String s) {
    Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
    server.send(sendData, 512);
}
}
```

Lab #8_2 UDP

```
public class UDPClientTest {
    UDPSocket client = null;
    byte[] sendData = new byte[512];
    public UDPClientTest(String ip, int port) {    client = new UDPSocket(ip, port);      }
    public String receive() {
        byte[] receiveData = client.receive(512);
        return Utility.getReceivedString(receiveData);
    }
    public void send(String s) {
        Utility.paddingBytesArray(sendData, 512, s.getBytes(), s.length());
        client.send(sendData, 512);
    }
}
```

Lab #8_3 HTTP

- Lab#8_3에서는 HTTPClientTest 클래스를 구현한다.

```
public class HTTPClientTest {
    private void requestGet(String url) throws Exception {
    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();
    con.setRequestMethod("GET"); con.setRequestProperty("User-Agent", USER_AGENT);
    int responseCode = con.getResponseCode();
    System.out.println("Response Code : " + responseCode);
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())
    StringBuffer response = new StringBuffer();
    String line = "";
    while ((line = in.readLine()) != null) { response.append(line); }
    in.close(); System.out.println(response.toString());
    }
}
```

Lab #8_4 ImageTransfer

- Lab#8_4에서는 이미지파일을 Client -> Server 보내는 클래스를 구현한다.

```
public class ImageSenderClientTest {
    TCPClient client = null;
    byte[] imageInByte;
    byte[] sizeBuf = new byte[SIZE_OF_INT];
    public ImageSenderClientTest(String ip, int port) { client = new TCPClient(ip, port); }
    public void run() {
        if (client != null) {
            imageInByte = Utility.convertImageFileToByteArray(imagefile));
            sizeBuf = Utility.convertIntegerToByteArray(SIZE_OF_INT, imageInByte.length);
            client.write(sizeBuf, SIZE_OF_INT, true);
            client.write(imageInByte, imageInByte.length, true);
            client.close();
        }
    }
}
```

Lab #8_4 ImageTransfer

```
public class ImageReceiverServerTest {
    TCPServer server = null; byte[] imageInByte; byte[] sizeBuf;
    public ImageReceiverServerTest(int port) {    server = new TCPServer(port);  }
    public TCPClient listen() { return server.checkForNewConnections();    }
    public void run() {
        TCPClient client = listen();
        if(client != null) {
            int imageSize = 0;
            int nread = client.read(sizeBuf, SIZE_OF_INT, true);
            if (nread == SIZE_OF_INT) { imageSize = Utility.convertByteArrayToInteger(sizeBuf); }
            if (imageSize > 0) {
                imageInByte = new byte[imageSize];
                nread = client.read(imageInByte, imageSize, true);
                Utility.convertByteArrayToImageFile(imageInByte, "outputfilename.jpg");
            }
        }
    }
}
```

Lab #8_4 ImageDownloader

- Lab#8_4에서는 이미지파일을 Server -> Client 보내는 클래스를 구현한다.

```
public class ImageReceiverClientTest {
    TCPClient client = null;
    byte[] imageInByte;
    byte[] sizeBuf = new byte[SIZE_OF_INT];
    public ImageReceiverClientTest(String ip, int port) { client = new TCPClient(ip,
port); }
    public void run() {
        if (client != null) {
            // read sizeBuf, imageInByte
        }
    }
}
```

Lab #8_4 ImageDownloader

- Lab#8_4에서는 이미지파일을 Server -> Client 보내는 클래스를 구현한다.

```
public class ImageSenderServerTest {
    TCPServer server = null; byte[] imageInByte; byte[] sizeBuf = new byte[SIZE_OF_INT];
    public ImageSenderServerTest(int port) {   server = new TCPServer(port);    }
    public TCPClient listen() { return server.checkForNewConnections();    }
    public void run() {
        TCPClient client = listen();
        if(client != null) {
            // write sizeBuf, imageInByte
        }
    }
}
```

Lab #8_4 ImageDownloaderWithFilename

- Lab#8_4에서는 이미지파일을 Server -> Client 보내는 클래스를 구현한다.

```
public class ImageReceiverClientTest2 {
    TCPClient client = null;
    byte[] imageInByte;
    byte[] sizeBuf = new byte[SIZE_OF_INT];
    byte[] nameBuf = new byte[SIZE_OF_FILENAME];
    public ImageReceiverClientTest2(String ip, int port) { client = new TCPClient(ip,
port); }
    public void run() {
        if (client != null) {
            // read nameBuf, sizeBuf, imageInByte...
        }
    }
}
```

Lab #8_4 ImageDownloaderWithFilename

- Lab#8_4에서는 이미지파일을 Server -> Client 보내는 클래스를 구현한다.

```
public class ImageSenderServerTest2 {
    TCPServer server = null; byte[] imageInByte; byte[] sizeBuf = new byte[SIZE_OF_INT];
    byte[] nameBuf = new byte[SIZE_OF_FILENAME];
    public ImageSenderServerTest2(int port) {   server = new TCPServer(port);   }
    public TCPClient listen() { return server.checkForNewConnections();    }
    public void run() {
        TCPClient client = listen();
        if(client != null) {
            // write nameBuf, sizeBuf, imageInByte...
        }
    }
}
```

Lab #8_4 MultipleImageTransfer

- Lab#8_4에서는 여러장 이미지파일을 Client -> Server 보내는 클래스를 구현한다.

```
Public class MultipleImageSenderClientTest {
    TCPClient client = null;
    byte[] imageInByte;
    byte[] sizeBuf = new byte[SIZE_OF_INT];
    byte[] nameBuf = new byte[SIZE_OF_FILENAME];
    public MultipleImageSenderClientTest(String ip, int port) { client = new
TCPClient(ip, port); }
    public void run() {
        if (client != null) {
            // write NUM_FILES (# of images)
            for (int i=0; i < NUM_FILES; i++) {
                // write nameBuf, sizeBuf, imageInByte
            }
        }
    }
}
```

Lab #8_4 MultipleImageTransfer

```
public class MultipleImageReceiverServerTest {
    TCPServer server = null; byte[] imageInByte; byte[] sizeBuf;
    public MultipleImageReceiverServerTest(int port) {   server = new TCPServer(port); }
    public TCPClient listen() { return server.checkForNewConnections();    }
    public void run() {
        TCPClient client = listen();
        if(client != null) {
            // read NUM_FILES (# of images)
            for (int i=0; i < NUM_FILES; i++) {
                // read nameBuf, sizeBuf, imageInByte
            }
        }
    }
}
```

Lab #8_4 ThreadedMultipleImageTransfer

- Lab#8_4에서는 스레드를 사용하여 여러장 이미지파일을 Client -> Server 보내는 클래스를 구현한다.

```
public class ThreadedMultipleImageSenderClientTest {
    public void run() {
        for (int i=0; i < numThread; i++) {
            ImageSenderClient client = new ImageSenderClient(new TCPClient(ip, port), ...);
            new Thread(client).start();
        }
    }
}
public class ImageSenderClient implements Runnable {
    public ImageSenderClient(TCPClient c, String dir, int si, int ei, int num) { ... }
    public void run() {
        // write image (si ~ ei) : numFiles, for {nameBuf, sizeBuf, imageInByte }
    }
}
```

Lab #8_4 ThreadedMultipleImageTransfer

```
public class ThreadedMultipleImageReceiverServerTest {
    public ImageReceiverClient listen() {
        TCPClient client = server.checkForNewConnections(); // connected by # of threads
        return new ImageReceiverClient(client, dir);
    }
    public void run() {
        ImageReceiverClient client = listen();
        new Thread(client).start();
    }
}
public class ImageReceiverClient implements Runnable {
    public ImageReceiverClient(TCPClient c, String dir) { ... }
    public void run() {
        // read image (si ~ ei) : numFiles, for {nameBuf, sizeBuf, imageInByte }
    }
}
```

Lab8_FileTransfer

- **Lab8_FileTransfer에서는 ThreadedMultipleImageTransfer를 참고하여 디렉토리 안에 있는 모든 파일(*.*)을 Client -> Server 보내는 프로그램을 작성한다.**

과제 제출

- 모든 Lab 코드와 보고서를 전체적으로 묶어서 e-learning에 과제 제출
- 각 Lab마다 **본인이 추가로 작성한 코드**와 설명을 중점적으로 보고할 것!