Chapter 09.UDP 프로그래밍

Originally made by Prof. Hanku Lee
Modified by Mingyu Lim

Collaborative Computing Systems Lab.
School of Internet & Multimedia Engineering
Konkuk University, Seoul, Korea

1. UDP를 이용한 프로그래밍 방법

❑ UDP 기본 개요

- UDP (User Datagram Protocol)는 데이터그램 프로토콜이다.
- 비연결성 (connectionless) 프로토콜
- 데이터 그램은 패킷(packet)이라고도 한다.
  : 패킷이란 인터넷 내에서 운반되어야 할 메시지 단위
- 패킷을 보낼때마다 수신측의 주소와 로컬 파일 설명자를 함께 전송
  (전송해야할 데이터 외에 추가적인 데이터가 함께 전송)
- 비연결성이기 때문에 패킷이 수신 측에 전송한 순서로 도달하지 않는 문제 내포
- UDP의 경우에는 데이터의 크기가 64KB로 제한
- 연결되지 않은 상태로 전송되기 때문에 좀더 빠르게 데이터 전송

1. UDP를 이용한 프로그래밍 방법(계속)

❑ TCP vs UDP

	TCP(Transfer Control Protocol)	UDP(User Datagram Protocol)
특징	• 접속지향(Connection oriented) 프로토콜 • 가장 많이 사용됨. • 두 프로그램간의 통신이 종료 시까지 계속 연결 유지	• 비접속(connection-less) 프로토콜 • 연결을 설정하지 않고 데이터를 전송
장점	• 양방향 모두 가능 • 데이터의 신뢰성있는 전송(데이터의 손실이 적다.) • 안정적이다. • 같은 장소에 많은 정보량을 전송할 때 유용 • FIFO 방식	• 메시지 : 손실, 중복, 비순서적으로 도착할 수 있음 • 작은 data를 때 보낼 때 유용 • TCP 에 비해 전송 속도가 빠르다. • 멀티미디어 작업시 주로 사용.
단점	• 속도가 느리다. • 재전송으로 인한 대역폭 소비	• 수신된 메시지의 순서무시 • 데이터의 전송 보장 못함 • 패킷 파괴 및 손실 검출 불가능
용도	• 일반적인 서버-클라이언트 구조로 구성된 중소규모의 네트워크 • 예> HTTP	• 빠른 데이터 전송을 필요로하는 1대1 또는 서버-클라이언트 구조의 네트워크 • 예> SNMP

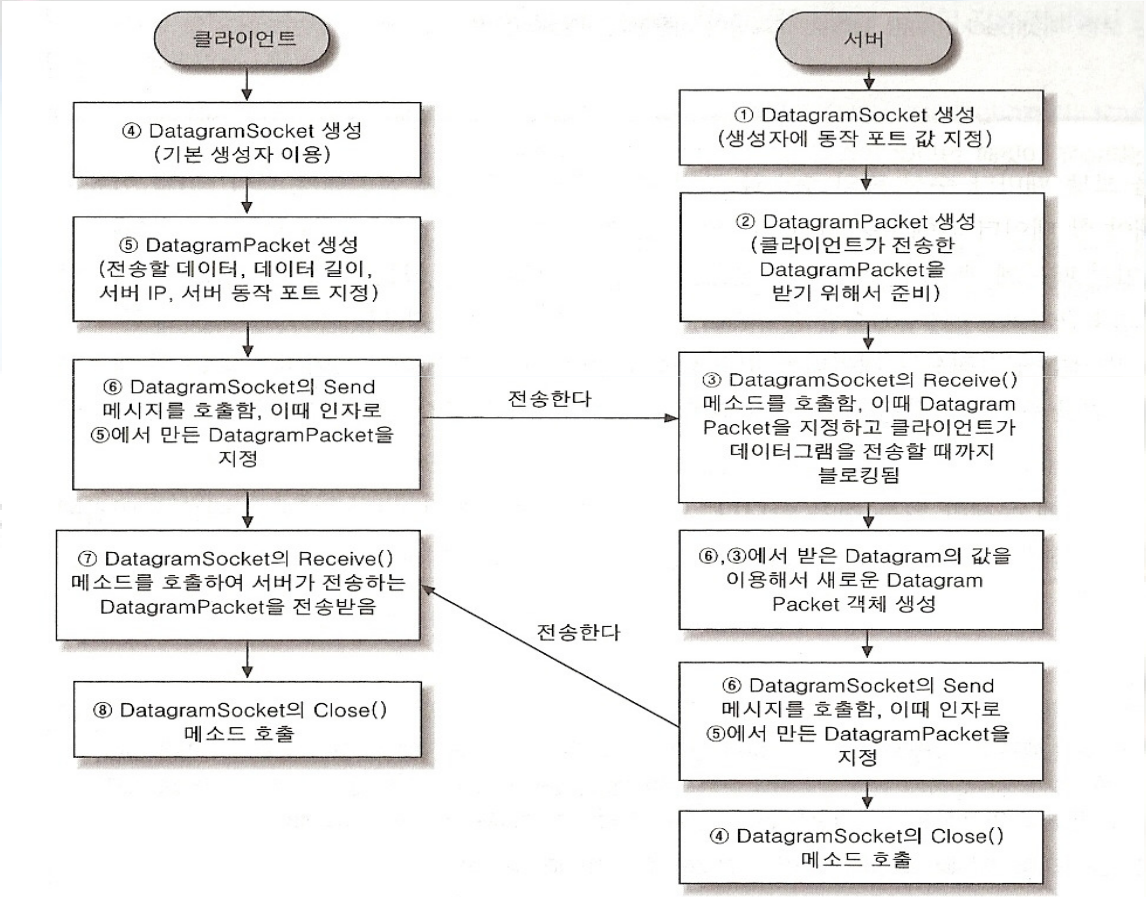
1. UDP를 이용한 프로그래밍 방법(계속)

❑ UDP 클라이언트 / 서버 동작 순서

1. 서버는 클라이언트로부터 정보를 받아들이기 위해 특정 포트에서 동작하는 DatagramSocket 생성
   : receive() 메소드에 비어있는 DatagramPacket 객체를 인자로 지정후 클라이언트에서 패킷을 보낼 때까지 대기
2. 클라이언트는 데이터를 전송하기 위해 DatagramSocket 생성 (포트 지정 안함)
   : 데이터그램 패킷에 서버의 DataSocket의 동작포트, 서버의 IP, 전송데이터, 전송할 데이터 길이 지정후 send() 이용해서 전송
3. 서버는 전달된 패킷을 receive() 메소드를 통해 읽고 패킷을 새롭게 구성 (클라이언트의 IP, 동작포트, 데이터, 데이터 길이)
4. 서버쪽에서는 DatagramPacket에 있는 send() 이용해서 클라이언트에게 전송
5. 클라이언트도 서버와 마찬가지로 receive() 이용해서 서버가 반송한 데이터를 전달 받는다.
6. 모든 작업이 끝나면 서버와 클라이언트는 close() 메소드 호출

1. UDP를 이용한 프로그래밍 방법(계속)

❑ UDP 클라이언트 / 서버 동작 순서



[그림 9-1] UDP 방식의 클라이언트와 서버의 동작 순서

2. UDP 에코 클라이언트 / 서버 프로그래밍

❑ 에코 서버 프로그래밍(예제)

```
import java.net.*;

public class UDPEchoServer {

    public static void main(String[] args) {
        if(args.length != 1){
            System.out.println("사용법 : java UDPEchoServer port");
            System.exit(1);
        }
        int port = 0;
        try{
            port = Integer.parseInt(args[0]);
        }catch(Exception ex){
            System.out.println("port 번호는 양의 정수로 입력하여 주세요.");
            System.exit(1);
        }
        DatagramSocket dsock = null;
        try{
            System.out.println("접속 대기상태입니다.");
            dsock = new DatagramSocket(port);
            String line = null;
            while(true){
                // 받기
                byte[] buffer = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
                dsock.receive(receivePacket);
                String msg = new String(receivePacket.getData(),0, receivePacket.getLength());
                System.out.println("전송받은 문자열 : " + msg );
                if(msg.equals("quit"))
                    break;
                // 전송
                DatagramPacket sendPacket = new DatagramPacket(
                        receivePacket.getData(), receivePacket.getData().length,
                        receivePacket.getAddress(), receivePacket.getPort());
                dsock.send(sendPacket);
            } // while
            System.out.println("UPDEchoServer를 종료합니다.");
        }catch(Exception ex){
            System.out.println(ex);
        }finally{
            if(dsock != null)
                dsock.close();
        }
    } // main
} // class
```

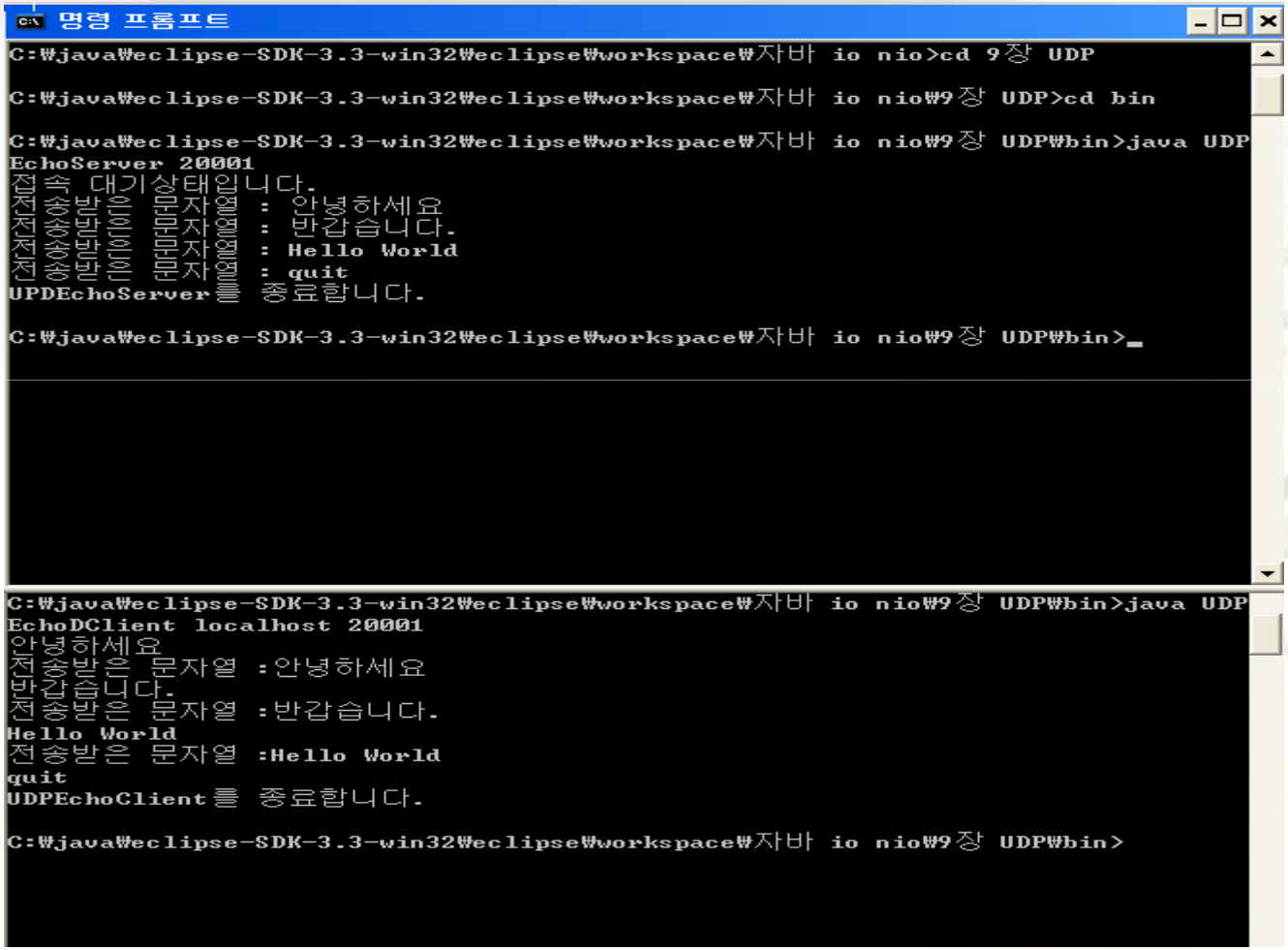
2. UDP 에코 클라이언트 / 서버 프로그래밍

❑ 에코 클라이언트 프로그래밍(예제)

```
import java.net.*;
import java.io.*;

public class UDPEchoDClient {

    public static void main(String[] args) {
        if(args.length != 2){
            System.out.println("사용법 : java UDPEchoClient ip port");
            System.exit(1);
        }
        String ip = args[0];
        int port = 0;
        try{
            port = Integer.parseInt(args[1]);
        }catch(Exception ex){
            System.out.println("port 번호는 양의 정수로 입력하여 주세요.");
            System.exit(1);
        }
        InetAddress inetaddr = null;
        try {
            inetaddr = InetAddress.getByName(ip);
        } catch (UnknownHostException e) {
            System.out.println("잘못된 도메인이나 ip입니다.");
            System.exit(1);
        }
        DatagramSocket dsock = null;
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            dsock = new DatagramSocket();
            String line = null;
            while((line = br.readLine()) != null){

                // 전송
                DatagramPacket sendPacket = new DatagramPacket(line.getBytes(), line.getBytes().length, inetaddr, port);
                dsock.send(sendPacket);
                if(line.equals("quit"))
                    break;
                // 받기
                byte[] buffer = new byte[line.getBytes().length];
                DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
                dsock.receive(receivePacket);
                // 받은 결과 출력.
                String msg = new String(receivePacket.getData(), 0, receivePacket.getData().length);
                System.out.println("전송받은 문자열 :" + msg);
            } // while
            System.out.println("UDPEchoClient를 종료합니다.");
        }catch(Exception ex){
            System.out.println(ex);
        }finally{
            if(dsock != null)
                dsock.close();
        }
    } // main
} // class
```

2. UDP 에코 클라이언트 / 서버 프로그래밍

❑ 에코 클라이언트 프로그래밍(실행결과)



3. UDP 타임 서버/클라이언트 작성

❑ 타임 서버

- 여러 대의 컴퓨터를 운용하며 각각의 컴퓨터 시간이 항상 같게 설정하기 위해 사용
- 각 클라이언트 컴퓨터들은 서버로부터 시간을 읽어들여 서로 동기화

❑ 타임 서버 (예제 9-3)

- DatagramSocket을 생성하여 클라이언트로부터 요청 DatagramPacket 수신

```
…
DatagramSocket dsock = null;
try{
  System.out.println("접속 대기상태입니다.");
  dsock = new DatagramSocket(port);
  String line = null;
  while(true){
    // 받기
    byte[] buffer = new byte[1024];
    DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
    dsock.receive(receivePacket);
    …
```

3. UDP 타임 서버/클라이언트 작성

❑타임 서버 (예제 9-3)

- 현재 시간을 "yyyy-MM-dd HH:mm:ss a" 형식으로 얻기

```
java.text.SimpleDateFormat format
= new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss a");
String sdate = format.format(new Date());
```

- 수신된 패킷으로부터 클라이언트의 InetAddress, 포트번호 얻기
- 해당 클라이언트로 특정형식의 시간 정보 전송

```
  System.out.println(receivePacket.getAddress().getHostAddress() + " 에 현  재시간 "
+ sdate + " 을 전송합니다.");
  DatagramPacket sendPacket = new DatagramPacket(sdate.getBytes(),
sdate.getBytes().length, receivePacket.getAddress(), receivePacket.getPort());
  dsock.send(sendPacket);
} // while
```

3. UDP 타임 서버/클라이언트 작성

❑ 타임 클라이언트 (예제 9-4)

- 서버 주소 정보

```
String ip = args[0];
…
InetAddress inetaddr = null;
try {
  inetaddr = InetAddress.getByName(ip);
…
```

- 자신의 주소,포트를 알리기 위해 서버에게 DatagramPacket 전송

```
…
DatagramSocket dsock = null;
try{
  dsock = new DatagramSocket();
  String line = null;
  // 전송
  DatagramPacket sendPacket = new DatagramPacket("".getBytes(),
"".getBytes().length, inetaddr, port);
  dsock.send(sendPacket);
…
```

3. UDP 타임 서버/클라이언트 작성

□타임 클라이언트 (예제 9-4)

- 서버에서 현재 시간 읽어들여 출력

```
byte[] buffer = new byte[200];
DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
dsock.receive(receivePacket);
// 받은 결과 출력.
String msg = new String(receivePacket.getData(), 0,
receivePacket.getData().length);
System.out.println("서버로 부터 전달받은 시간 :" + msg.trim());
```

- 실행