

5강

TCP 소켓 프로그래밍 III

1. UDP Echo 서버/클라이언트
2. UDP Time 서버/클라이언트

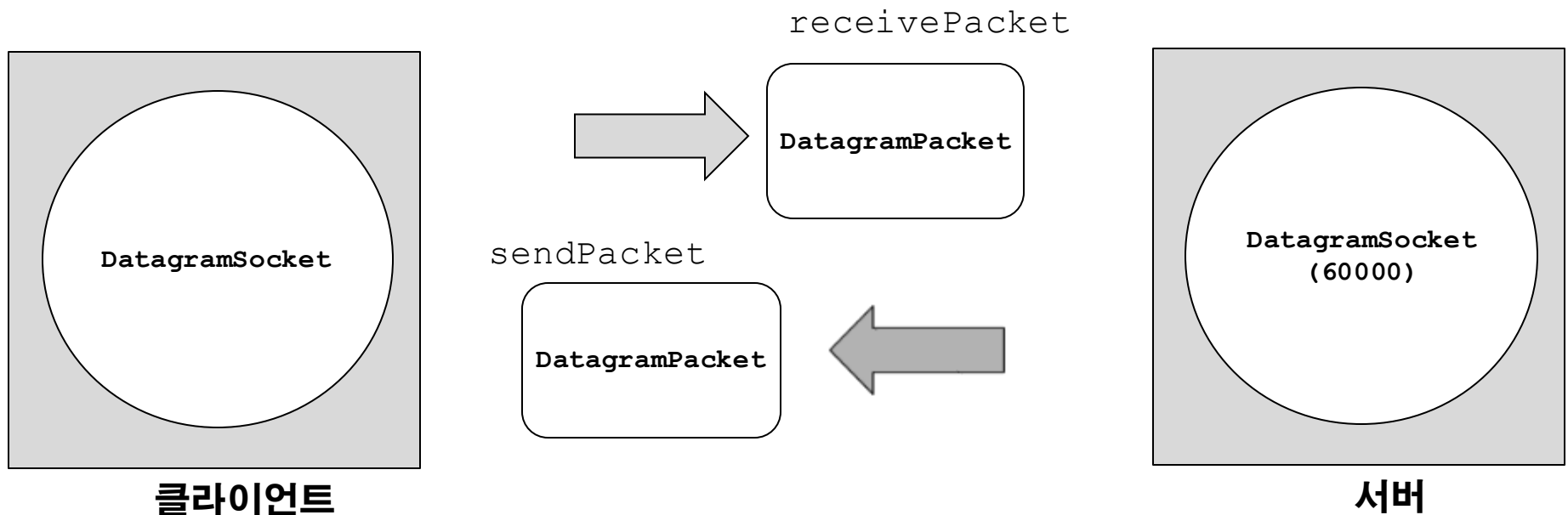
1. UDP Echo 서버/클라이언트

□ UDP 소켓 프로그래밍

1. 비 연결 지향 프로그래밍

2. TCP와 달리 연결되지 않은 상태로 데이터 통신을 하기 때문에 패킷이 유실될 가능성이 있다

3. 속도 면에서는 큰 장점이 있다. (처음 반응속도가 빠르다)



1. UDP Echo 서버/클라이언트

□ [실습] UDP Echo 서버/클라이언트 만들기

[1] UDPEchoServer : packet 받기 작성

```
// 1. UDP 소켓 생성
datagramSocket = new DatagramSocket( PORT );

// 2. 수신 대기
log( "packet 수신 대기" );
DatagramPacket receivePacket = new DatagramPacket( new byte[BUFFER_SIZE], BUFFER_SIZE );
datagramSocket.receive( receivePacket );
```

- **DatagramSocket** 객체를 **PORT**번호와 함께 생성한다.
- 수신 패킷을 생성한다. 생성할 때는 수신할 데이터의 버퍼를 미리 잡아준다.
(**UDP** 패킷 통신에는 버퍼사이즈가 고정되어 있다)
- 대기상태인지 코드를 작성하고 실행 해보자.

1. UDP Echo 서버/클라이언트

□ [실습] UDP Echo 서버/클라이언트 만들기

[2] UDPEchoClient : packet 보내기 작성

```
// 1. UDP 소켓 생성
datagramSocket = new DatagramSocket();

// 2. 데이터 보내기
String message = "Hello World";
byte[] sendData = message.getBytes( "UTF-8" );

DatagramPacket sendPacket =
    new DatagramPacket( sendData, sendData.length, new InetSocketAddress( SERVER_ADDRESS, SERVER_PORT ) );
datagramSocket.send( sendPacket );
```

- **DatagramSocket** 객체를 기본 생성자로 생성한다.
- 송신 패킷을 생성한다. 생성할 때는 보낼 데이터의 바이트 배열과 배열의 크기 그리고 서버 IP와 서비스 포트가 필요하다.
- **DatagramSocket** 객체의 **send** 메서드로 보낸다.
- 실행 시키면 서버가 대기 상태에서 **packet**을 받는지 확인해 보자.

1. UDP Echo 서버/클라이언트

□ [실습] UDP Echo 서버/클라이언트 만들기

[3] UDPEchoServer : packet 받고 다시 보내기(echo)

```
//3. 수신 데이터 출력
String message = new String( receivePacket.getData(), 0, receivePacket.getLength(), "UTF-8" );
log( "packet 수신:" + message );

// 4. 데이터 보내기
DatagramPacket sendPacket = new DatagramPacket( receivePacket.getData(), receivePacket.getLength(),
receivePacket.getAddress(), receivePacket.getPort() );

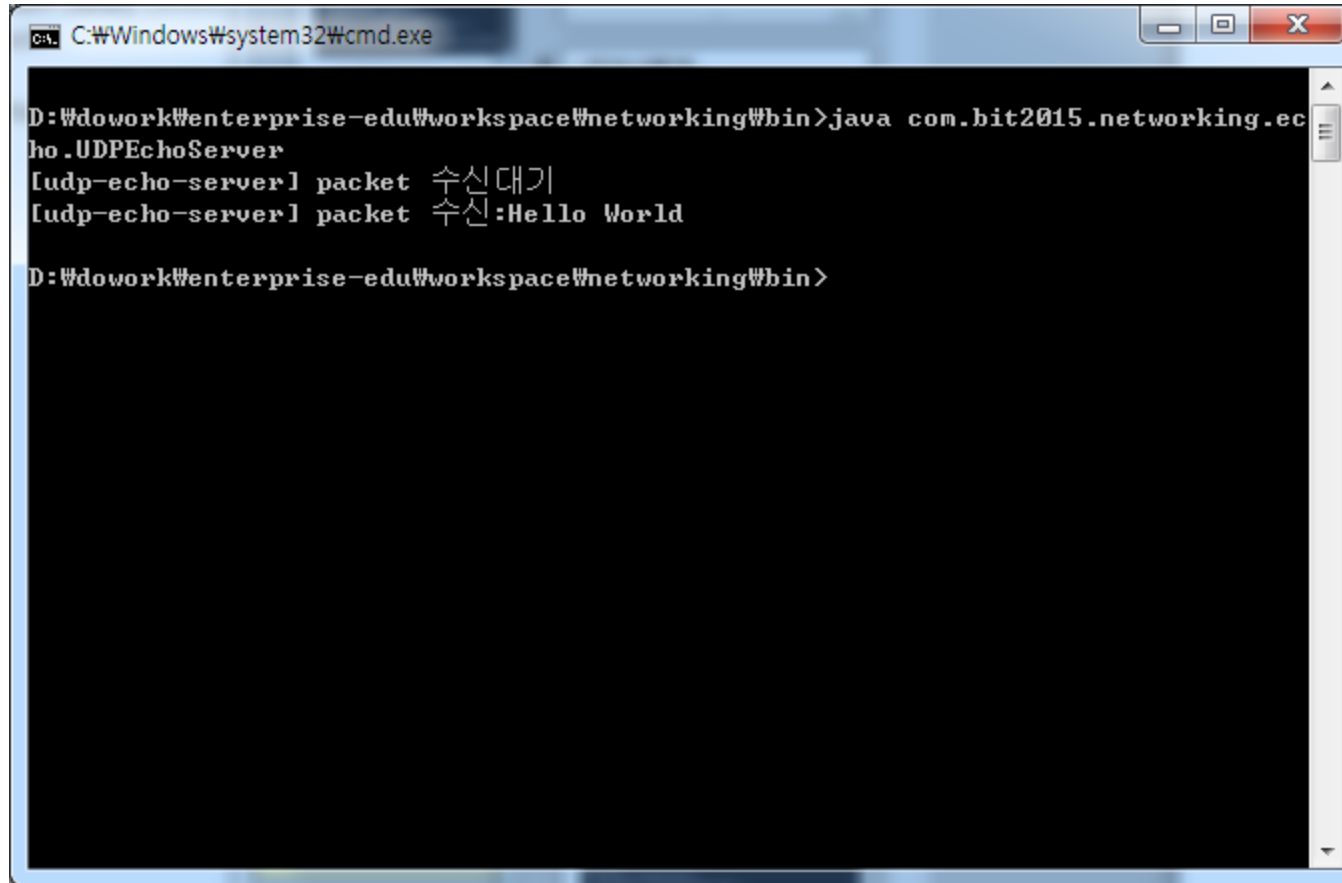
datagramSocket.send( sendPacket );
```

- 수신 패킷에서 클라이언트가 보낸 데이터를 바이트 배열로 가져올 수 있다.
- 수신 패킷에 있는 주소와 **port**는 클라이언트가 읽기 대기 중 인 **DatagramSocket**의 정보이다.
- 송신 패킷을 만들 때 데이터와 함께 사용하게 된다.
- 수신 데이터를 화면에 출력해서 확인하고 다시 송신 패킷을 보내 보자.

1. UDP Echo 서버/클라이언트

❑ [실습] UDP Echo 서버/클라이언트 만들기

[3] UDPEchoServer : packet 받고 다시 보내기(echo)



```
C:\Windows\system32\cmd.exe

D:\work\enterprise-edu\workspace\networking\bin>java com.bit2015.networking.ec
ho.UDPEchoServer
[udp-echo-server] packet 수신대기
[udp-echo-server] packet 수신:Hello World

D:\work\enterprise-edu\workspace\networking\bin>
```

1. UDP Echo 서버/클라이언트

❑ [실습] UDP Echo 서버/클라이언트 만들기

[4] UDPEchoClient : packet 받기 작성

```
// 3. 데이터 받기
```

```
DatagramPacket receivePacket = new DatagramPacket( new byte[ BUFFER_SIZE ], BUFFER_SIZE );  
datagramSocket.receive( receivePacket );
```

```
// 4. 메시지 출력
```

```
String message = new String( receivePacket.getData(), 0, receivePacket.getLength(), "UTF-8" );  
System.out.println( "<<" + data );
```

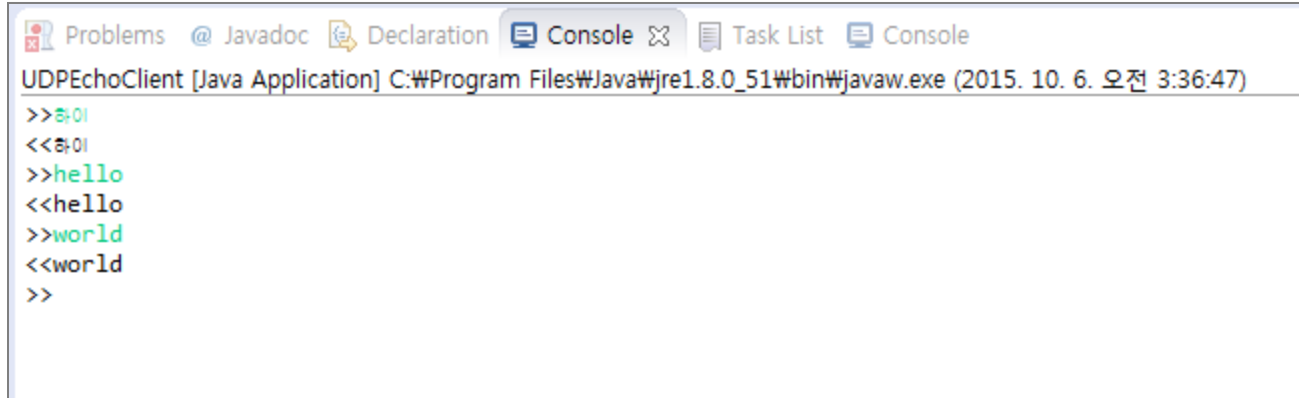
Problems @ Javadoc Declaration Console Task List Console

<terminated> UDPEchoClient [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 10. 6. 오전 3:26:27)

<<Hello World

1. UDP Echo 서버/클라이언트

□ [과제] UDP Echo 서버/클라이언트 완성



```
Problems @ Javadoc Declaration Console Task List Console
UDPEchoClient [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 10. 6. 오전 3:36:47)
>>하이
<<하이
>>hello
<<hello
>>world
<<world
>>
```

- 위와 같이 계속 에코서비스를 받을 수 있도록 서버/클라이언트 를 완성한다.
(**Loop** 처리와 **Scanner** 를 사용한다.)
- 서버에 스레드 기반에 다중처리를 하지 않고 다중 처리 여부를 확인해 본다.

2. UDP Time 서버/클라이언트

□ 기능

1. 클라이언트는 서버에 서버 시간을 요청한다.
2. 서버는 서버 시간을 클라이언트에게 제공해 준다.
3. 데이터가 간단하고 서버가 빠르게 반응해야 하므로 타임 서비스에는 **UDP**를 쓰는 것이 좋다.

2. UDP Time 서버/클라이언트

□ [과제] UDP 타임 서버/클라이언트 만들기

1. 클라이언트가 요청하면 서버는 다음과 같은 포맷으로

“2000-04-28 23:20:15 오후” 클라이언트에게 타임을 제공해 준다.

2. 다음 코드는 현재 서버 시간을 위와 같은 포맷의 **String**을 만드는 코드이다.

테스트 하고 서버에 응용한다.

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

3. 패키지는 **com.kosta.udp.time**

4. 서버/클라이언트 클래스 이름은 **UdpTimeServer / UdpTimeClient** 로 한다.

5. 클라이언트가 서버에 전송하는 요청 메시지는 “” 이다.