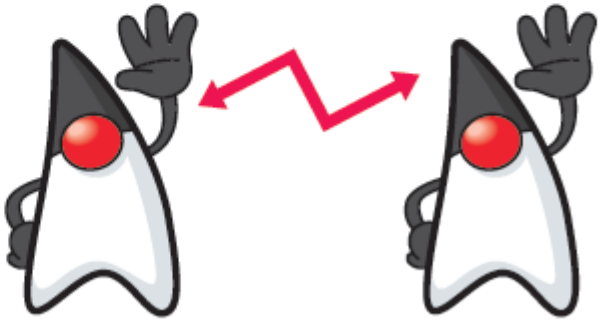
17장 네트워크 프로그래밍

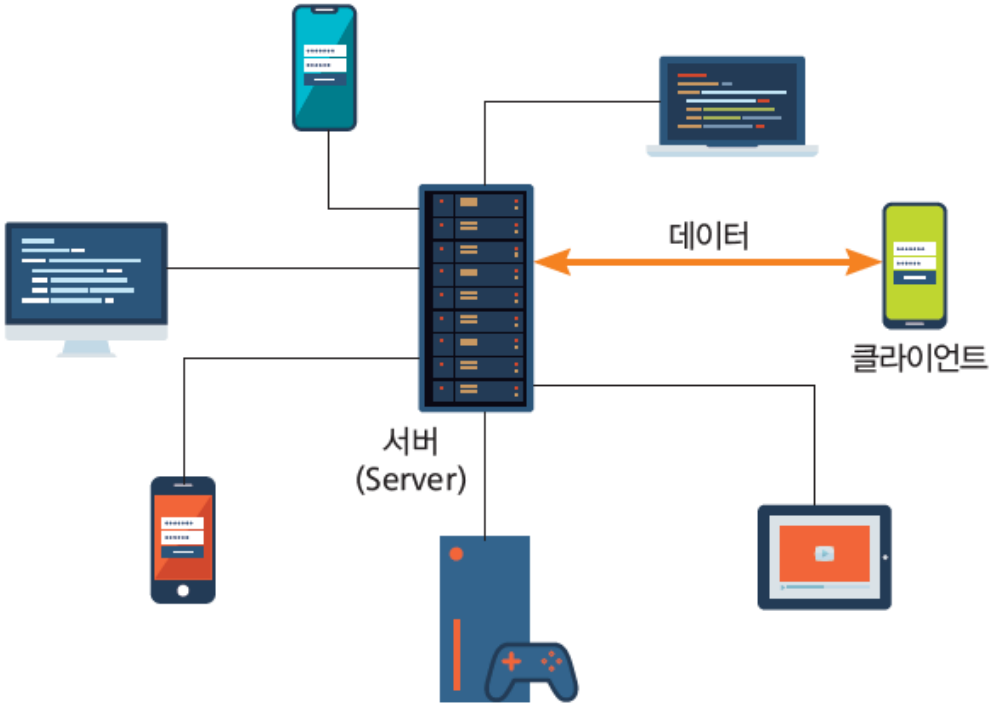
17장의 목표

1. www.google.com과 같은 호스트의 IP 주소를 얻는 방법을 알고 있습니까?
2. 온라인 웹 페이지의 데이터를 표시할 수 있습니까?
3. 소켓을 이용한 네트워킹 프로그래밍을 작성할 수 있습니까?
4. 무연결 UDP 프로그래밍을 수행할 수 있습니까?
5. 간단한 채팅 프로그램을 만들 수 있나요?



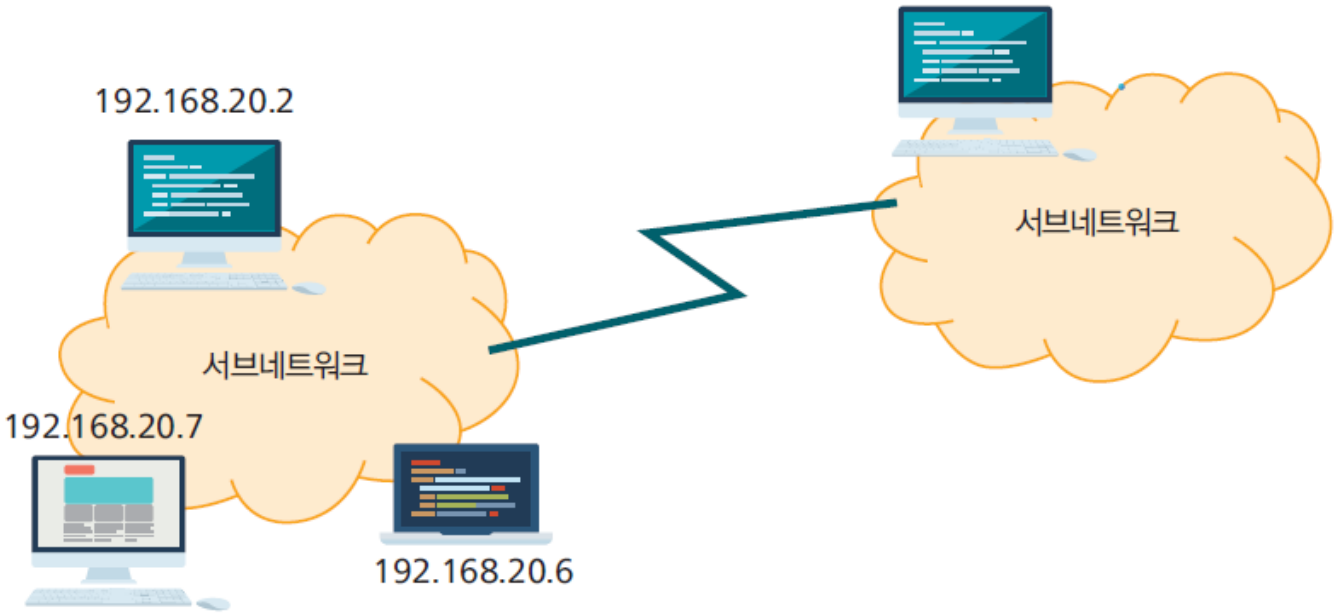
서버와 클라이언트

- 서버는 여러 명의 사용자들에게 서비스를 제공하는 컴퓨터이고 클라이언트는 서비스를 요청해서 사용하는 컴퓨터를 의미한다



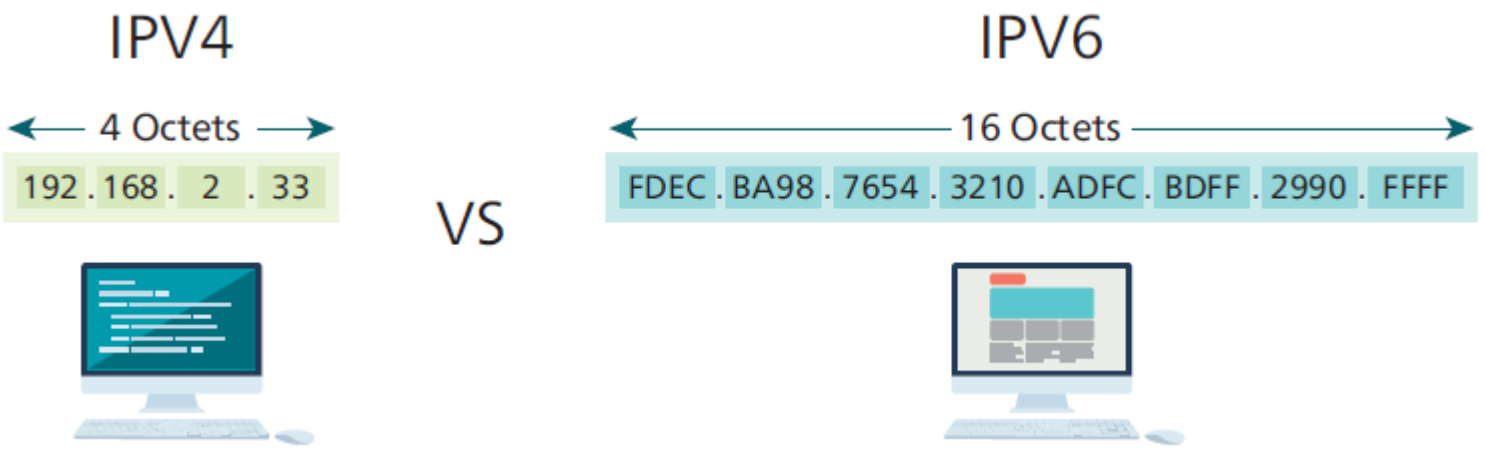
IP 주소(IPv4)

- IP 주소(IP address)는 네트워크에 존재하는 컴퓨터를 유일하게 식별하는 숫자이다.



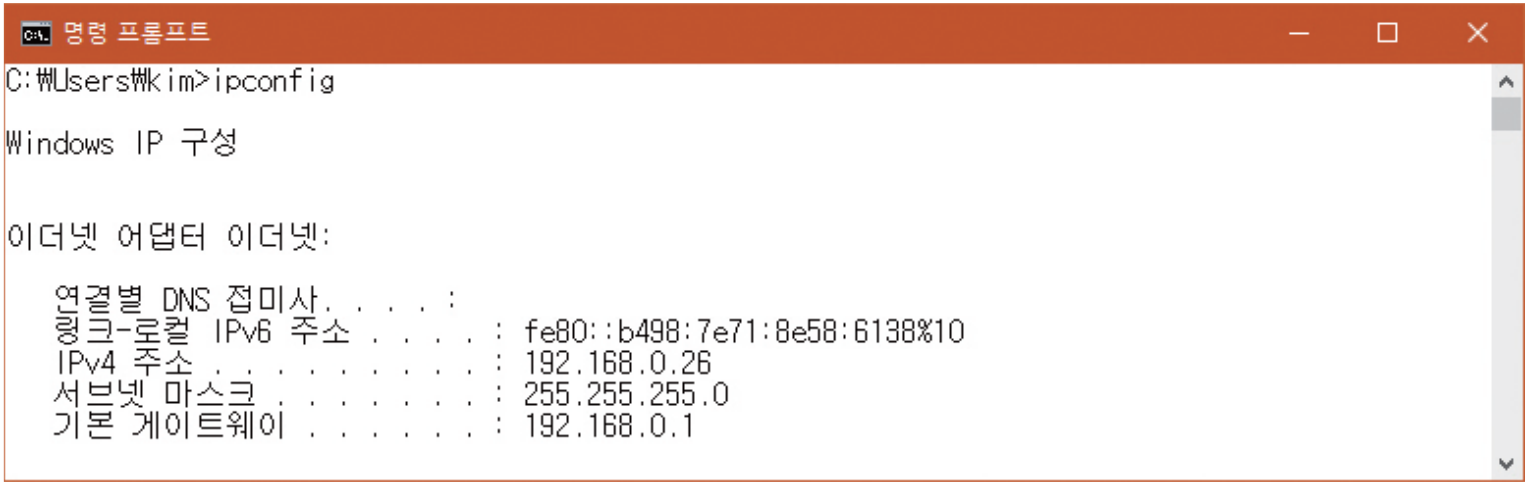
IP 주소(IPv6)

- 최근에는 인터넷상의 컴퓨터 증가로 128비트를 사용하는 IPv6 규격이 사용된다.



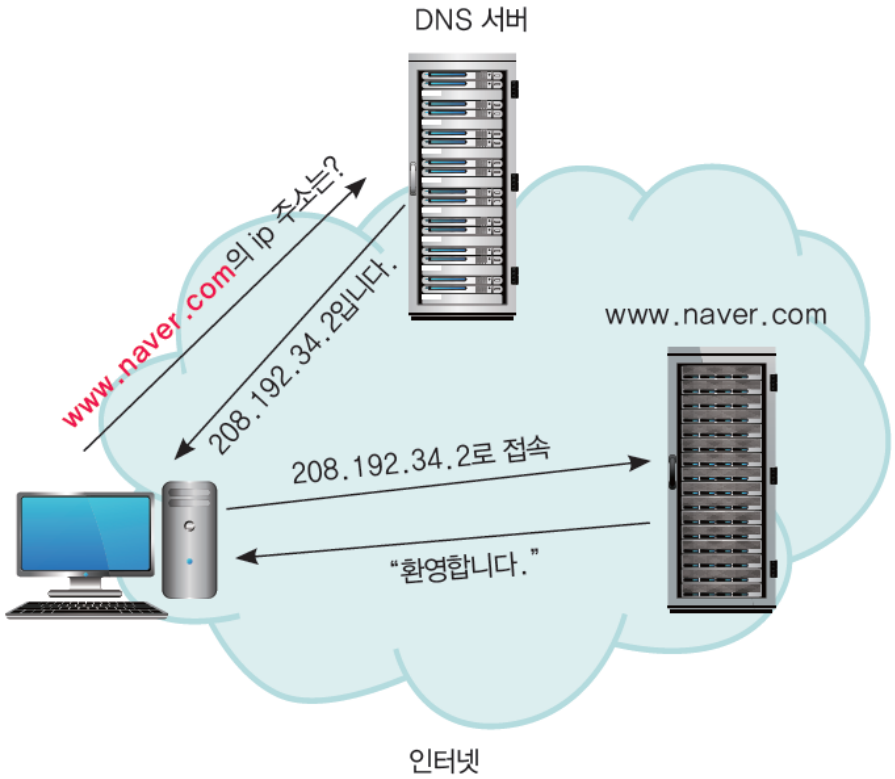
IP 주소

- 자기 컴퓨터의 IP 주소를 알아보려면 다음과 같이 ipconfig 명령어를 실행시키면 된다.



호스트 이름, DNS, URL

- DNS(Domain Name System): 숫자 대신 기호를 사용하는 주소
- DNS 서버: 기호 주소를 숫자 주소가 변환해주는 서버
- URL(Uniform Resource Locator): 인터넷 상의 자원을 나타내는 약속



URL

- **URL(Uniform Resource Locator)**은 인터넷 상의 파일이나 데이터베이스같은 자원에 대한 주소를 지정하는 방법이다

예제: 호스트 이름 -> IP 주소 프로그램

- 호스트 이름을 받아서 IP 주소를 반환하는 프로그램을 작성해보자. 인터넷 주소는 InetAddress 클래스가 담당한다

```
public class host2ip
{
  public static void main ( String[] args ) throws IOException   {
    String hostname = "www.naver.com";

    try     {
      InetAddress address = InetAddress.getByName(hostname);
      System.out.println("IP 주소: " + address.getHostAddress());
    }
    catch ( UnknownHostException e )  {
      System.out.println(hostname + "의 IP 주소를 찾을 수 없습니다. " );
    }
  }
}
```

```
IP 주소: 125.209.222.142
```

중간점검

중간점검

1. IP 주소와 호스트 이름은 어떻게 다른가?
2. 자신을 가리키는 IP 주소는 무엇인가?
3. DNS 서버가 하는 역할은 무엇인가?

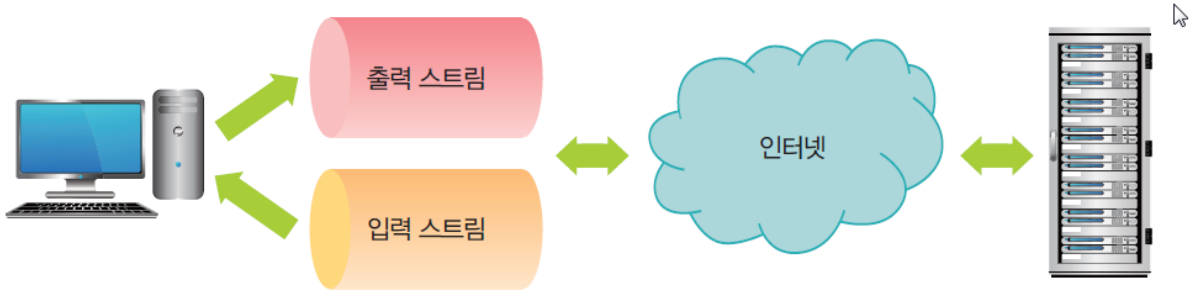
웹에서 파일 다운로드

- **java.net.URL을 이용하여** 우리의 프로그램과 인터넷 상의 원격 컴퓨터를 연결한다. 그리고 원격 컴퓨터가 가지고 있는 자원에 접근할 수 있다.

형식

```
try {
    URL testURL = new URL("http://www.naver.com/");
    // 여기에 더 많은 코드가 들어간다.

} catch (MalformedURLException e){
    // 예외 처리
}
```



예제: URLConnection 사용

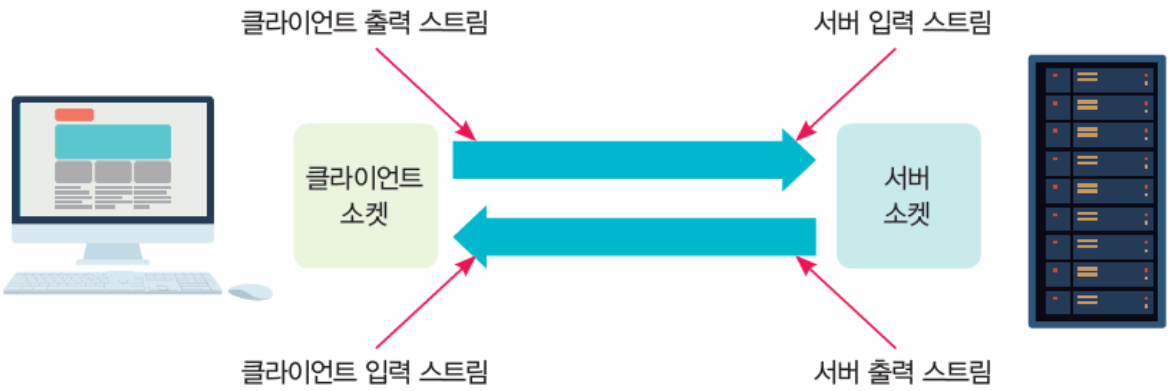
```
public class URLConnectionReader {
    public static void main(String[] args) throws Exception {
        URL site = new URL("https://www.naver.com/");
        URLConnection url = site.openConnection();
        BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                            url.getInputStream()));
        String inLine;

        while ((inLine = in.readLine()) != null)
            System.out.println(inLine);
        in.close();
    }
}
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
...
```

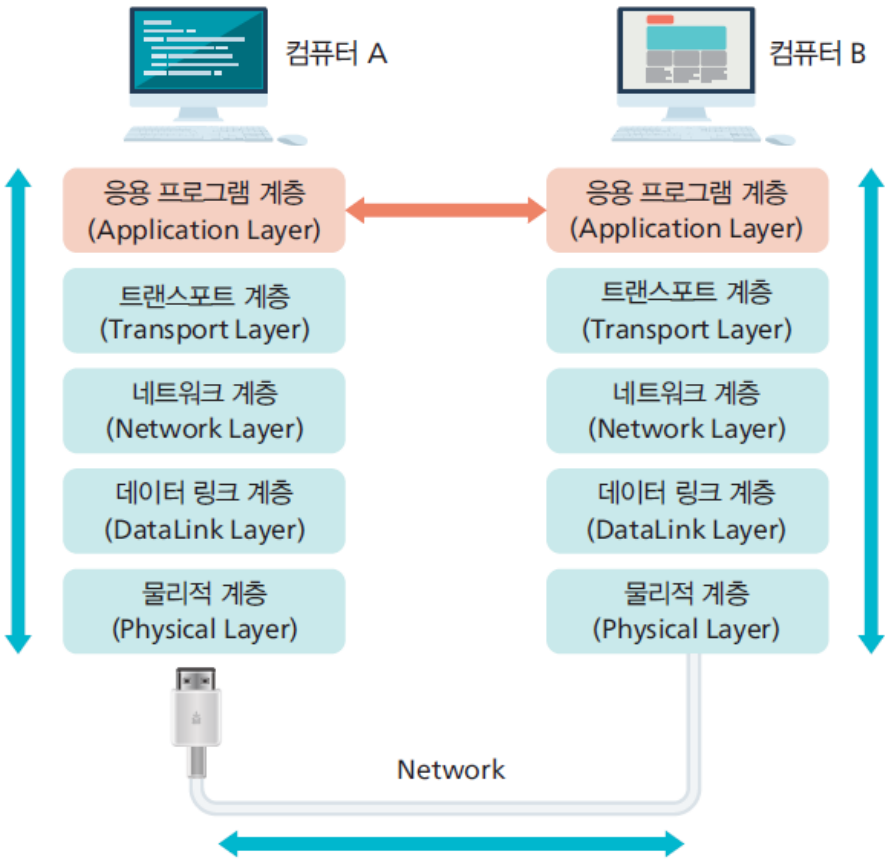
TCP를 이용한 통신

- 소켓(Socket)과 같은 저수준의 네트워크 통신 기능이 필요한 경우도 있다. 예를 들면 자바로 클라이언트-서버 응용 프로그램을 만드는 경우이다



프로토콜

- 프로토콜(Protocol)은 컴퓨터 간에 상호통신을 할 때 데이터를 원활하고 신뢰성 있게 주고 받기 위해 필요한 약속을 규정하는 것이다.



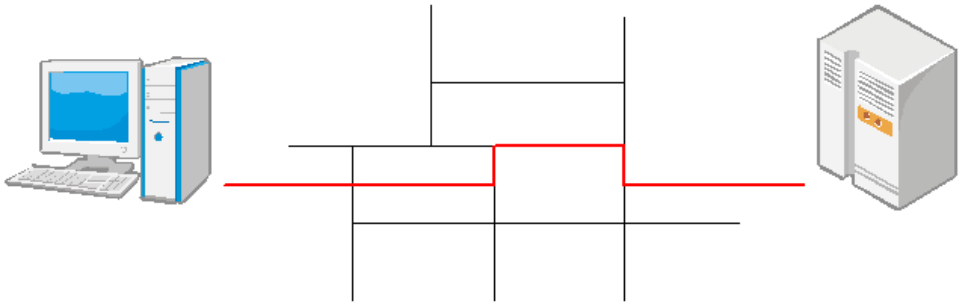
프로토콜

응용 프로그램 계층 (application layer)	• 응용 프로그램 계층은 네트워크 통신이 필요한 응용 프로그램이 있는 곳이다. • 이러한 애플리케이션의 예로는 이메일 클라이언트 및 웹 브라우저가 있다. • 이러한 응용 프로그램은 전송 계층을 사용하여 원격 호스트에 연결하라는 요청을 보낸다.
전송 계층 (transport layer)	• 전송 계층은 서로 다른 호스트에서 실행 중인 응용 프로그램 사이의 연결을 설정한다. • 안정적인 연결에는 TCP를 사용하고 빠른 연결에는 UDP를 사용한다. • 포트 번호를 할당하여 상위 응용 프로그램에서 실행 중인 프로세스를 추적하고 네트워크 계층을 사용하여 TCP/IP 네트워크에 접근한다.
네트워크 계층(network layer)	• 네트워크 계층은 패킷을 생성하여서 서로 다른 네트워크로 전송한다. • IP 주소를 사용하여 패킷의 소스와 대상을 식별한다.
데이터 링크 계층(datalink layer)	• 프레임(frame)을 생성하고, 점대점 방식으로 프레임을 이동시킨다. • 이 프레임은 캡슐화된 패킷을 감싸고 있으며, MAC 주소를 사용하여 소스와 대상을 식별한다.
물리적 계층(physical layer)	• 프레임 안의 비트들이 전기, 빛, 무선 신호 등으로 부호화된다. • 예를 들어 RS-232, SONET, WiFi와 같은 프로토콜이 여기에 속한다.

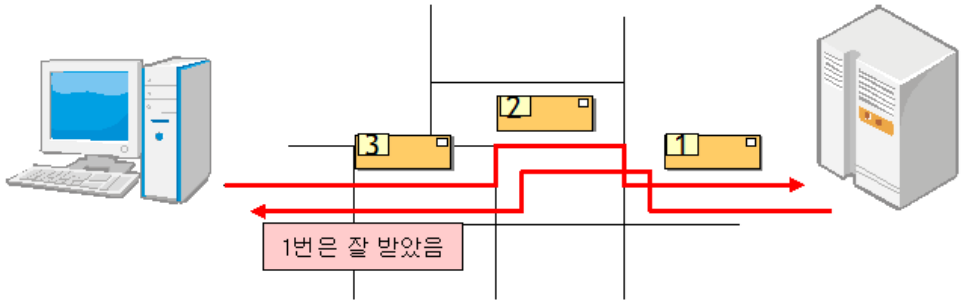
TCP

- TCP(Transmission Control Protocol)는 신뢰성있게 통신하기 위하여 먼저 서로 간에 연결을 설정한 후에 데이터를 보내고 받는 방식

(1) 먼저 가능한 경로 중에서 하나가 결정된다.



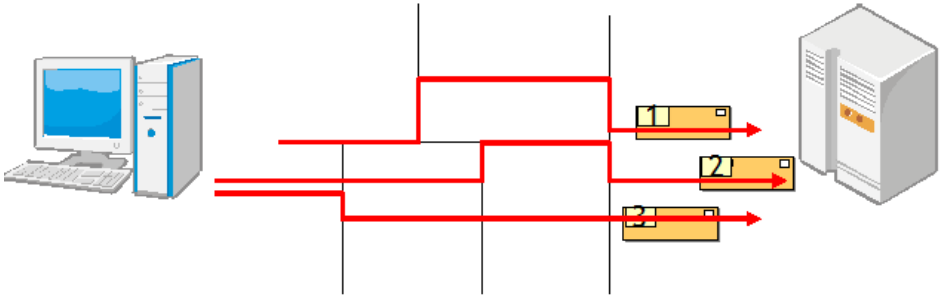
(2) 데이터는 패킷으로 나누어지고 패킷에 주소를 붙여서 전송한다.



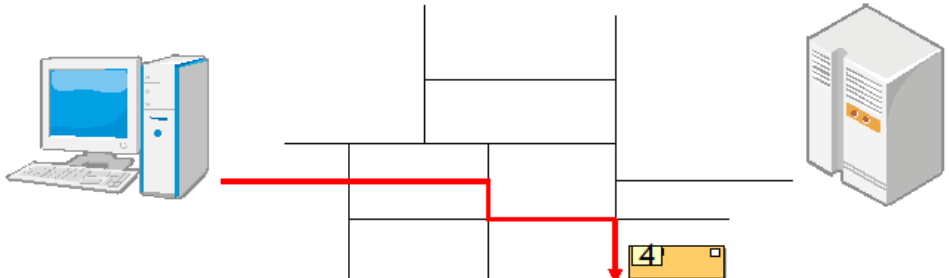
UDP

- UDP(User Datagram Protoocol)는 데이터를 몇 개의 고정 길이의 패킷(다이어그램이라고 불린다)으로 분할하여 전송

(1) 데이터를 패킷으로 나누어서 패킷에 주소를 붙이고 전송한다.

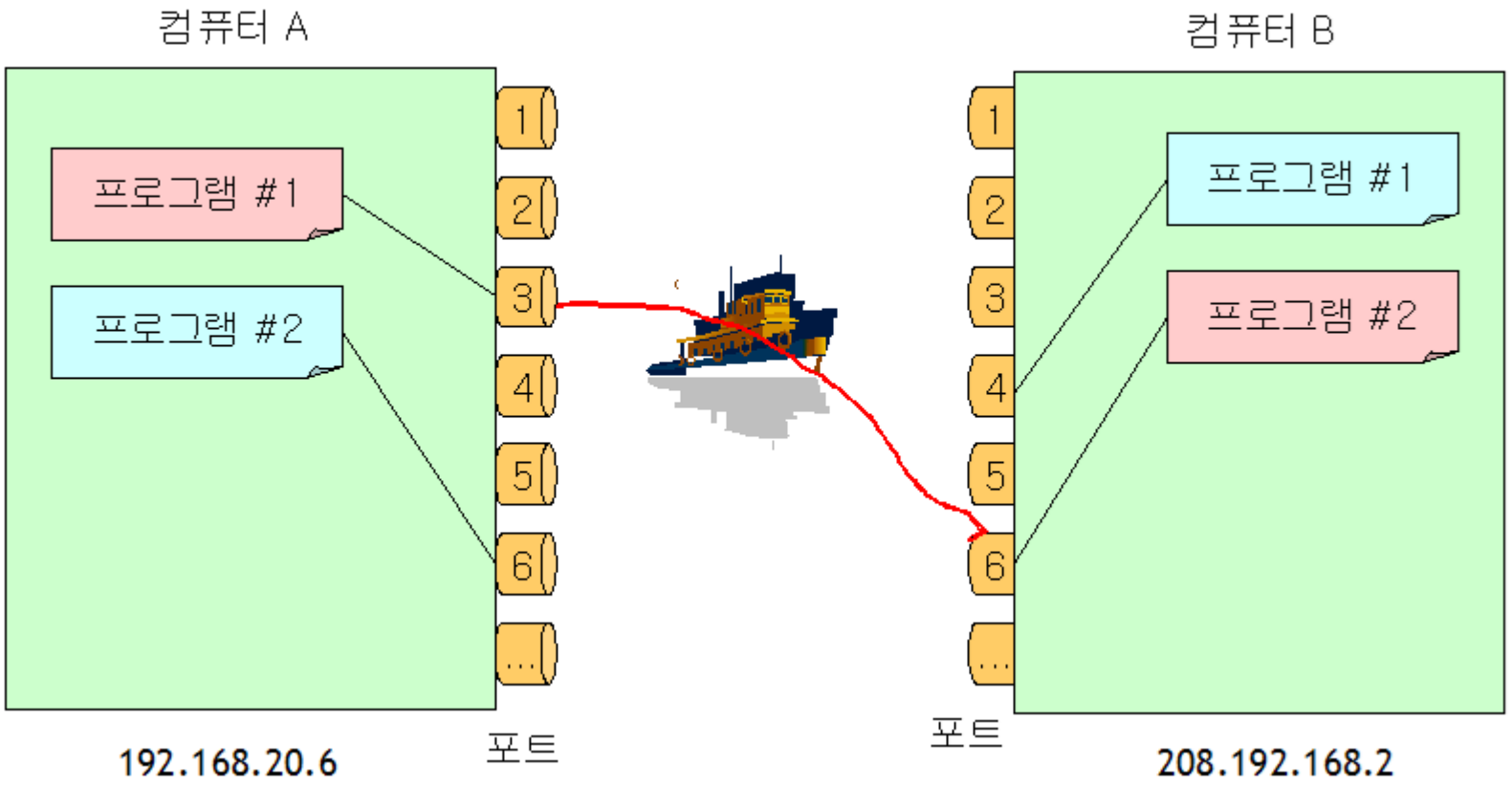


(2) 페킷의 순서가 지켜지지 않으며 패킷이 분실될 수도 있다.



포트

- 포트(port): 가상적인 통신 선로



Socket 클래스

- 소켓(socket): TCP를 사용하여 응용 프로그램끼리 통신을 하기 위한 연결 끝점(end point)

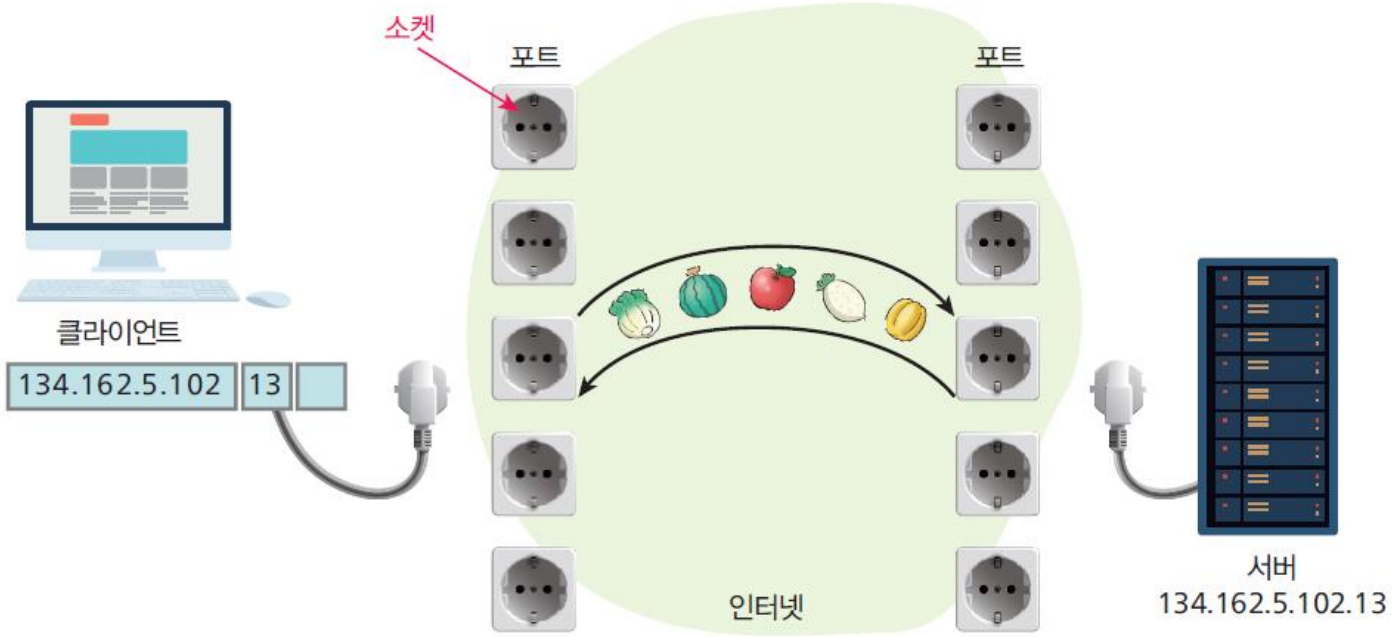


그림 17.1 소켓의 개념

예제: 날짜 서버에 연결하기

- 인터넷에 보면 정확한 현재 시각을 알려주는 서버들이 존재한다(원자 시계로 측정된 시각이다). 여기서는 미국 시각을 알려주는 서버(NIST 서버)에 소켓을 이용하여 접속해 보자. 소켓은 Socket 클래스에 의하여 제공된다. Socket 클래스 생성자의 첫 번째 인수는 사이트 주소이고 두 번째 인수가 바로 포트 번호이다. NIST 서버를 지정하는 소켓을 생성하면 바로 서버와 연결된다. 소켓으로부터 입력 스트림을 얻어서 스트림에 읽으면 현재 시각을 알 수 있다.

```
59453 21-08-27 06:34:46 50 0 0  65.7 UTC(NIST) *
```

예제: 날짜 서버에 연결하기

```
// 소스를 입력하고 Ctrl+Shift+O를 눌러서 필요한 파일을 포함한다.
public class SocketTest {
        public static void main(String[] args) throws IOException {
                try (Socket s = new Socket("time-c.nist.gov", 13)) {
                        InputStream inStream = s.getInputStream();
                        Scanner in = new Scanner(inStream);

                        while (in.hasNextLine()) {
                                String line = in.nextLine();
                                System.out.println(line);
                        }
                }
        }
}
```

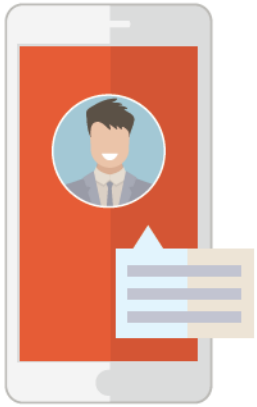
```
59453 21-08-27 06:34:46 50 0 0  65.7 UTC(NIST) *
```

중간점검

1. TCP 통신에서 새로운 연결이 만들어지는 과정을 설명하라.
2. accept() 메소드가 반환하는 값은 무엇인가?
3. 소켓에서 입력 스트림 객체를 얻는 메소드 이름은 무엇인가?

서버와 클라이언트 제작

- 두 사람이 간단한 채팅을 할 수 있는 프로그램을 작성해보자. GUI도 없고, 프로토콜도 “quit” 뿐이다.



```
C> java Server Enter↵

연결을 기다리고 있음
클라이언트와 연결되었음
클라이언트가 보낸 문자열: hello
클라이언트로 보낼 문자열을 입력하고
엔터키를 치세요: hello too
클라이언트에서 연결을 종료하였음
```

Server

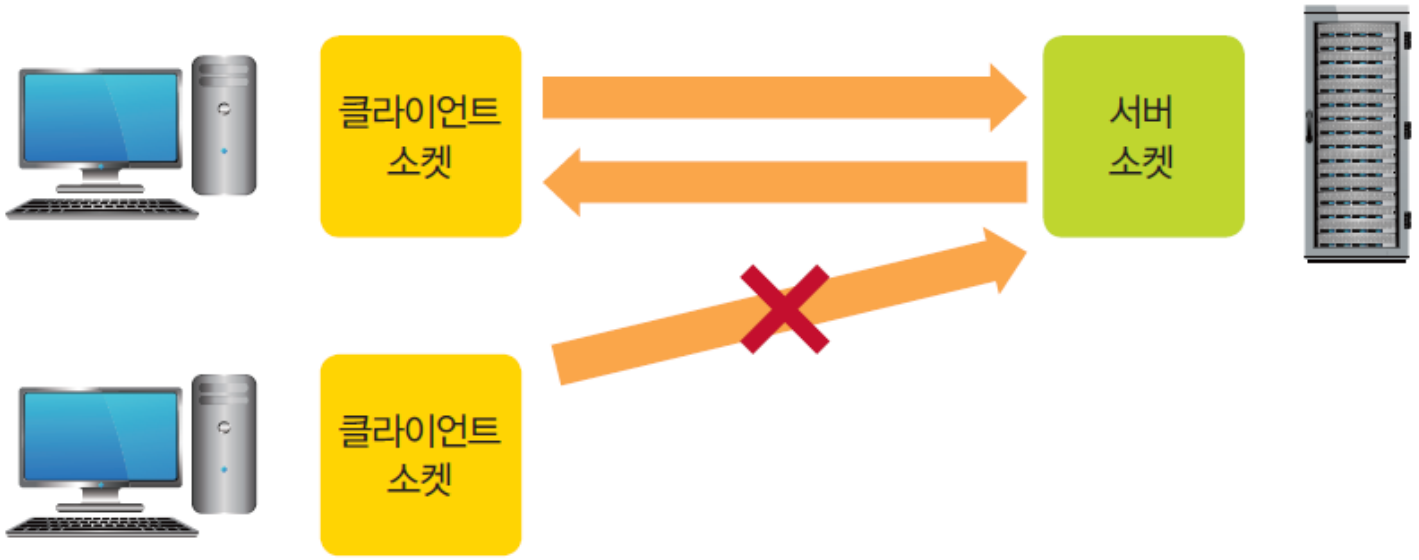
```
C> java Client Enter↵

서버로 보낼 문자열을 입력하고 엔터키를
치세요: hello
서버로부터 온 메시지: hello too
서버로 보낼 문자열을 입력하고 엔터키를
치세요: quit
```

Client

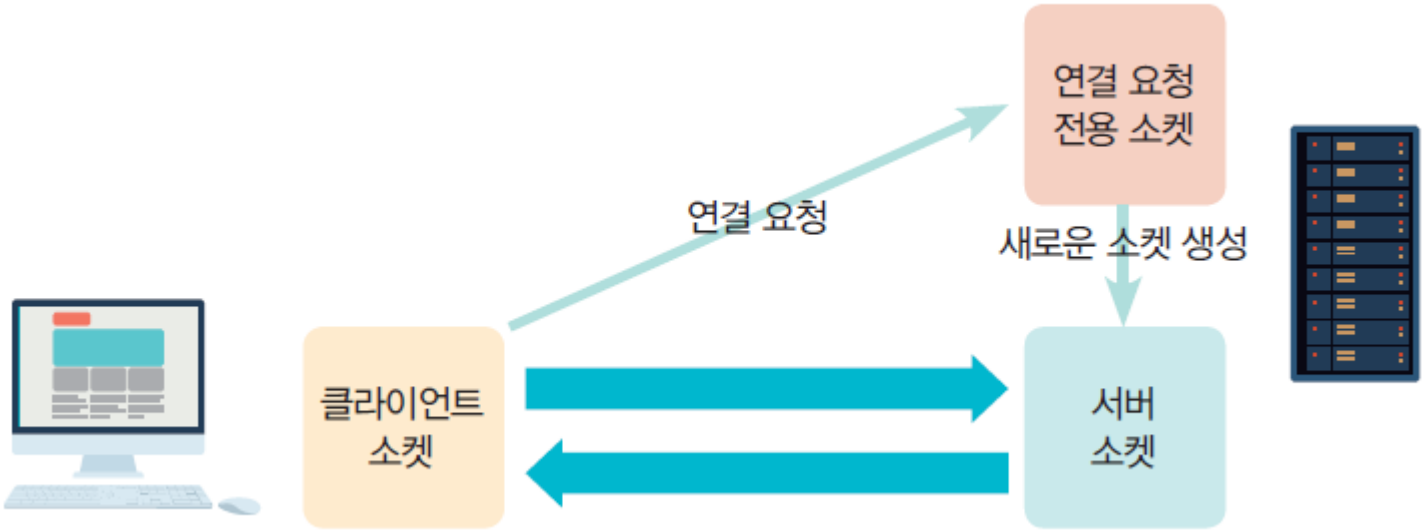
서버와 클라이언트 제작

- 서버가 하나의 소켓만을 사용한다면 문제가 발생한다.



Well-Known Socket

- 서버는 연결 요청만을 받는 소켓을 따로 가지고 있다. 모든 클라이언트는 이곳으로 연결 요청을 하여야 한다. 연결 요청이 승인되면 서버는 해당 클라이언트를 상대하는 새로운 소켓을 만든다. 클라이언트는 이 새로운 소켓을 사용하여 데이터를 주고받는다



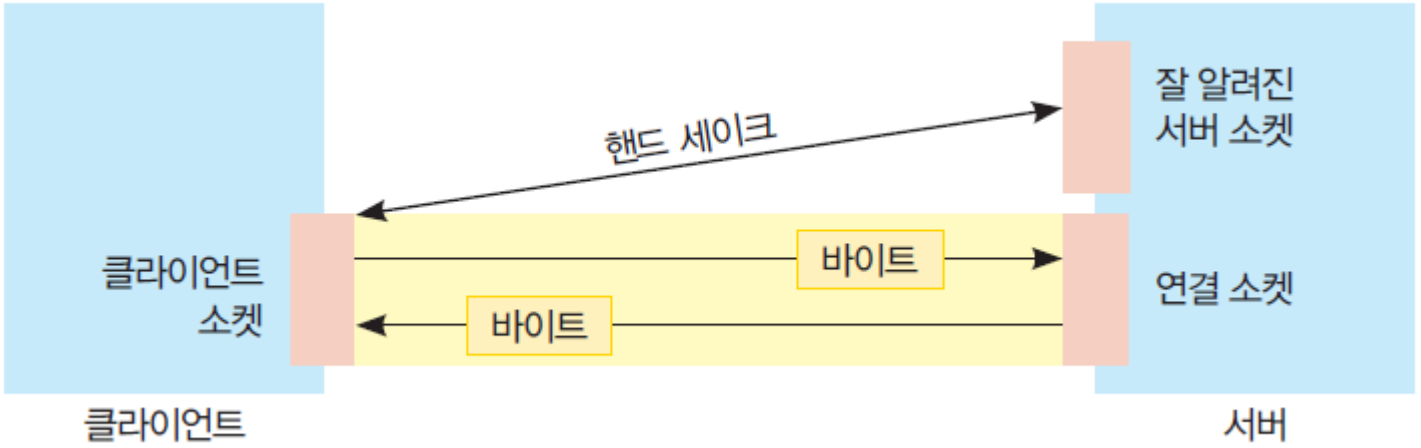
ServerSocket과 Socket 클래스

1. ServerSocket 객체 생성
   - ServerSocket server = new ServerSocket(portNumber, queueLength);
   - 위와 같이 ServerSocket 생성자를 호출하면 포트 번호가 portNumber인 포트를 기반으로 하는 소켓을 생성한다. queueLength는 서버에 연결되기를 기다리는 클라이언트의 최대 개수이다. portNumber는 클라이언트가 서버 컴퓨터에서 서버 애플리케이션을 찾기 위하여 필요하다. 각 클라이언트는 이 포트 번호를 이용하여 서버에게 연결을 요청하여야 한다

ServerSocket과 Socket 클래스

2. accept() 메소드 호출

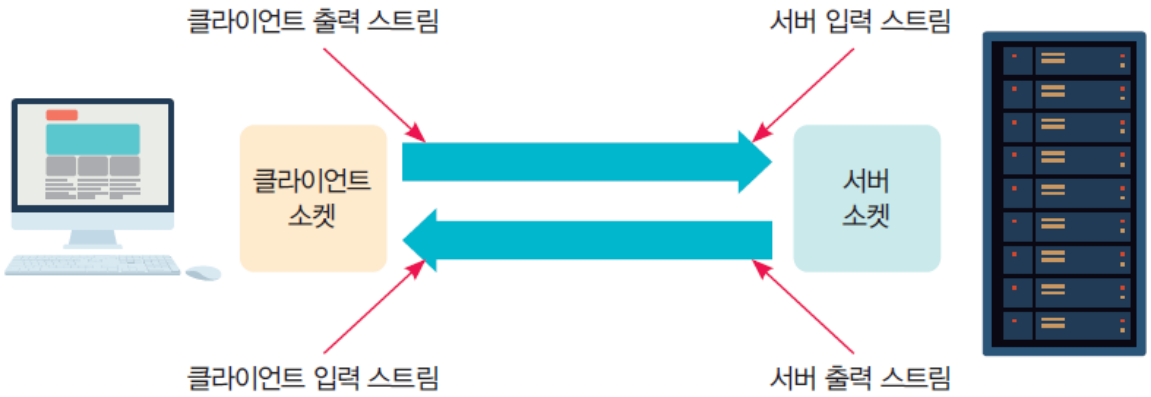
- 서버는 클라이언트가 연결을 시도하기를 기다린다. 이것은 ServerSocket의 메소드인 accept()를 호출하면 된다.
- Socket clientSocket = server.accept();



ServerSocket과 Socket 클래스

3. 소켓으로부터 스트림 객체를 얻는다.

- InputStream input = clientSocket.getInputStream();
- OutputStream output = clientSocket.getOutputStream();



채팅 서버 제작

```
public class Server {
        public static void main(String[] args) {
                ServerSocket serverSocket=null;
                Socket clientSocket=null;
                BufferedReader in=null;
                PrintWriter out=null;
                Scanner sc = new Scanner(System.in);

                try {
                        serverSocket = new ServerSocket(5000);
                        System.out.println("연결을 기다리고 있음");
                        clientSocket = serverSocket.accept();
                        out = new
PrintWriter(clientSocket.getOutputStream());
                        in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                        System.out.println("클라이언트와 연결되었음");
```

채팅 서버 제작

```
                while (true) {
                    String msg = in.readLine();
                    if (msg.equalsIgnoreCase("quit")) {
                        System.out.println("클라이언트에서 연결을 종료하였음");
                        break;
                    }
                    System.out.println("클라이언트가 보낸 문자열: " + msg);
                    System.out.print("클라이언트로 보낼 문자열을 입력하고 엔터키를 치세요: ");
                    String omsg = sc.nextLine();
                    out.write(omsg + "\n");
                    out.flush();
                }
                out.close();
                clientSocket.close();
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

채팅 클라이언트 제작

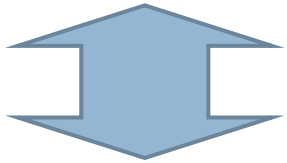
```
public class Client {
        public static void main(String[] args) throws IOException {
                Socket clientSocket = null;
                BufferedReader in = null;
                PrintWriter out = null;
                final Scanner sc = new Scanner(System.in);
                try {
                        clientSocket = new Socket("localhost", 5000);
                        out = new
PrintWriter(clientSocket.getOutputStream());
                        in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                        String msg;
```

채팅 클라이언트 제작

```
                while (true) {
                    System.out.print("서버로 보낼 문자열을 입력하고 엔터키를 치세요: ");
                    msg = sc.nextLine();
                    if (msg.equalsIgnoreCase("quit")) {
                        out.println(msg);
                        out.flush();
                        break;
                    }
                    out.println(msg);
                    out.flush();
                    msg = in.readLine();
                    System.out.println("서버로부터 온 메시지: " + msg);
                }
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                out.close();
                clientSocket.close();
            }
        }
}
```

서버와 클라이언트 프로그램 실행

```
C> java Client
서버로 보낼 문자열을 입력하고 엔터키를 치세요: hello
서버로부터 온 메시지: hello too
서버로 보낼 문자열을 입력하고 엔터키를 치세요: quit
```

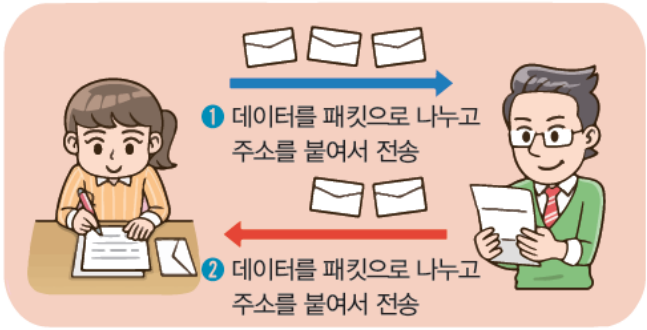


```
C> java Server
연결을 기다리고 있음
클라이언트와 연결되었음
클라이언트가 보낸 문자열: hello
클라이언트로 보낼 문자열을 입력하고 엔터키를 치세요: hello too
클라이언트에서 연결을 종료하였음
```

UDP를 이용한 서버와 클라이언트

- DatagramSocket 클래스
  - DatagramSocket()은 UDP 프로토콜을 사용하는 소켓을 생성
- DatagramPacket 클래스
  - DatagramPacket()은 UDP 패킷을 생성한다.

TCP 프로토콜



UDP 프로토콜

예제: UDP를 사용하여서 데이터 보내고 받기

- 문자열 1개를 UDP를 이용하여 보내고 받는 프로그램을 가지고 UDP 통신을 설명해보자.

우리는 여전히 우리 운명의 주인이다.

Sender 클래스

```
public class Sender {
    public static void main(String[] args) throws IOException {

        DatagramSocket socket = null;
        socket = new DatagramSocket();
        String s = "우리는 여전히 우리 운명의 주인이다.";
        byte[] buf = s.getBytes();

        // "address"의 "port"에 있는 클라이언트에게 데이터를 보낸다.
        InetAddress address = InetAddress.getByName("127.0.0.1"); // 로컬 호스트
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address,
                5000);
        socket.send(packet);
        socket.close();
    }
}
```

Receiver 클래스

```
public class Receiver {
    public static void main(String[] args) throws IOException {

        byte[] buf = new byte[256];

        DatagramSocket socket = new DatagramSocket(5000);  // 포트 번호: 5000
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);
        System.out.println(new String(buf));
    }
}
```

서버와 클라이언트의 실행

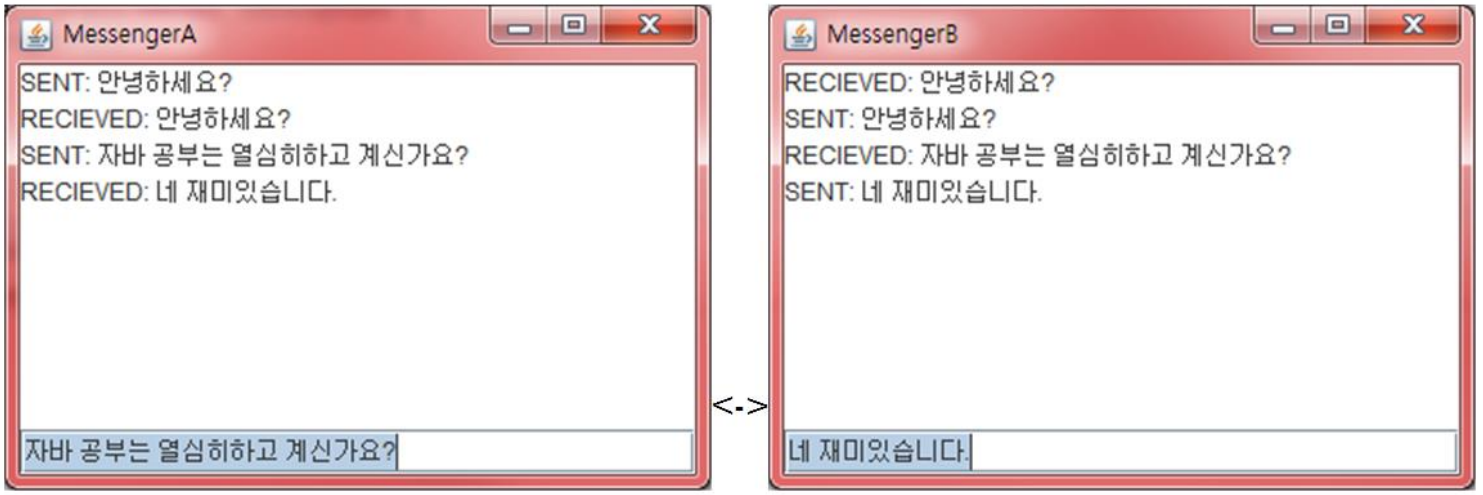
- 두개의 프로그램을 동시에 실행하여야 한다.


```
C> java Receiver
```

```
C> java Sender
우리는 여전히 우리 운명의 주인이다.
```

UDP를 이용한 서버와 클라이언트 작성

- 예제로 UDP 통신을 이용하여서 간단한 채팅을 할 수 있는 메신저를 작성하여 보자. 이 메신저는 정해진 상대와 텍스트를 주고 받을 수 있다.



MessengerA 클래스

```
public class MessengerA {
    protected JTextField textField;
    protected JTextArea textArea;
    DatagramSocket socket;
    DatagramPacket packet;
    InetAddress address = null;
    final int myPort = 5000;                    // 수신용 포트 번호
    final int otherPort = 6000;   // 송신용 포트 번호

    public MessengerA() throws IOException {
        MyFrame f=new MyFrame();
        address = InetAddress.getByName("127.0.0.1");
        socket = new DatagramSocket(myPort);
    }
```

MessengerA 클래스

```
// 패킷을 받아서 텍스트 영역에 표시한다.
    public void process() {
        while (true) {
            try {
                byte[] buf = new byte[256];
                packet = new DatagramPacket(buf, buf.length);
                socket.receive(packet); // 패킷을 받는다.
                // 받은 패킷을 텍스트 영역에 표시한다.
                textArea.append("RECIEVED: " + new String(buf) + "\n");
            }
            catch (IOException ioException) {
                ioException.printStackTrace();
            }
        }
    }
```

MessengerA 클래스

```
// 내부 클래스 정의
    class MyFrame extends JFrame implements ActionListener {

        public MyFrame() {
            super("MessengerA");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            textField = new JTextField(30);
            textField.addActionListener(this);

            textArea = new JTextArea(10, 30);
            textArea.setEditable(false);

            add(textField, BorderLayout.PAGE_END);
            add(textArea, BorderLayout.CENTER);
            pack();
            setVisible(true);
        }
```

MessengerA 클래스

```
public void actionPerformed(ActionEvent evt) {
            String s = textField.getText();
            byte[] buffer = s.getBytes();
            DatagramPacket packet;

            // 패킷을 생성한다.
            packet = new DatagramPacket(buffer, buffer.length, address,
                    otherPort);
            try {
                socket.send(packet);          // 패킷을 보낸다.
            } catch (IOException e) {
                e.printStackTrace();
            }
            textArea.append("SENT: " + s + "\n");
            textField.selectAll();
            textArea.setCaretPosition(textArea.getDocument().getLength());
        }
    }

    public static void main(String[] args) throws IOException {
        MessengerA m = new MessengerA();
        m.process();
    }
}
```

MessengerB 클래스

```
// 다음의 몇 개의 문장만 제외하고 MessengerA와 동일
....
public class MessengerB {
    ...
    final int myPort = 6000;
    final int otherPort = 5000;

    public MessengerB() throws IOException {
       ...
    }
    public static void main(String[] args) throws IOException {
        MessengerB m = new MessengerB();
        m.process();
    }
}
```

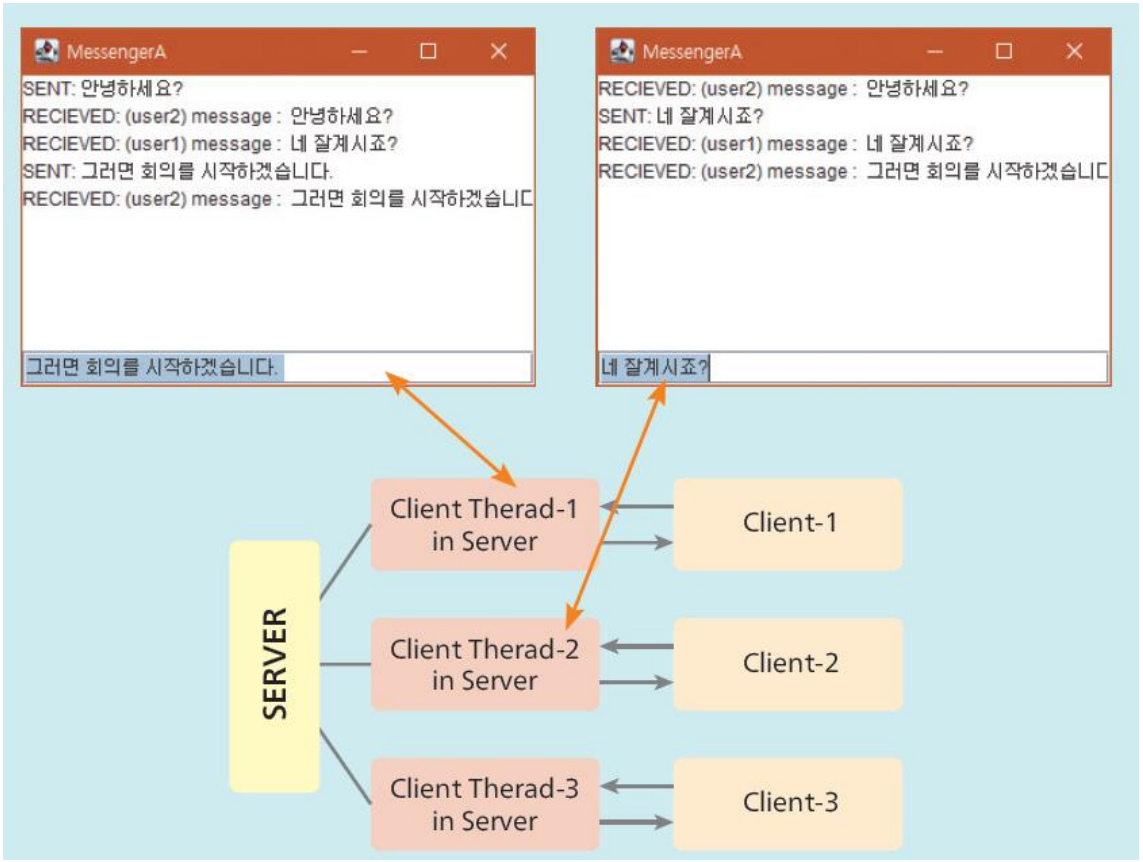
중간점검

중간점검

1. UDP의 장점과 단점은 무엇인가?
2. UDP에서는 패킷을 받을 상대방을 어떻게 지정하는가?
3. DatagramSocket 클래스에서 패킷을 보내고 받는 메소드 이름은?

Mini Project: 다자 회의 시스템

- 최근에 코로나 바이러스가 유행하면서 줌과 같은 인터넷 회의 시스템이 많이 사용되고 있다. 하나의 서버가 여러 클라이언트를 모아서 회의를 할 수 있는 프로그램을 작성해보자



Summary

- 네트워크에는 서버와 클라이언트 컴퓨터가 존재한다.
- IP 주소는 네트워크에서 컴퓨터를 식별하는 숫자이다.
- URL(Uniform Resource Locator)은 인터넷상의 파일이나 데이터베이스 같은 자원에 대한 주소를 지정하는 방법이다.
- 자바에서 InetAddress 클래스의 getByName()을 호출하면서 호스트 이름을 전달하면 IP 주소를 저장하고 있는 객체가 반환된다.
- 웹에서 파일을 다운로드하려면 java.net.URL 클래스를 사용한다.
- TCP(Transmission Control Protocol)는 신뢰성 있게 통신하기 위하여 먼저 서로 간에 연결을 설정한 후에 데이터를 보내고 받는 방식이다.
- TCP를 사용하여 응용 프로그램끼리 통신을 하기 위해서는 먼저 연결을 해야 한다. 연결을 하기 위해서는 연결 끝점(end point)이 있어야 하는데 그것이 바로 소켓이다.
- 자바에서 Socket 클래스와 ServerSocket 클래스는 클라이언트측과 서버측을 구현하는 데 사용된다. accept() 메소드는 클라이언트와 연결이 되면 새로운 Socket 객체를 생성하여 반환한다.
- UDP(User Datagram Protocol) 프로토콜을 이용한 방식은 편지와 비슷하다. 자바에서는UDP를 DatagramPacket과 DatagramSocket 클래스로 지원한다.

Q & A