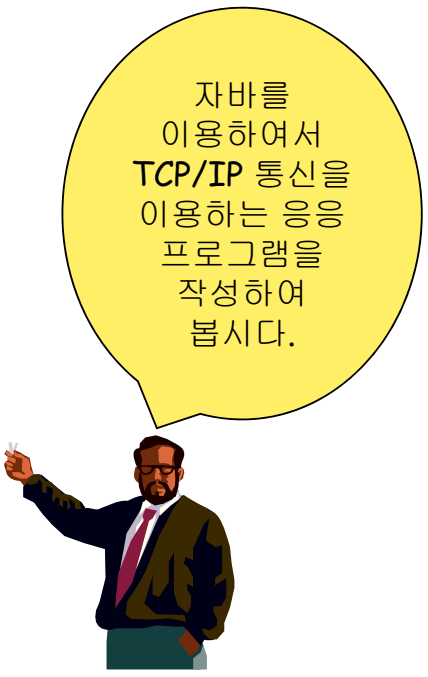
Power Java

제25장 네트워크 프로그래밍

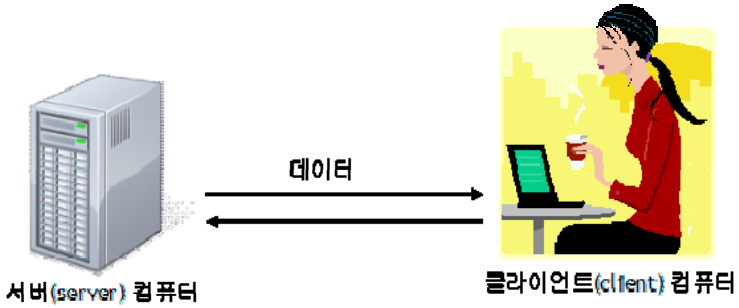
이번 장에서 학습할 내용

- 네트워크 프로그래밍의 개요
- URL 클래스
- TCP를 이용한 통신
- TCP를 이용한 서버 제작
- TCP를 이용한 클라이언트 제작
- UDP를 이용한 통신



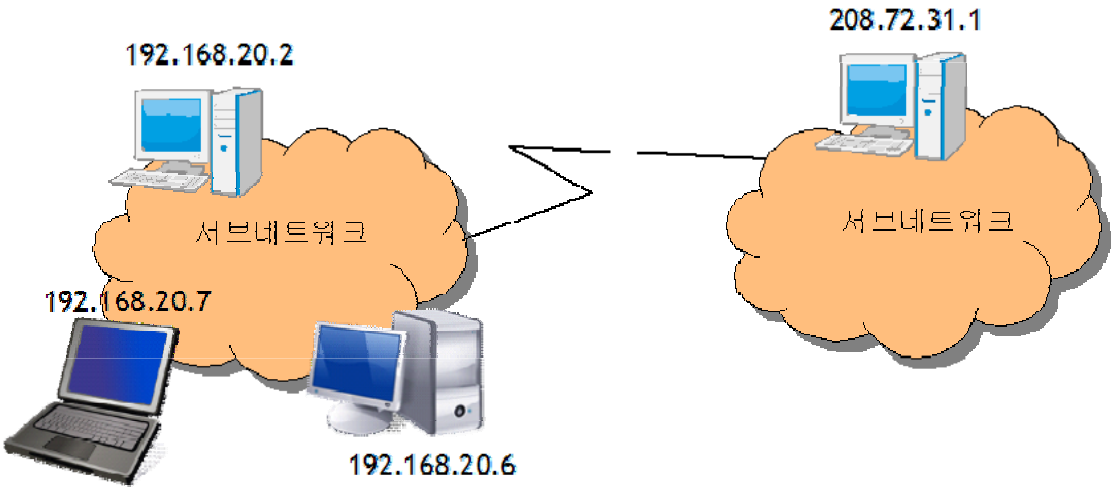
서버와 클라이언트

- 서버(Server): 사용자들에게 서비스를 제공하는 컴퓨터
- 클라이언트(Client): 서버에게 서비스를 요청해서 사용하는 컴퓨터
- (예) 웹서버와 클라이언트



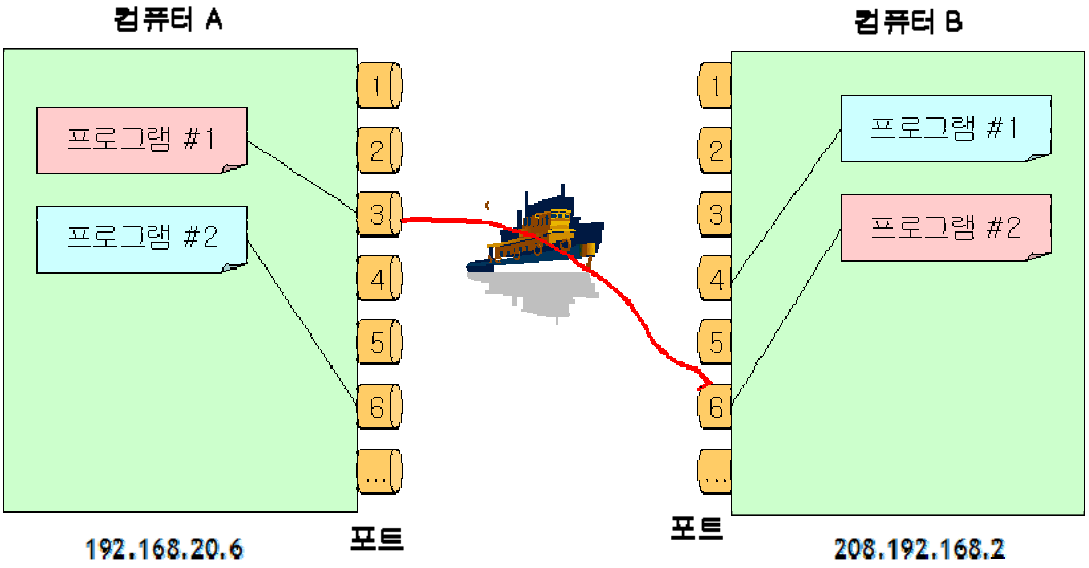
IP 주소

- IP 주소: 인터넷에서 컴퓨터의 주소



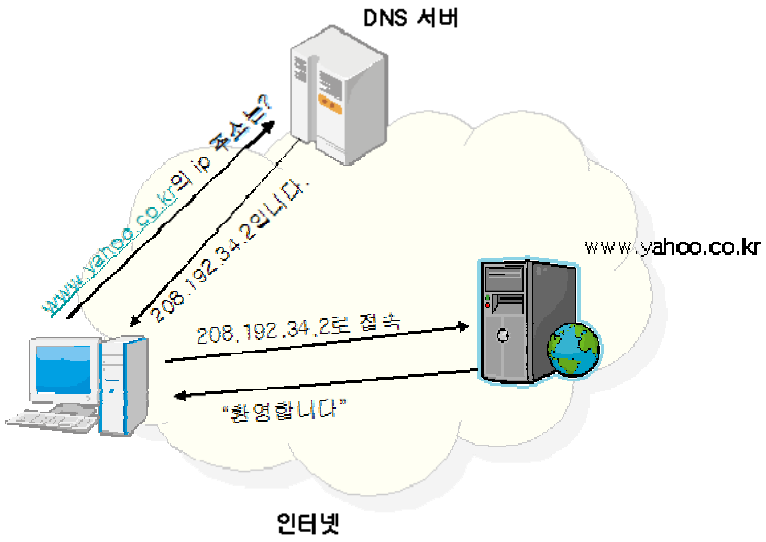
포트

- 포트(port): 가상적인 통신 선로



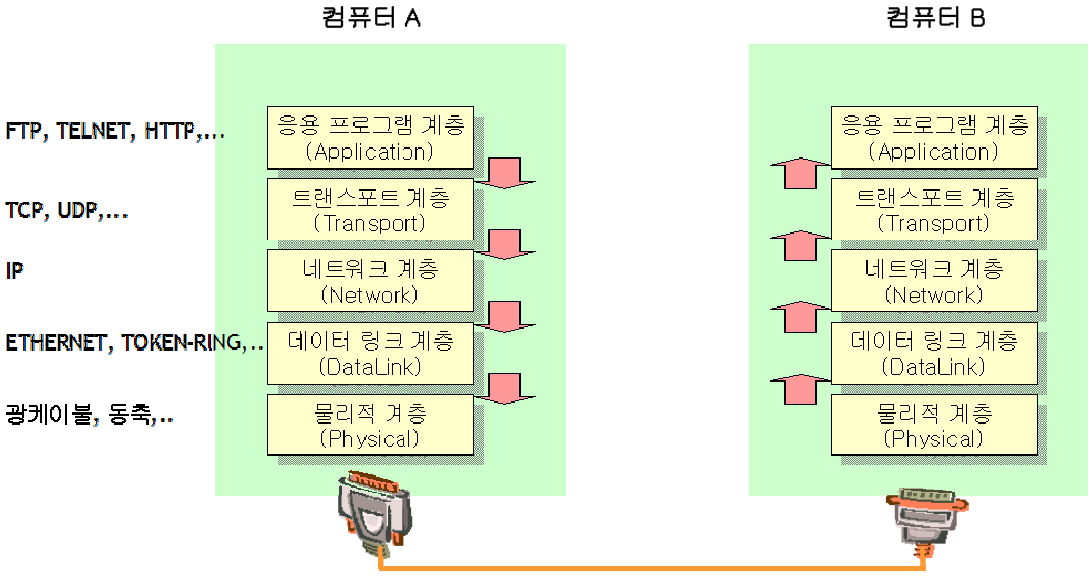
호스트 이름, DNS, URL

- 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



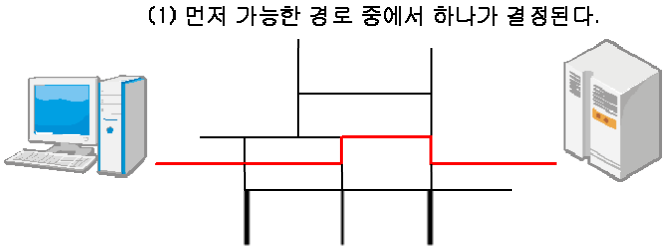
프로토콜

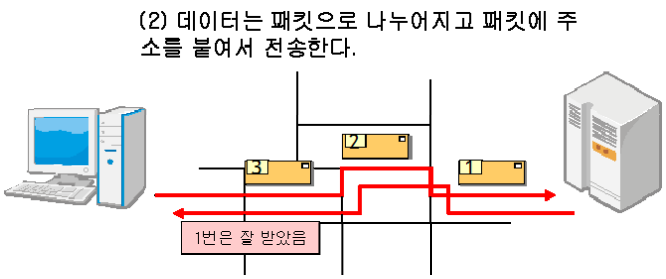
- 프로토콜(protocol): 통신을 하기 위한 약속



TCP

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

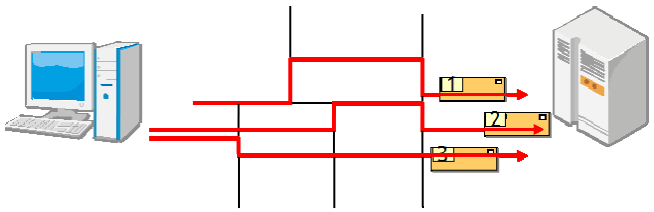




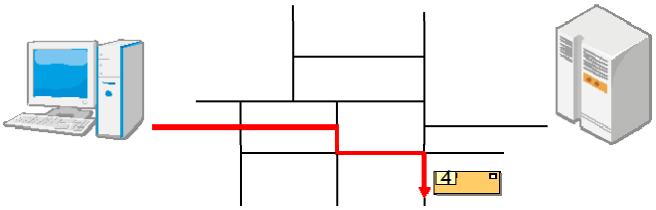
UDP

- 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) 패킷의 순서가 지켜지지 않으며 패킷이 분실될 수도 있다.



자바와 네트워크

- 네트워크 프로그래밍을 위한 패키지는 java.net
- TCP를 위한 클래스
  - URL
  - URLConnection
  - Socket
  - ServerSocket
- UDP를 위한 클래스
  - DatagramPacket
  - DatagramSocket
  - MulticastSocket

중간 점검 문제

1. IP 주소와 도메인 이름은 어떻게 다른가?

2. 전화와 비슷한 전송 프로토콜은 _____이고 편지와 비슷한 프로토콜은 _____이다.

3. TCP/IP에서 자신을 가리키는 주소는?

URL 클래스

- URL java = **new** URL("http://java.sun.com");// 절대 경로
- URL reference = **new** URL(java, "reference.html");// 상대 경로

예제

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

public class ParseURLExample {
    public static void main(String[] args) throws Exception {
        URL myURL = new URL("http://java.sun.com:80/docs/books/tutorial"
                            + "/index.html?name=database#TOP");
        System.out.println("protocol = " + myURL.getProtocol());
        System.out.println("authority = " + myURL.getAuthority());
        System.out.println("host = " + myURL.getHost());
        System.out.println("port = " + myURL.getPort());
        System.out.println("path = " + myURL.getPath());
        System.out.println("query = " + myURL.getQuery());
        System.out.println("filename = " + myURL.getFile());
        System.out.println("ref = " + myURL.getRef());
    }
}
```


예제

```
protocol = http
authority = java.sun.com:80
host = java.sun.com
port = 80
path = /docs/books/tutorial/index.html
query = name=database
filename = /docs/books/tutorial/index.html?name=database
ref = TOP
```

Socket 클래스

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

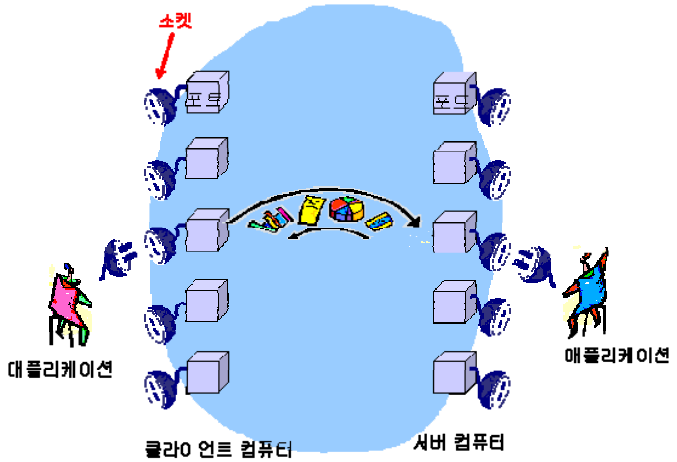


그림 14.2 소켓의 개념

ServerSocket과 Socket

소켓의 종류

- ServerSocket 클래스: 서버를 위한 소켓


서버(server) 컴퓨터

- Socket 클래스: 클라이언트를 위한 소켓


클라이언트(client) 컴퓨터

Socket 클래스

생성자	설명
Socket(String host, int port)	호스트 이름이 host이고 포트 번호가 port인 새로운 소켓을 생성한다.
Socket(InetAddress address, int port)	InetAddress에 기술된 주소로 새로운 소켓을 생성한다.

메소드	설명
InputStream getInputStream()	소켓이 사용하는 입력 스트림을 반환한다.
OutputStream getOutputStream()	소켓이 사용하는 출력 스트림을 반환한다.
Inetaddress getInetAddress()	소켓이 연결되어 있는 인터넷 주소를 반환한다.
public int getLocalPort()	소켓이 연결되어 있는 포트 번호를 반환한다.
public int getPort()	원격 컴퓨터의 포트 번호를 반환한다.
public InetAddress getLocalAddress()	소켓이 연결되어 있는 인터넷 주소를 반환한다.

ServerSocket 클래스

생성자	설명
public ServerSocket(int port) throws IOException	포트 번호 port에 대해 ServerSocket의 새로운 인스턴스를 만든다. 포트 번호 0는 비어있는 포트 번호를 사용한다는 의미이다. queue는 서버가 받을 수 있는 입력 연결의 개수를 의미한다.(디폴트는 50 연결이다.) addr는 컴퓨터의 인터넷 주소를 나타낸다.
public ServerSocket(int port, int queue)	
public ServerSocket(int port, int queue, InetAddress addr)	

메소드	설명
public Socket accept()	접속 요청을 받는다.
public void close()	ServerSocket을 닫는다.
public Inetaddress getInetAddress()	소켓이 연결되어 있는 인터넷 주소를 반환한다.
public int getSoTimeout()	소켓에 대한 타임아웃 값을 밀리 초로 반환하거나 설정한다.

소켓을 이용한 서버 제작

1. ServerSocket 객체 생성
   - ServerSocket server = **new** ServerSocket(portNumber, queueLength);
2. accept() 메소드 호출
   - Socket clientSocket = server.accept();
3. 소켓으로부터 스트림 객체를 얻는다.
   - InputStream input = clientSocket.getInputStream();
   - OutputStream output = clientSocket.getOutputStream();
4. 상호 대화 단계
   - read()와 write() 사용
5. 종료
   - close() 사용

TCP 예제: 퀴즈 서버와 클라이언트

퀴즈 클라이언트:
퀴즈에 대한 답을
보낸다.

퀴즈 서버:
퀴즈를 출제한다.


QuizServer

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

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

        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(5555);
        } catch (IOException e) {
            System.err.println("다음의 포트 번호에 연결할 수 없습니다: 5555");
            System.exit(1);
        }

        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            System.err.println("accept() 실패");
            System.exit(1);
        }
```



QuizServer

```
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                            clientSocket.getInputStream()));
        String inputLine, outputLine;
        QuizProtocol qp = new QuizProtocol();

        outputLine = qp.process(null);
        out.println(outputLine);

        while ((inputLine = in.readLine()) != null) {
            outputLine = qp.process(inputLine);
            out.println(outputLine);
            if (outputLine.equals("quit"))
              break;
        }
        out.close();
        in.close();
        clientSocket.close();
        serverSocket.close();
    }
}
```

예제

QuizProtocol 클래스

```
class QuizProtocol {
    private static final int WAITING = 0;
    private static final int PROBLEM = 1;
    private static final int ANSWER = 2;

    private static final int NUMPROBLEMS = 3;

    private int state = WAITING;
    private int currentProblem = 0;

    private String[] problems = { "네트워크 처리 패키지는?", "자바의 최신버전은?", "
      인터넷에서 컴퓨터를 식별하는 주소는?" };
    private String[] answers = { "java.io",
                                 "1.6",
                                 "IP 주소" };
```


QuizProtocol 클래스

```
public String process(String theInput) {
        String theOutput = null;
         if (state == WAITING) {
            theOutput = "퀴즈를 시작합니다(y/n)";
            state = PROBLEM;
        } else if (state == PROBLEM) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = problems[currentProblem];
                state = ANSWER;
            } else {
                state = WAITING;
                theOutput = "quit";
            }
        } else if (state == ANSWER) {
            if (theInput.equalsIgnoreCase(answers[currentProblem])) {
                theOutput = "정답입니다. 계속하시겠습니까? (y/n)";
                state = PROBLEM;
            } else {
                state = PROBLEM;
                theOutput = "오답입니다. 계속하시겠습니까? (y/n)";
            }
            currentProblem = (currentProblem+1)% NUMPROBLEMS;
        }
        return theOutput;
```


QuizIClient 클래스

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

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

            Socket quizSocket = null;
            PrintWriter out = null;
            BufferedReader in = null;

            try {
                  quizSocket = new Socket("localhost", 5555);
                  out = new PrintWriter(quizSocket.getOutputStream(), true);
                  in = new BufferedReader(new InputStreamReader(quizSocket
                              .getInputStream()));
            } catch (UnknownHostException e) {
                  System.err.println("localhost에  접근할 수 없습니다.");
                  System.exit(1);
            } catch (IOException e) {
                  System.err.println("입출력 오류");
                  System.exit(1);
            }
```

QuizIClient 클래스

```
        BufferedReader user = new BufferedReader(new InputStreamReader(
                System.in));
        String fromServer;
        String fromUser;

        while ((fromServer = in.readLine()) != null) {
            System.out.println("서버: " + fromServer);
            if (fromServer.equals("quit"))
                break;

            fromUser = user.readLine();
            if (fromUser != null) {
                System.out.println("클라이언트: " + fromUser);
                out.println(fromUser);
            }
        }

        out.close();
        in.close();
        quizSocket.close();
    }
}
```

서버와 클라이언트의 실행

- 두개의 프로그램을 동시에 실행하여야 한다.

```
C> java QuizServer Enter
...
```

```
C> java QuizClient Enter
...
```

```
서버: 퀴즈를 시작합니다(y/n)
y
클라이언트: y
서버: 네트워크 처리 패키지는?
java.io
클라이언트: java.io
서버: 정답입니다. 계속하시겠습니까? (y/n)
n
클라이언트: n
서버: quit
```

다중 클라이언트를 지원하려면

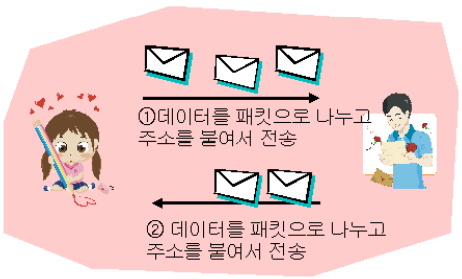
- 각각의 클라이언트를 별도의 스레드로 처리하여야 한다

```
while(true){
    연결 요청을 수락한다;
    클라이언트를 대응하는 쓰레드를 만든다;
}
```


UDP를 이용한 서버와 클라이언트

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

TCP 프로토콜



UDP 프로토콜


Sender 클래스

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

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 호스
	트
		DatagramPacket packet = new DatagramPacket(buf, buf.length, address,
				5000);
		socket.send(packet);
		socket.close();
	}
}
```



Receiver 클래스

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

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.
```


Q & A