



JAVA

네트워크 프로그래밍



네트워크 프로그래밍

1. URL 클래스

2. TCP를 이용한 통신

3. TCP→서버 제작

4. TCP→클라이언트 제작

5. UDP를 이용한 통신



서버와 클라이언트

네트워크란?

데이터 교환을 목적으로 로컬 컴퓨터와 원격 컴퓨터 간에 데이터의 흐름을 나타내는 구조라고 정의할 수 있다.

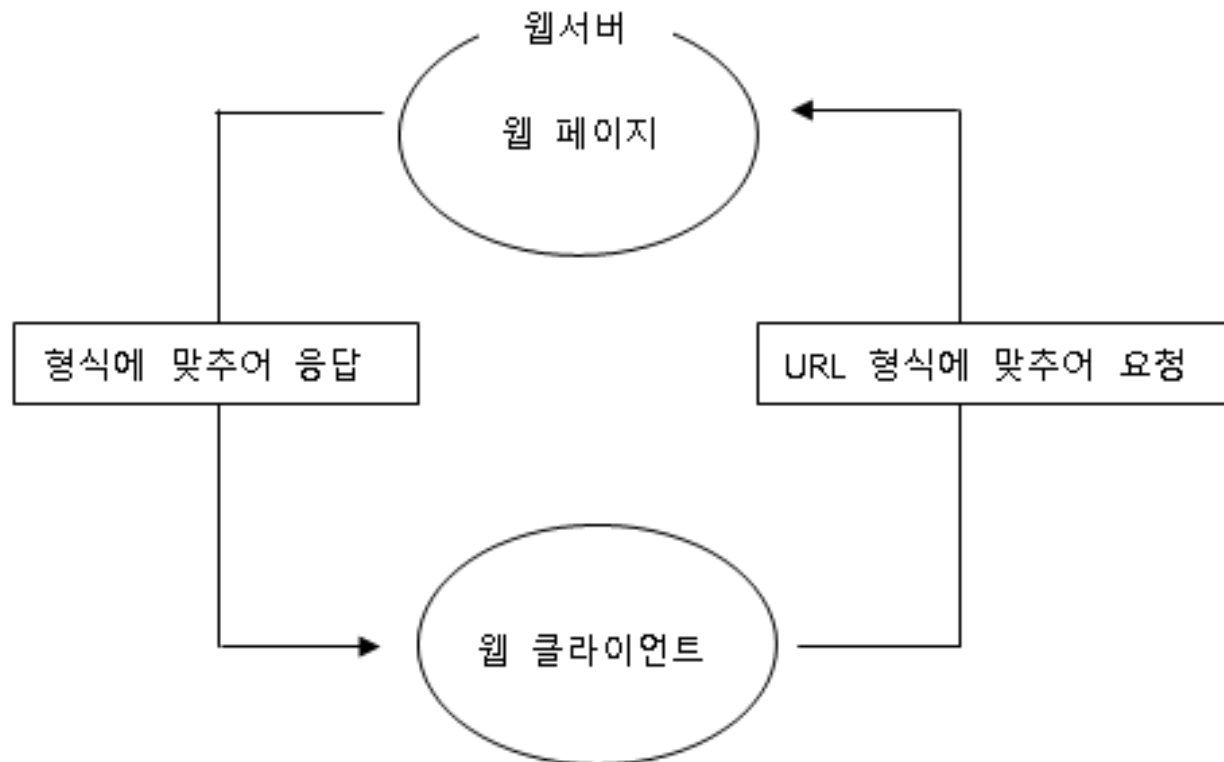
(여러 대의 컴퓨터를 통신 회선으로 연결한 것)

네트워크에서 자원이나 데이터의 공유하는 방법에 따라 다양한 네트워크 모델이 있다. 가장 흔한 방식이 서버(Server)와 클라이언트(Client) 모델이다

- 서버(Server)
 - : 사용자들에게 서비스를 제공(응답)하는 컴퓨터
- 클라이언트(Client)
 - : 서버에게 서비스를 요청해서 사용하는 컴퓨터

서버와 클라이언트

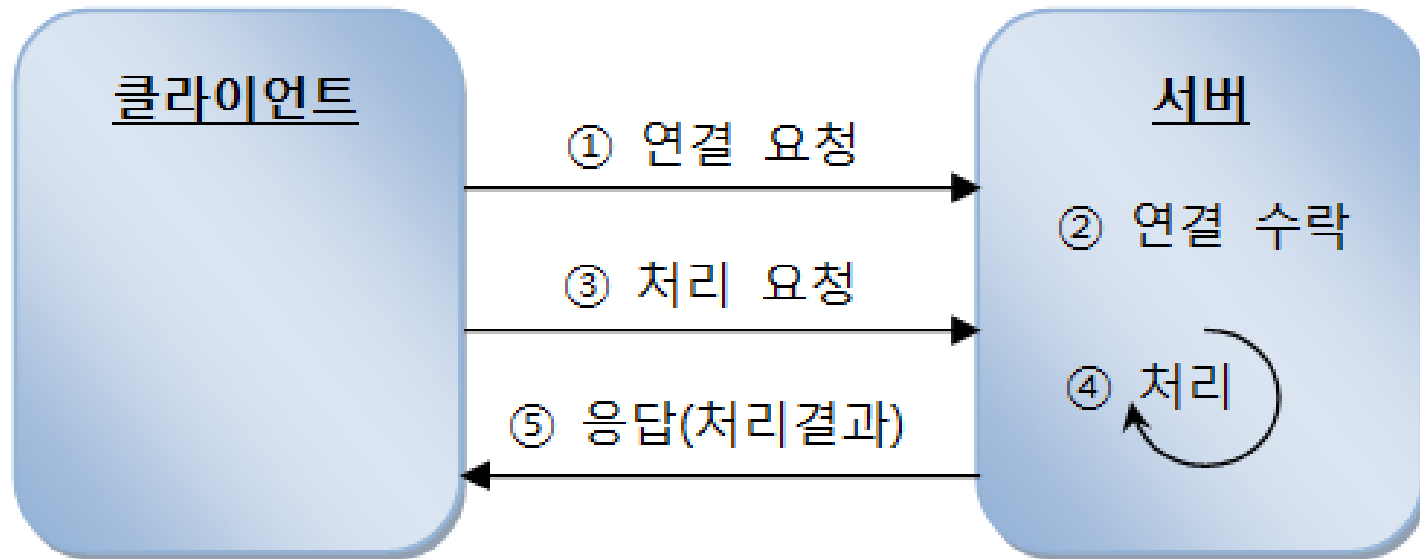
웹 서비스는 대표적인 서버-클라이언트이다. 웹 클라이언트는 자신이 원하는 페이지의 경로(URL)를 사용해서 웹 서비스를 요청한다. 그러면 웹 서버는 웹페이지라는 단위(HTML, JSP)의 서비스를 웹 클라이언트에게 제공한다. 이렇게 웹 서비스가 실시간으로 이루어진다.



서버와 클라이언트

❖ 서버와 클라이언트

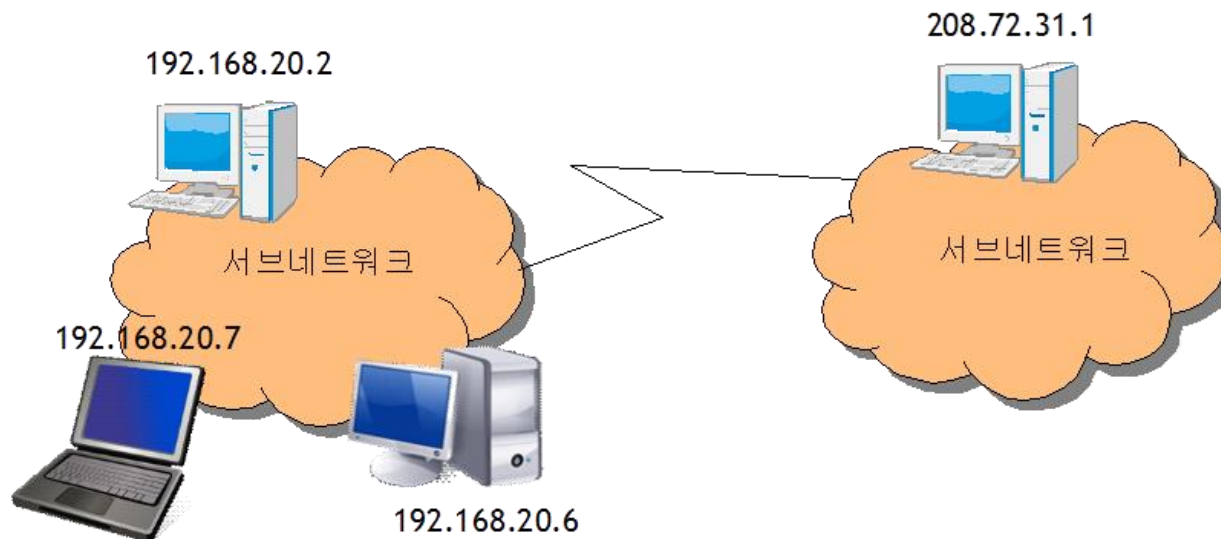
- 서버: 서비스를 제공하는 프로그램
 - 웹 서버, FTP서버, DBMS, 메신저 서버
 - 클라이언트의 연결을 수락하고, 요청 내용 처리한 후 응답 보내는 역할
- 클라이언트: 서비스를 받는 프로그램
 - 웹 브라우저, FTP 클라이언트, 메신저
 - 네트워크 데이터를 필요로 하는 모든 애플리케이션이 해당(모바일 앱 포함)



IP(Internet Protocol) 주소

IP 주소: 하나의 컴퓨터가 다른 컴퓨터와 통신을 하려면 그 컴퓨터의 주소를 알아야 한다. IP 주소(IP Address)는 네트워크에 존재하는 컴퓨터를 유일하게 식별하는 주소(숫자)이다.

- 네트워크상에서 컴퓨터를 식별하는 번호
- 네트워크 어댑터(랜 (Lan) 카드) 마다 할당





IP(Internet Protocol) 주소

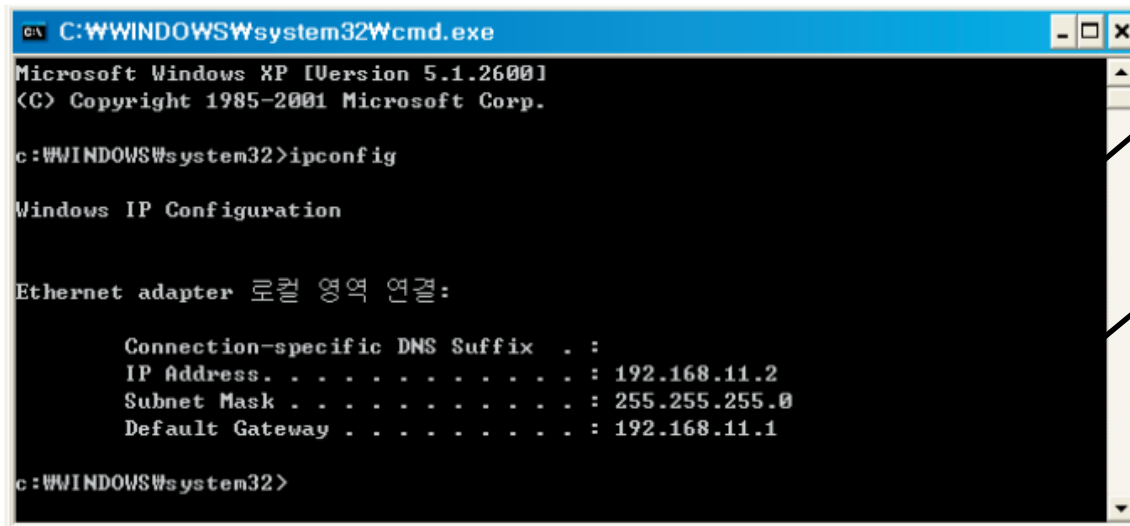
IP 주소란 통신할 때, 송신자와 수신자를 구별하는 고유의 주소를 말한다.

인터넷에 연결된 모든 통신망과 이 통신망에 연결된 컴퓨터에 부여되는 고유의 식별 주소를 의미한다. IP 주소는 IPv4와 IPv6 두가지가 있으면 일반적으로 IP 주소라 하면 IPv4 주소를 말한다.

IPv4 주소는 내부에 32비트(4바이트)로 기억되지만 표시할 때는 4개의 10진수를 점(.)으로 구분하여 표시한다. IP 주소는 명령 프롬프트에서 ipconfig 명령을 입력하면 다음과 같이 확인할 수 있다. ipconfig 명령은 네트워크 인터페이스의 설정 정보를 알아보거나 IP주소나 서브넷 마스크 등의 설정을 변경할 때 사용한다.

IP 주소

- IP 주소는 32비트를 8 비트씩 끊어서 표현하고, 각 자리는 1바이트로 0~255 까지의 범위를 갖게 된다.
- 32비트의 주소 체계를 IP 버전 4(IPv4) 주소라고 한다.
- IPv4는 포화 상태이고, 이를 극복하고자 나온 것이 IP 버전 6(IPv6)이다.
- IPv6는 128 비트의 주소 체계를 관리하고 있으며, 16비트씩 8부분으로 나누어 16진수 표시한다.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\WINDOWS\system32>ipconfig

Windows IP Configuration

Ethernet adapter 로컬 영역 연결:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 192.168.11.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.11.1

c:\WINDOWS\system32>
```

ipconfig

Ipconfig /all



도메인 네임

호스트이름은 네트워크상에 존재하는 컴퓨터 이름이다.

IP 주소는 숫자로 표현되어 외우기 어렵고 혼동되기 쉽다. 그래서 **문자로 표현한 것이 도메인 네임(domain name)**이다. 그리고 **도메인 네임을 IP주소로 자동으로 변환해 주는 서버(DNS, domain name server)**가 별도로 있다. 따라서 사용자는 둘 중 하나만 알아도 서버에 연결할 수 있다.

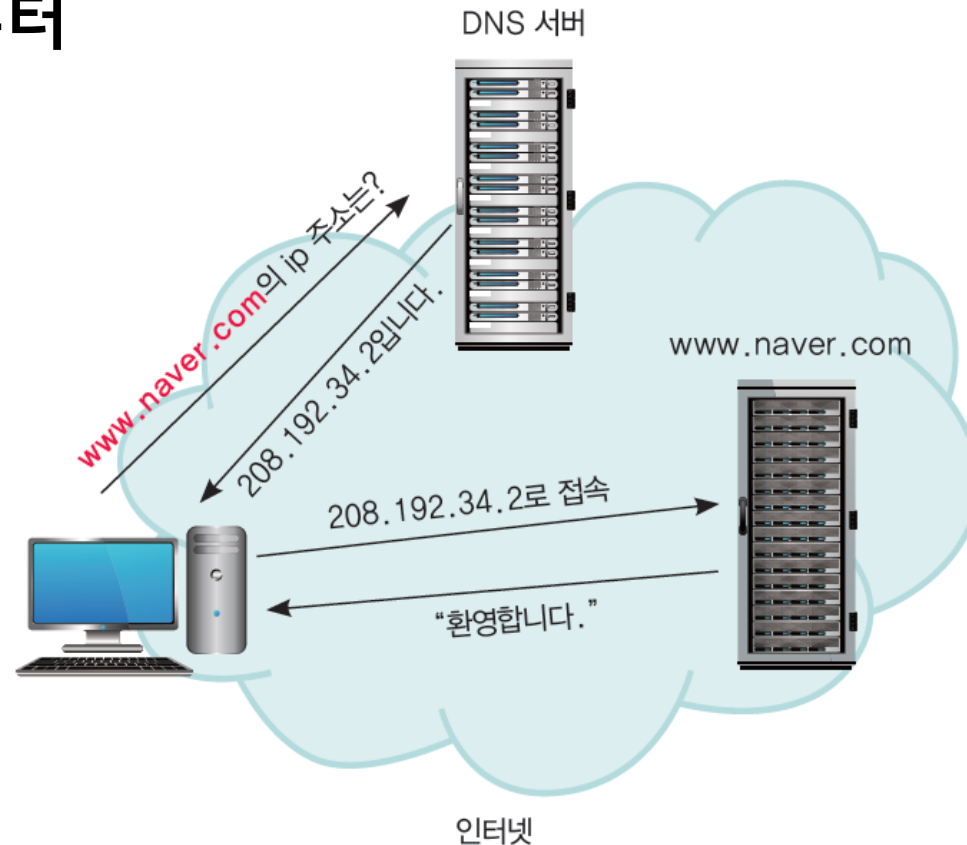
DNS가 작동하지 않을 때에도 IP 주소를 사용하여 연결할 수 있다. 네이버를 예로 들면 다음과 같다.

네이버의 서버 IP 주소: 125.209.222.142

네이버의 서버 도메인 네임: www.naver.com

호스트 이름, DNS, URL

DNS(Domain Name System) 서버: 네트워크 상에서 사람이 기억하기 쉽게 문자로 만들어진 도메인을 컴퓨터가 처리할 수 있는 숫자로 된 인터넷주소(IP)로 바꾸는 역할을 하는 서버컴퓨터





도메인 네임

자바는 IP 주소를 다루도록 InetAddress 클래스를 제공한다. 이 클래스를 생성자가 없다. 정적(static) 클래스에서 정보를 가져오는 형식으로 사용한다.

메서드명	설명
getLocalHost()	현재 컴퓨터의 InetAddress 객체를 반환한다.
getByName (String hostName)	전달받은 이름(hostName)으로 지정된 컴퓨터의 InetAddress 객체를 반환한다.
getAllByName (String hostName)	전달받은 이름(InetAddress)으로 지정된 모든 컴퓨터의 InetAddress객체를 배열로 반환한다.



도메인 네임

- InetAddress 클래스 메소드

메서드명	설명
<code>equals</code> (<code>InetAddress other</code>)	인수로 전달받은 컴퓨터(<code>other</code>)와 IP 주소가 같으면 <code>true</code> , 아니면 <code>false</code> 를 반환한다.
<code>getAddress()</code>	IP 주소를 나타내는 4개의 바이트 배열을 반환한다.
<code>getHostAddress()</code>	호스트의 IP 주소를 나타내는 문자열을 반환한다
<code>getHostName()</code>	호스트의 이름을 나타내는 문자열을 반환한다.



InetAddress 클래스

InetAddress 클래스는 Internet Protocol(IP) Address를 표현하기 위한 클래스이다.

```
InetAddress ita = InetAddress.getByName("URL값");
```



호스트 이름, DNS, URL

URL(Uniform Resource Locator)

URL(Uniform Resource Locator)은 인터넷에서 각종 서비스를 제공하면서 서버 내 파일과 같은 자원의 위치를 나타내고자 하는 표준적인 논리 주소이다. URL은 인터넷에서 자원이 어디 있는지를 알려주고자 사용하는 규약이다.

웹 브라우저는 인터넷에서 다양한 형태의 서비스를 지원한다. 즉 HTTP와 FTP, 이메일 등의 서비스를 지원한다. 이렇게 다양한 서비스를 제공하는 서버로부터 필요한 데이터를 가져오려면 이들의 위치를 표시하는 체계가 필요하다 이 체계가 URL이다. URL(Uniform Resource Locator)은 인터넷에 있는 자원의 위치를 나타내기 위한 규약이다.



호스트 이름, DNS, URL

URL(Uniform Resource Locator)은 네트워크에 연결된 서버에 서비스를 요청하는 정보로써 다음과 같은 형식을 표현한다.

URL(Uniform Resource Locator)은 서비스의 종류, 서버의 위치(도메인네임), 파일의 위치를 포함한다. 이 URL을 사용해서 서비스를 제공하는 각 서버 내 파일의 위치를 나타내게 된다. 일반적으로 URL 체계(syntax)는 다음과 같이 구성된다.

프로토콜://서버정보:포트번호/디렉토리명/파일명



호스트 이름, DNS, URL

흔히 URL(Uniform Resource Locator)은 웹 사이트 주소로 알고 있지만 컴퓨터와 네트워크의 자원을 모두 나타낼 수 있다. 해당 주소에 연결하려면 해당 URL(Uniform Resource Locator)에 맞는 프로토콜을 알아야 하고 그와 동일한 프로토콜로 연결해야 한다. FTP 프로토콜인 경우에는 FTP 클라이언트를 이용해야 하고 HTTP인 경우에는 웹 브라우저를 이용해야 하며, 텔넷인 경우에는 텔넷 프로그램을 이용해야 한다.

모든 컴퓨터는 자기 자신을 가리키는 특별한 호스트 이름과 IP 주소를 가지고 있다. localhost와 127.0.0.1이 그것이다.



URL 클래스

URL 클래스는 위치 정보를 관리하기 위한 클래스이다.

URL은 두 부분으로 되어있는데 첫 번째 부분은 자원에 접근할 때 사용하는 프로토콜(protocol)을, 두 번째 부분은 자원의 이름을 나타낸다.

<http://www.hanbit.co.kr:80/index.html>

프로토콜 식별자

호스트 이름

포트

파일명



URL 클래스

자바에서는 URL 객체를 이용하여 URL의 주소를 나타낸다.

```
URL hanbit = new URL("http://www.hanbit.co.kr/");
```

절대경로를 사용하여 URL을 표현한 것이고, 현재의 디렉토리에 상대적으로 URL을 지정할 수 있는데 이때도 먼저 기준이 되는 URL 객체를 만들어야 한다.

```
URL hanbit = new URL("http://www.hanbit.co.kr/");  
URL book = new URL(hanbit, "book/bookmain.html");
```



URL 클래스

- URL 클래스의 생성자

생성자명	설명
URL(String urlString)	전달받은 문자열로부터 URL 객체를 생성한다.
URL(String protocol, String host, String file)	전달받은 프로토콜과 호스트, 파일명으로부터 URL 객체를 생성한다.
URL(String protocol, String host, int port, String file)	전달받은 프로토콜과 호스트, 포트번호, 파일명으로부터 URL 객체를 생성한다.



URL 클래스

- URL 클래스의 메소드

메소드	설명
getProtocol()	URL 주소의 프로토콜을 반환한다.
getHost()	URL 주소의 호스트 이름을 반환한다
getPort()	URL 주소의 포트 번호를 반환한다
getPath()	URL 주소의 경로 요소를 반환한다.
getQuery()	URL 주소의 쿼리 요소를 반환한다.
getFile()	URL 주소의 파일 이름을 반환한다.
getRef()	URL 주소의 참조 요소를 반환한다.



URLConnection 클래스

URL 클래스는 URL을 이용하여 연결 및 입력 스트림을 형성할 수 있지만, URLConnection 클래스는 URL을 이용하여 참조된 자원에 대해 읽고 쓰는 작업 즉 입력과 출력 스트림을 형성할 수 있다.

이때 URL 인스턴스 생성 후 **openStream()**으로 입력 스트림 형성한다.

```
URL url = new URL("http://www.hanbit.co.kr/");  
InputStream is = url.openStream();
```



URLConnection 클래스

URLConnection 클래스는 실제로 추상클래스로서 URLConnection 타입을 갖는 인스턴스는 java.net.URL 클래스의 openConnection() 메소드를 통해서 구할 수 있다. URL 클래스로부터 URLConnection 인스턴스를 구하게 되면 URLConnection.getInputStream() 메소드를 사용하여 원격 자원으로부터 데이터를 읽어올 수 있게 된다.

```
URL url = new URL("http://www.naver.com");  
URLConnection con = url.openConnection();  
InputStream is = con.getInputStream();
```



URLEncoder 클래스와 URLDecoder 클래스

이 두 클래스는 운영체제마다 일부 문자를 인식하는 방법이 다르기 때문에 이를 해결하기 위해서 사용한다.

URLEncoder 클래스는 데이터를 웹서버에서 요구하는 자료 형으로 변환하는 역할을 한다. 웹서버가 요구하는 형식이란 'x-www-form-urlencoded'라고 불리는 MIME 형식을 말한다. MIME 형식에 대한 변환 규칙은 다음과 같다.

- 아스키문자(0-9, a-z, A-Z), '.', '-', '_' 등은 그대로 전달된다.
- 공백은 '+'로 전달된다.
- 기타 문자는 %XX와 같이 전달된다. 이때 %XX는 아스키보드를 16진수로 나타낸 것이다.

URLDecoder 클래스는 URLEncoder 클래스와 반대되는 역할을 한다. 'x-www-form-urlencoded' 형식의 MIME 형식을 일반 문자열로 변환하는 역할이다.



프로토콜

프로토콜(protocol)이란 약속 또는 규약을 의미한다.

네트워크상의 두 종점(end-to-end)간에 데이터 통신을 정확하고 효율적으로 수행하기 위한 약속의 집합이라고 하겠다. 이러한 약속에는 무엇을, 어떻게, 언제 통신할 것인지가 정의되어 있다.

다시 말해 클라이언트와 서버간의 통신 규약이다. 통신 규약이란 상호 간의 접속이나 통신방식, 주고받을 데이터의 형식, 오류검출 방식, 코드변환방식, 전송속도 등에 대하여 정의하는 것을 말한다.

같은 프로토콜을 사용하면 컴퓨터의 기종이 달라도 컴퓨터 상호간에 통신할 수 있고 데이터의 의미를 일치시켜 원하는 동작을 시킬 수 있게 된다.



프로토콜

데이터 통신은 다른 대상 간에 프로토콜에 맞추어 이루어져야 한다. 여기서 대상이란 하드웨어뿐 아니라 소프트웨어도 포함된다. 둘 모두에 대해 규약이 정의되어야 하며 규약에 맞게 데이터를 전송하고 처리해야 한다.

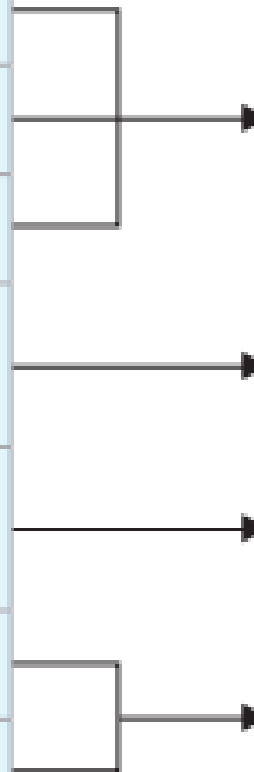
ISO(국제표준화기구)에서는 다른 네트워크 간의 상호 호환을 위해 필요한 표준 아키텍처를 정리해 개방형 연결에 적합하도록 참조 모델을 개발하였다. 이 모델을 OSI 7계층(Layer)이라 한다. OSI 7계층(Layer)은 하드웨어부터 소프트웨어까지 지켜야할 규약을 7단계 계층화해서 정의하고 있다.

프로토콜

OSI 7계층



TCP/IP





OSI 7계층(Layer)

계 층	기 능
응용 계층	응용 프로세스와의 직접 관련하여 일반적인 응용 서비스를 수행한다. 응용 프로세스 간의 정보 교환, 전자 메일, 파일 전송 등의 서비스를 제공한다 프로토콜: DNS, FTP, HTTP
표현 계층	응용 계층으로부터 받은 데이터를 하위 계층인 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 역할을 수행한다. 코드 변환, 구문 검색, 데이터 압축 및 암호화 등의 기능을 수행한다.
세션 계층	양 끝단의 응용 프로세스가 통신을 관리하는 방법을 제공한다. 통신 세션을 구성하며 포트번호를 기반으로 연결한다. 프로토콜: SSH, TLS



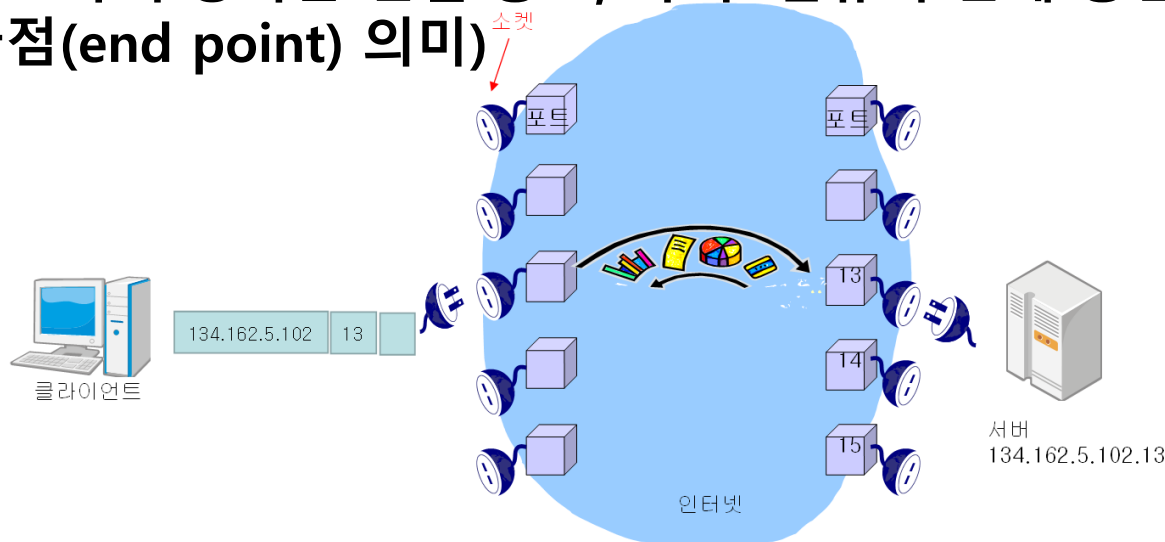
OSI 7계층(Layer)

전송 계층	패킷의 전송이 유효한지 확인하고 전송에 실패된 패킷을 다시 보내는 것과 같은 신뢰성 있는 통신을 보장한다. 주소 설정, 오류 및 흐름 제어, 다중화를 수행한다. 프로토콜: TCP, UDP
네트워크 계층	다중 네트워크 링크에서 발신지로부터 목적지까지 전달할 책임을 가진다.
데이터 링크 계층	두개의 개방 시스템 간의 효율적이고 신뢰성 있는 정보 전송을 위한 오류의 검출과 회복을 위한 오류 제어 기능을 수행한다.
물리 계층	물리적 매체를 통해서 데이터 비트를 전송하기 위해 요구되는 기능들을 정의하며 케이블, 연결장치 등 전송에 필요한 두 장치간의 실제 접속과 같은 기계적이고 전기적인 특성 등의 물리적 특성에 대한 규칙을 정의한다.

소켓

우리가 통신할 때에 전송할 패킷(데이터)이 컴퓨터에서 랜 케이블로 나간다. 그리고 그 랜 케이블에서 라우터, DNS 순으로 패킷이 이동한다. 그 이동된 패킷들은 다시 라우터를 타고 목표로 하는 컴퓨터로 전송이 돼서 프로그램을 찾아 통신이 이루어진다. 그런데 우리는 이런 일련의 전송 형태를 다 설정을 하지 않는다.

이 통신 형태들은 OS 단계에서 설정되고(OSI 7계층), 프로그램을 작성할 때는 소켓을 통해서 통신이 된다.(서버와 클라이언트 간에 통신할 수 있도록 추상화한 연결 통로, 즉 두 컴퓨터 간에 통신을 하기 위한 연결 끝점(end point) 의미)





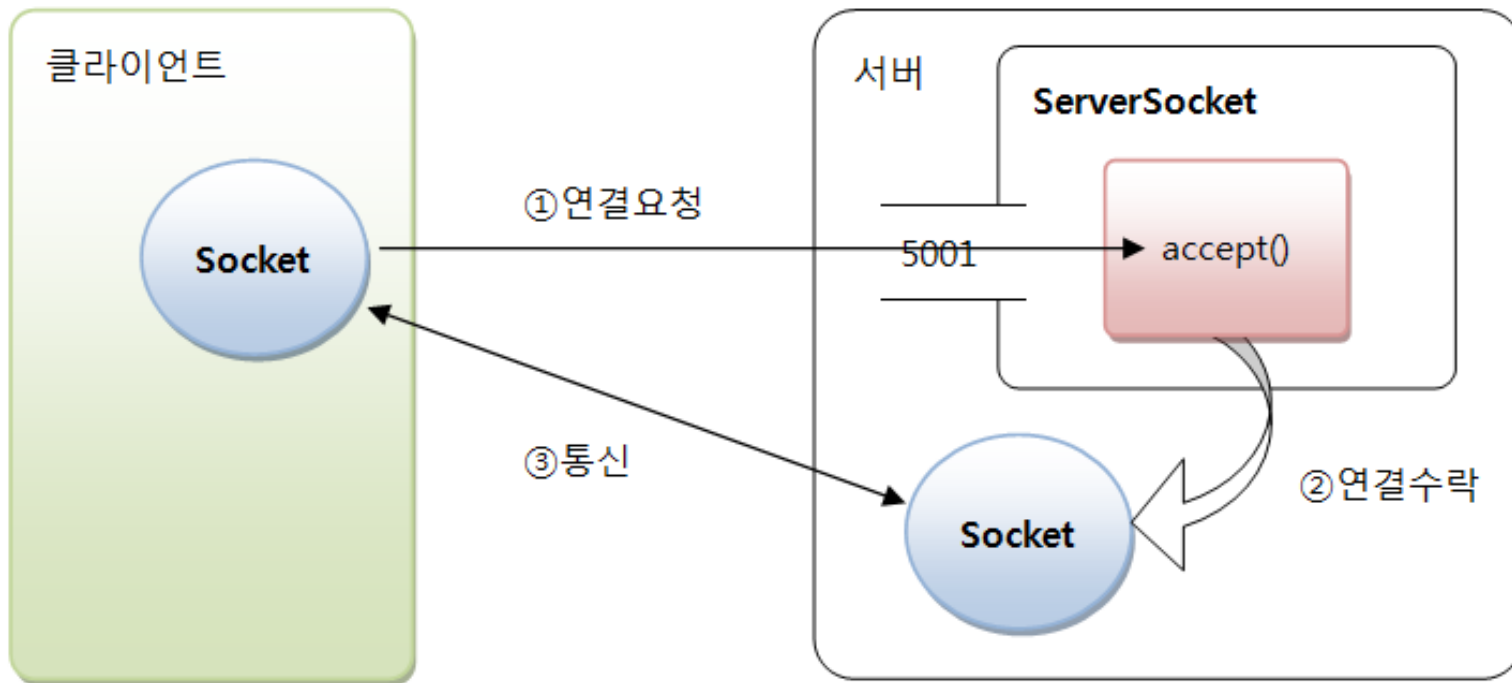
TCP와 UDP

데이터를 주고 받을 때 이 데이터는 일정 크기로 쪼개져서 '패킷(packet)'이라는 약속된 형태의 틀에 담겨져서 각각 이동한다. 패킷에는 데이터를 보낼 목적지, 데이터 내용 등의 정보가 담겨 있다. 패킷을 주고받는 형식에 따라 TCP, UDP 이렇게 크게 두 종류로 나누어진다.

TCP와 UDP의 차이점은 송수신하는 데이터의 안정성 보장 여부이다. TCP 통신은 송수신하는 데이터의 손실이 없도록 안전을 보장하는 방식이고, UDP는 데이터의 안정성을 보장하지 않아 데이터 손실이 발생할 수도 있다. 따라서 TCP는 안전성을 보장하기 때문에 전송 속도가 느리다. 이에 비해 UDP는 안전성을 체크하지 않아 전송 속도가 빠르다는 장점이 있다.

TCP

TCP(Transmission Control Protocol)는 연결 지향이며, 자체 오류를 처리한다. 네트워크 전송 중 순서가 뒤바뀐 메시지를 재조합해준다. 연결지향이란 데이터를 전송하는 측과 전송받는 측에서 전용의 데이터 전송 선로를 만든다는 의미이다. 데이터의 신뢰도가 중요하다고 판단될 때 주로 사용한다. Socket 클래스와 ServerSocket 클래스가 있다.



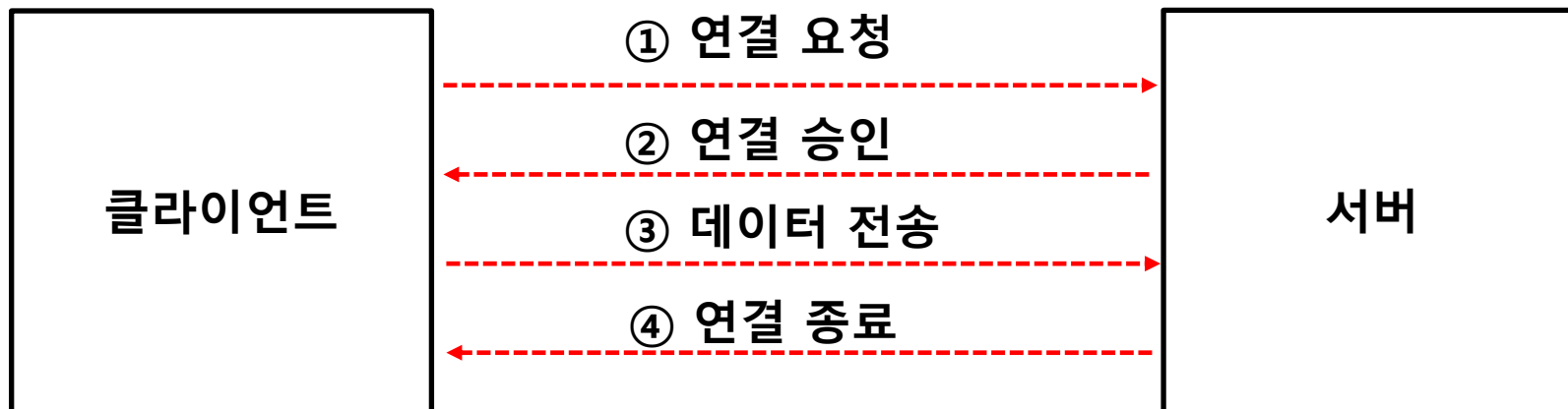
TCP

- TCP의 장점

- TCP는 신뢰성 있게 데이터를 보낼 수 있다.
- TCP는 데이터를 받는 순서가 데이터를 보내는 순서와 동일하게 관리된다.

- TCP의 단점

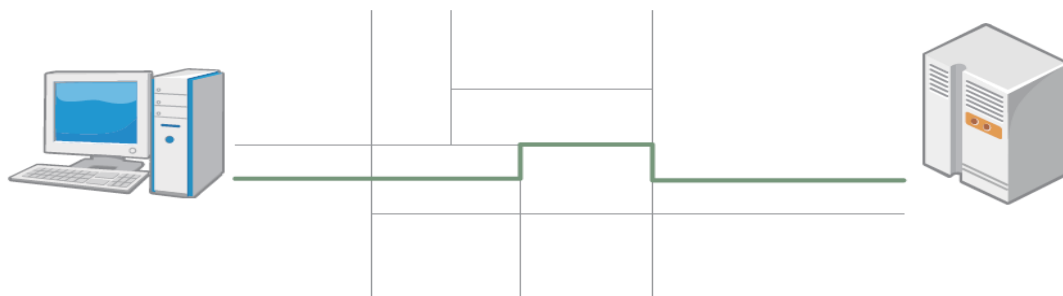
- 연결을 하는 과정과 연결을 해제하는 과정에 상당히 시간이 많이 소요된다는 점이다.



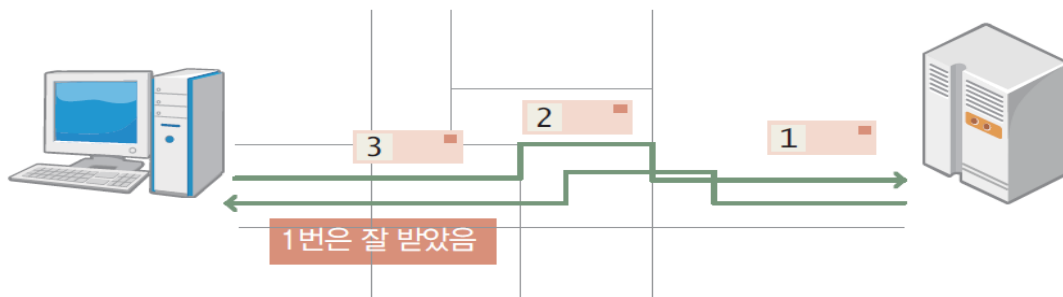
TCP

HTTP(HyperText Transfer Protocol), FTP(File Transfer Protocol) 등이 모두 TCP를 사용한다. TCP를 사용해야만 데이터의 순서가 보장되기 때문이다.

(1) 먼저 가능한 경로 중에서 하나가 결정된다.



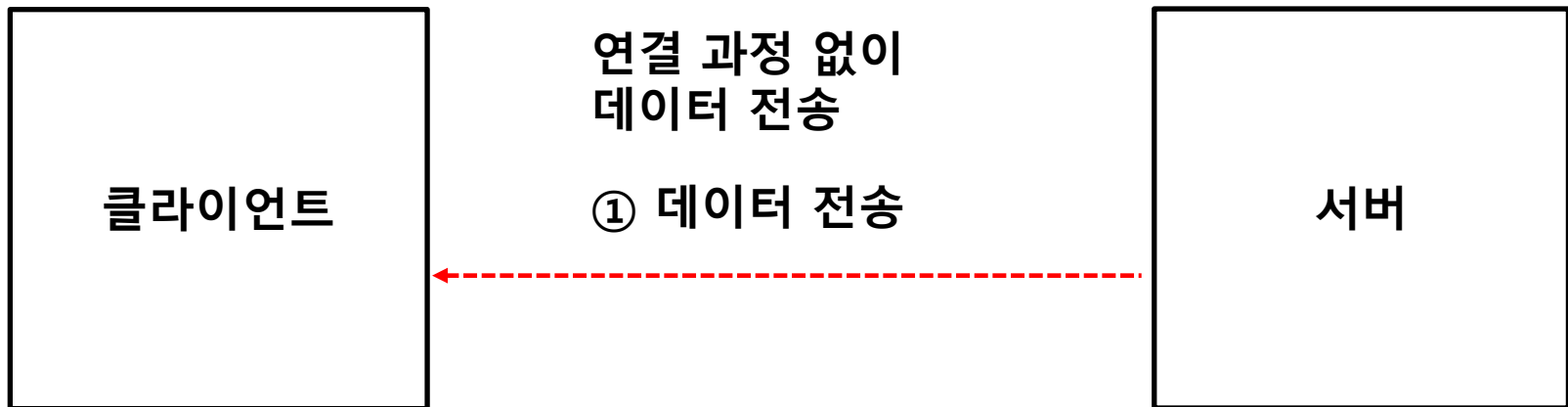
(2) 데이터는 패킷으로 나누어지고 패킷에 주소를 붙여서 전송한다.



TCP/IP는 인터넷 표준 프로토콜이다. 따라서 인터넷 기반의 응용 프로그램은 모두 TCP/IP 기반으로 통신을 한다.

UDP

UDP(User Datagram Protocol)는 비연결 지향이며, 오류를 처리하지 않는다. 메시지 순서를 재조합해 주지도 않는다. 단순히 데이터를 전송하거나 받기만 하는 프로토콜이다. UDP는 실시간으로 멀티미디어 데이터를 처리할 때 사용한다.





UDP

UDP(User Datagram Protoocol)는TCP와 달리 연결하지 않고 데이터를 몇 개의 고정 길이의 패킷(네트워크를 통해 전송하기 쉽도록 자른 데이터의 전송단위)으로 분할하여 전송하는 방식이다. UDP는 높은 신뢰도가 필요하지 않은 통신을 위하여 쓰인다.

- UDP의 장점

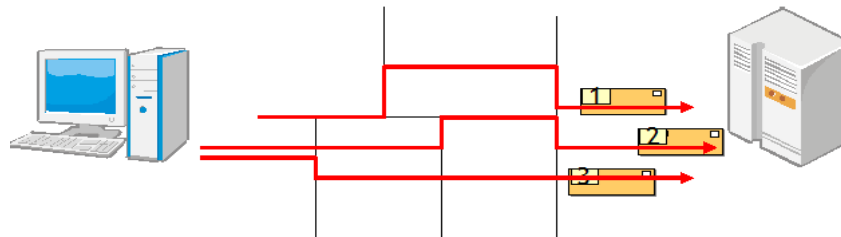
연결 절차가 없으므로 빠르고 효율적인 통신이 가능하다는 것이다. UCC와 같은 인터넷 상의 동영상 서비스는 일반적으로 UDP로 서비스를 제공한다. 약간의 손실이 있어도 동영상을 보는데 지장이 없기 때문이다.

UDP

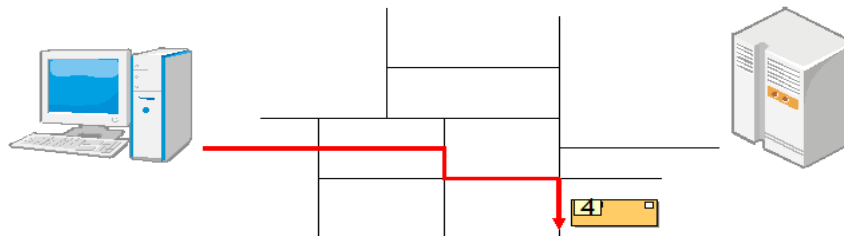
UDP(User Datagram Protocol)

- 통신 선로가 고정적이지 않음
 - 데이터 패킷들이 서로 다른 통신 선로 통해 전달될 수 있음
 - 먼저 보낸 패킷이 느린 선로 통해 전송될 경우, 나중에 보낸 패킷보다 늦게 도착 가능

(1) 데이터를 패킷으로 나누어서 패킷에 주소를 붙이고 전송한다.



(2) 패킷의 순서가 지켜지지 않으며 패킷이 분실될 수도 있다.





포트번호

포트(port) 번호는 인터넷이나 기타 다른 네트워크에서 메시지가 컴퓨터에 도달하였을 때, 컴퓨터에서 전달되어야 할 특정 프로세스를 지정하려는 목적으로 사용한다. 다시 말해서 **IP 주소는 해당 컴퓨터를 찾을 때 필요한 주소이며, 포트 번호는 컴퓨터에서 해당 프로세스를 찾을 때 필요한 번호이다.**

예를 들어, 컴퓨터의 IP 주소가 192.168.10.20이라고 하면 컴퓨터에서 FTP로 데이터를 받을 때 사용하는 IP 주소는 192.168.10.20이고, 채팅을 할 때 사용하는 IP 주소도 192.168.10.20이다. 그런데 FTP 서버와 채팅 서버가 각각 동일 IP를 사용함으로써 서로 혼동이 생기게 된다. IP 주소를 사용하는 프로세스가 하나가 아닌 것이다.



포트번호

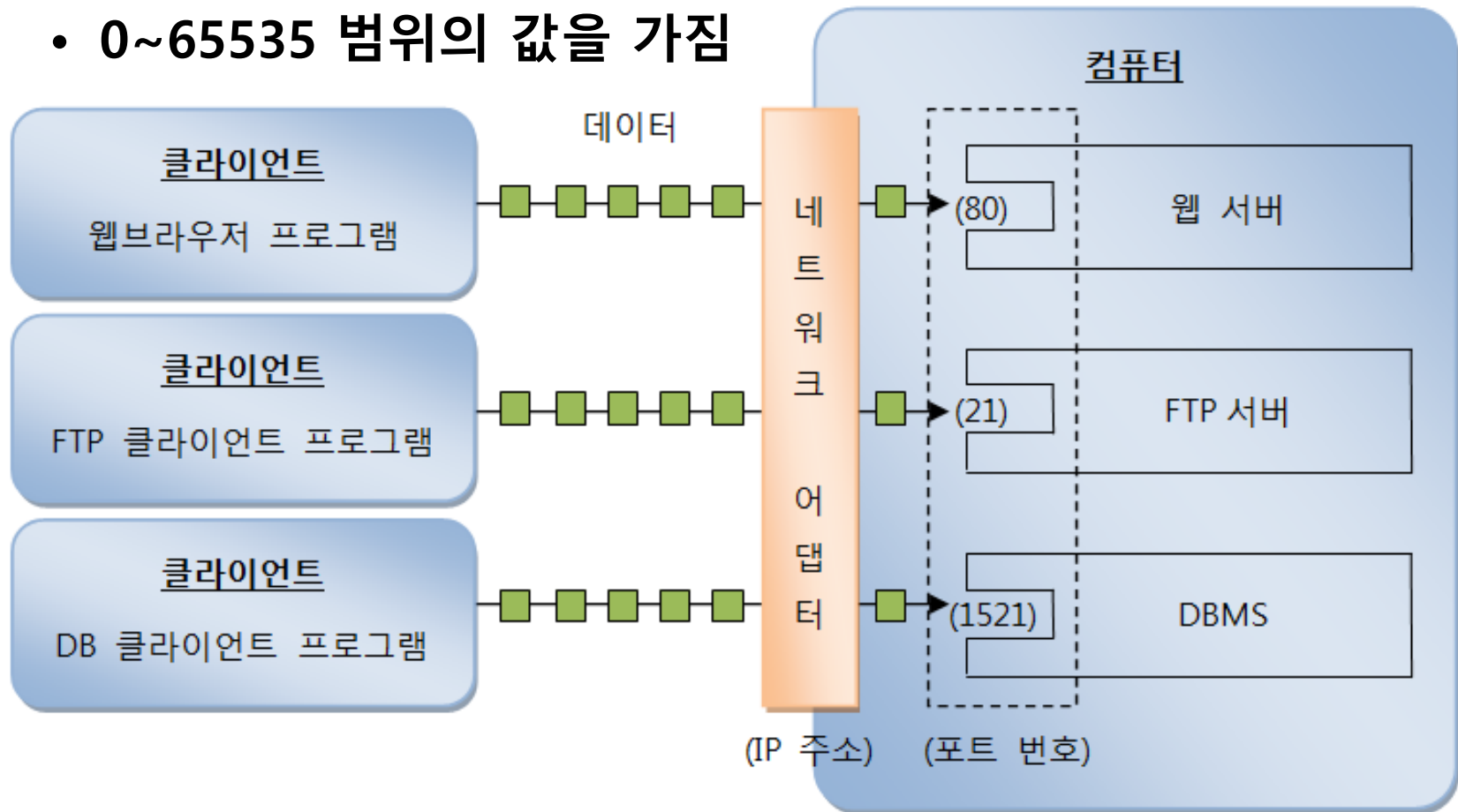
그래서 포트 번호가 도입되었다. FTP 서버가 포트 9000번을 사용하면 채팅 서버는 포트 9001번을 사용함으로써 프로세스를 구분한다. **컴퓨터의 주소가 IP 주소이며, 프로세스의 주소가 포트번호이다.**

응용 프로그램은 인터넷 번호 할당 허가 위원회(LANA)에 의해 미리 지정된 포트 번호를 가지고 있다. 이런 포트 번호는 '잘 알려진 포트'라고 불린다. 다른 응용 프로그램의 프로세스는 연결할 때마다 포트 번호가 새로 부여된다. 포트 번호는 0부터 65,535까지이며, 이 중 0부터 1023까지는 특정 서비스가 사용하도록 예약되어 있다. HTTP 서비스에는 80 포트가 지정된다.

포트

포트 : 컴퓨터 사이에 데이터를 주고받을 수 있는 통로

- 포트는 같은 컴퓨터 내에서 프로그램을 식별하는 번호.
- 클라이언트는 서버 연결 요청 시 IP 주소와 port 같이 제공
- 0~65535 범위의 값을 가짐





포트

네트워크를 통하여 전달되는 모든 데이터의 주소는 컴퓨터를 가리키는 **32비트의 IP 주소**와 **16비트의 포트번호**로 구성된다.

프로토콜	설명	포트 번호
HTTP (hyper text transfer protocol)	웹 서비스	80
SMTP (simple mail transfer protocol)	메일 서버 간 메일을 송수신하는 프로토콜	25
FTP (file transfer protocol)	컴퓨터 간 파일을 주고받을 프로토콜	21

TCP를 이용한 통신

TCP를 이용한 서버와 클라이언트 간의 통신은 1:1 소켓통신이다. 즉 자바에서는 클라이언트와 서버는 `java.net.Socket` 클래스를 사용해 상호 데이터를 전송하게 된다. 서버가 먼저 실행되어서 클라이언트의 연결 요청을 기다리고 있어야 한다. 한 포트에 하나의 서버 소켓만 연결할 수 있다.

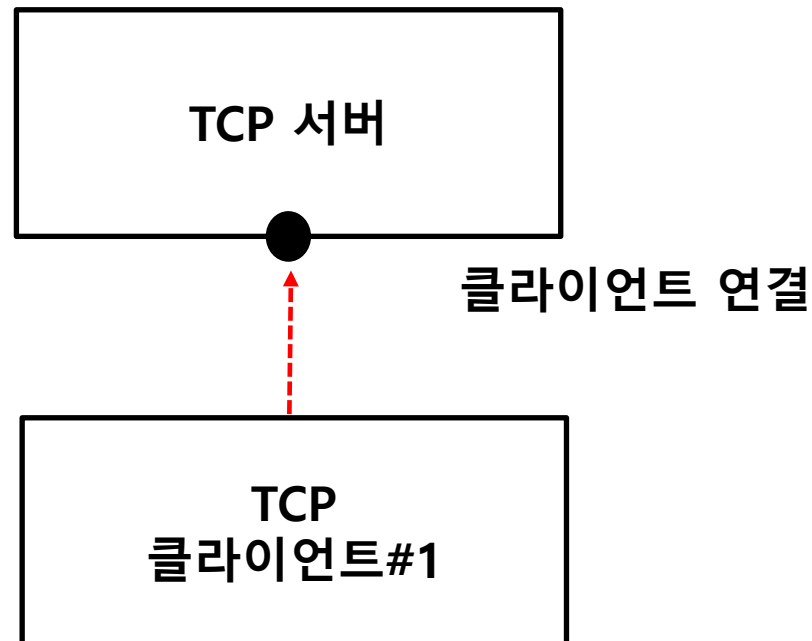
다음은 TCP 통신 절차를 설명한 것이다.

① 서버는 서버 소켓을 사용해서 서버 컴퓨터의 특정 포트에서 클라이언트의 연결 요청을 처리할 준비를 한다. 즉 서버는 먼저 실행되어서 클라이언트의 연결을 기다린다.



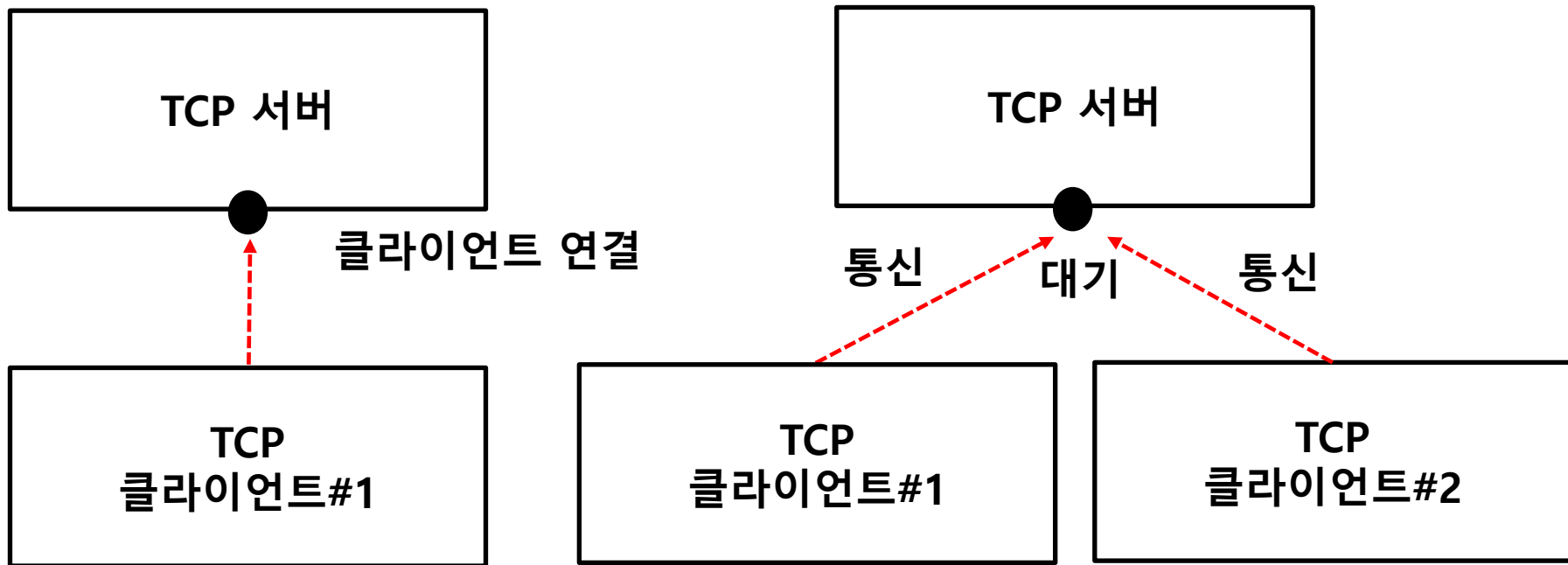
TCP를 이용한 통신

② 클라이언트는 연결할 서버의 IP와 포트번호로 소켓을 생성해서 서버에 연결을 요청한다. TCP 프로토콜이 연결 지향이라서 데이터 전송 전에 반드시 연결 과정이 있다.



TCP를 이용한 통신

③ 서버소켓은 클라이언트의 요청을 받으면 새로운 소켓을 생성해서 클라이언트의 소켓과 연결되도록 한다. 이제 클라이언트와 소켓과 새로 생성된 서버의 소켓은 원래 서버소켓과 관계없이 1:1 통신이 가능해진다.





TCP를 이용한 통신

- ④ 서버는 클라이언트가 보낸 데이터를 받아서 처리한다
- ⑤ 서버는 처리한 데이터를 클라이언트에 보낸다
- ⑥ 클라이언트는 서버가 보낸 데이터를 받아서 적절하게 처리한다

자바는 TCP 소켓 통신에서 네트워크에서의 역할에 따라 다음과 같이 다른 소켓 클래스를 제공한다.

- **ServerSocket 클래스**: 서버에서 사용하는 클래스이다. 포트와 연결되어 외부의 연결 요청을 기다린다. 연결 요청이 오면 소켓을 새로 생성해준다. 이를 통해 통신을 이루어진다.
- **Socket 클래스**: 클라이언트와 서버 모두에 사용하는 클래스이다. 프로세서 간의 통신을 담당한다. 입력 스트림과 출력 스트림을 가지고 있다가 스트림을 사용해 프로세스 간의 통신을 이루어진다

TCP를 이용한 통신



클라이언트
IP: x.x.x.x



서버
IP: y.y.y.y
ServerSocket

① 서버가 시작된다.

Local Port: 25
Local IP: y.y.y.y
Remote port: 5555
Remote IP: x.x.x.x

④ 소켓이 생성된다.

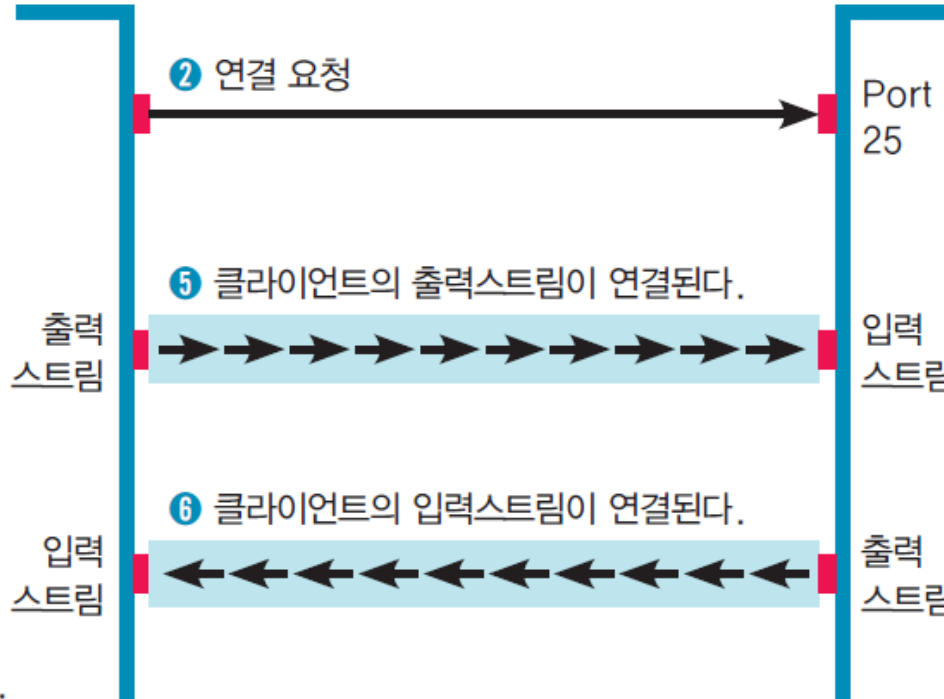
② 연결 요청

⑤ 클라이언트의 출력스트림이 연결된다.

⑥ 클라이언트의 입력스트림이 연결된다.

Local Port: 5555
Local IP: x.x.x.x
Remote port: 25
Remote IP: y.y.y.y

③ 소켓이 생성되고
연결이 만들어진다.





ServerSocket 클래스

- ServerSocket 클래스의 생성자 및 메서드

생성자	설명
ServerSocket(int port)	포트 번호(port)에 대해 ServerSocket의 새로운 인스턴스를 만든다. 포트번호(port)는 비어있는 포트 번호를 사용한다. queue는 서버가 받을 수 있는 입력 연결의 개수를 의미한다. (디폴트는 50연결이다) addr는 컴퓨터의 인터넷 주소를 나타낸다.
ServerSocket(int port, int queue)	
ServerSocket(int port, int queue, InetAddress addr)	
메서드	설명
public Socket accept()	클라이언트의 연결 요청을 받아 Socket 객체를 생성한다.
public void close()	서버 소켓을 닫는다.
public InetAddress getInetAddress()	소켓이 연결되어 있는 인터넷 주소를 반환한다



Socket 클래스

• Socket 클래스의 생성자 및 메서드

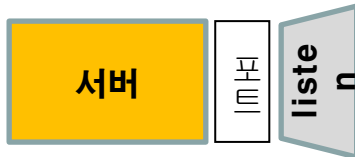
생성자	설명
Socket(String host, int port)	호스트 이름이 host이고 포트번호가 port인 새로운 소켓을 생성한다.
Socket(InetAddress address, int port)	InetAddress에 기술된 주소로 새로운 소켓을 생성한다.

메서드	설명
InputStream getInputStream()	소켓에서 입력 스트림 객체를 반환한다.
OutputStream getOutputStream()	소켓에서 출력 스트림 객체를 반환한다.
InetAddress getInetAddress()	원격 컴퓨터의 InetAddress 객체를 가져온다.
InetAddress getLocalAddress()	로컬 컴퓨터의 InetAddress 객체를 가져온다.
void close()	소켓을 닫는다.

소켓을 이용한 서버 제작

클라이언트와 서버 연결

- 서버는 서버 소켓으로 들어오는 연결 요청을 기다림



- 클라이언트가 서버에게 연결 요청



- 서버가 연결 요청 수락하고 새로운 소켓을 만들어 클라이언트와 연결 생성





소켓을 이용한 서버 제작

– 서버 소켓 생성

```
ServerSocket serverSocket = new ServerSocket(5550);
```

- 이미 사용 중인 포트 번호를 지정하면 오류가 발생

– 클라이언트로부터 접속 기다림

```
Socket socket = serverSocket.accept();
```

- accept() 메소드는 연결 요청이 오면 새로운 Socket 객체 반환
- 서버에서 클라이언트와의 데이터 통신은 새로 만들어진 Socket 객체를 통해서 이루어짐
- ServerSocket 클래스는 Socket 클래스와 달리 주어진 연결에 대해 입출력 스트림을 만들어주는 메소드가 없음

– 네트워크 입출력 스트림 생성

```
BufferedReader in = new BufferedReader(  
    new InputStreamReader(socket.getInputStream()));  
BufferedWriter out = new BufferedWriter(  
    new OutputStreamWriter(socket.getOutputStream()));
```



소켓을 이용한 서버 제작

- accept() 메소드에서 얻은 Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 데이터 스트림 생성
- 일반 스트림을 입출력하는 방식과 동일하게 네트워크 데이터 입출력

– 송신과 수신

```
out.write(data);  
in.read();
```

– 네트워크 접속 종료

```
socket.close();
```

– 서버 응용프로그램 종료

```
serverSocket.close();
```



UDP를 이용한 서버와 클라이언트

UDP는 연결 설정 없이 IP주소만으로 데이터를 전송한다.
UDP 소켓은 신뢰할 수 없는 전송방법을 제공한다. 수신상태를 확인할 수 없어서 데이터가 제대로 전달되었는지 알 수 없기 때문이다. 신뢰성보다는 빠른 통신이 주된 목적일 때 사용한다. 자바에서는 UDP 방식을 **DatagramSocket** 클래스, **DatagramPacket** 클래스로 처리한다.

DatagramSocket: UDP 통신을 위한 클래스이다. UDP 패킷을 송수신하는 기능을 제공한다. 즉 비연결 프로토콜인 UDP의 특성에 따라 대기/접속/해제/검증 등의 설정 절차 없이 **DatagramPacket**을 상대방에게 전달하거나 전달받는 기능을 제공한다. UDP는 서버 소켓과 소켓의 구분 없으며, **DatagramSocket**이 두 가지 역할을 동시에 한다.

UDP를 이용한 서버와 클라이언트

DatagramSocket 클래스

- DatagramSocket()은 UDP 프로토콜을 사용하는 소켓을 생성한다.
. TCP 프로토콜 소켓과는 다르게 서버 소켓과 클라이언트 소켓의 구분이 없다.

DatagramSocket 클래스의 생성자

생성자	설명
DatagramSocket(int port, InetAddress laddr)	지정된 주소에 연결된 데이터그램 소켓을 생성한다.





UDP를 이용한 서버와 클라이언트

DatagramPacket 클래스

- DatagramPacket은 UDP 프로토콜을 사용하여 데이터를 보내거나 전송받기 위한 클래스이다.

DatagramPacket 클래스의 생성자

생성자	설명
DatagramPacket(byte[] buf, int length)	length 크기의 버퍼를 사용하는 DatagramPacket 객체 생성한다
DatagramPacket(byte[] buf, int length, InetAddress address, int port)	length 크기의 버퍼를 사용하는 address 주소의 port 번호로 보내는 DatagramPacket 객체 생성한다



Thank You

