

2021.06.08 Java

〈Network = socket 통신〉

```
import java.net.MalformedURLException;
import java.net.URL;
// 이걸 java.net에 들어있네
// 지금까지 보통 java.util들에 많이 있었음.
public class _2nd_NetworkURL {
    public static void main(String[] args) throws MalformedURLException {
        //Malform - 악성코드
        //악성코드로 이상한 url로 링크 태워서 공격 하는 것에대한 방어조치
        // www.daum.net / http://www.daum.net
        // url은 반드시 후자로 줘야 됨. 전자의 경우에는 악성코드의 공격이 가능함.
        URL myURL = new URL( spec: "http://www.loanconsultant.or.kr/source/index.jsp?t=20191216");

        // Protocol: HTTP(웹 애플리케이션 전용 프로토콜)
        // 여기서 프로토콜만 보이면됨 / 나머지는 그냥 해본 것들.
        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());
    }
}
```

- 외부 라이브러리를 가져와
사용할 때 그 라이브러리가
network통신을 요구 할 수도 있기
때문에 network 어느정도 사용 할 줄
알아야 한다.

◀ 20210607_Java project에 있음

TCP	192.168.0.5:3836	172.217.175.106:443	ESTABLISHED
TCP	192.168.0.5:3870	216.58.220.142:443	ESTABLISHED
TCP	192.168.0.5:6038	140.82.114.25:443	ESTABLISHED
TCP	192.168.0.5:6039	211.115.106.202:80	CLOSE_WAIT
TCP	192.168.0.5:6892	113.29.139.238:443	ESTABLISHED
TCP	192.168.0.5:8236	3.219.243.226:443	ESTABLISHED
TCP	192.168.0.5:9441	185.199.109.133:443	ESTABLISHED
TCP	192.168.0.5:9648	52.78.231.108:443	ESTABLISHED
TCP	192.168.0.5:9757	211.115.106.202:80	CLOSE_WAIT
TCP	192.168.0.5:11484	223.130.195.200:443	ESTABLISHED
TCP	192.168.0.5:11647	40.115.22.134:443	ESTABLISHED
TCP	192.168.0.5:11651	117.52.156.63:80	ESTABLISHED
TCP	192.168.0.5:11652	117.52.156.63:80	ESTABLISHED
TCP	192.168.0.5:11655	15.165.110.115:443	ESTABLISHED
TCP	192.168.0.5:11656	211.115.106.202:80	ESTABLISHED

- http의 기본 포트는 80이다.
- https의 기본 포트는 443이다.

모르고 있었다면 간단한 실험으로도 알 수 있다.

- http 프로토콜
 - ☐ <http://www.google.com> - http 프로토콜. 구글이 망하지만 않는다면 잘 접속된다.
 - ☐ <http://www.google.com:80> - 80 포트를 명시했다. 잘 접속된다.
 - ☒ <http://www.google.com:81> - 81 포트를 명시했다. 목적이 다른 포트이므로 접속이 안 된다.
- https 프로토콜
 - ☐ <https://www.google.com> - https 프로토콜. 잘 접속된다.
 - ☐ <https://www.google.com:443> - 443 포트를 명시했다. 잘 접속된다.
 - ☒ <https://www.google.com:444> - 444 포트를 명시했다. 목적이 다른 포트이므로 접속이 안 된다.

이는 80과 443이 기본 포트 번호이기 때문이다. 포트 번호를 생략하면 기본 포트를 사용하게 된다.

• 결론 및 요약

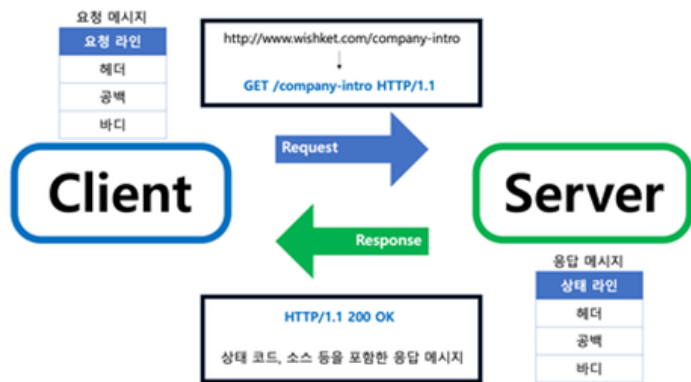
- 80
 - HTTP가 문서화되기 이전부터 보통 사용하지 않는 빈 포트 번호였다.
 - 1991년 HTTP 0.9 버전에서 처음으로 문서화되면서 기본 포트가 지정되었다.
- 443
 - RFC 1700 이전까지는 빈 포트 번호였다.
 - Kipp E.B. Hickman의 요청으로 1994년 10월에 RFC 1700 문서에 443이 추가되었다.
 - 443인 이유는 빈 칸에 순서대로 배정하다 보니 그렇게 된 것 같다.

추가개념: <http://blog.wishket.com/http-%EA%B7%B8%EB%A6%AC%EA%B3%A0-https%EC%9D%98-%EC%9D%B4%ED%95%B4/>

출처: <https://johngrib.github.io/wiki/why-http-80-https-443/>

HTTP(HYPERTEXT TRANSFER PROTOCOL)

- 웹 상에서 클라이언트와 서버가 서로 정보를 주고받을 수 있도록 하는 규약 (Protocol의 사전적 의미: 규약).



위의 내용을 정리하면 HTTP를 통해 이런 일이 이루어집니다.

1. 클라이언트가 보고 싶은 정보를 서버에게 HTTP를 통해 요청.
2. 서버는 알맞은 응답 메시지 및 정보를 클라이언트에게 전달.
3. 응답 메시지 및 정보 중 HTTP바디(이걸 설명하자니 내용이 또 길어서... '실제로 필요한 정보' 정도가 적당하겠습니다. 추후 따로 해설해드릴게요.) 내용이 클라이언트가 설정한 클라이언트의 용처에 도달한다.

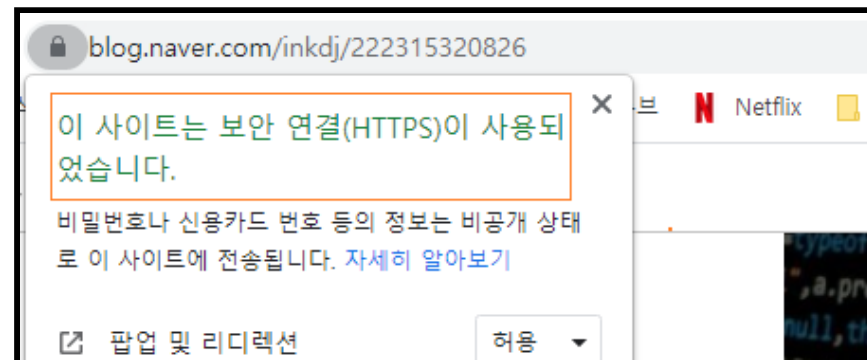


HTTP(HyperText Transfer Protocol) / 80port 사용

- 인터넷 상에서 자료전송(개인정보)시 자료가 보안처리 없이 그대로 전송. 해킹에 쉽게 노출.

HTTPS(HyperText Transfer Protocol over Secure Socket Layer) / 443 port 사용

- 인터넷 상에서 자료전송(개인정보)시 암호화되어 전송이 됨. 중간에 해킹이 되어도 안전.



port번호는 ip주소와 함께 쓰여 해당하는 프로토콜에 의해 사용된다.

인터넷에서 말하는 port는 추상적인 개념으로, 통상적으로 소프트웨어적인 입출력 인터페이스를 의미하기도 한다.

결국 port번호는, 컴퓨터 내의 프로세스를 구별/식별하는 수단이 된다.

- 쉽게 예를 들면 우편물은 집주소와 더불어 받는 사람 이름까지 적어줘야 정확히 배송된다.

이때 집주소를 ip / 받는 사람 이름을 port라고 예를 들 수 있다.

〈Server 구동〉

자세한 코드 설명은 다음 수업 때

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class _2nd_SocketServer {
    public static void main(String[] args) {
        int port = Integer.parseInt("33333");

        try{
            ServerSocket servSock = new ServerSocket(port);
            System.out.println("Server: Listening - " + port);

            while(true){
                Socket sock = servSock.accept();
                System.out.println "[" + sock.getInetAddress() + "] client connected");

                OutputStream out = sock.getOutputStream();
                PrintWriter writer = new PrintWriter(out, autoFlush: true);

                writer.println(new Date().toString());

                InputStream in = sock.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));

                System.out.println("msg: " + reader.readLine());
            }
        } catch (IOException e){
            System.out.println("Server Exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```
_2nd_SocketServer x
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\AppData\Local\JetBrains\Toolbox\app
Server: Listening - 33333

MINGW64:/c/Users/Samuel

Samuel@DESKTOP-VVE1E8K MINGW64 ~
$ netstat -na | grep 33333
TCP    0.0.0.0:33333      0.0.0.0:0        LISTENING
TCP    [::]:33333        [::]:0           LISTENING

Samuel@DESKTOP-VVE1E8K MINGW64 ~
$ |
```

<div> <div>_2nd_SocketServer</div> <div> <div>"C:\Program Files\Java\jdk-16\bin\javaw.exe"</div> <div>Server: Listening - 33333</div> <div>구동중</div> </div> </div>	Google Chrome(23)		0.1%	370.6MB	0.1MB/s
	IntelliJ IDEA(6)		1.0%	687.3MB	0MB/s
	20210608_Java - _2nd_SocketS...		1.0%	570.3MB	0MB/s
	Filesystem events processor		0%	0.1MB	0MB/s
	Java(TM) Platform SE binary		0%	15.6MB	0MB/s
	Java(TM) Platform SE binary		0%	89.8MB	0MB/s
	콘솔 창 호스트		0%	5.7MB	0MB/s
	콘솔 창 호스트		0%	5.7MB	0MB/s

<div> <div>_2nd_SocketServer</div> <div> <div>"C:\Program Files\Java\jdk-16\bin\javaw.exe"</div> <div>Server: Listening - 33333</div> <div>서버 구동중지</div> <div>Process finished with exit code -1</div> </div> </div>	IntelliJ IDEA(4)		0.2%	704.5MB
	20210608_Java - _2nd_SocketS...		0.2%	609.0MB
	Filesystem events processor		0%	0.1MB
	Java(TM) Platform SE binary		0%	89.7MB
	콘솔 창 호스트		0%	5.7MB

〈Client〉

```
import java.io.*;
import java.net.Socket;
import java.net.UnknownHostException;
public class _2nd_SocketClient {
    public static void main(String[] args) {
        // 사실망이라 해킹 걱정 ㄴㄴ
        // router ip 알아야 해킹할 수 있음
        String hostname = "192.168.1.100"; // gitbash에서 ipconfig -all
        int port = 33333;

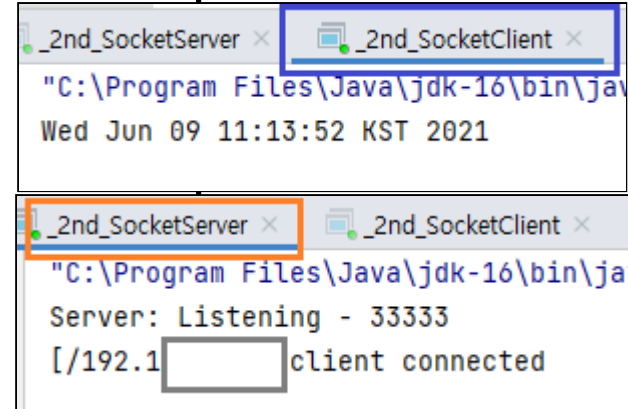
        for (int i = 0; i < 10; i++) {
            try {
                Socket sock = new Socket(hostname, port);
                OutputStream out = sock.getOutputStream();

                String str = "Hello Network Programming";

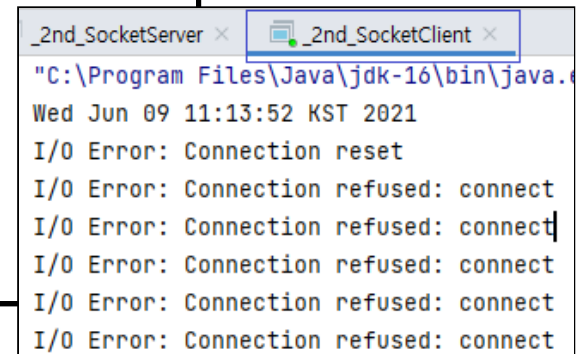
                out.write(str.getBytes());

                InputStream in = sock.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));

                String time = reader.readLine();
                System.out.println(time);
            } catch (UnknownHostException e) {
                System.out.println("Server Not Found: " + e.getMessage());
            } catch (IOException e) {
                System.out.println("I/O Error: " + e.getMessage());
            }
        }
    }
}
```



server를 구동한 상태로
Client코드 돌리면
connected 된 것을 볼 수 있음.

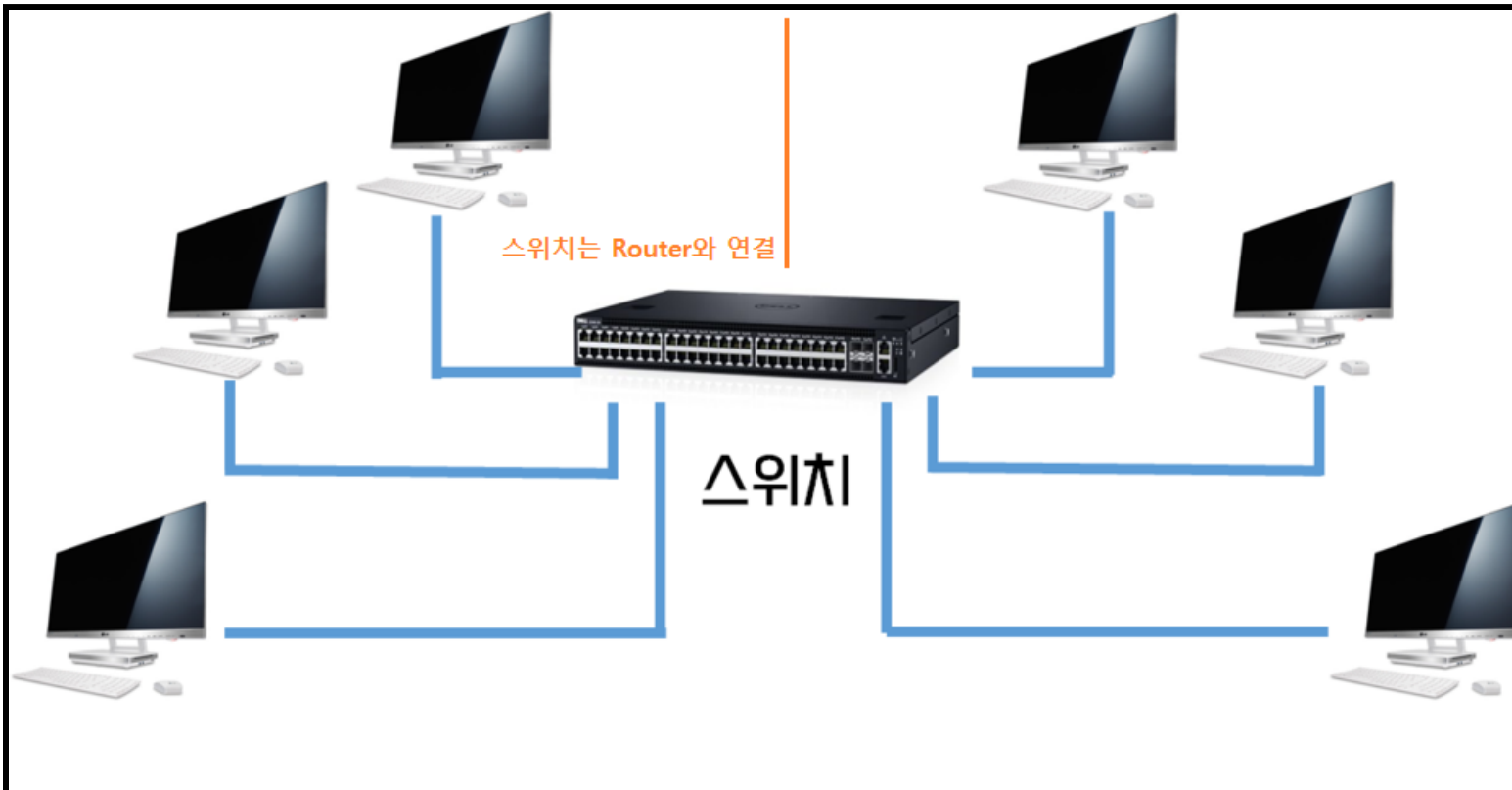


◀ server 구동
중지 되었을 때

```
Samuel@DESKTOP-VVE1E8K MINGW64 ~  
$ netstat -na | grep 3333  
TCP      0.0.0.0:33333      0.0.0.0:0          LISTENING  
TCP      192.168.1.1:4814   192.168.1.1:33333   ESTABLISHED  
TCP      192.168.1.1:4816   192.168.1.1:33333   ESTABLISHED  
TCP      192.168.1.1:33333  192.168.1.1:4814   ESTABLISHED  
TCP      192.168.1.1:33333  192.168.1.1:4816   ESTABLISHED  
TCP      [::]:33333        [::]:0             LISTENING
```

Listening: 누군가 접속하길 기다리는 중
Established: 누군가 접속해서 연결되었음

〈네트워크 흐름에 대한 개념〉



스위치:

전원만 연결하면 기본적인 설정없이 바로 사용할 수 있는 네트워크 연결 장비

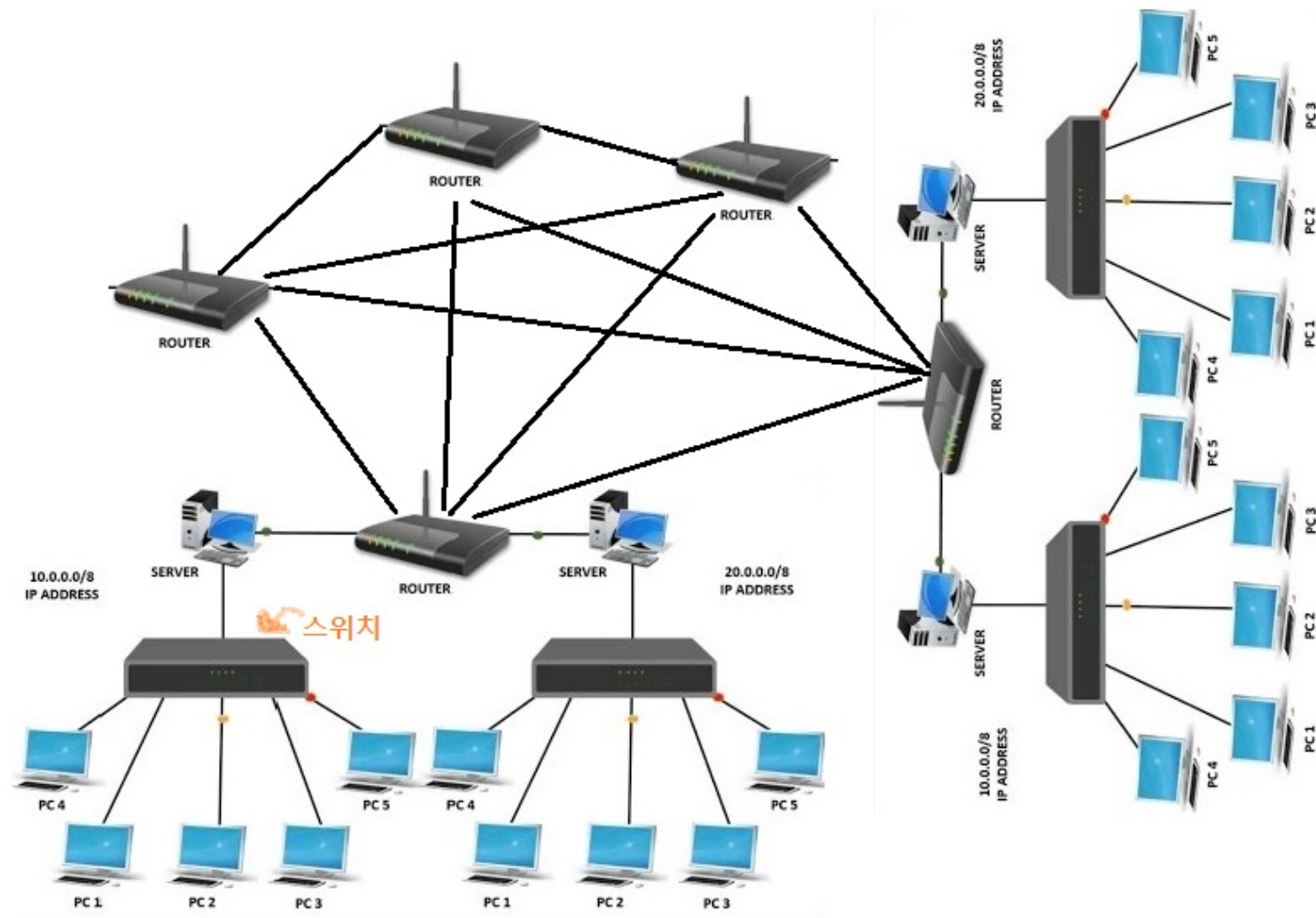
스witch는 네트워크 환경에 접속되어 있는 장비들에게 각각의 고유한 대역폭을 할당한다.

전체 대역폭을 나누어 공유하는 것이 아니라

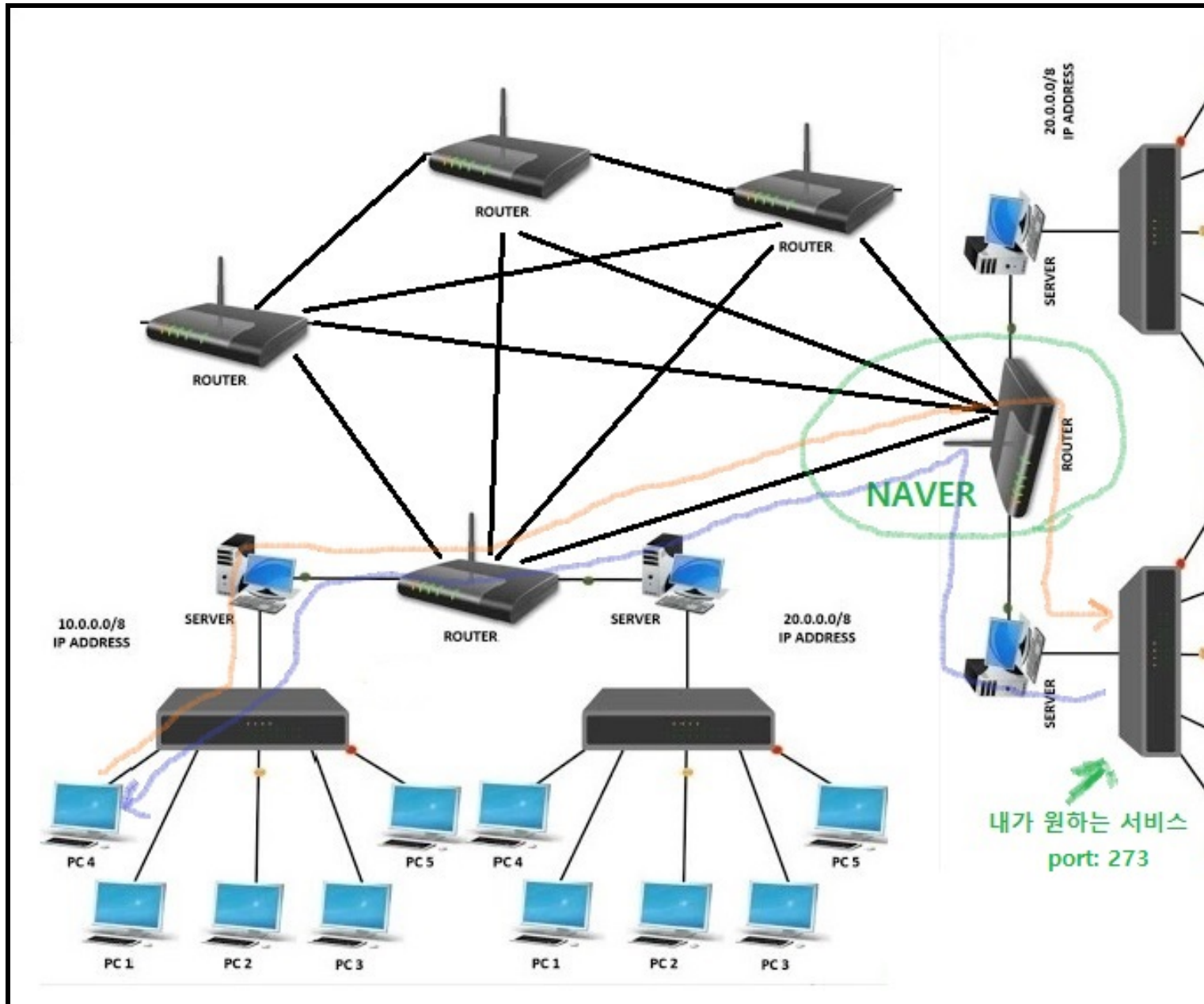
전체 대역폭 그대로를 연결된 포트에 각각 할당해 주는 것.

Broadcasting: 스위치가 전 구간으로 data를 뿌리는 것

이런식으로 연결되어 있음.



만약 내가 어떤 NAVER의 어떤 서비스를 원한다면 그 흐름은 >>



전체적인 흐름을 이해하되

내가 그 흐름에서 알아야 할 중요한 것은 IP / PORT 번호에 대한 개념.

《 추가적인 개념 》

- IP에는 공인 IP / 사설 IP 두 종류가 있으며 인터넷을 하려면 **공인 IP**가 필요하다.
- 라우터/공유기/스위치는 하나의 공인 IP를 가지고 여러대의 사설 IP가 인터넷을 사용할 수 있게 함.

- **스위치(switch)**는 네트워크 회선과 서버컴퓨터를 연결하는 네트워크 장비.

자기에게 연결된 장비들의 IP와 MAC주소를 모두 가지고 있기때문에. 장치에서 패킷이 오면 그

패킷의 목적지를 파악해서그 장치로만 패킷을 보내준다. (모든 장치에 패킷을 다 보내는 HUB에 비해 네트워크 속도가 빨라짐)

- **라우터(router)**는 패킷의 위치를 추출하여 그 위치에 대한 최적의 경로를 지정하며 이 경로를 따라 데이터 패킷을 다음 장치로 전향시키는 장치.

참고

>> <https://blog.naver.com/skdaksdptn/221902361141>

>> <https://blog.naver.com/bizblocklll/222149701305>