# TCP/IP Protocal & Socket

유튜브: https://youtu.be/T5EzeM1TPk8

사이버보안트랙 1971362 이준희

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

## <목차>

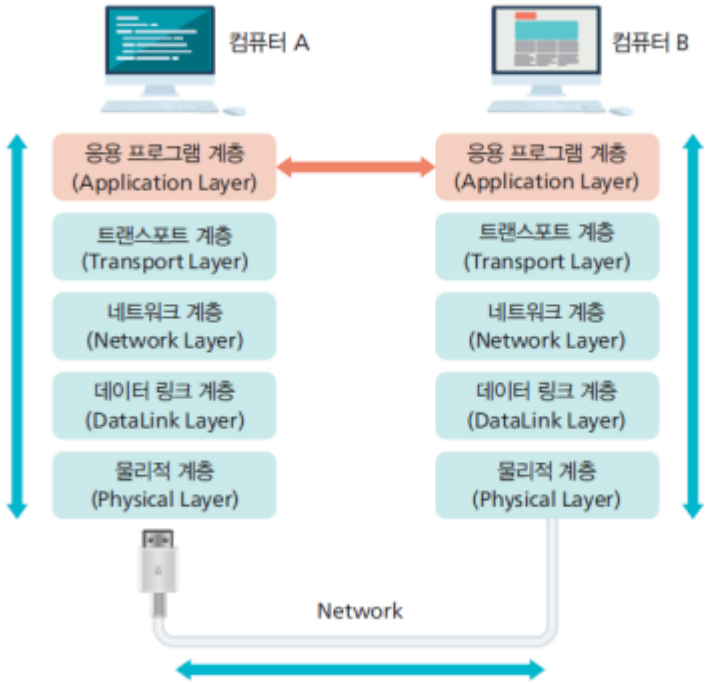


# TCP/IP 프로토콜

- TCP/IP 프로토콜
  - TCP는 Transmission Control Protocol
  - 두 시스템 간에 신뢰성 있는 데이터의 전송을 관장하는 프로토콜
  - TCP에서 동작하는 응용프로그램 사례
    - – e-mail, FTP, 웹(HTTP) 등
- IP (Internet Protocoal)
  - 패킷 교환 네트워크에서 송신 호스트와 수 신 호스트가 데이터를 주고 받는 것을 관장 하는 프로토콜
  - TCP보다 하위 레벨 프로토콜

# 프로토콜(Protocal)

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



# IP 주소

- IP 주소
  - 네트워크 상에서 유일하게 식별될 수 있는 컴퓨터 주소
    - 숫자로 구성된 주소
    - 4개의 숫자가 '.'으로 연결
      - 예) 192.156.11.15
  - 숫자로 된 주소는 기억하기 어려우므로 www.naver.com과 같은 문자열로 구성된 도메인 이름으로 바꿔 사용
    - DNS(Domain Name System)
      - 문자열로 구성된 도메인 이름을 숫자로 구성된 IP 주소로 자동 변환
  - 현재는 32비트의 IP 버전 4(IPv4)가 사용되고 있음
    - IP 주소 고갈로 인해 128비트의 IP 버전 6(IPv6)이 점점 사용되는 추세

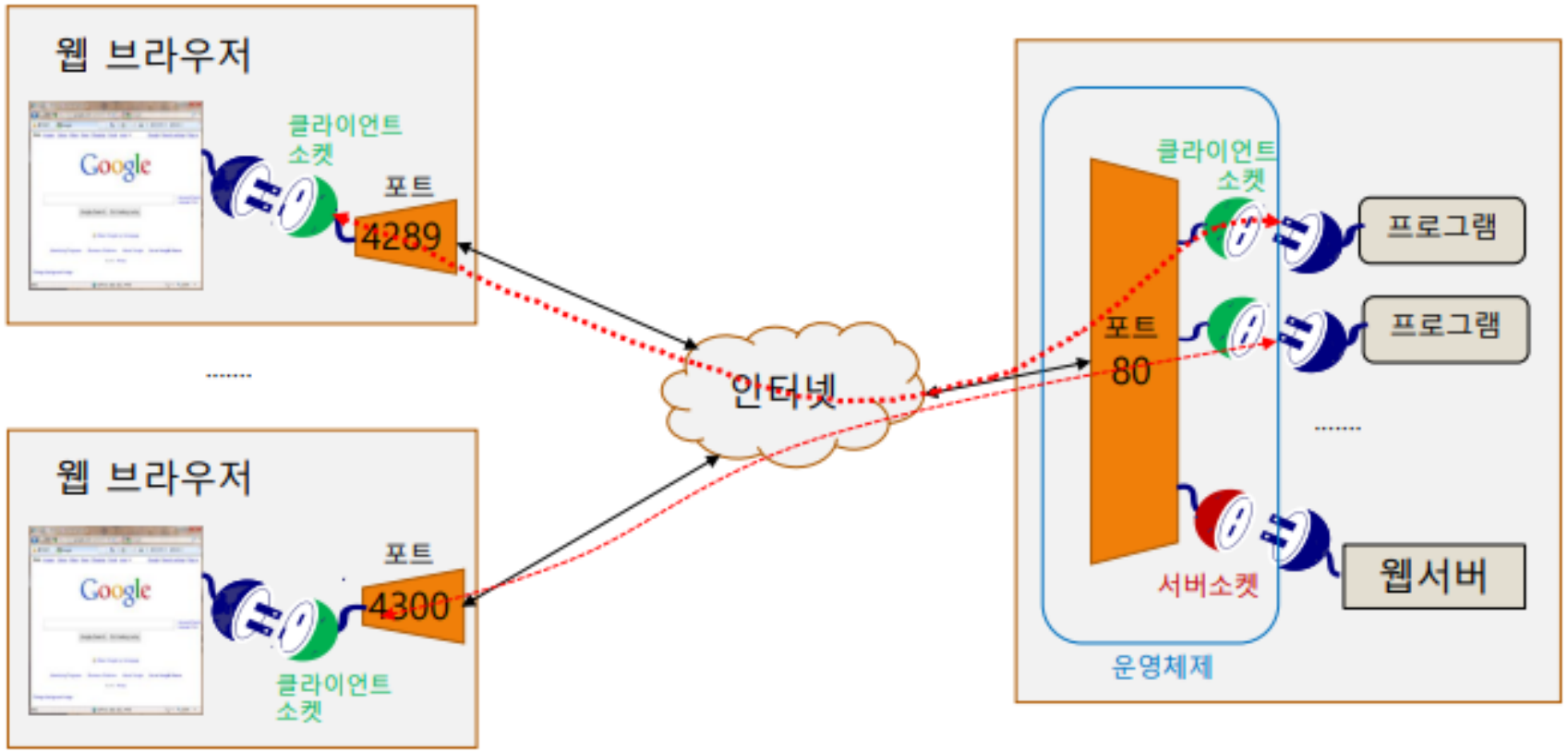
# 포트 (Port)

- 포트 (Port)
  - 통신하는 프로그램 간에 가상의 연결단 포트 생성
    - IP 주소는 네트워크 상의 컴퓨터 또는 시스템을 식별하는 주소
    - 포트 번호를 이용하여 통신할 응용프로그램 식별
  - 모든 응용프로그램은 하나 이상의 포트 생성 가능
    - 포트를 이용하여 상대방 응용프로그램과 데이터 교환
  - 잘 알려진 포트(well-known ports)
    - 시스템이 사용하는 포트 번호
    - 잘 알려진 응용프로그램에서 사 용하는 포트 번호
      - 0부터 1023 사이의 포트 번호
      - ex) SSH 22, HTTP 80, FTP 21
    - 잘 알려진 포트 번호는 개발자가 사용하지 않는 것이 좋음
    - (0부터 1023 사이의 포트 번호를 제외한 나머지 ~65535까지의 포트 사용)
      - => 충돌 가능성 있기 때문에

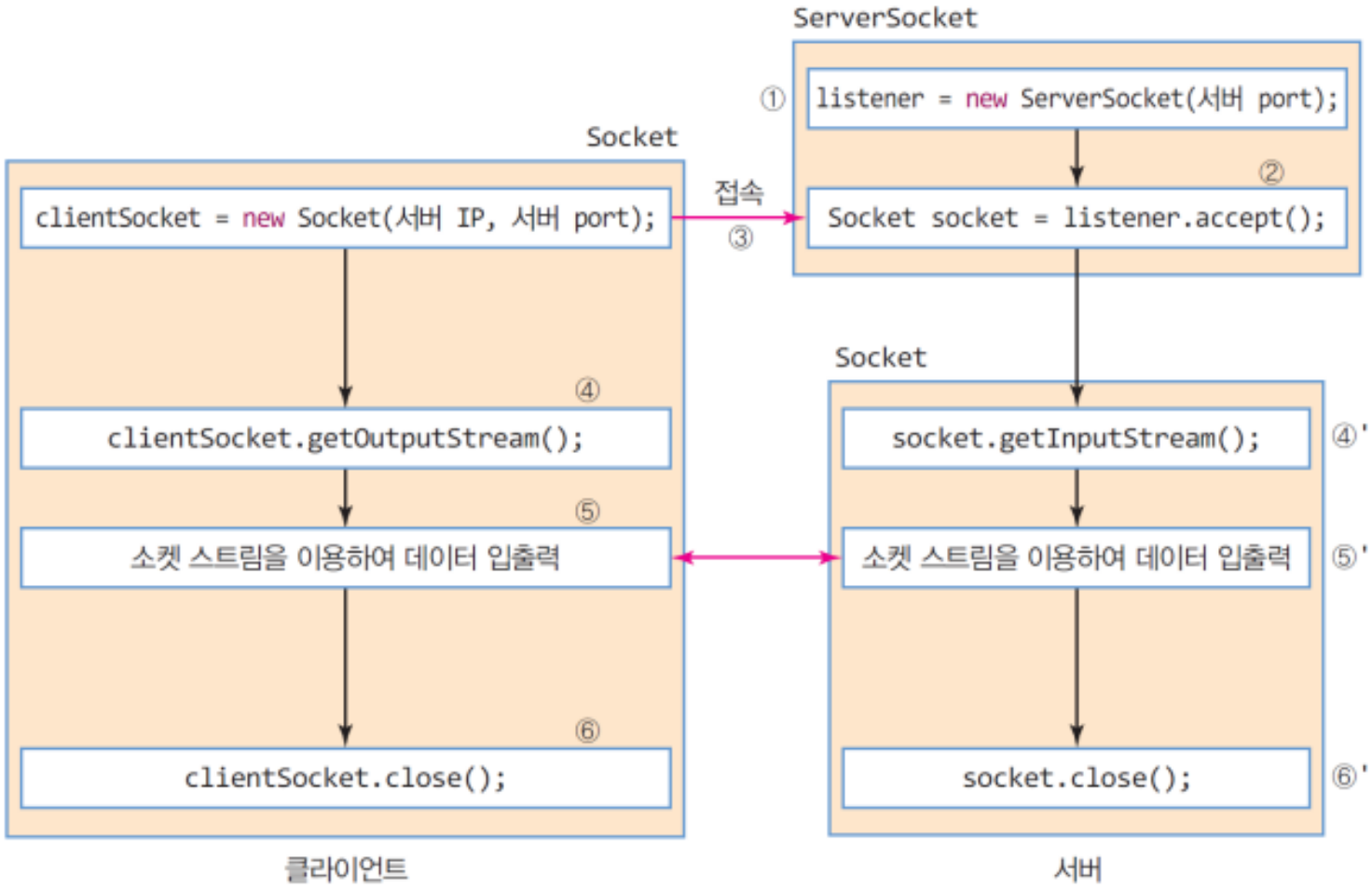
# 소켓 (Socket)

- 소켓 (Socket)
  - TCP/IP 네트워크를 이용하여 쉽게 통신 프로그램을 작성하도록 지원하는 기반 기술
  - 두 응용프로그램 간의 양방향 통신 링크의 한쪽 끝 단 소켓끼리 데이터를 주고 받음
  - 소켓은 특정 IP 포트 번호와 결합
  - 자바로 소켓 통신할 수 있는 라이브러리 지원
  - 소켓 종류 : 서버 소켓과 클라이언트 소켓
  - 소켓은 콘센트와 같은 역할 수행
  - 소켓을 연결하기만 하면 하위의 복잡한 프로토콜이나 네트워크 상태를 알 필요 없이 연결된 소켓을 통해서 데이터를 주고받을 수 있음 서버 소켓 클래스와 (클라이언트) 소켓 클래스

# 소켓을 이용한 웹 서버와 클라이언트 사이의 통신 사례



# 소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조



# Socket 클래스, 클라이언트 소켓

- Socket 클래스
  - 클라이언트 소켓에 사용되는 클래스
  - java.net 패키지에 포함

| 생성자 | 설명 |
|---|---|
| Socket | 연결되지 않은 상태의 소켓을 생성 |
| Socket(InetAddress address, int port) | 소켓을 생성하고, 지정된 IP 주소(addresss)와 포트 번호(port)에서 대기하는 원격 응용프로그램의 소켓에 연결 |
| Socket(String host, int port) | 소켓을 생성하여 지정된 호스트(host)와 포트 번호(port)에 연결한다. 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정 |

| 메소드 | 설명 |
|---|---|
| void bind(SocketAddress bindpoint) | 소켓에 로칼 IP 주소와 로컬 포트 지정 |
| void close() | 소켓을 닫는다. |
| void connect(SocketAddress endpoint) | 소켓을 서버에 연결 |
| InetAddress getInetAddress() | 소켓에 연결된 서버 IP 주소 반환 |
| InputStream getInputStream() | 소켓에 연결된 입력 스트림 반환. 이 스트림을 이용하여 소켓이 상대편으로부터 받은 데이터를 읽을 수 있음 |
| InetAddress getLocalAddress() | 소켓이 연결된 로컬 주소 반환 |
| int getLocalPort() | 소켓의 로컬 포트 번호 반환 |
| int getPort() | 소켓에 연결된 서버의 포트 번호 반환 |
| OutputStream getOutputStream() | 소켓에 연결된 출력 스트림 반환. 이 스트림에 출력하면 소켓이 서버로 데이터 전송 |
| boolean isBound() | 소켓이 로컬 주소에 연결되어있으면 true 반환 |
| boolean isConnected() | 소켓이 서버에 연결되어 있으면 true 반환 |
| boolean isClosed() | 소켓이 닫혀있으면 true 반환 |
| void setSoTimeout(int timeout) | 데이터 읽기 타임아웃 시간 지정. 0이면 타임아웃 해제 |

# 클라이언트에서 소켓으로 서버에 접속하는 코드

- 클라이언트 소켓 생성 및 서버에 접속
  - ex) Socket clientSocket = new Socket("128.12.1.1", 5550);
    - => Socket 객체의 생성되면 곧 바로 128.12.1.1의 주소의 5550포트에 자동 접속
- 소켓으로부터 데이터를 전송할 입출력 스트림 생성
  - ex) BufferedReader in = new BufferedReader( new InputStreamReader(clientSocket.getInputStream()));
  - BufferedWriter out = new BufferedWriter( new OutputStreamWriter(clientSocket.getOutputStream()));
- 서버로 데이터 전송
  - flush()를 호출하면 스트림 속에 데이터를 남기지 않고 모두 전송
  - ex) out.write("hello"+"\n"); out.flush();
- 서버로부터 데이터 수신
  - String line = in.readline(); //서버로부터 한 행의 문자열 수신
- 네트워크 접속 종료
  - clientSocket.close();

# ServerSocket 클래스, 서버 소켓

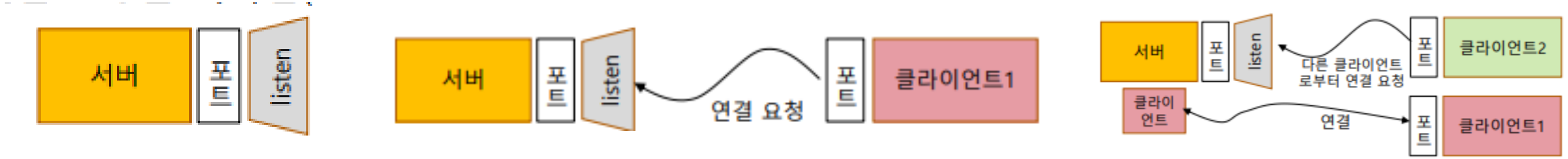
- ServerSocket 클래스
  - 서버 소켓에 사용되는 클래스
  - java.net 패키지에 포함

| 생성자 | 설명 |
|---|---|
| ServerSocket(int port) | 지정된 포트 번호(port)와 결합된 소켓 생성 |

| 메소드 | 설명 |
|---|---|
| Socket accept() | 클라이언트로부터 연결 요청을 기다리다 요청이 들어오면 수락하고 클라이언트와 데이터를 주고받을 새 Socket 객체를 반환 |
| void close() | 서버 소켓을 닫는다. |
| InetAddress getInetAddress() | 서버 소켓의 로컬 IP 주소 반환 |
| int getLocalPort() | 서버 소켓의 로컬 포트 번호 반환 |
| boolean isBound() | 서버 소켓이 로컬 주소에 연결되어있으면 true 반환 |
| boolean isClosed() | 서버 소켓이 닫혀있으면 true 반환 |
| void setSoTimeout(int timeout) | accept()가 대기하는 타임아웃 시간 지정. 0이면 무한정 대기 |

# 서버에 클라이언트가 연결되는 과정

- 1. 서버는 서버 소켓으로 들어오는 연결 요청을 기다림(listen)
- 2. 클라이언트가 서버에게 연결 요청
- 3. 서버가 연결 요청 수락(accept)
  - 새로운 클라이언트 소켓을 만들어 클라이언트와 통신하게 함
  - 그리고 다시 다른 클라이언트의 연결을 기다림



# 서버가 클라이언트와 통신하는 과정

- 서버 소켓 생성
  - ex) ServerSocket serverSocket = new ServerSocket(5550);
    - => 서버는 접속을 기다리는 포트로 5550 선택
- 클라이언트로부터 접속 기다림
  - ex) Socket socket = serverSocket.accept();
    - accept() 메소드는 연결 요청이 오면 새로운 Socket 객체 반환
    - 접속 후 새로 만들어진 Socket 객체를 통해 클라이언트와 통신
- 네트워크 입출력 스트림 생성
  - BufferedReader in = new BufferedReader( new InputStreamReader(socket.getInputStream()));
  - BufferedWriter out = new BufferedWriter( new OutputStreamWriter(socket.getOutputStream()));
  - Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 입출력 데이터 스트림 생성

# 스트림 (Stream)

- 스트림 입출력
  - 버퍼를 가지고 순차적으로 이루어지는 입출력
- 자바의 입출력 스트림
  - 응용프로그램과 입출력 장치를 연결하는 소프트웨어 모듈
    - 입력 스트림 : 입력 장치로부터 자바 프로그램으로 데이터를 전달
    - 출력 스트림 : 출력 장치로 데이터 출력

# 자바의 입출력 스트림 특징

- 스트림의 양끝에 입출력장치와 자바 응용프로그램 연결
- 스트림은 단방향
  - 입력과 출력을 동시에 하는 스트림 없음
- 입출력 스트림 기본 단위
  - 바이트 스트림의 경우 : 바이트
  - 문자 스트림의 경우 : 문자(자바에서는 문자1개 : 2 바이트)
- 선입선출 구조 (FIFO)

# 자바의 입출력 스트림 종류

- 바이트 스트림과 문자 스트림
  - 바이트 스트림
    - 입출력되는 데이터를 단순 바이트로 처리
      - 예) 바이너리 파일을 읽는 입력 스트림
  - 문자 스트림
    - 문자만 입출력하는 스트림
    - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
      - 예) 텍스트 파일을 읽는 입력 스트림
- JDK는 입출력 스트림을 구현한 다양한 클래스 제공

# Q & A