mimunlee 스터디 4주차 발표 자료

💚 섹션 1. 인터넷 네트워크

IP

인터넷 프로토콜 (Internet Protocol)

IP의 역할

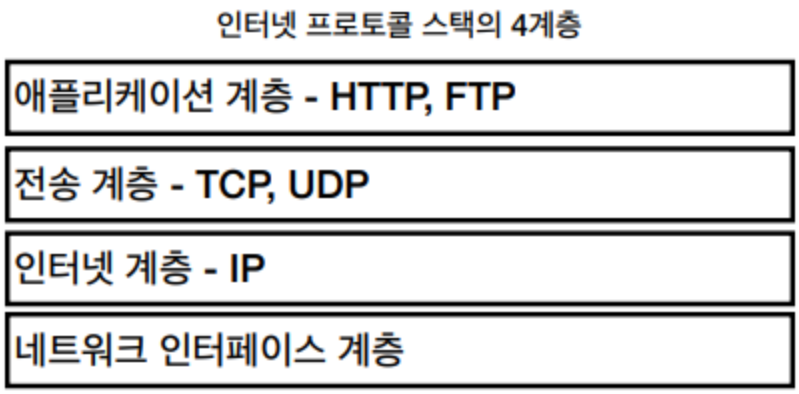
- 지정한 IP 주소(IP Address)에 데이터 전달
- 패킷(Packet)이라는 통신 단위로 데이터 전달

IP 프토토콜의 한계

- 비연결성
  - 패킷을 받을 대상이 없거나 서비스 불능 상태여도 패킷 전송
    - ex) 패킷을 받을 대상이 꺼져있어도 패킷을 보내는 클라이언트는 대상의 상태를 알 수 없다.
- 비신뢰성
  - 중간에 패킷이 사라지면?
    - ex) 중간에 랜 케이블이 끊겨있어, 통신이 안될 때 패킷을 보내면 패킷이 정상적으로 전송되지 않고 소실된다.
  - 패킷이 순서대로 안오면?
    - ex) hello와 world를 순서대로 보냈는데, world가 먼저 도착할 수도 있다.

      → 순서가 보장이 안된다!
- 프로그램 구분
  같은 IP를 사용하는 서버에서 통신하는 애플리케이션이 둘 이상이면?

인터넷 프로토콜 스택의 4계층



통신 과정

애플리케이션 계층

1. 프로그램이 Hello, world! 메시지 생성
2. SOKET 라이브러리를 통해 전달

전송 계층

3. TCP 정보 생성, 메시지 데이터 포함

인터넷 계층

4. IP 패킷 생성 (TCP 데이터 포함)

네트워크 인터페이스 계층

5. 생성된 IP 패킷에 인터넷 프레임을 씌운 뒤, 서버에 전송

☝️ **여기서 패킷이란?**

정보 기술에서 패킷 방식의 컴퓨터 네트워크가 전달하는 데이터의 형식화된 블록이다.

쉽게 말하면 pack과 bucket의 합성어로,
우체국에서 화물을 적당한 덩어리로 나눠 행선지를 표시하는 꼬리표를 붙이듯이 **정보를 보낼 때 특정 형태를 맞추어서 보내자고 약속한 규칙**이다.

IP 패킷

IP 패킷이 가지는 정보

- 출발지 IP
- 목적지 IP
- 기타

TCP/IP 패킷

TCP/IP 패킷이 가지는 정보

- IP 패킷
- 출발지 PORT
- 목적지 PORT
- 전송 제어
- 순서
- 검증 정보

TCP

전송제어 프로토콜(Transmission Control Protocol)

TCP 특징

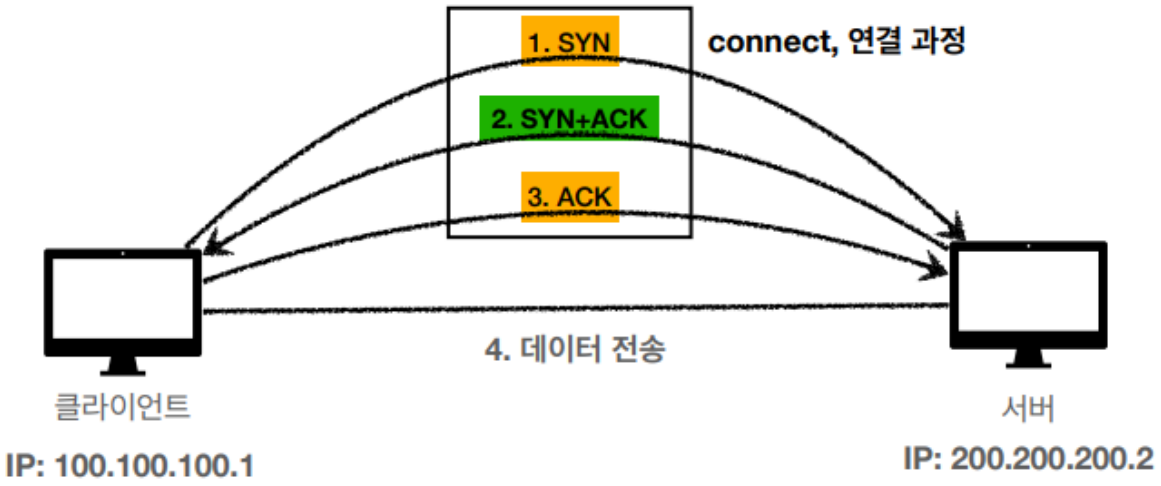
- 연결 지향 - TCP 3 way handshake **(가상 연결)**
  - 연결이 되었는지 확인 후 데이터 전송

    → IP 프로토콜의 비연결성 한계 해결
- 데이터 전달 보증
  - 대상에게 데이터를 전송했을 때, 대상이 데이터를 잘 받았는지 응답이 돌아온다.

    → IP 프로토콜의 데이터 소실 문제 해결
- 순서 보장

이러한 TCP의 특징은 TCP가 전송 제어 정보, 순서 정보, 검증 정보 등 다양한 정보를 가지고 있기 때문에 가능하다.

TCP 3 way handshake (가상연결)



- SYN : 접속 요청
- ACK : 요청 수락

참고 : 3. ACK와 함께 데이터 전송 가능

3 way handshake가 가상연결인 이유?

클라이언트와 서버가 서로 SYN, ACK로 상태를 체크하고 이상이 없다면, 연결이 되었다고 생각하는 개념

✅ 이와 같은 이유로 **TCP는 신뢰할 수 있는 프로토콜**이다.

현재는 대부분 TCP를 사용한다.

UDP

사용자 데이터그램 프로토콜 (User Datagram Protocol)

UDP 특징

- 하얀 도화지와 유사하다. (기능이 거의 없음)
- TCP 처럼
  - 연결지향 → X
  - 데이터 전달 보증 → X
  - 순서 보장 → X

그런데 UDP를 왜 쓸까?

UDP는 IP에 **PORT**와 체크섬 정도만 추가된 것이다.

PORT 정보가 있기 때문에 IP의 문제점 중 **같은 IP를 사용하는 서버에서 통신하는 애플리케이션이 둘 이상이면 구분을 할 수 없다는 한계**를 해결한다.

TCP는 여러 정보가 있고 여러 과정을 거쳐 데이터를 전송하기 때문에 속도가 느리고 데이터가 복잡하지만 **UDP는 그에 비해 단순하고 빠르다.**

하지만! 아직까지는 TCP를 대부분 사용한다는 것~!

PORT

같은 IP 내에서 프로세스를 구분하기 위한 번호.

- 0 ~ 65535 할당 가능
- 0 ~ 1023 : 잘 알려진 포트, 사용하지 않는 것이 좋음
  - FTP - 20, 21
  - TELNET - 23
  - HTTP - 80
  - HTTPS - 443

DNS

도메인 네임 시스템 (Domain Name System)

- 전화번호부와 같이 **IP 주소를 도메인 명과 함께 저장**한다.
- IP의 기억하기 어렵다는 점과 변경될 수 있다는 점을 해결하기 위해 사용한다.
- 도메인 명으로 DNS 서버에 요청하면 IP주소를 응답해주고 해당 IP 주소로 접속하는 식으로 사용한다.

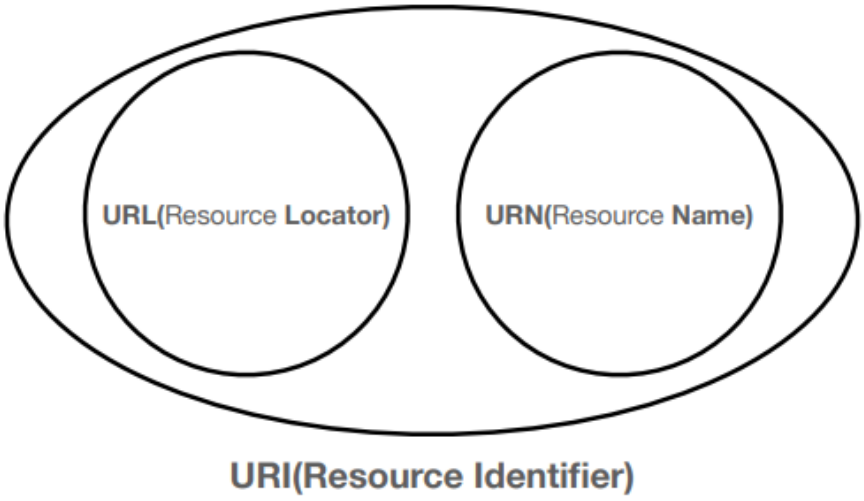
💙 섹션 2. URI와 웹 브라우저 요청 흐름

URI

- **U**niform : 리소스 식별하는 통일된 방식
- **R**esource : 자원, URI로 식별할 수 있는 모든 것 (제한 없음)
- **I**dentifier : 다른 항목과 구분하는데 필요한 정보

URI 구조

URI는 로케이터, 이름 또는 둘다 추가로 분류될 수 있다.



URL, URN

- URL - Uniform Resource Locator : 리소스가 있는 위치를 지정
- URN - Uniform Resource Name : 리소스에 이름을 부여
- URN 이름만으로 실제 리소스를 찾을 수 있는 방법이 보편화 되지 않음 → 거의 사용 X
- **고로 URI와 URL은 거의 같은 의미라고 봐도 된다.**

URL 문법

🔗 문법

scheme://[userinfo@]host[:port][/path][?query][#fragment]

실제 예

https://www.google.com:443/search?q=hello&hl=ko

• 프로토콜 (https)
• 호스트명 (www.google.com)
• 포트 번호 (443)
• 패스 (/search)
• 쿼리 파라미터 (q=hello&hl=ko)

scheme

scheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- 주로 프로토콜 사용

- **프로토콜**: 어떤 방식으로 자원에 접근할 것인가 하는 약속 규칙
  - 예) http, https, ftp 등등
- http는 80 포트, https는 443 포트를 주로 사용, 포트는 생략 가능
  - https는 http에 보안 추가 (HTTP Secure)

userinfo

scheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- URL에 사용자정보를 포함해서 인증
- 거의 사용하지 않음

host

sscheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- 호스트명
- 도메인명 또는 IP 주소를 직접 사용가능

PORT

scheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- 접속 포트
- 일반적으로 생략, 생략시 http는 80, https는 443

path

scheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- 리소스 경로(path), 계층적 구조
- 예)
  - /home/file1.jpg
  - /members
  - /members/100, /items/iphone12

query

scheme://[userinfo@]host[:port][/path][?query][#fragment]

https://www.google.com:443/search?q=hello&hl=ko

- key=value 형태

- ?로 시작, &로 추가 가능 ?keyA=valueA&keyB=valueB
- **query parameter, query string 등으로 불림**, 웹서버에 제공하는 파라미터, 문자 형태

fragment

> **scheme://[userinfo@]host[:port][/path][?query][#fragment]**
>
> **https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started.introducing-spring-boot**

- html 내부 북마크 등에 사용
- 서버에 전송하는 정보 아님

웹 브라우저 요청 흐름

https://www.google.com:443/search?q=hello&hl=ko

- **구글에 hello 검색 한국어로 요청하는 URL**

1. DNS 조회 → IP 알아냄
2. 포트 알아냄 (443)
3. HTTP 요청 메시지 생성

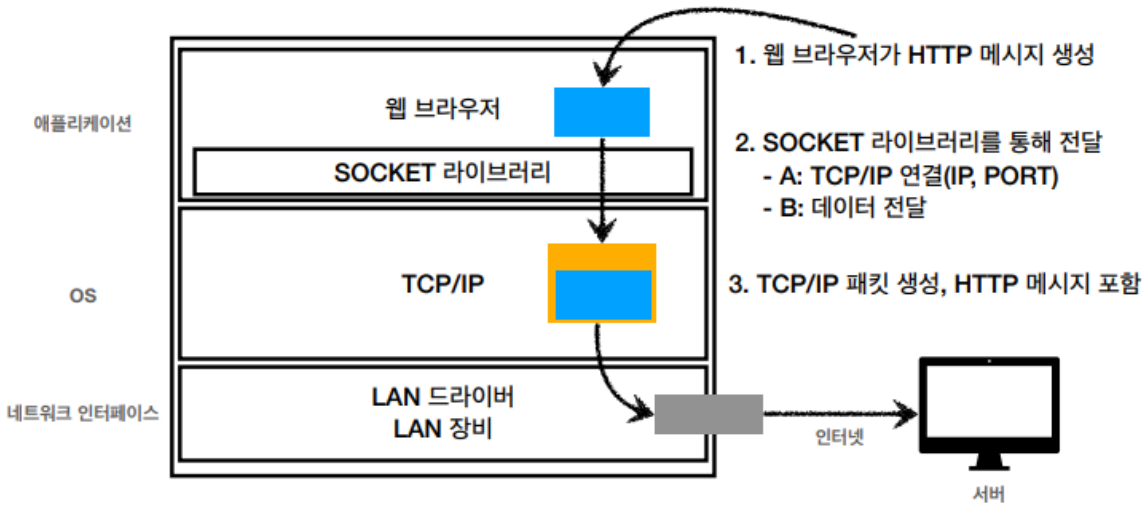
GET /search?q=hello&hl=ko
HTTP/1.1
Host: www.google.com

4. HTTP 메시지 전송
5. 패킷 생성

6. 요청 패킷 전송
7. 아래와 같은 HTTP 응답 메시지 포함된 응답 패킷 전달받음

💬 HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8
Content-Length: 3423
<html>
<body>...</body>
</html>

패킷 전송 흐름


애플리케이션
웹 브라우저
SOCKET 라이브러리
OS
TCP/IP
네트워크 인터페이스
LAN 드라이버
LAN 장비
1. 웹 브라우저가 HTTP 메시지 생성
2. SOCKET 라이브러리를 통해 전달
- A: TCP/IP 연결(IP, PORT)
- B: 데이터 전달
3. TCP/IP 패킷 생성, HTTP 메시지 포함
인터넷
서버