2025.1.3.금

<모든 개발자를 위한 HTTP 웹 기본 지식>

섹션2. 인터넷 네트워크

1. 인터넷 통신: 클라이언트 서버 사이에 인터넷 있음

인터넷망을 통해 헬로월드 메시지 보내야 함. 인터넷이 단순하지 않음. 해저 광케이블 있을 수 있고 인공위성 통해서 내려갈 수도 있고. 어떤 규칙으로 어떻게 목적지까지 도착?

그러면 ip에 대해 학습해야함

2. IP(인터넷 프로토콜)

헬로월드 한국-> 미국으로 보내야 함. ip주소를 통해 가능해짐. 클라이언트인 내가 아이피주소를 부여받아야 함. 친구도 서버에 아이피주소가 있어야 함.

메시지를 그냥 보내는게 아니라 아이피패킷이란 규칙이 있음. 아이피패킷 만들어서 던짐

친구도 똑같이 함.

내가 던질 때랑 친구가 나에게 던질 때랑 요청 응답할 때 서로 다른 곳으로 전달될 수도 있음. 인터넷망 복잡하기 때문에.

프로그램구분. 게임하면서 음악도 듣고. 같은 아이피인데 어떻게 구분?

패킷 도착했는데 서버가 꺼질수도

패킷사이즈 1500바이트 정도. 끊어서 보냄.

MTU 설명 참고: <https://skstp35.tistory.com/284>

MTU (Maximum Transmission Unit 최대 전송 단위)

데이터링크에서 하나의 프레임 또는 패킷에 담아 운반 가능한 최대 크기

운영체제별 MTU 확인

- 윈도우 : netsh interface ip show interface - Linux : ifconfig



중간에 다른 노드 탈 수도

TCP가 해결해줌

3. TCP, UDP

패킷: 패키지 + 버킷(덩어리) 합성어.

TCP 특징

연결지향- 연결여부. 소켓이 연결됐다. 진짜 연결된 게 아니라 개념적으로 연결된 것.

SYN 싱크로나이즈

UDP 특징

기능이 없음

PORT: 게임용, 음악용 패킷 구분해주는 용도

TCP는 3 way handshake 하려면 시간이 걸림.

TCP는 손을 못 댐. UDP 최근 각광받고 있음.

4. PORT

애플리케이션 구분 위해.

IP가 아파트라면 PORT는 몇동 몇호.

5. DNS

중간에 전화번호부 같은 서버를 제공함.

도메인 사서 등록.

DNS 서버 있으면 두 문제 다 해결됨

섹션3. URI와 웹 브라우저 요청 흐름

1. URI

URI – 소스를 식별한다.

URL – 김영한 위치, URN – 김영한 그 자체.

URN 찾을 수 없음. 거의 안 씀

다 문자열 형태로 넘어가기 때문에 쿼리스트링이라고 함

2. 웹 브라우저 요청 흐름

섹션4. HTTP 기본

1. 모든 것이 HTTP

HTTP 역사: 1.1에 거의 대부분의 기능 들어가 있음

HTTP2, 3은 성능 개선한 것

크롬브라우저 열기 -> F12 -> 네트워크 -> 프로토콜 열기. H1.1, h2 확인 가능

2. 클라이언트 서버 구조

과거에는 클라이언트, 서버 구분되지 않았음

클라이언트 서버 구조를 만들게 되면 양쪽이 독립적으로 진행, 각각 발전 가능

3. Stateful, Stateless

서버가 클라이언트의 상태를 보존하지 않는다는 의미

상태유지 - 점원c는 고객의 컨텍스트를 모름

무상태 – 고객이 필요한 데이터를 그 때 그 때 점원에게 다 넘겨줌

무상태로 설계하면 무한한 서버 증설 가능

무상태 실무 한계 – 데이터를 너무 많이 보냄

2025.1.13.월

4. 비 연결성(connectionless)

스테이스리스를 기억하자

- 첫페이지는 로그인도 필요없는 정적 페이지 뿌리기. 거기서 놀게 한 다음에 이벤트 참여하도록.

5. HTTP 메시지

공백 무조건 있어야 함

요청 메시지 : DELETE는 삭제. PUT은??

HTTP 헤더: OWS - 띄어쓰기 허용

**HTTP란?**

**HTTP**(HyperText Transfer Protocol)는 **월드 와이드 웹(WWW)** 상에서 클라이언트와 서버 간에 데이터를 주고받기 위해 사용되는 프로토콜입니다. 웹 브라우저(클라이언트)가 서버에 요청(request)을 보내고, 서버는 이에 대한 응답(response)을 보내는 형태로 작동합니다.

**HTTP의 주요 특징**

1. **비연결성**
   * 클라이언트와 서버가 요청과 응답을 주고받으면 연결이 끊어짐.
   * 요청마다 새로운 연결이 생성되어 처리됨.
2. **무상태성 (Stateless)**
   * 서버는 클라이언트의 이전 요청 정보를 기억하지 않음.
   * 필요할 경우 쿠키, 세션, 토큰 등을 사용해 상태를 유지.
3. **텍스트 기반 프로토콜**
   * 요청 및 응답은 사람이 읽을 수 있는 텍스트 형태로 전송됨.
4. **HTTP 메서드**
   * **GET**: 데이터 요청
   * **POST**: 데이터 전송
   * **PUT**: 데이터 수정
   * **DELETE**: 데이터 삭제
   * 등 다양한 메서드를 지원.

**HTTP의 발전**

HTTP/1.0 → HTTP/1.1 → HTTP/2 → HTTP/3로 발전하면서 더 빠르고 효율적인 통신이 가능해졌습니다.

**HTTP와 비슷한 다른 기술들**

HTTP는 웹 통신의 표준이지만, 다음과 같은 유사한 기술도 있습니다:

1. **HTTPS (HTTP Secure)**
   * HTTP에 **SSL/TLS** 암호화를 추가해 보안성을 강화한 버전.
   * 민감한 데이터(예: 로그인 정보) 보호에 사용.
2. **FTP (File Transfer Protocol)**
   * 파일 전송에 특화된 프로토콜.
   * 주로 파일 업로드/다운로드에 사용되며, HTTP보다 단순한 구조를 가짐.
3. **SMTP/IMAP/POP3**
   * 이메일 전송 및 관리에 사용되는 프로토콜들.
     + **SMTP**: 메일 전송.
     + **IMAP**: 메일 서버에서 메일을 읽음.
     + **POP3**: 메일을 로컬로 다운로드.
4. **WebSocket**
   * 양방향 실시간 통신이 가능한 프로토콜.
   * HTTP와는 다르게 클라이언트와 서버 간 연결이 지속됨.
5. **MQTT (Message Queuing Telemetry Transport)**
   * IoT(사물 인터넷) 기기 간 통신에 사용.
   * 경량화되어 네트워크 자원을 적게 소모함.
6. **CoAP (Constrained Application Protocol)**
   * IoT 환경에서 HTTP를 대신해 사용되는 경량 프로토콜.
   * HTTP와 비슷한 요청/응답 구조를 가지지만 더 간단함.

**간단 요약**

* HTTP는 클라이언트-서버 간 데이터를 주고받는 표준 프로토콜.
* HTTPS, FTP, WebSocket, MQTT 등 유사한 기술이 있으며, 각 기술은 특정 상황에 맞춰 사용됩니다.