**네트워크 개념 정리**

**전산기본**

***OSI 7*계층이란*?***

> 네트워크 통신이 일어나는 과정을 7단계로 나눈것

**7 계층 : Application Layer(응용 계층)**

> 사용자 인터페이스의 역할을 담당하는 계층

( 즉, 사용자가 이용하는 네트워크 응용프로그램)

EX) Internet Explorer

**6 계층 : Presentation Layer(표현 계층)**

> 전송하는 데이터의 Format(구성 방식)을 결정하는 계층

> 다양한 데이터 Format을 일관되게 상호 변환, 압축 및 암호화, 복호화 기능을 수행

**5 계층 : Session Layer(세션 계층) ★중요**

> 네트워크 상에서 통신을 할 경우 양쪽 host간 최초 연결이 되게 하고 통신 중 연결이 지속되도록 시켜주는 역할을 하는 계층.

> 통신을 하는 두 host사이에 세션을 열고, 닫고, 관리하는 기능을 담당.

**7 6 5계층을 통하여 data가 만들어 짐.**

===============================================================

**4 계층 : Transport Layer(전송 계층)**

> 정보를 분할하고, 상대편에 도달하기 전에 다시 합치는 과정을 담당하는 계층

4계층의 단위 : Segment

> 목적지 컴퓨터에서 발신지 컴퓨터 간의 통신에 있어 Error control(에러제어)와 Flow control(흐름 제어)를 담당.

**3 계층 : Network Layer(네트워크 계층)**

> Logical address(IP, IPX)를 담당하고, 패킷의 이동경로를 결정하는 계층

3계층의 단위 : Packet

> 경로선택, 라우팅, 논리적인주소(IP)를 정의하는 계층

**2 계층 : Data Link Layer(데이터링크 계층)**

> 물리적 계층을 통한 데이터 전송에 신뢰성을 제공

> 두가지 하위 게층이 존재

Logical Link contorl - 통신 장치간의 연결을 설정하고 관리하는 책임

Media Access Control - 다중 장치가 같은 미디어 채널을 공유, 제어(Block ID + Device ID)

**1 계층 : Physical Layer(물리 계층)**

> 네트워크 통신을 위한 물리적인 표준을 정의하는 계층

**OSI 7계층으로 나눈이유?**

> 계층을 나눈 이유는 통신이 일어나는 과정을 단계별로 파악할 수 있기 때문이다.

즉 7단계 중 특정한 곳에 이상이 생기면 다른 단계의 장비 및 소프트웨어를 건들이지 않고도 이상이 생긴 단계만 고칠 수 있기 때문이다.

***TCP/IP 4*계층**

**TCP/IP**

데이터가 의도된 목적지에 닿을 수 있도록 보장해주는 통신 규약이다.

TCP와 IP 두가지의 프로토콜로 이루어져 있다.

IP기반에 TCP가 사용되서 이렇게 불린다고 한다.

TCP가 데이터의 추적을, IP가 배달을 처리한다고 보면 된다.

**TCP**

두 호스트가 교환하는 데이터와 승인 메세지의 형식을 정의하여, 서버와 클라이언트간의 데이터를 신뢰성있게 전달하기 위해 만들어진 규약이다.

**IP**

컴퓨터와 컴퓨터간에 데이터를 전송하기 위해서, 각 컴퓨터의 주소가 필요하다. Internet Protocol은 4 바이트로 이루어진 컴퓨터의 주소이며, 192.168.9.255 와 같이 3개의 마침표로 나뉘어진 숫자로 표시된다.

**DNS란?**

Domain Name System 의 줄임말로, 사람이 이해하기 쉬운 www.naver.com과 같은 **Domain name** 을 사용해 요청했을 때, 호스트의 **IP주소** ex) 66.249.66.60 로 변환하거나 그 반대의 변환을 수행하기 위해서 개발된 시스템이다

**DNS 동작 원리**

1. 사용자가 브라우저에 www.naver.com 을 입력하는 순간 클라이언트에 저장된 Local DNS 서버 IP에게 www.naver.com 의 ip 주소를 물어본다. (사용자가 브라우저에 주소창을 입력하는 순간 클라이언트에 저장된 Local DNS 서버 IP에게 주소창의 ip 주소를 물어본다.)
2. Local DNS 서버는 우선 자신의 캐시에 해당 Domain의 ip 주소가 저장되어있는지 확인하고, 없다면 Root DNS 서버에 www.naver.com의 ip를 물어본다. Root DNS 서버는 TLD 서버들의 IP주소 테이블을 참조해, 이 중 com 네임서버의 ip주소를 Local DNS 서버에 보낸다.(Local DNS 서버는 우선 자신의 캐시에 해당 Domain ip주소가 저장되어있는지 확인하고, 없다면 Root DNS 서버에 주소창의 ip를 물어본다. Root DNS 서버는 TLD 서버들의 IP주소 테이블을 참조해 이중 com 네임서버의 ip주소를 Local DNS 서버로 보낸다.)
3. Local DNS는 응답받은 com 네임서버로 www.naver.com을 보낸다. 응답받은 com 네임서버도 마찬가지로 com 네임서버를 사용하는 도메인들의 ip테이블을 참조해 naver.com 네임서버의 ip주소를 보낸다.
4. 마지막으로 naver.com 네임서버에 www.naver.com 요청을 보낸다. www.naver.com , d2.naver.com 등 naver.com을 사용하는 모든 도메인 중 www.naver.com에 대한 ip 주소를 찾아 Local DNS 서버에게 응답한다.
5. Local DNS는 응답받은 ip주소를 캐시에 저장하고 클라이언트에게 보낸다.

**TCP/UDP**

TCP : 연결형 서비스를 지원하는 전송 계층 프로토콜로써, 인터넷 환경에서 기본으로 사용합니다.

호스트간 신뢰성 있는 데이터 전달과 흐름제어를 합니다.

UDP : 비연결형 서비스를 지원하는 전송계층 프로토콜로써, 인터넷상에서 서로 정보를 주고받을 때 정보를 보낸다는 신호나 받는다는 신호 절차를

거치지 않고,보내는 쪽에서 일방적으로 데이터를 전달하는 통신 프로토콜입니다.

|  |  |
| --- | --- |
| **TCP(Transfer Control Protocol)** | **UDP(User Datagram Protocol)** |
| **연결이 성공해야 통신 가능(연결형 프로토콜)** | **비연결형 프로토콜(연결 없이 통신이 가능)** |
| 데이터의 경계를 구분하지 않음(Byte-Stream Service) | 데이터의 경계를 구분함(Datagram Service) |
| **신뢰성 있는 데이터 전송(데이터의 재전송 존재)** | **비신뢰성 있는 데이터 전송(데이터의 재전송 없음)** |
| 일 대 일(Unicast) 통신 | 일 대 일, 일 대 다(Broadcast), 다 대 다(Multicast) 통신 |

**HTTP 프로토콜**

HTTP(Hypertext Transfer Protocol)는 인터넷상에서 데이터를 주고 받기 위한 **서버/클라이언트 모델**을 따르는 프로토콜이다.

애플리케이션 레벨의 프로토콜로 **TCP/IP위에서 작동**한다.

HTTP는**어떤 종류의 데이터든지 전송**할 수 있도록 설계돼 있다.

HTTP로 보낼 수 있는 데이터는 **HTML문서, 이미지, 동영상, 오디오, 텍스트 문서** 등 여러종류가 있다.

하이퍼텍스트 기반으로(Hypertext) 데이터를 전송하겠다(Transfer) = **링크기반으로 데이터에 접속**하겠다는 의미이다.

**HTTP와 HTTPS의 차이점**

**HTTP**의 보안 취약점을 해결하기 위한 프로토콜이 **HTTPS**다. **HTTPS**는 **HTTP**에 S(Secure Socket)을 추가한 것이다. 기본 골격이나 사용 목적 등은 **HTTP와** 거의 동일하지만, 데이터를 주고 받는 과정에 '보안'요소가 추가된다는 것이 가장 큰 **차이점**이다.

**HTTPS의 동작방식**

HTTPS의 핵심은 **공개키 암호화 방식**이다. 암호화와 복호화를 할 수 있는 2개의 키가 존재한다. 만약 1번 키로 데이터를 암호화 했다면, 다른 2번 키로만 복호화할 수 있다. 반대로 2번 키로 암호화 했다면, 1번 키로 복호화할 수 있는 방식이다.

생성된 2개의 키 중 하나는 공개 키 저장소에 존재한다. 또한 다른 하나는 서버가 소지하고 있다

클라이언트가 서버에 데이터 처리를 요청할 때, 공개 키 저장소에 있던 키를 통해 데이터 암호화를 진행한다.

공개 키를 통해 암호화 된 데이터를 포함하는 HTTPS 요청이 들어오면, 서버는 이에 대응하는 다른 키를 통해 복호화를 진행한다.

복호화된 데이터를 가지고 클라이언트의 요청을 처리하고 다시 클라이언트에 서버가 가진 키를 통해 암호화를 진행한 결과를 응답한다. 클라이언트는 이를 다시 공개 키를 통해 복호화를 진행해 데이터를 얻는다.

**HTTP Header 구조**

**HTTP 헤더**는 **HTTP** 본문(body) 및 요청/응답에 대한 정보를 포함한다. 본문에 대한 정보는 본문 컨텐츠 길이 등 본문 컨텐츠와 관련이 있다. 요청/응답에 대한 정보는 요청/응답에 대한 일반(General) 정보를 포함한다 (예 : 요청 시간, 요청에 사용 된 브라우저 등).

**Get과 Post의 차이점**

GET은 **서버로부터 정보를 조회하기 위해** 설계된 메소드

GET은 요청을 전송할 때 필요한 데이터를 Body에 담지 않고, 쿼리스트링을 통해 전송합니다.

POST는 **리소스를 생성/변경하기 위해 설계**되었기 때문에 GET과 달리 전송해야될 데이터를 HTTP 메세지의 Body에 담아서 전송합니다.

GET은 Idempotent, POST는 Non-idempotent하게 설계되었습니다.

멱등이라는 것은 **동일한 연산을 여러 번 수행하더라도 동일한 결과**가 나타나야 합니다.

여기서 GET이 Idempotent하도록 설계되었다는 것은 GET으로 **서버에게 동일한 요청을 여러 번 전송하더라도 동일한 응답이 돌아와야 한다는 것**을 의미합니다.

GET은 설계원칙에 따라 서버의 데이터나 상태를 변경시키지 않아야 Idempotent하기 때문에 **주로 조회를 할 때에 사용**해야합니다.

반대로 POST는 Non-idempotent하기 때문에 **서버에게 동일한 요청을 여러 번 전송해도 응답은 항상 다를 수 있습니다**. 이에 따라 POST는 서버의 상태나 데이터를 변경시킬 때 사용됩니다.

**쿠키와 세션**

쿠키는 클라이언트 **로컬에 저장되는 키와 밸류가 들어있는 저용량 데이터 파일이다.**

쿠키는 **일정시간**동안만 데이터를 저장할 수있다. 그래서 로그인 상태유지 같은곳에서 쓰인다.

쿠키는 클라이언트의 상태 정보를 로컬에 저장하였다가 참조한다.

세션은 세션은 클라이언트와 웹서버 간 네트워크 연결이 지속 유지되고 있는 상태를 말합니다.

쿠키를 기반으로 하고 있지만, 사용자 정보 파일을 브라우저에 저장하는 쿠키와 달리 세션은 서버 측에서 관리.

(다시말해 웹 브라우저를 통해 웹서버에 접속한 이후부터 그 브라우저를 종료 할떄까지 상태 유지)

**쿠키와 세션의 차이(\*\*\*)**

1.저장위치 쿠키 : 클라이언트에 파일로 저장 세션 : 서버에 저장

2.보안 쿠키 : 클라이언트 로컬에 저장되기 떄문에 변질되거나 request에서 스나이핑당할 우려가 있어서 보안 취약 세션 : 쿠키를 이용해서 세션id만 저장하고 그것으로 구분해서 서버에서 처리하기 떄문에 비교적 안전 (보안 면에서 쿠키보다 우수)

**3.라이프 사이클(\*\*) 쿠키 : 만료시간은 있지만 파일로 저장되기 떄문에 브라우저를 종료해도 계속해서 정보가 남아 있을수 있음. 만료기간을 넉넉히 잡아두면 쿠키삭제 할 때까지 유지될수도 있음. 세션 : 만료기간을 정할수는 있지만 브라우저가 종료되면 그 에 상관없이 삭제됌.**

4.속도 쿠키 : 쿠키에 정보가 있기 떄문에 서버에 요청시 속도가 빠르다. 세션 : 정보가 서버에 있기 떄문에 처리가 요구되어 비교적으로 느림