0️⃣

Chapter 0. 서버 처음 해보기

📝 학습 목표

1. 클라이언트와 서버가 데이터를 주고받는 것을 이해한다.
2. Web Server와 WAS의 차이를 이해한다.

📸 잠깐 ! 스터디 인증샷은 찍으셨나요?📸

* 스터디리더께서 대표로 매 주차마다 한 장 남겨주시면 좋겠습니다!👰💗
(사진을 저장해서 이미지 임베드를 하셔도 좋고, 복사+붙여넣기해서 넣어주셔도 좋습니다!)

📄 0주차 주제

서버에 대해 처음 시작하는 여러분들이 가장 먼저 드는 의문은 보통 서버란 무엇인가?에 대한 내용일 것입니다.

그에 대한 가장 간단한 대답은

서버란 클라이언트의 요청에 대해 **적절한 응답**을 **주는 것**이라고 답할 수 있을 것입니다.

(더 정확히 서버를 설명하기 위해서는 소켓 프로그래밍, IPC 등 더 많은 내용들이 필요하지만 0주차에 담기에는 내용이 너무 많을 것 같아 **부록. 서버란 무엇인가(소켓&멀티 프로세스)**에 담아놓았으니 필요하신 분들은 확인하여주세요! **(부록은 필수가 아닙니다!)**)

위 문장의 포인트는 두 가지가 있는 데 첫 번째는 어떻게 **요청과 응답을 주는 것**인지, 두 번째는 **적절한 응답을 찾는 것**일 것입니다.

두 번째에 대한 내용은 2주차에서 배울 내용인 SQL과 그 이후 주차에서 함께 알아볼 것 이구요.

이번 주에 배워볼 내용은 클라이언트와 서버가 어떻게 통신을 하는 것인지, 더 나아가서는 그 응답을 주는 서버는 어떠한 종류를 가지고 있는지에 대해 배워보고 마지막에는 직접 로컬 서버를 띄워서 요청을 보내고 응답을 받는 것까지 해보겠습니다.

🤌 이번 주에 익힐 내용은..

클라이언트와 서버가 어떻게 통신하는 가 그리고 서버의 종류에 대해 배워볼 예정입니다.

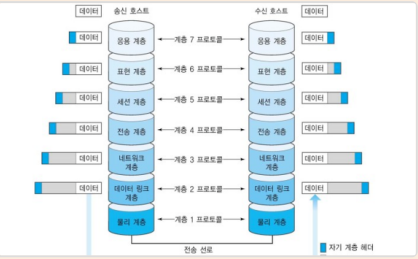
스터디 시작 전 꼭 읽어보기

▼ TCP/IP 계층

OSI 7계층과 TCP/IP 4계층 모델

OSI 7계층과 TCP/IP 4계층 모델 우리가 우리의 컴퓨터에서 다른 사람의 컴퓨터로 통신할 때, 네트워크 통신을 하게 된다. 모든 컴퓨터는 네트워크 통신을 할 때

https://ariz1623.tistory.com/327



본격적인 설명 전 Layered Architecture, OSI 7계층, TCP/IP 5계층을 학습하고 와주세요.

(본문은 위 블로그에서 나오는 내용인 TCP/IP 4계층 중 첫 계층인 NetWork Interface Layer를 Physical Layer와 Data Link Layer로 나눈 5계층을 기준으로 설명해 보겠습니다.)

0주차 본문

이번 주의 목표 중 하나는 클라이언트와 서버가 어떻게 서로의 정보를 주고받는가에 대해 아는 것입니다.

이를 위해 기본적으로 인터넷이 무엇인지, 어떻게 구성이 되는지 그리고 클라이언트와 서버가 어떻게 통신하는지를 알아야 합니다.

인터넷

인터넷이란 Data를 전달하는 장치들이 이루는 거대한 network 망을 의미하며 Application에게 통신 서비스를 제공하는 존재를 의미합니다. (요약하자면 데이터를 전달해주는 망이라고 생각해 주시면 될 것 같아요!)

인터넷 구성

인터넷은 여러 형태의 network와 그 안에 sub-network로 구성이 됩니다. 또한 network는 다양한 entity인 Network Edge, Access network, Network Core 등으로 이루어져 있습니다.

Network Edge

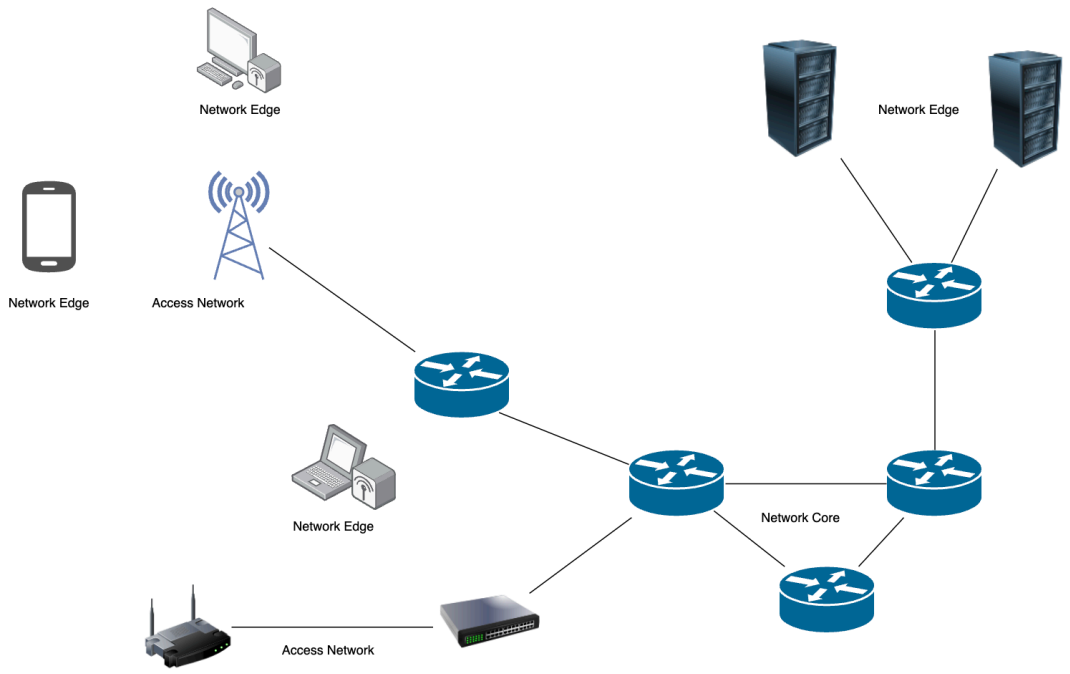
Network Edge란 끝에 있는 entity를 의미하며 End System으로 구성이 되어있습니다. 대표적인 End System으로는 web browser, email client, 스마트폰 그리고 우리가 앞으로 개발할 server 등 실제 application이 여기에 포함이 됩니다.

Access Network

Access Network란 End system이 Internet의 첫 부분과 연결되는 구간으로 대표적으로 우리가 인터넷에 접속할 때 사용하는 랜선, 와이파이 등이 있습니다.

Network Core

Network Core란 Network의 핵심 부분으로 End system의 정보를 실어 나르는 역할을 합니다. 대표적으로 router가 있습니다. (router는 받은 데이터를 적절히 전달하는 역할을 합니다.)



각 용어들을 자세히 외울 필요는 없고 간단하게 인터넷의 구성하는 요소들에는 이러한 것들이 있고 각 구성요소들이 어떠한 역할을 하는구나 정도만 이해해 주시면 될 것 같습니다.

인터넷 통신

이제 인터넷이 어떠한 것으로 구성이 되는지 알아보았으니 본격적으로 어떻게 클라이언트와 서버가 서로 통신을 하는지 알아보겠습니다.

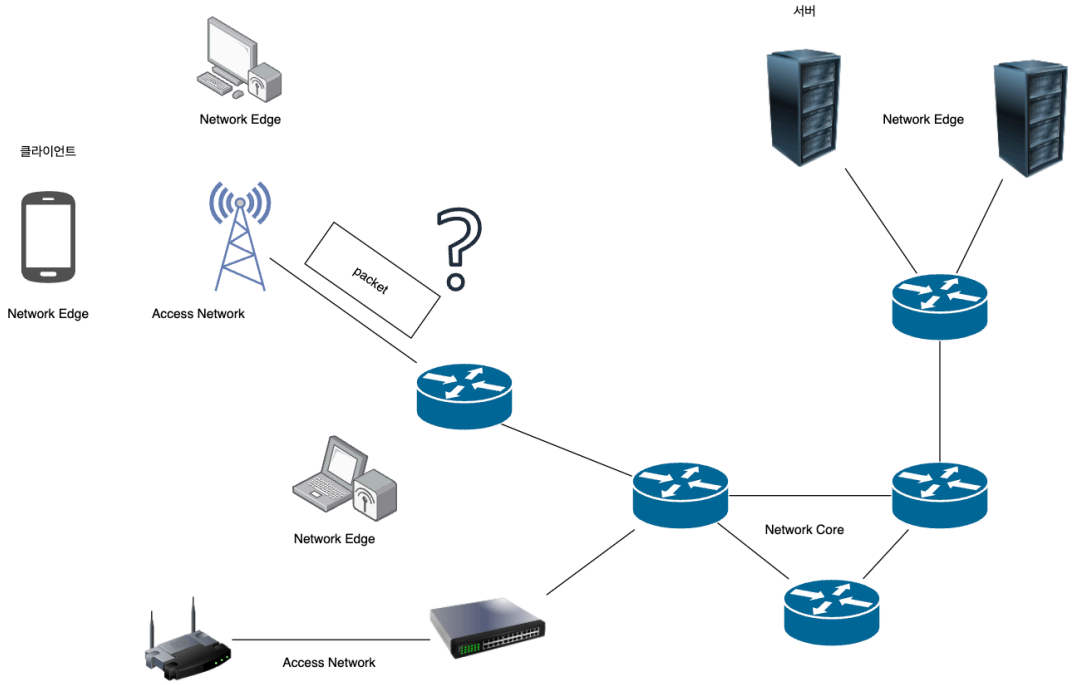
인터넷 통신을 간단히 말하자면 end system 간 패킷(=Datagram)을 주고받는 것이라고 말할 수 있습니다.

Packet(=Datagram)

Packet은 인터넷상에서 장치들이 서로 통신할 때 전송하는 데이터 조각입니다.

간단하게 설명하자면 상대에게 전송할 데이터가 담긴 것이라고 생각하시면 될 것 같습니다. (TCP, UDP 인지에 따라 IPv4, IPv6에 인지에 따라 형태가 달라지기 때문에 정확한 형태는 게시하지 않겠습니다.)

근데 전 세계에는 정말 다양한 기기들이 있는데 어떻게 상대 기기를 식별하여서 패킷을 보낼 수 있는 것일까요?



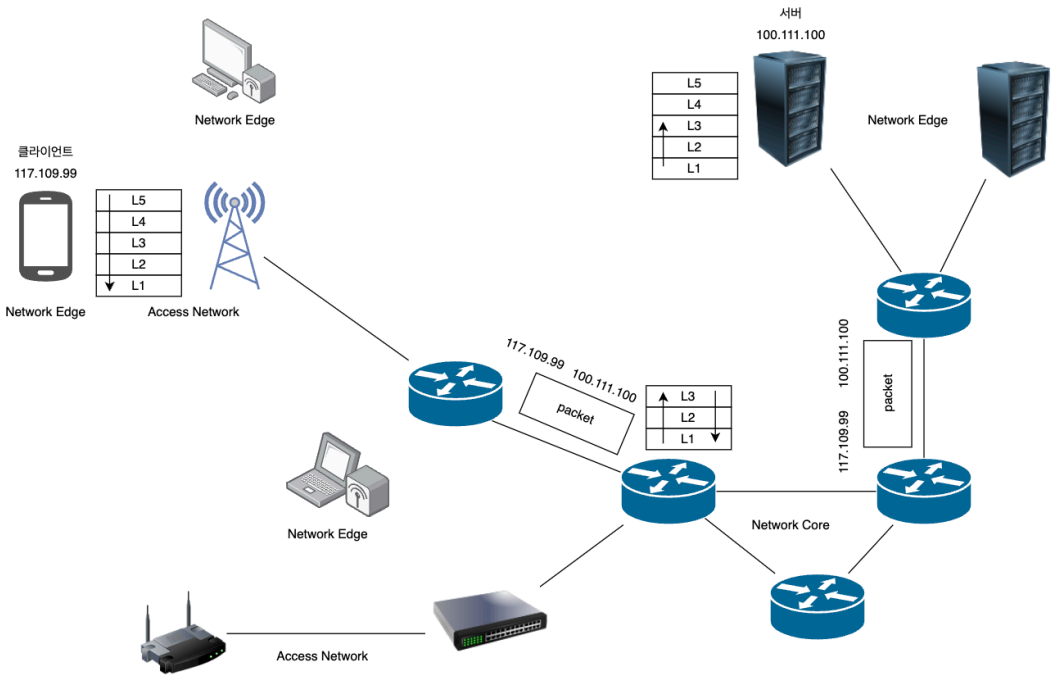
이에 대해 알기 위해서는 **IP**에 대한 개념이 필요합니다.

IP

IP란 Internet Protocol의 약자로 Network Layer에서 작용을 하며 인터넷상에서 유일하게 상대를 식별할 수 있는 수단입니다. (간단하게 말하자면 인터넷상의 주소라고 생각하시면 될 것 같아요)

클라이언트(end system)는 요청을 보낼 때 패킷에 IP 주소를 담고 Access Network를 통해 Network core로 보냅니다.

패킷을 받은 router는 자신의 Network Layer에서 패킷 안에 있는 IP를 확인 후 더 알맞은 위치의 router로 보내게 되고 그렇게 최종적으로 Network Edge에 도착하면 Network Layer에서 패킷을 확인 후 도착지의 IP가 자신의 IP가 옳다면 Transport Layer로 올리고 아니라면 다른 곳으로 보내게 됩니다.



(위의 그림은 번호가 0.0.0 이런 식으로 나누었는데 IPv4가 가장 많이 쓰이기 때문에 실제로 프로젝트를 할 때는 0.0.0.0 이런 식의 형태가 많이 나올 것입니다.)

위에 동작들을 보면 router들이 다른 정보 없이 패킷의 정보만을 해석하여 목적지를 추적한다는 것을 알 수 있습니다.

위와 같은 방식으로 패킷이 이동을 한다면 다른 기기의 정보를 알 수 없기 때문에 여러 가지 문제점을 생각할 수 있을 겁니다.

1. **만약 상대의 컴퓨터가 꺼져있다면?**
2. **만약 중간에 패킷이 소실이 된다면?**
3. **뒤에 보낸 패킷이 먼저 도착한다면?**
4. **컴퓨터에 프로그램이 여러 개가 켜져 있다면?**

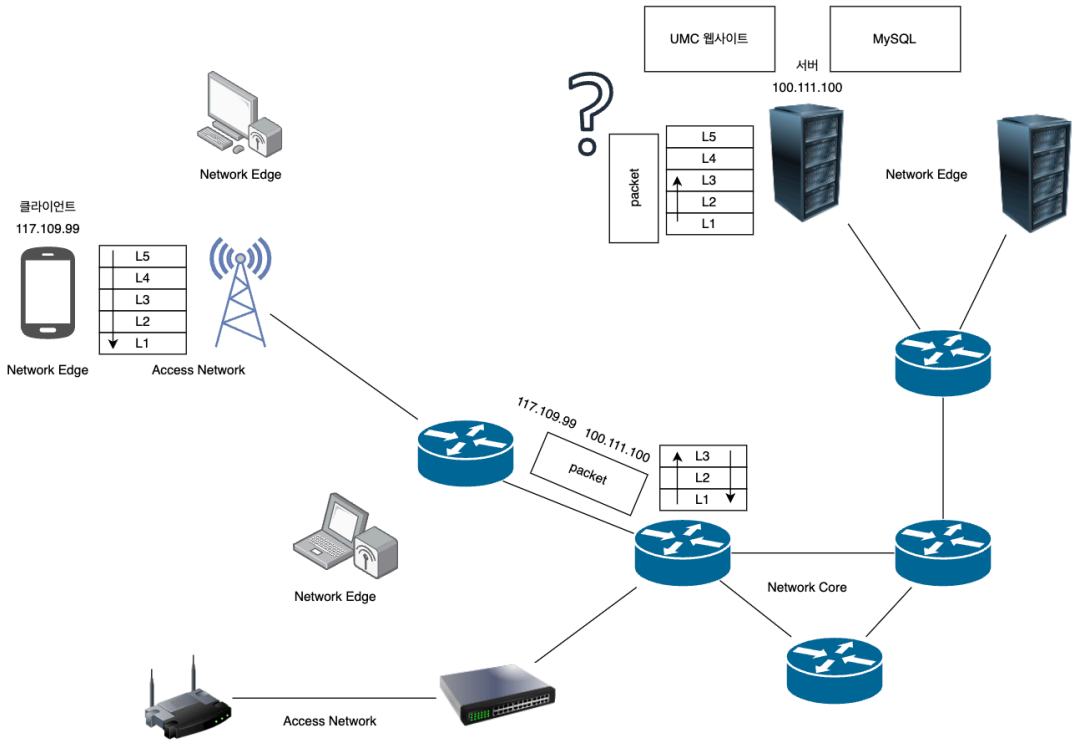
이를 해결하기 위해서는 PORT와 TCP, UDP가 필요합니다.

PORT

(원래는 TCP, UDP 프로토콜 안에 PORT가 포함되는 것인데 PORT를 먼저 설명하는 것이 더 쉬울 것 같아 먼저 설명하도록 하겠습니다.)

문제 4번처럼 UMC 웹사이트와 MySQL이라는 프로그램이 같이 켜져 있다고 가정해 봅시다.

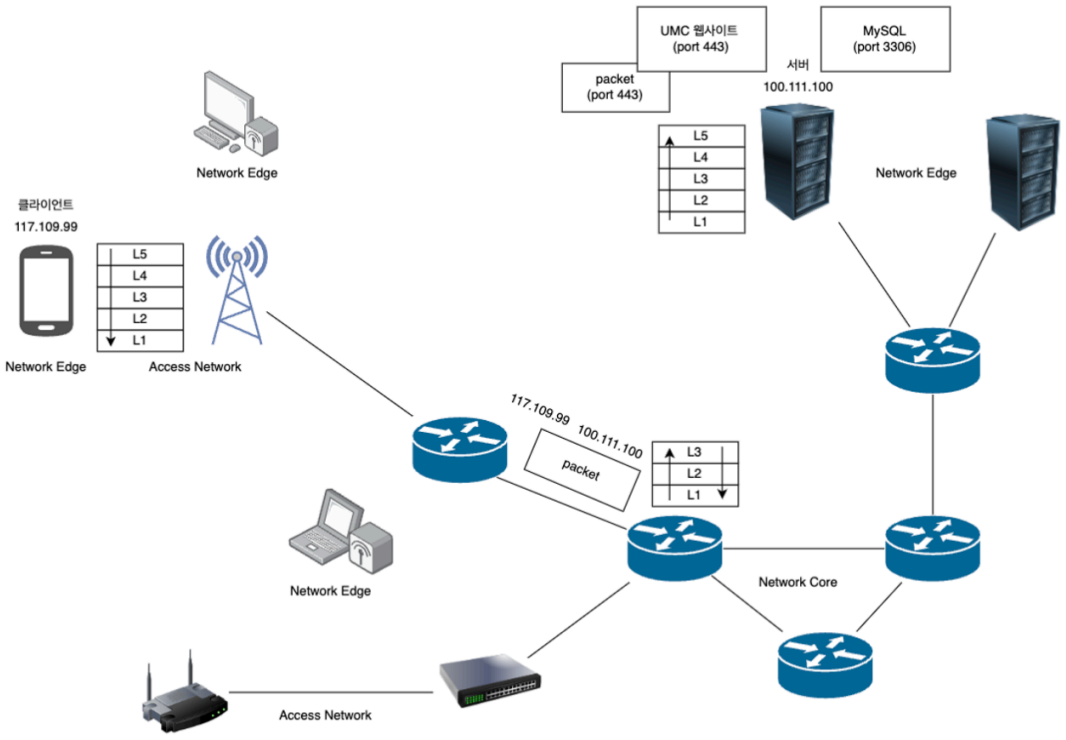
Network Layer에서 자신의 IP 임을 확인하였지만 어느 프로그램으로 packet을 이동시켜야 하는지 알 수 없습니다.



이때 필요한 것이 PORT입니다.

PORT란 같은 IP 내에서 프로세스를 구분하는 데 사용하는 번호를 의미합니다. 대표적으로 HTTP의 80 HTTPS의 443 등이 존재합니다.

위의 상황에서 PORT 번호를 적용하여서 본다면 밑의 그림처럼 해결이 가능합니다. (실제 웹사이트인 https://neordinary.co.kr:443 을 눌러 보시면 neordinary 웹사이트로 연결되는 모습을 볼 수 있습니다. (우리가 보는 원래 웹사이트에도 포트 번호가 존재하는 데 http의 80이나 https 443은 이미 정해져있기 때문에 생략한 것입니다.))



위 그림으로 보자면 UMC 웹사이트의 주소는 https://100.111.100:443 일 것이고 이 중 100.111.100은 IP, 443은 PORT가 됩니다.

참고로 https://neordinary.co.kr:443 주소에서 neordinary.co.kr은 도메인(IP주소를 우리가 보기 쉽게 문자 등으로 바꾼 것)주소이고 443이 포트 입니다!

TCP

아직 1, 2, 3번 문제가 남았습니다.

이 문제들을 해결 가능한 것이 TCP입니다.

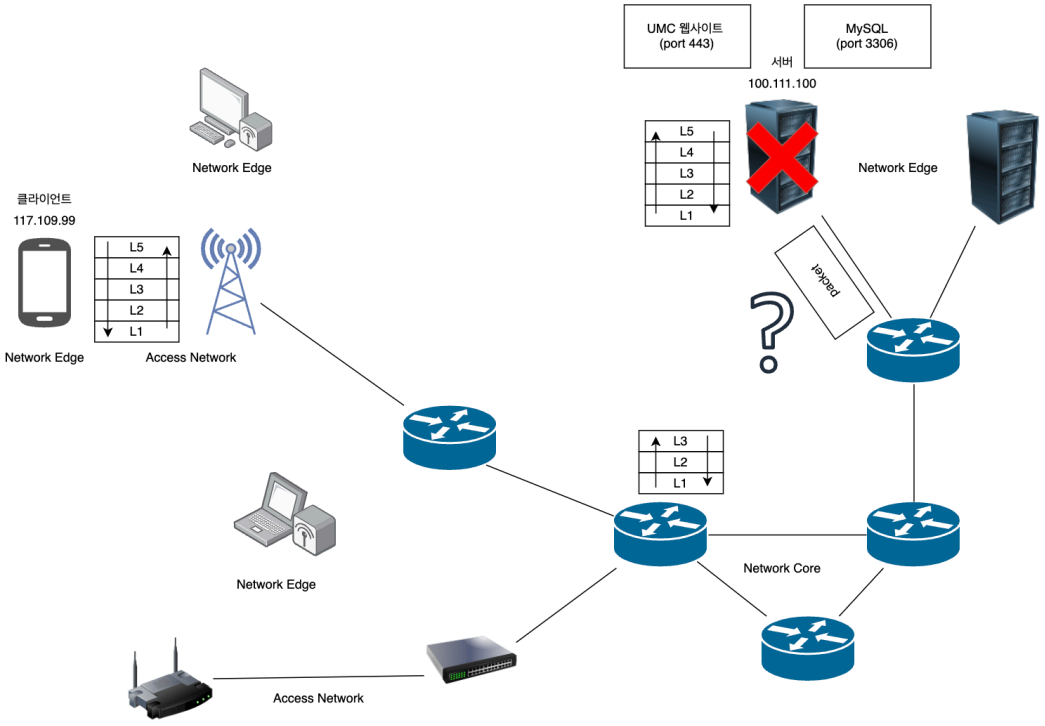
TCP란 Transmission Control Protocol(전송 제어 프로토콜)의 약자로 연결 보증을 해줌으로써 위의 1, 2, 3번 문제를 해결할 수 있습니다. (Transport Layer에서 사용이 됩니다.)

3 way-handshake

1번 문제처럼 상대방의 컴퓨터 서버가 꺼져있다고 가정해 보겠습니다.

보내는 입장에서는 packet에 도착지의 IP와 PORT만 넣어서 보내고 router는 IP와 PORT를 보고 목적지로 보냅니다.

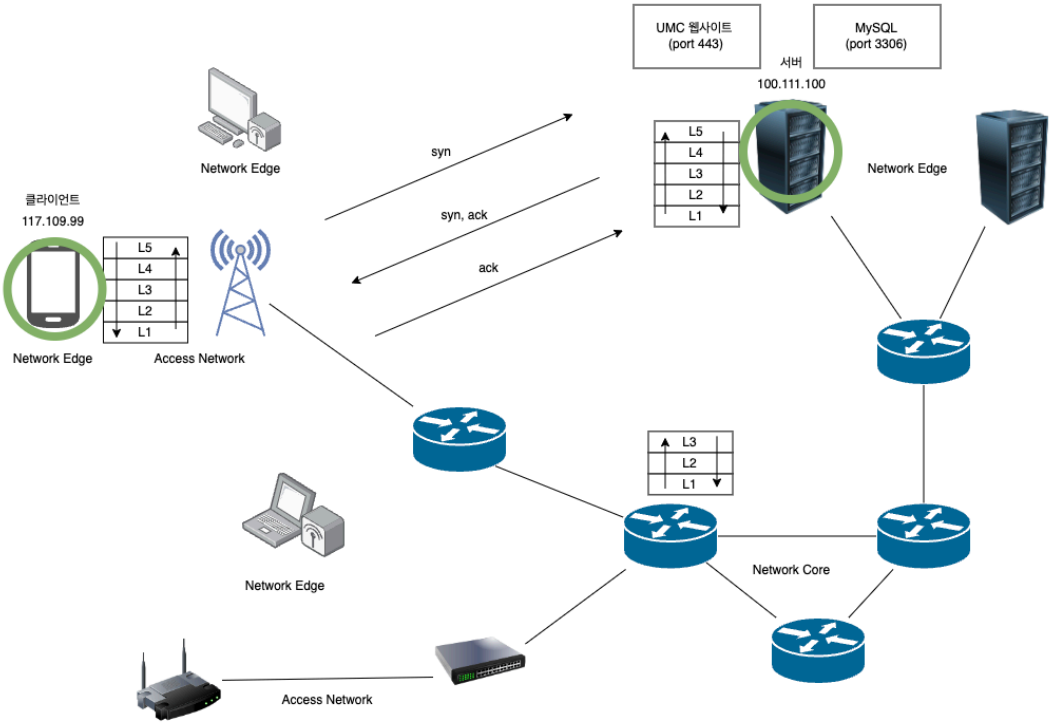
하지만 목적지의 컴퓨터는 꺼져있기 때문에 메시지를 받을 수 없습니다.



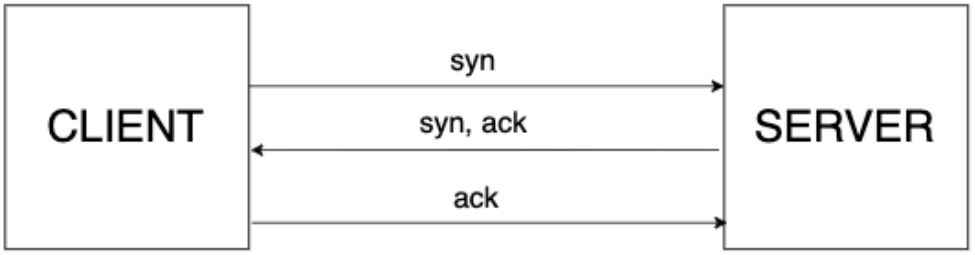
그렇기에 데이터를 보내기에 앞서 클라이언트는 상대방이 있는지 확인해야 합니다.

그때 사용하는 것이 3 way-handshake입니다.

클라이언트는 받는 곳에게 syn을 쏘고 그것을 받은 상대방은 받을 수 있는 상태라는 ack와 보내는 쪽도 받는 것이 가능한가를 물어보는 syn를 보냅니다. 클라이언트가 syn을 받고 온전한 상태라는 ack를 보내게 되면 3 way-handshake가 완성이 되며 서로 받을 수 있는 상태라는 것이 보증이 되게 됩니다. (syn와 ack는 서로의 상태가 괜찮은 지를 묻고 답하는 거라고 생각하시면 될 것 같아요!)



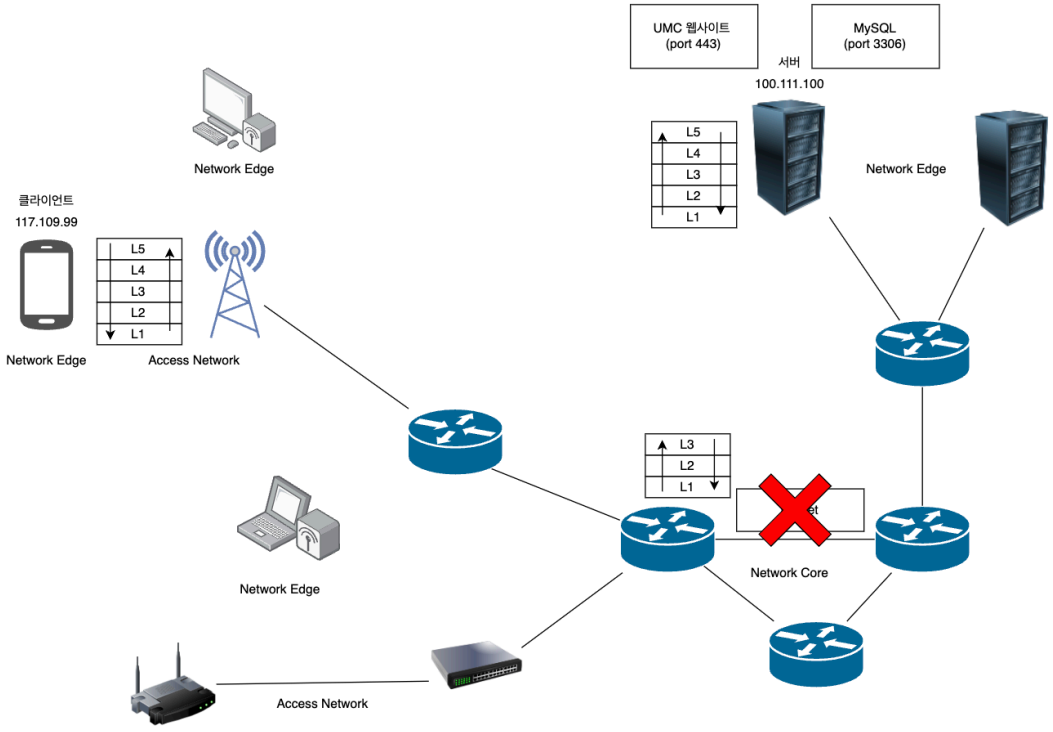
실제로는 router를 타고 가야 하지만 보기 쉽게 화살표로 표시하였습니다.



3 way handshake만 요약하면 이렇습니다.

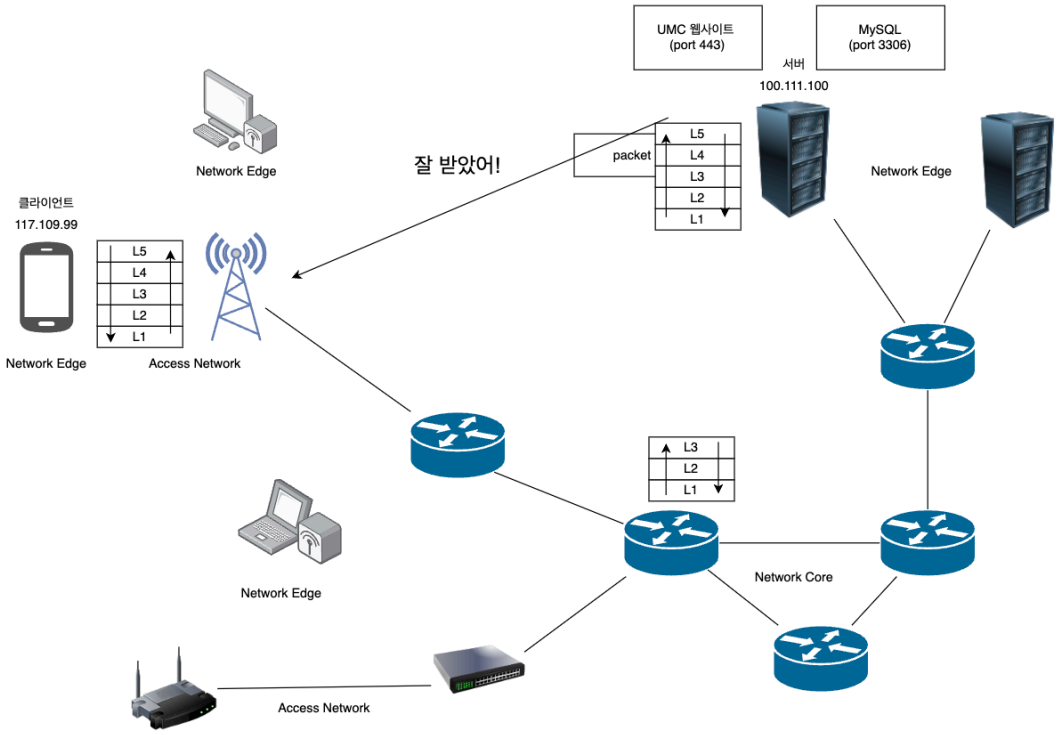
데이터 전달 보증

2번 문제처럼 Packet이 중간에 소실되어 있다고 가정해 보겠습니다.



보내는 입장이나 받는 입장에서는 패킷이 소실되었는지 아직 도착을 안한 건지 알 수 없습니다.

그래서 받았음을 알려줌으로써 이를 해결합니다.

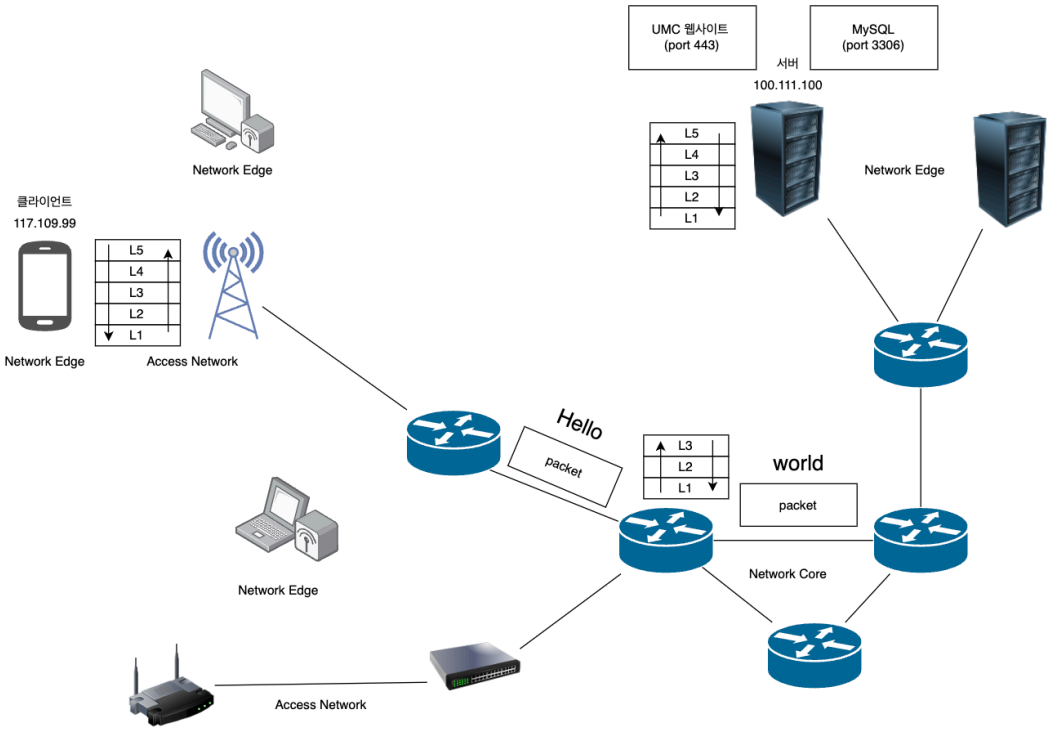


순서 보장

보통 Packet을 보낼 때 일정량 이상이 된다면 끊어서 보냅니다.

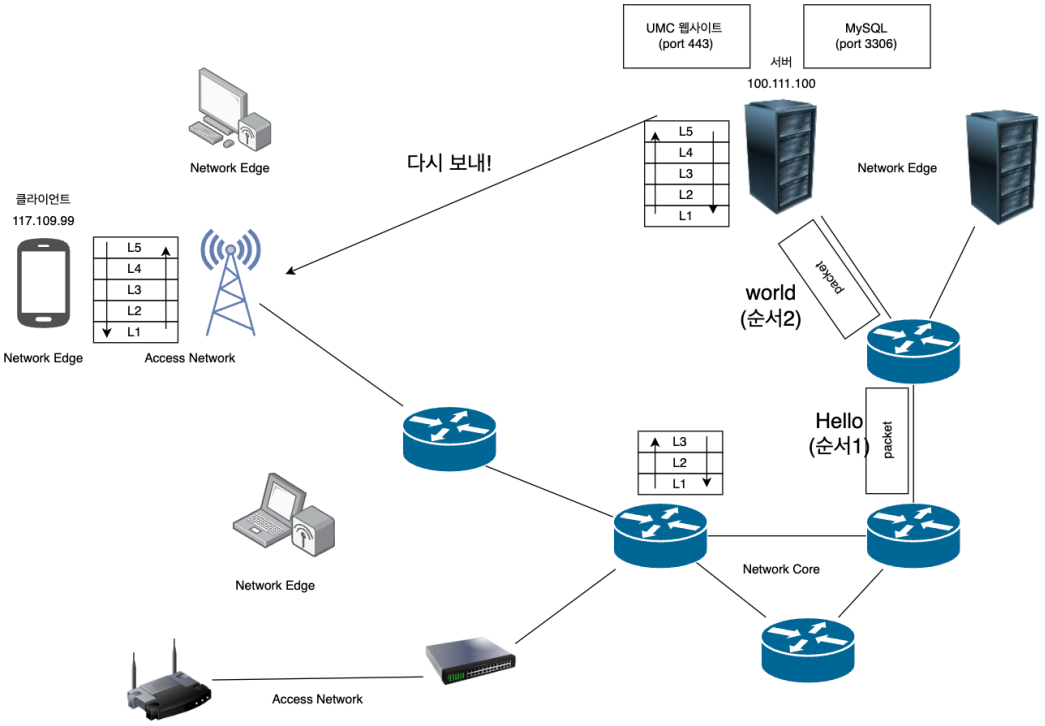
이때 3번 문제처럼 뒤에서 보낸 packet이 먼저 도착할 수도 있습니다.

예를 들어 Hello World라는 정보를 서버로 보내는 데 크기가 너무 커서 Hello와 World로 나누어서 보낸다고 가정해 봅시다. 그때 만약 순서가 뒤집힌다면 올바른 정보가 가지 못할 것입니다.



TCP는 패킷에 sequence number라는 순서를 붙임으로써 이 문제를 해결합니다.

도착한 패킷의 sequence number를 분석해서 만약 순서가 이상 하다면 다시 보내달라고 하여서 문제를 해결합니다.

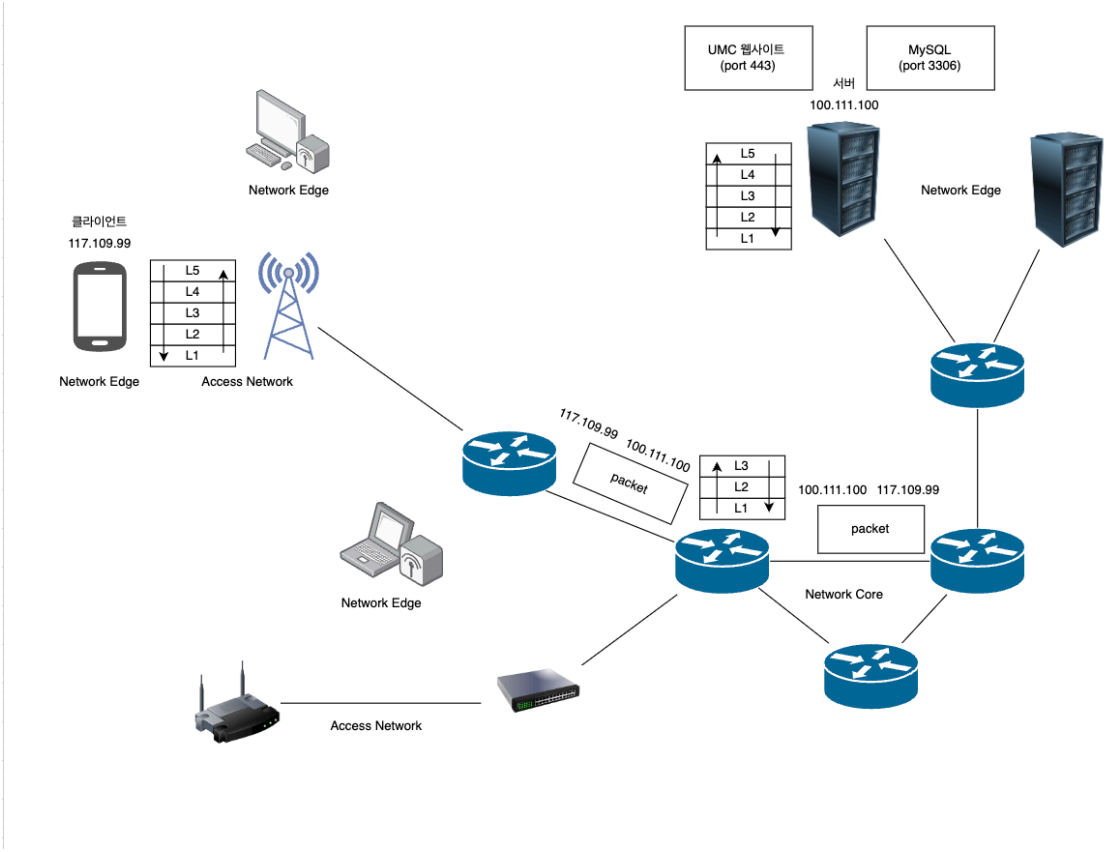


UDP

UDP란 User Datagram Protocol(사용자 데이터그램 프로토콜)의 약자로 TCP와는 달리 3 way handshake나 데이터 전달 보증, 순서 보장 없이 기존의 IP에 PORT와 체크 섬(데이터가 맞는 지만 확인)만 추가한 프로토콜입니다. TCP에 비해 신뢰성이 떨어지지만 검증하는 부분이 적어지기에 그만큼 더 빠르다는 장점이 있습니다.

(UMC에서는 웹 서버가 많이 사용하는 TCP를 주로 다루기에 UDP의 더 정확한 설명은 하지 않겠습니다.)

전체적인 흐름을 요약하자면 클라이언트가 서버에게 3 way handshake를 보내고 연결이 되면 패킷에 데이터와 IP, PORT 등 여러 정보를 넣어서 Access Network를 통해 인터넷 망으로 보냅니다. 그 후 Network Core로 들어가면 router의 network Layer에서 패킷의 IP 주소를 보고 다음 router로 보내고 마침내 Network Edge에 도달하게 되면 Physical Layer, Data Link Layer를 거쳐서 Network Layer에서 IP를 확인 후 옳다면 Transport Layer에서 PORT 번호를 확인 후 알맞은 애플리케이션으로 보내서 처리하게 됩니다. (그 후 응답을 해야한다면 다시 서버에서 클라이언트로 패킷을 보내게 됩니다.)



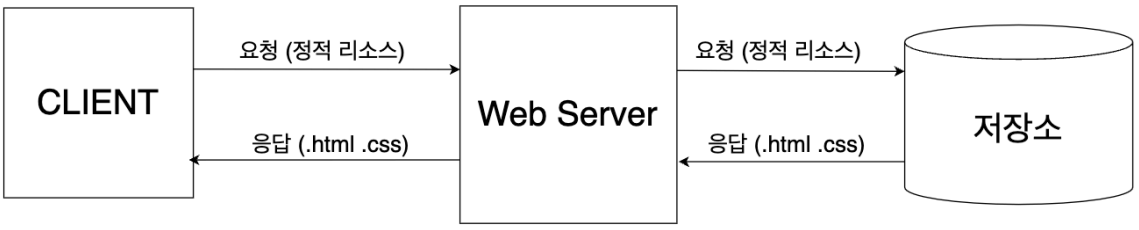
첫 번째 목표인 서버란 무엇인지 어떠한 일을 하는지 그리고 어떻게 통신하는지를 알아보았으니 이제 두 번째 목표인 서버의 종류에 대해 알아보겠습니다.

Web Server와 WAS

서버의 종류를 크게 나누자면 정적 리소스를 처리하는 Web Server와 동적 리소스를 처리 가능한 WAS가 존재합니다.

Web Server

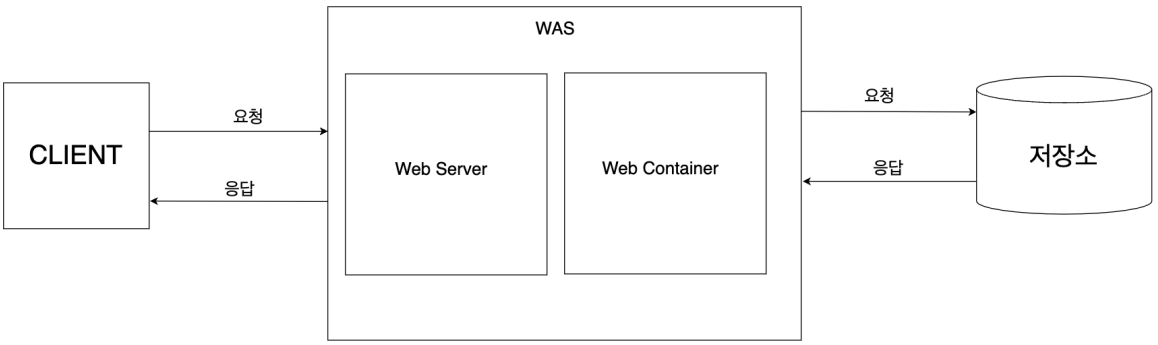
Web Server란 위에서 설명하였듯이 정적 리소스를 처리해 주는 서버를 말합니다. (정적 리소스란 HTML, CSS, 이미지처럼 정적인 자원을 의미합니다.)



대표적으로 Apache, Nginx 등이 있습니다.

WAS

WAS란 동적 리소스를 처리해 주는 서버를 말합니다. (동적 리소스란 DB 조회나 다양한 로직 처리를 하는 것을 의미합니다.)



참고로 WAS에는 Web Server가 포함되어 있기에 정적 리소스까지 처리할 수 있습니다.

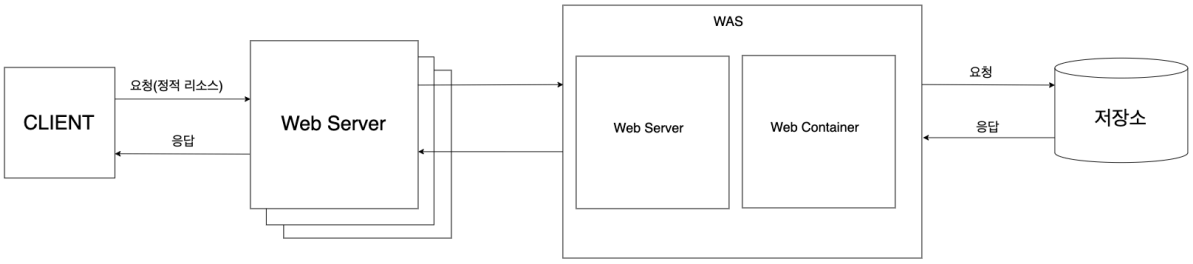
대표적으로는 Spring Boot의 내장 서버인 Tomcat이 있습니다.

한편 WAS는 정적 리소스까지 처리 가능하다고 하였는데 Web Server는 오늘날까지 많이 쓰이고 있습니다. 왜 그러한 것일까요?

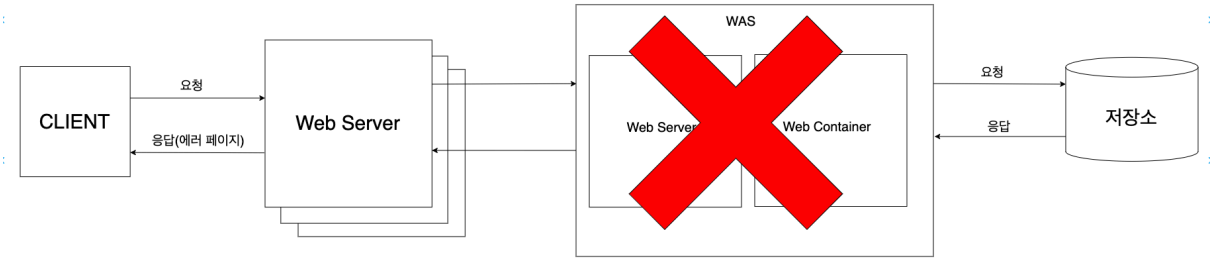
Web Server와 WAS를 같이 쓰는 이유

기본적으로 WAS는 Web Server에 비해 더 비싸고 에러가 많이 납니다.

그래서 정적 리소스를 많이 사용할 때는 Web Server만 늘리는 방식으로 절약할 수 있습니다.



또한 만약 WAS가 에러가 날 시 Web Server는 요청을 WAS로 보내지 않고 에러 페이지를 띄우게 할 수 있습니다.



이외에도 Web Server에서 Web Server에서 추가로 보안 처리를 할 수 있는 등 많은 장점이 있어서 Web Server를 WAS 앞단에서 많이 사용합니다.

Web Server에서 WAS로 넘어가는 흐름을 이해하기 위해서는 reverse proxy라는 개념이 필요한데 그것은 **부록. Web Server & Web Application Server(WAS), Reverse Proxy에** Web Server, WAS의 더 자세한 내용과 함께 넣어 놓았으니 필요하신 분들은 참고하여 주세요. **(부록은 필수가 아닙니다!)**

실습

🌱 Spring Boot 실습

> 💡 Spring Boot 서버를 처음 시작해보는 간단한 실습입니다!

🌱 Chapter 0. 서버 처음 해보기 - Spring Boot 실습

📦 Node.js 실습

> 💡 Node.js 서버를 처음 시작해보는 간단한 실습입니다!

📦 Chapter 0. 서버 처음 해보기 - Node.js 실습

🎯핵심 키워드

> 💡 주요 내용들에 대해 조사해보고, 자신만의 생각을 통해 정리해보세요!
> 레퍼런스를 참고하여 정의, 속성, 장단점 등을 적어주셔도 됩니다.
> 조사는 공식 홈페이지
> **Best**, 블로그(최신 날짜) **Not Bad**

▼ IP

- Internet Protocol

- 인터넷 프로토콜을 사용하여 컴퓨터 네트워크 내에서 상호 연결된 각 장치에 지정된 숫자 레이블
- 이러한 IP주소는 데이터 채킷을 의도한 목적지로 라우팅하기 때문에 인터넷에서 디바이스 간의 통신을 원활하게 하는 데 필수적인 요소
- <기능>
    1. 데이터 라우팅 : 데이터를 올바른 경로로 전송하기 위해 라우터로 데이터를 전달함.
    2. 패킷화 : 데이터를 패킷으로 나눠 전송한 뒤, 수신 측에서 다시 조합함.
    3. 유니크한 주소 부여 : 각 장치에 고유한 IP 주소를 할당해 식별할 수 있게끔 함.
- <유형>
    1. 공개/비공개
       1.1. 공개 : 공인 IP 주소는 컴퓨터를 해당 ISP와 나머지 인터넷에 식별함. 비공개 네트워크의 공유 액세스 포인트 역할을 할 수 있음.
       1.2. 비공개 : 로컬 네트워크 내의 컴퓨터를 식별하는 데 사용
    2. 정적/동적
       2.1. 정적 : 고정된 IP 주소로, 특정 컴퓨터에 반영구적으로 할당됨. 주로 서버나 중요한 장치에서 사용되며 변경되지 않음.
       2.2. 동적 : ISP에 의해 자동으로 할당되며 필요에 따라 변경 가능함.

▼ PORT

- 네트워크에서 데이터를 송수신할 때 특정 소프트웨어나 서비스를 식별하기 위해 사용하는 논리적인 통로
- 네트워크 연결이 시작되고 끝나는 가상 지점으로, 소프트웨어 기반이고 컴퓨터의 운영 체제에서 관리함.
- 각 포트는 특정 프로세스/서비스와 연결됨.
- 포트 번호 : 네트워크에 연결된 모든 장치에서 표준화되며 각 포트에는 번호가 할당됨. 이를 활용하면 해당 장치 내의 특정 서비스 또는 애플리케이션을 대상으로 지정할 수 있음.
- <사용 예>
    1. 브라우저로 웹 페이지를 열 때 : 브라우저는 서버의 IP 주소와 포트 80(HTTP) 또는 포트 443(HTTPS)로 요청을 보냄.
    2. 이메일 송수신 : 이메일 서버는 SMTP(25), IMAP(143), POP3(110)과 같은 프로토콜을 사용함.

- 포트와 방화벽 : 포트는 네트워크 보안과도 밀접한 관련이 있으므로, 이를 이용해 특정 포트를 열거나 닫아 데이터 흐름을 제어함.

▼ CIDR

- Classless Inter-Domain Routing
- 인터넷상의 데이터 라우팅 효율성을 향상시키는 IP 주소 할당 방법
- 보다 유연하게 IP 주소를 할당하고 디바이스 간에 데이터를 라우팅할 수 있음.
- <장점>
    1. IP 주소 낭비 감소 : IP 주소에 할당할 네트워크 및 호스트 식별자를 결정할 때 유연성을 제공함. 라우팅 테이블 항목이 줄어들고 데이터 패킷 라우팅이 단순화됨.
    2. 빠른 데이터 전송 : IP 주소를 보다 효율적으로, 여러 서브넷 구성이 가능해서 불필요한 경로를 사용하지 않고도 데이터가 대상 주소에 도달할 수 있음.
    3. Virtual Private Cloud(VPC) 생성 : VPC는 클라우드 내에서 호스팅되는 프라이빗 디지털 공간이고, 이를 이용해 데이터 패킷을 전송할 때 CDIR IP 주소를 사용함.
- 작동 방식 : 서브넷 마스크를 통해 네트워크와 호스틑 구분함. 이는 IP 주소 뒤에 / 기호와 서브넷 마스크 비트 수를 푸가하여 표기함.

▼ TCP와 UDP 차이

- 데이터를 전송하는 방법에서 근본적인 차이를 가지고 있음.
- TCP : 연결 지향적 프로토콜. 데이터 전송 전에 먼저 연결을 설정함. 데이터 패킷이 손실되거나 순서가 뒤바뀌는 경우, 이를 감지하고 재전송을 요청함으로써 데이터의 정확성을 보장함. ex) 이메일 전송, 웹 페이지 로딩 등
- UDP : 비연결 지향적 프로토콜. 최소한의 통신 매커니즘을 제공해, 데이터의 도착을 보장하거나 순서대로 데이터를 정렬하지 않음. 이 경우, 속도는 빠르지만 네트워크 상태에 따라 데이터 패킷 손실 가능성이 존재함. ex) 실시간 스트리밍, 온라인 게임

▼ Web Server와 WAS의 차이

- 웹 상에서 서비스를 제공하기 위해 사용되는 서버로, 역할과 기능에 차이가 있음.
- Web Server : 정적인 콘텐츠(HTML, CSS, 이미지 등)을 제공하는 서버로, HTTP 프로토콜을 이용해 클라이언트에게 웹 페이지를 제공함. 클라이언트는 URL을 통해 웹 페이지를 요청함. ex) Apache, Nginx 등으로, 홈페이지, 블로그, 사이트 등

- WAS : 동적인 콘텐츠(웹 애플리케이션)를 처리하고 제공하는 서버로, 웹 애플리케이션 실행 및 데이터 처리, 웹 서버와 클라이언트 간의 중계 역할을 수행함. 클라이언트의 요청에 따라 데이터베이스에서 정보를 가져오거나, 웹 애플리케이션을 실행해서 동적인 엡 페이지를 생성한 후 결과를 웹 서버에 전달함. 웹 서버는 이를 받아 클라이언트에게 전달하는 것. ex) Tomcat, JBoss, WebLogic 등으로, 온라인 쇼핑몰, 은행 인터넷 뱅킹, SNS 등

📣 학습 후기

- 이번 주차 워크북을 해결해보면서 어땠는지 회고해봅시다.
- 핵심 키워드에 대해 완벽하게 이해했는지? 혹시 이해가 안 되는 부분은 뭐였는지?

> 💡 실제로 실습을 해보고, 키워드를 정리해보면서 내가 평소에 많이 사용하던 서버와 데이터 전송 개념에 대해 알 수 있게 된 것 같다.

⚠️ 스터디 진행 방법

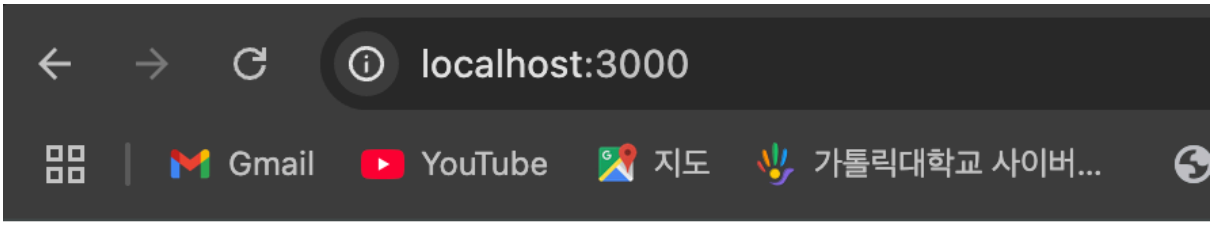
1. 스터디를 진행하기 전, 워크북 내용들을 모두 채우고 스터디에서는 서로 모르는 내용들을 공유해주세요.
2. 미션은 워크북 내용들을 모두 완료하고 나서 스터디 전/후로 진행해보세요.
3. 다음주 스터디를 진행하기 전, 지난주 미션을 서로 공유해서 상호 피드백을 진행하시면 됩니다.

✅ 실습 체크리스트

☑ ~~Node.js 서버 처음 해보기~~

☑️ 실습 인증

1. Node.js 서버 처음 해보기



Good Luck!

🔥 미션

1. 너디너리 홈페이지 접속하는 과정 적어보기 (소켓프로그래밍과 같은 개념 없이 TCP, IP, PORT 등의 개념 등 오늘 배운 내용으로 간단하게 적어주세요)
2. 깃허브 clone 받아서 실행하고 나온 페이지 스크린 샷 찍기

💪 미션 기록

> 🍀 미션 기록의 경우, 아래 미션 기록 토글 속에 작성하시거나, 페이지를 새로 생성하여 해당 페이지에 기록하여도 좋습니다!
>
> 하지만, 결과물만 올리는 것이 아닌, **중간 과정 모두 기록하셔야 한다는 점!** 잊지 말아주세요.

▼ **미션 기록**

[1번]

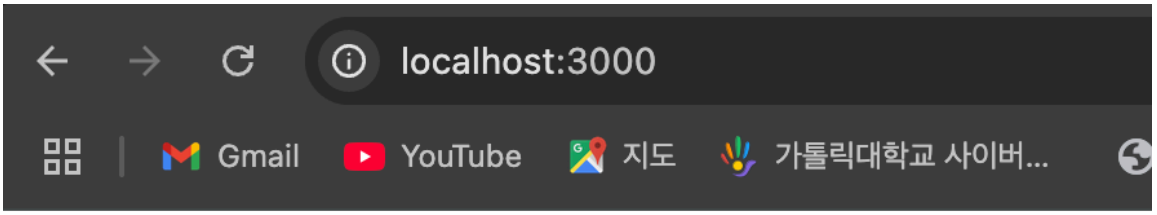
1. 브라우저에서 너디너리 홈페이지 URL (https://neordinary.co.kr:443) 입력
2. 서버에게 3way handshake를 보냄(TCP)
3. 패킷에 데이터와 IP(neordinary.co.kr:IP를 알기 쉽게 변경한 도메인), PORT(443)번호 등 여러 정보를 넣어 access network를 통해 인터넷망으로 전송
4. 이후 router를 거쳐 IP 주소 확인 후, 맞다면 PORT 번호에 맞는 알맞은 서버에 전송

5. 서버는 브라우저에 정보를 전달하고 사용자 화면에 홈페이지 표시

[2번]

1. 본인의 노트북에 git 명령어가 설치되어 있으므로 'git clone https://github.com/sudosubin/umc-7th-nodejs-first-run'를 이용해 터미널에서 Clone 다운로드 받기
2. VScode 터미널에서 'npm install' 명령어로 필요 파일 다운로드 하기
3. 성공적으로 설치된 것이 확인되었으니, 'npm run start'명령어를 통해 서버를 실행하기
4. 브라우저에서 url를 입력, 서버에 접속해 화면 캡처하기

(캡처화면)



Good Luck!

⚡ 트러블 슈팅

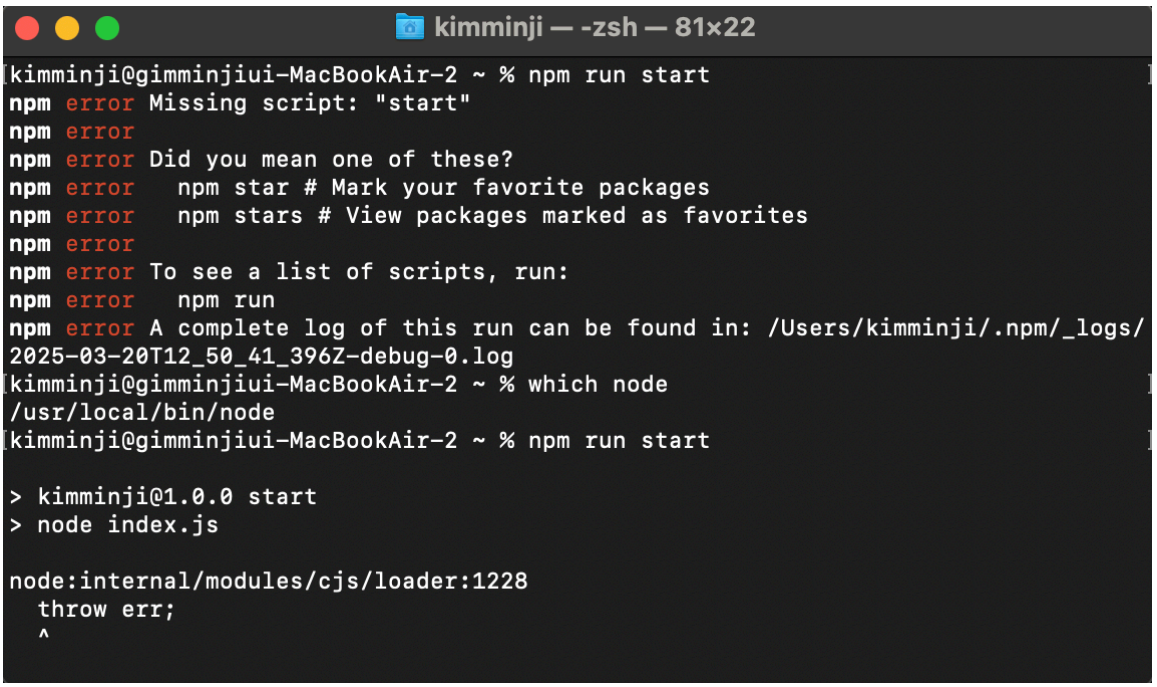
💡 실습하면서 생긴 문제들에 대해서, **이슈 - 문제 - 해결** 순서로 작성해주세요.

💡 스스로 해결하기 어렵다면? 스터디원들에게 도움을 요청하거나 **너디너리의 지식 IN 채널에 질문**해보세요!

▼ ⚡이슈 No.1

이슈

👉 npm run start 명령어를 터미널에 입력하였을 당시, 화면과 같은 오류가 발생함.



문제

👉 실행 시 필요한 package.json 파일 내용에 오류가 있는 것으로 보임.

해결

👉 package.json 파일 내용에 오류가 있어 보여 start 구문에 내용을 추가하였음. 그러나 문제는 해결되지 않았고, 라이브러리를 확인한 결과 package.json 파일이 현재 파일과 외부 파일 두 곳에 존재하는 것을 발견함. 때문에 node.js, clone을 모두 삭제하여 처음부터 다시 설치하였고, 문제가 해결됨.

참고레퍼런스

- https://stackoverflow.com/questions/31976722/start-script-missing-error-when-running-npm-start