TCP/IP와 피지컬 컴퓨팅을 활용한 출입관리 시스템

oralcoder@gmail.com
박주건

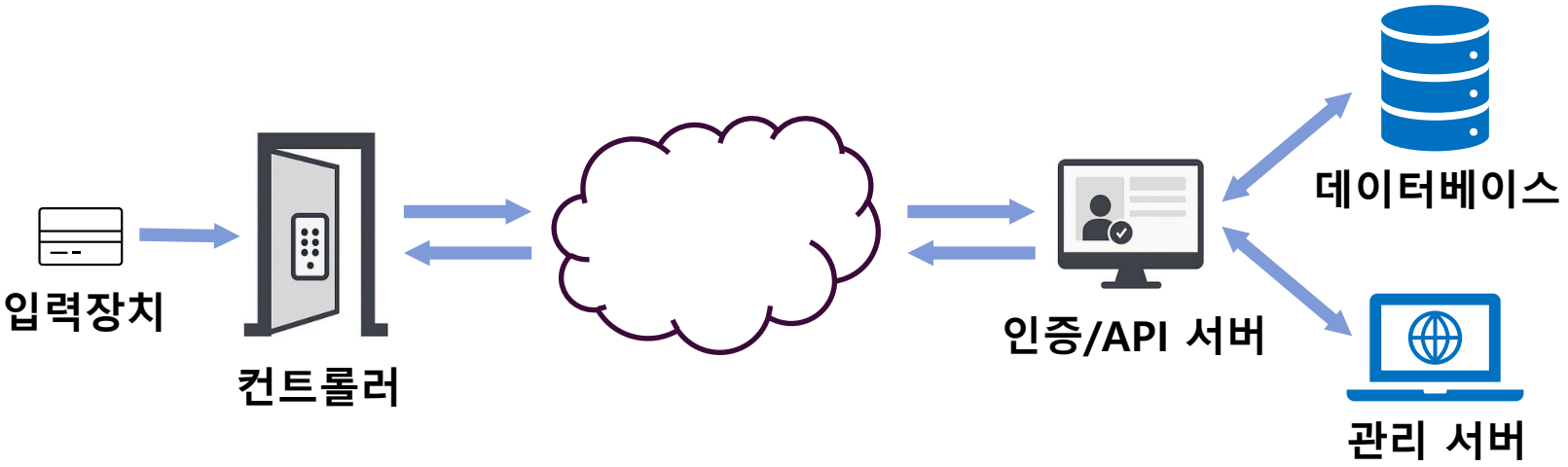
목차

출입관리 시스템 개요

- 특정 공간이나 시설에 대한 사람들의 출입을 체계적으로 통제하고 관리하는 보안 시스템
- 인가된 사람만 출입을 허용하고, 비인가자의 출입을 제한하여 보안과 안전을 확보
- 누가, 언제, 어디에 출입했는지를 기록하고 관리

출입관리 시스템 개요

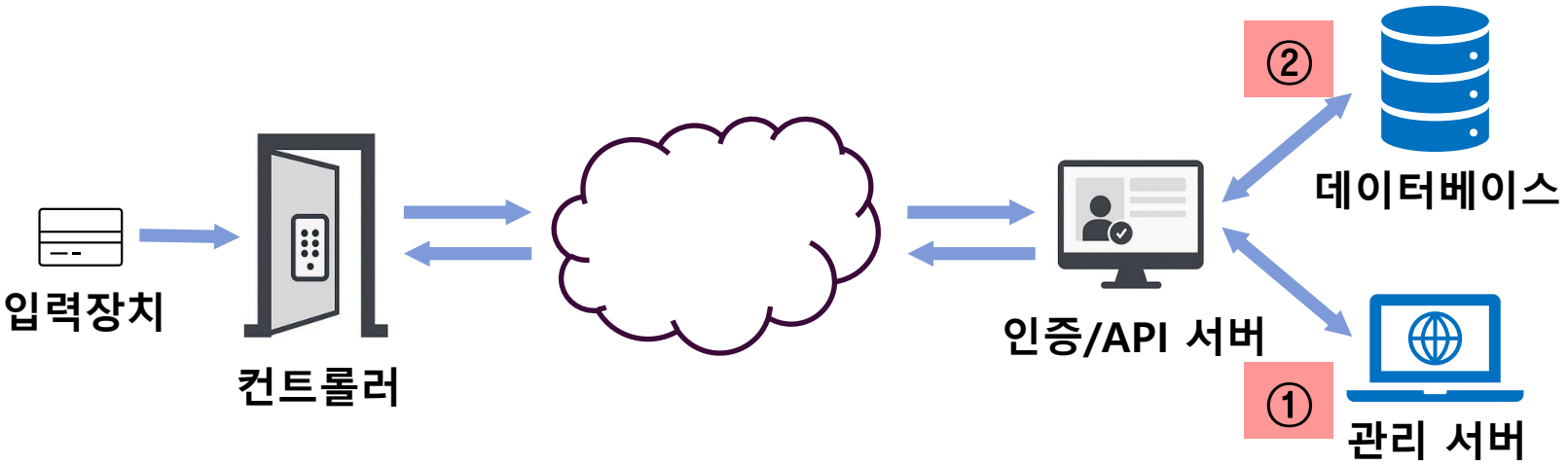
■ 시스템 구성도



출입관리 시스템 개요

■ 시스템 흐름

acc_id	gate_id
A12345	Gate01



① 관리자가 관리 서버(웹)를 통해 새로운 출입 정보 입력
관리 서버는 출입 정보를 인증/API 서버에게 전송

② 인증/API 서버는 관리 서버로부터 수신한 출입 정보를
데이터베이스에 저장

AccID (사용자 ID):
A12345

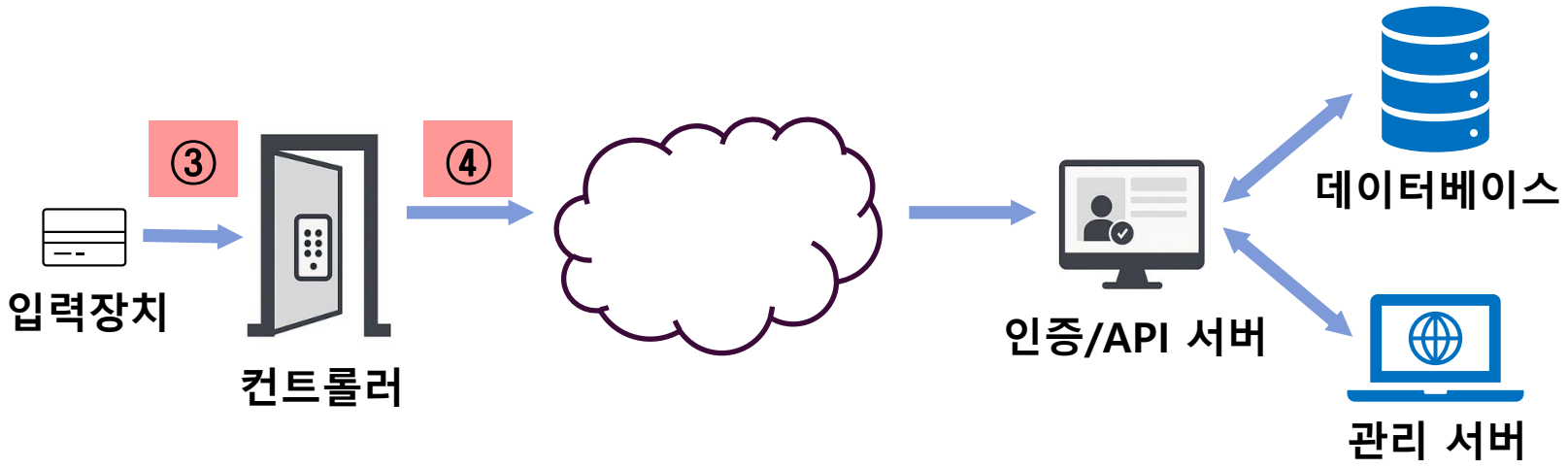
GateID (게이트 ID):
Gate01

전송

출입관리 시스템 개요

■ 시스템 흐름

acc_id	gate_id
A12345	Gate01



③ 사용자가 출입 카드를 태깅
카드에 저장된 acc_id가 컨트롤러에 입력

④ 컨트롤러는 해당 acc_id와 gate_id를 인증/API 서버에 전송

AccID (사용자 ID):
A12345

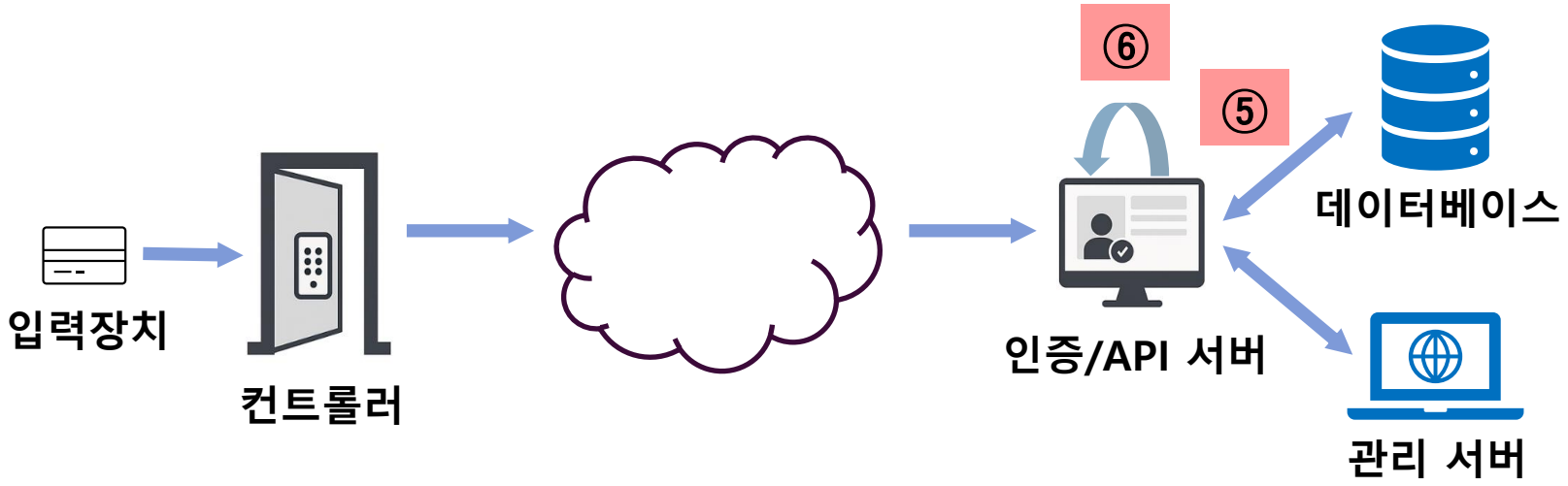
GateID (게이트 ID):
Gate01

전송

출입관리 시스템 개요

■ 시스템 흐름

acc_id	gate_id
A12345	Gate01



⑤ 인증/API 서버는 데이터베이스에 저장된 인증 정보 추출

⑥ 컨트롤러로부터 수신한 acc_id와
데이터베이스로부터 추출한 acc_id를 비교 인증

AccID (사용자 ID):
A12345

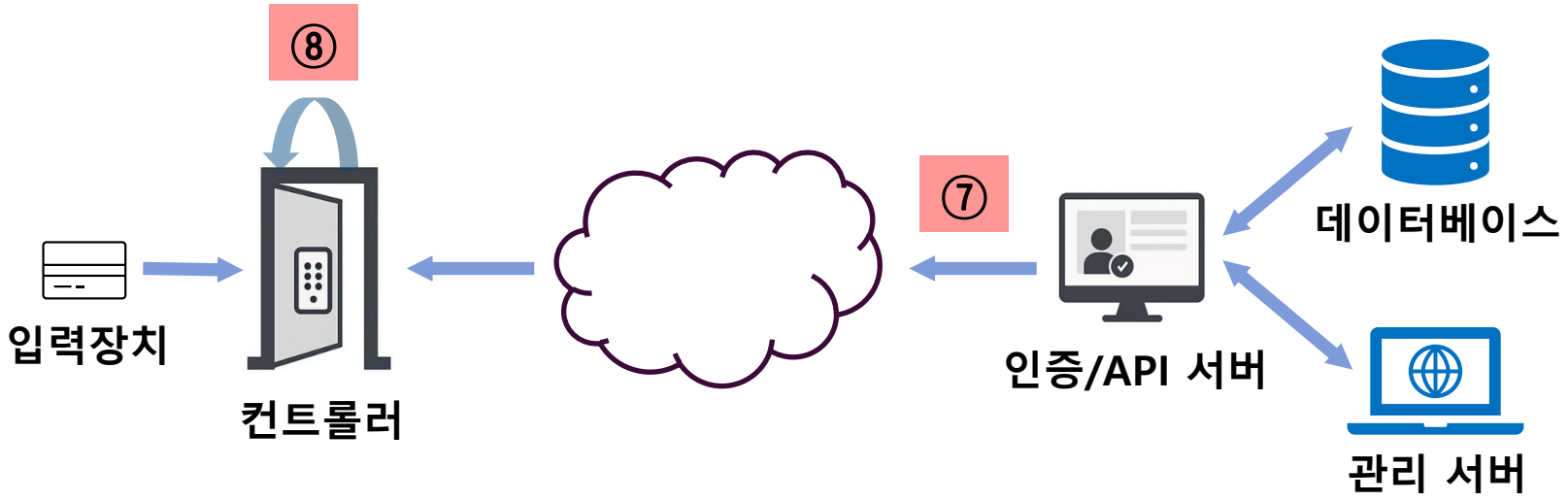
GateID (게이트 ID):
Gate01

전송

출입관리 시스템 개요

■ 시스템 흐름

acc_id	gate_id
A12345	Gate01



⑦ 인증 결과에 따라 개방 여부 전송

⑧ 컨트롤러는 수신한 명령에 따라 제어

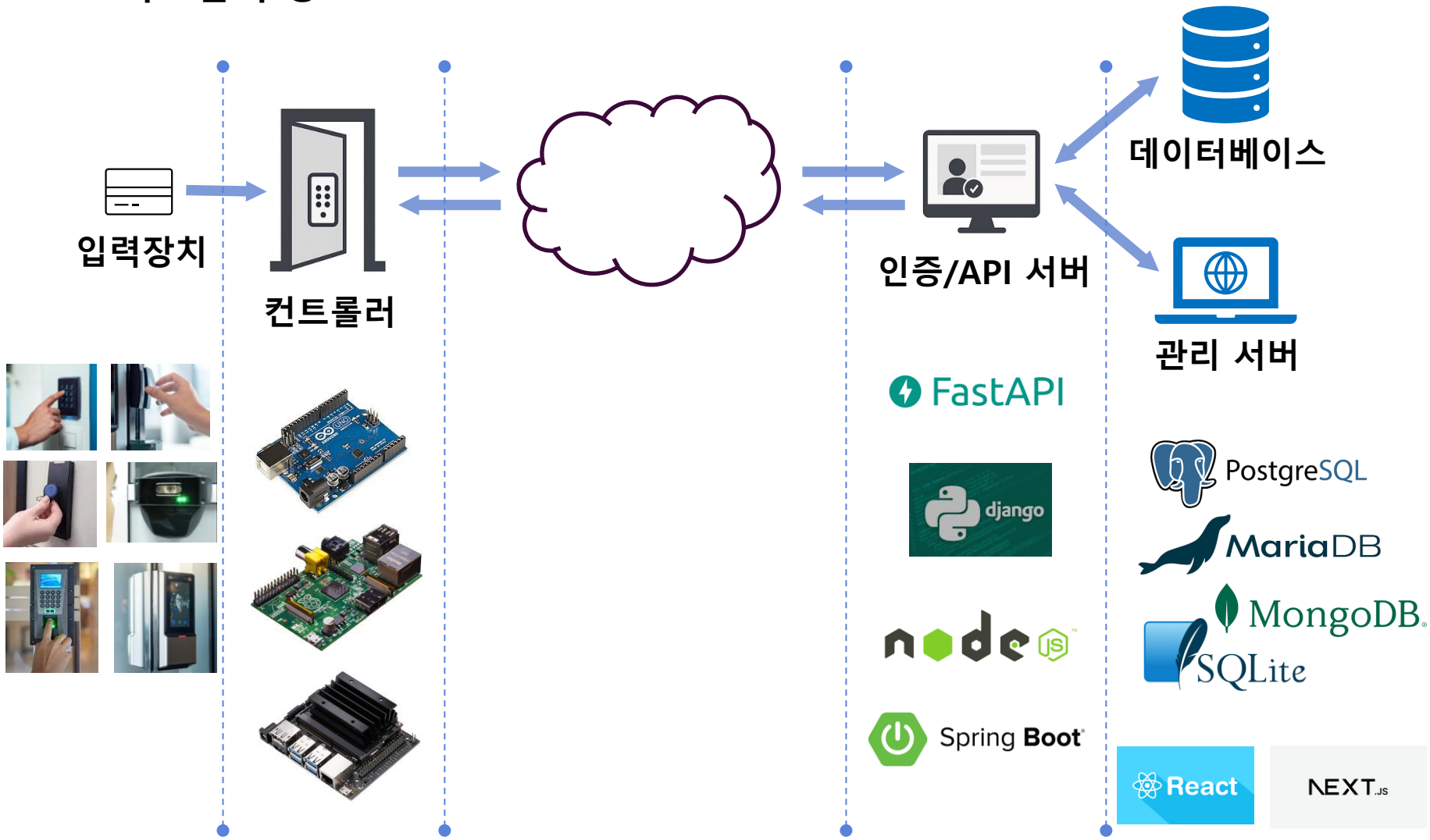
AccID (사용자 ID):
A12345

GateID (게이트 ID):
Gate01

전송

출입관리 시스템 개요

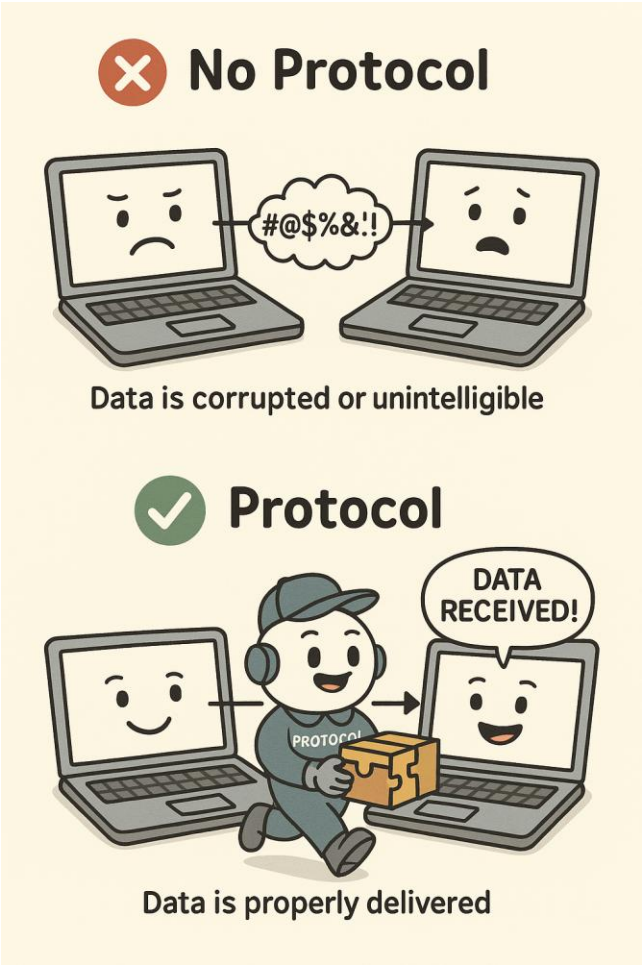
■ 시스템 구성 요소



프로토콜

■ **프로토콜(protocol)**

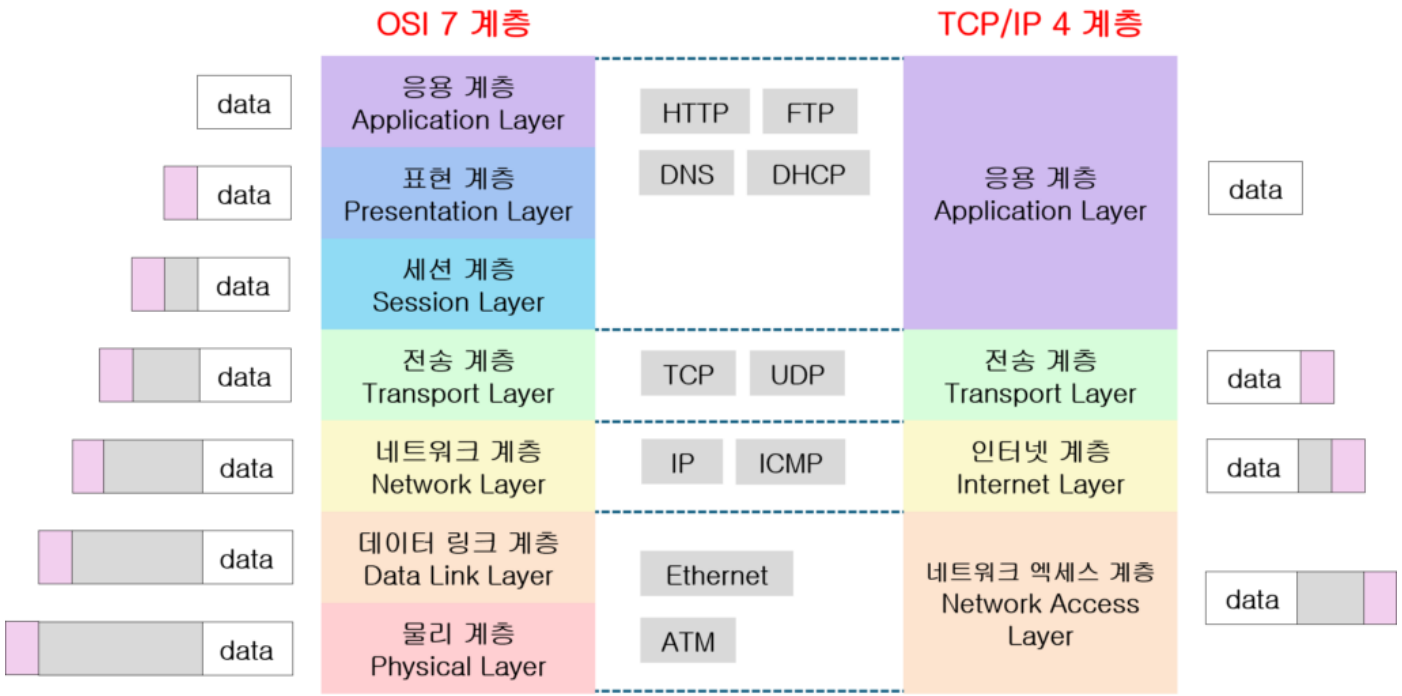
- 통신과 관련된 형식과 절차
- 서로 다른 장치나 시스템이 데이터를 주고받기 위해 정해진 규칙
- 메시지 종류, 형식, 교환 절차, 흐름 제어, 에러 제어 등에 관한 규칙



프로토콜

■ OSI 표준 모델 vs. TCP/IP 모델

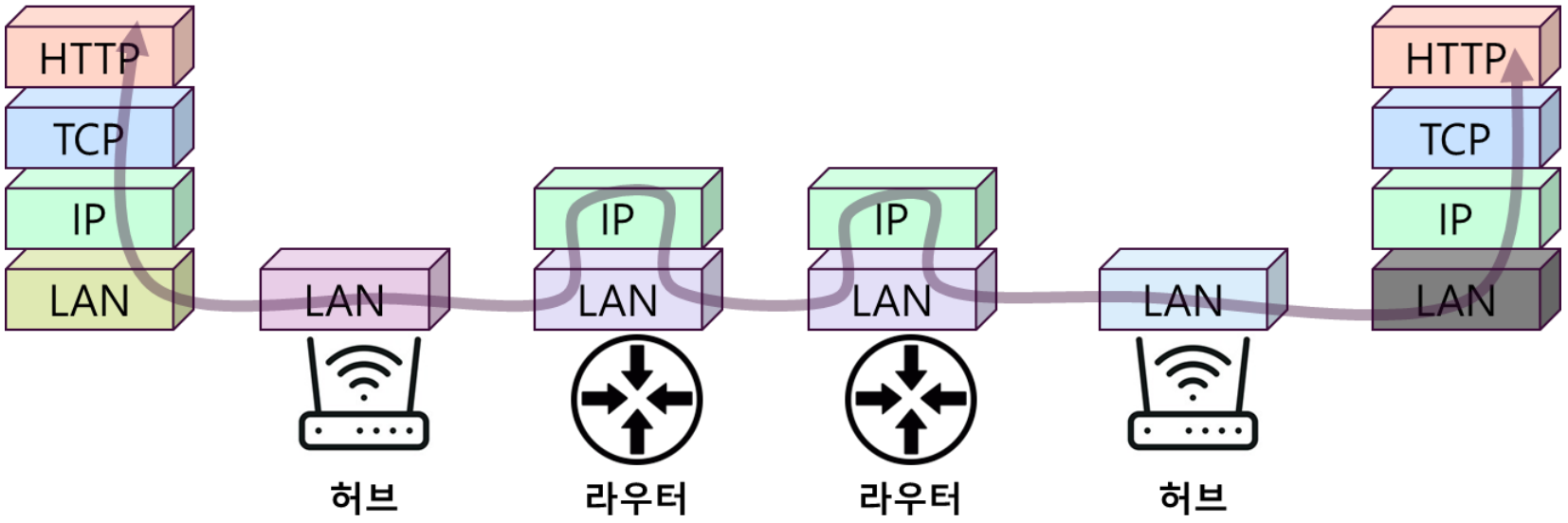
- OSI 표준 모델: **이론적인 체계를 제시**하여 상호 운용성을 확보하고 네트워크 통신 과정을 이해하기 쉽게 만드는 것을 목표
- TCP/IP 모델: **실제적인 구현**을 통해 신뢰성 있는 데이터 전송과 이기종 네트워크 간의 연결을 가능하게 하는 것을 목표



프로토콜

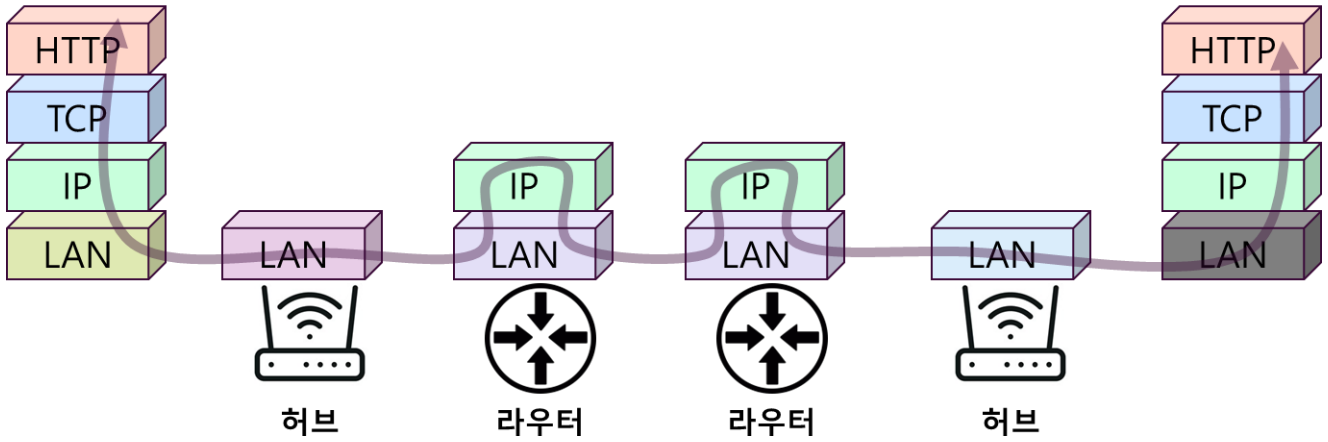
■ TCP/IP 모델의 데이터 흐름

- 인터넷 프로토콜 스택의 각 계층에서 데이터가 생성되고 캡슐화되어, 양 단말 간에 순차적으로 전달



프로토콜

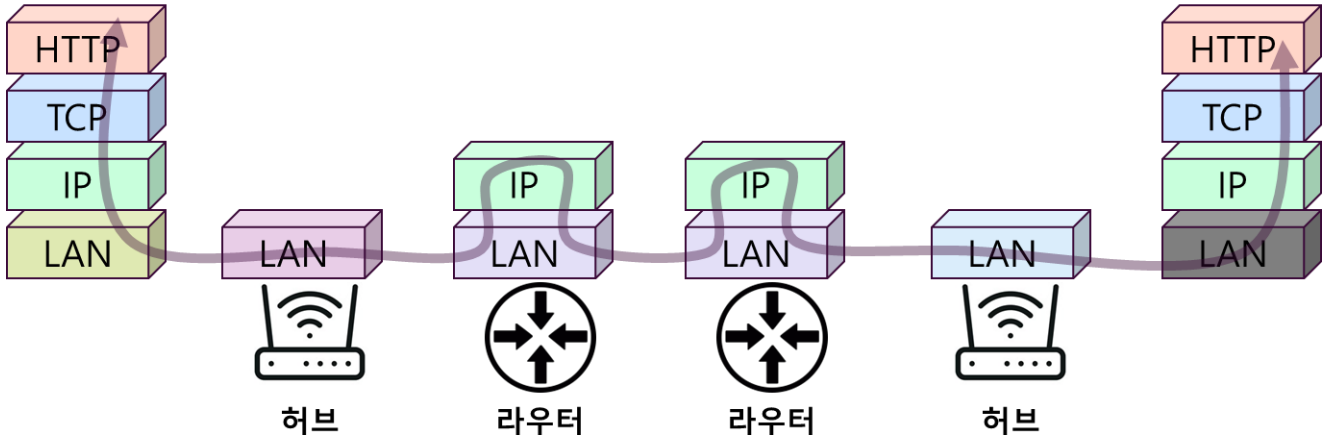
■ TCP/IP 모델의 데이터 흐름 – 웹 브라우저 예제



01. 사용자가 브라우저 주소창에 **http://naver.com/news** 라고 입력
02. 브라우저는 DNS에 IP 주소를 질의(naver.com의 IP 주소는 무엇?)
03. DNS는 naver.com의 IP 주소를 반환(223.130.195.200)
04. 브라우저는 TCP 연결 시도(3-way handshake: SYN → SYN-ACK → ACK)
    - TCP 레이어: **자신의 port(랜덤)**와 **목적지 port(80)** 추가
    - IP 레이어: **자신의 IP 주소(210.125.31.78)**와 **목적지 IP 주소(223.130.195.200)** 추가
    - Link 레이어: **자신의 MAC 주소**와 **라우터의 MAC 주소** 추가

프로토콜

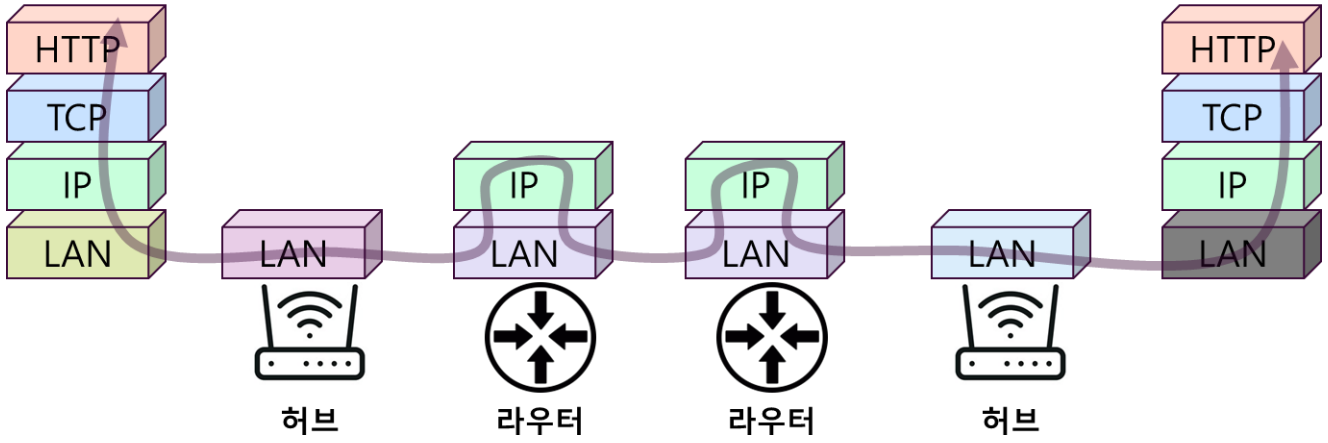
■ TCP/IP 모델의 데이터 흐름 – 웹 브라우저 예제



05. **허브**는 브라우저에서 보낸 패킷의 **MAC 주소**를 참조하여 패킷을 라우터에 전송
06. **라우터**는 패킷의 **IP 주소**를 참조하여 다음 라우터로 패킷을 전송(여러 라우터들을 경유)
07. 최종 목적지와 연결된 허브로 패킷 전송
08. **허브**는 라우터로부터 수신한 패킷의 **MAC 주소**를 보고 서버로 패킷 전송

프로토콜

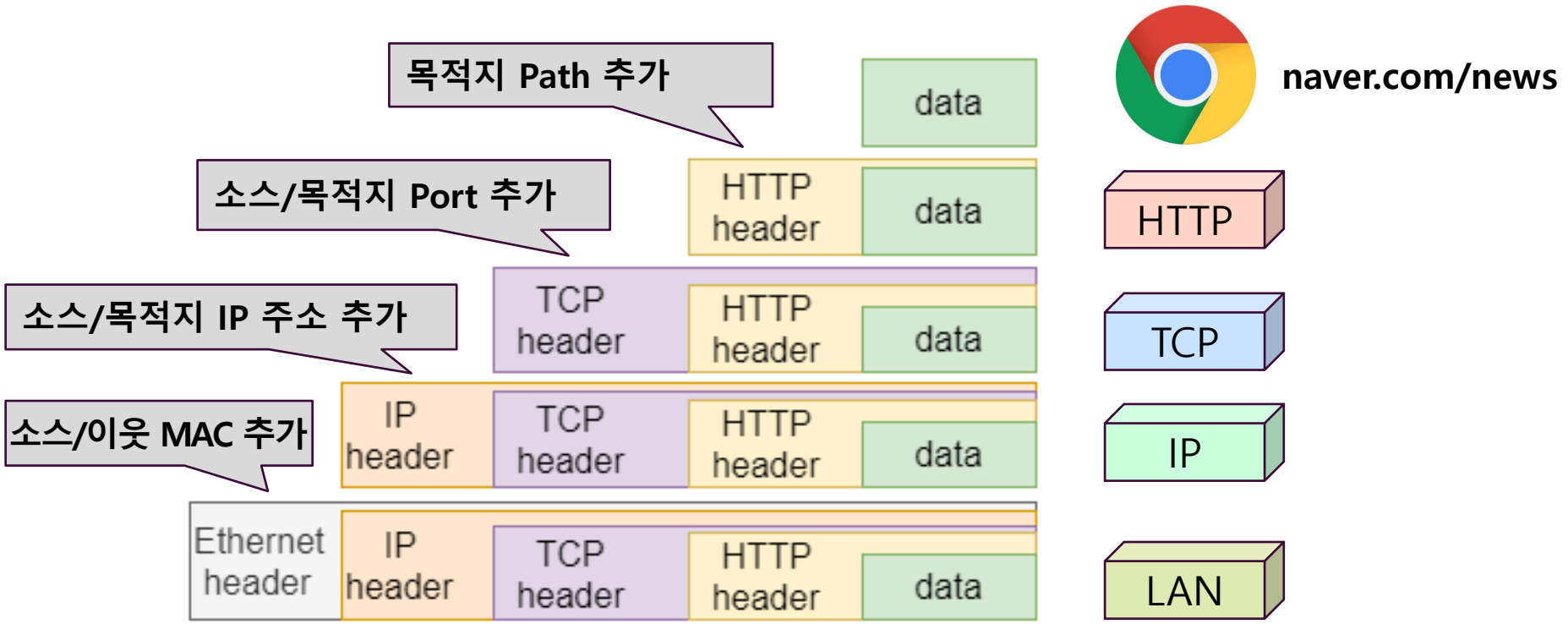
■ TCP/IP 모델의 데이터 흐름 – 웹 브라우저 예제



09. 서버는 패킷을 수신한 후 계층별로 디캡슐레이션 수행
    - Link 레이어: **자신의 MAC 주소**와 일치하는지 검사
    - IP 레이어: **자신의 IP 주소**와 일치하는지 검사
    - TCP 레이어: **port 번호** 확인 후 연결된 소켓(응용 서비스)에 전달
    - HTTP 계층: 최종적으로 브라우저의 요청인 /news 를 처리
10. HTTP 응답 생성 → 캡슐화 → 브라우저에 응답 전송

프로토콜

■ TCP/IP 모델의 데이터 흐름 – 웹 브라우저 예제



프로토콜

■ TCP/IP 모델의 데이터 흐름 – 웹 브라우저 예제

port 확인: netstat -ano

```
  TCP    192.168.0.8:53034      96.7.229.86:80         ESTABLISHED
  TCP    192.168.0.8:53035      142.250.206.227:80     ESTABLISHED
  TCP    192.168.0.8:53038      211.115.106.210:80     CLOSE_WAIT
  TCP    192.168.0.8:53039      96.7.229.86:80         ESTABLISHED
```

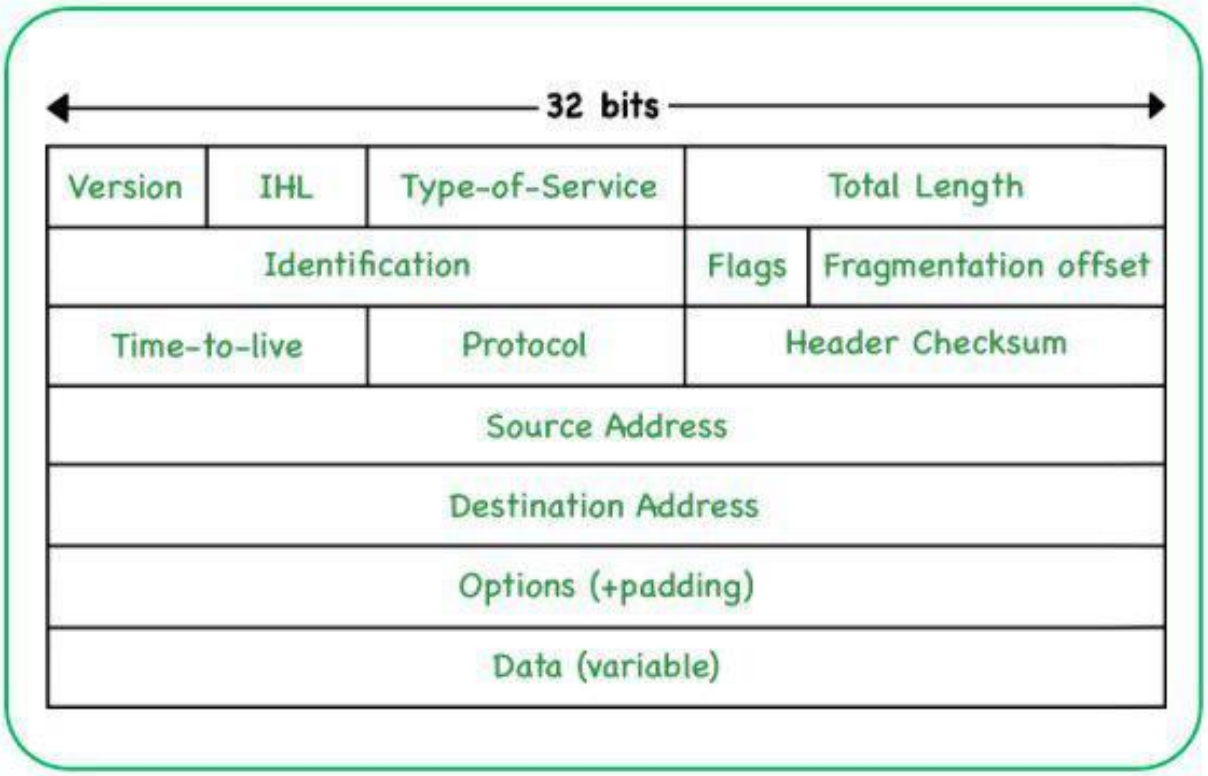
IP, MAC 주소 확인: ipconfig /all

```
무선 LAN 어댑터 Wi-Fi:

   연결별 DNS 접미사. . . . :
   설명. . . . . . . . . . . . . . : Realtek 8812BU Wireless LAN 802.11ac USB NIC
   물리적 주소 . . . . . . . . . . : 58-86-94-F1-47-B0
   DHCP 사용 . . . . . . . . . . . : 예
   자동 구성 사용. . . . . . . . . : 예
   링크-로컬 IPv6 주소 . . . . . . : fe80::579:c5be:77c5:df15%14(기본 설정)
   IPv4 주소 . . . . . . . . . . . : 192.168.0.8(기본 설정)
   서브넷 마스크 . . . . . . . . . : 255.255.255.0
   임대 시작 날짜. . . . . . . . . : 2025년 5월 7일 수요일 오전 12:47:19
   임대 만료 날짜. . . . . . . . . : 2025년 5월 7일 수요일 오전 2:47:19
   기본 게이트웨이 . . . . . . . . : 192.168.0.1
   DHCP 서버 . . . . . . . . . . . : 192.168.0.1
   DHCPv6 IAID . . . . . . . . . . : 676890260
   DHCPv6 클라이언트 DUID. . . . . : 00-01-00-01-29-47-70-8E-D8-BB-C1-57-24-69
   DNS 서버. . . . . . . . . . . . : 168.126.63.1
                                     168.126.63.2
```

■ IP (Internet Protocol)

- 인터넷 상에서 데이터를 주고받기 위한 규칙과 **주소 체계를 정의**



TCP/IP

■ IP 주소

- 컴퓨터의 주소를 숫자로 나타내는 방식(203.252.16.7)
- 인터넷에 연결되기 위해 각 단말기는 고유한 IP 주소를 가지고 있어야 함
- 32bits의 길이를 가짐(2^{32} ≈ 43억)
- IP 주소 부족 문제가 있음
  - ✓ IPv6 도입: 128bits 길이(2^{128} = 거의 무한대)
  - ✓ NAT(Network Address Translation) 사용: 사설 IP → 공인 IP로 변환

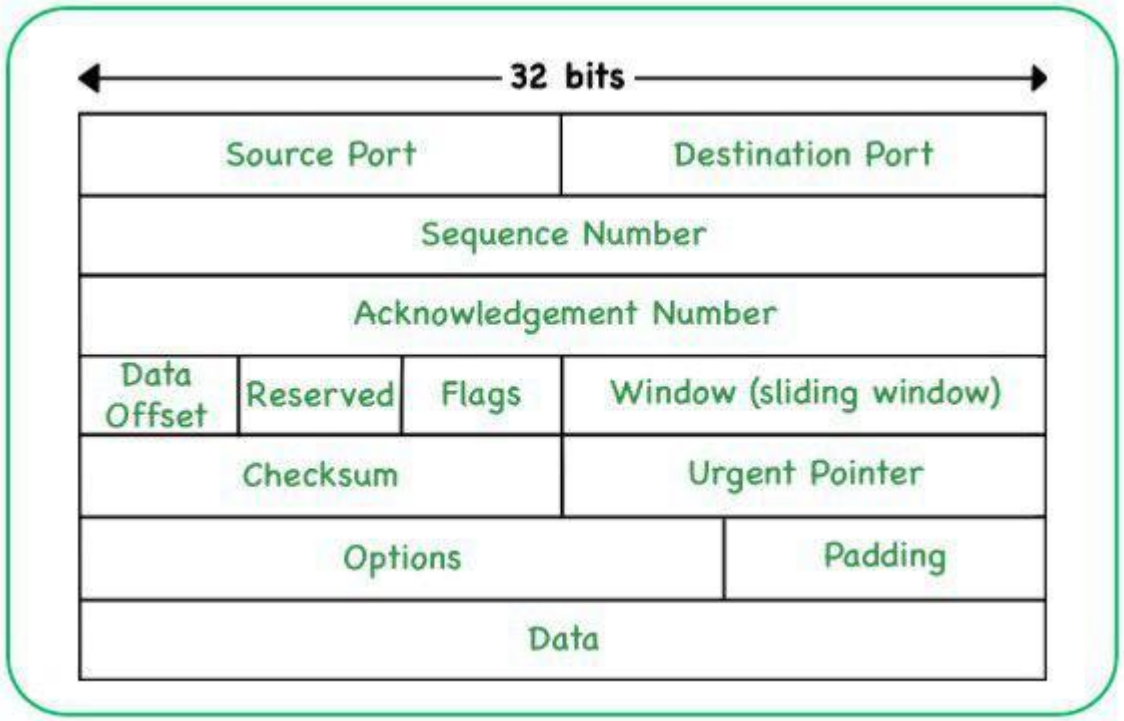
TCP/IP

■ TCP (Transmission Control Protocol)

- 데이터를 안정적이고 순서대로, 오류 없이 주고받을 수 있도록 관리
- 연결 지향형: 데이터를 보내기 전에 송신자와 수신자 간에 논리적인 연결을 설정(3-way handshake)
- 신뢰성: 데이터를 **세그먼트**라는 작은 단위로 나누어 전송하고, 각 세그먼트에 순서 번호(Sequence Number)를 부여. 만약 세그먼트가 손실되면 해당 세그먼트를 **재전송**
- **흐름 제어:** 수신 측의 데이터 처리 속도나 버퍼 용량을 고려하여 송신 측의 데이터 전송 속도 조절
- 혼잡 제어: 네트워크의 혼잡 상황을 감지하고, 혼잡이 발생하면 송신 측의 데이터 전송 속도를 줄여 **네트워크 과부하 방지**

TCP/IP

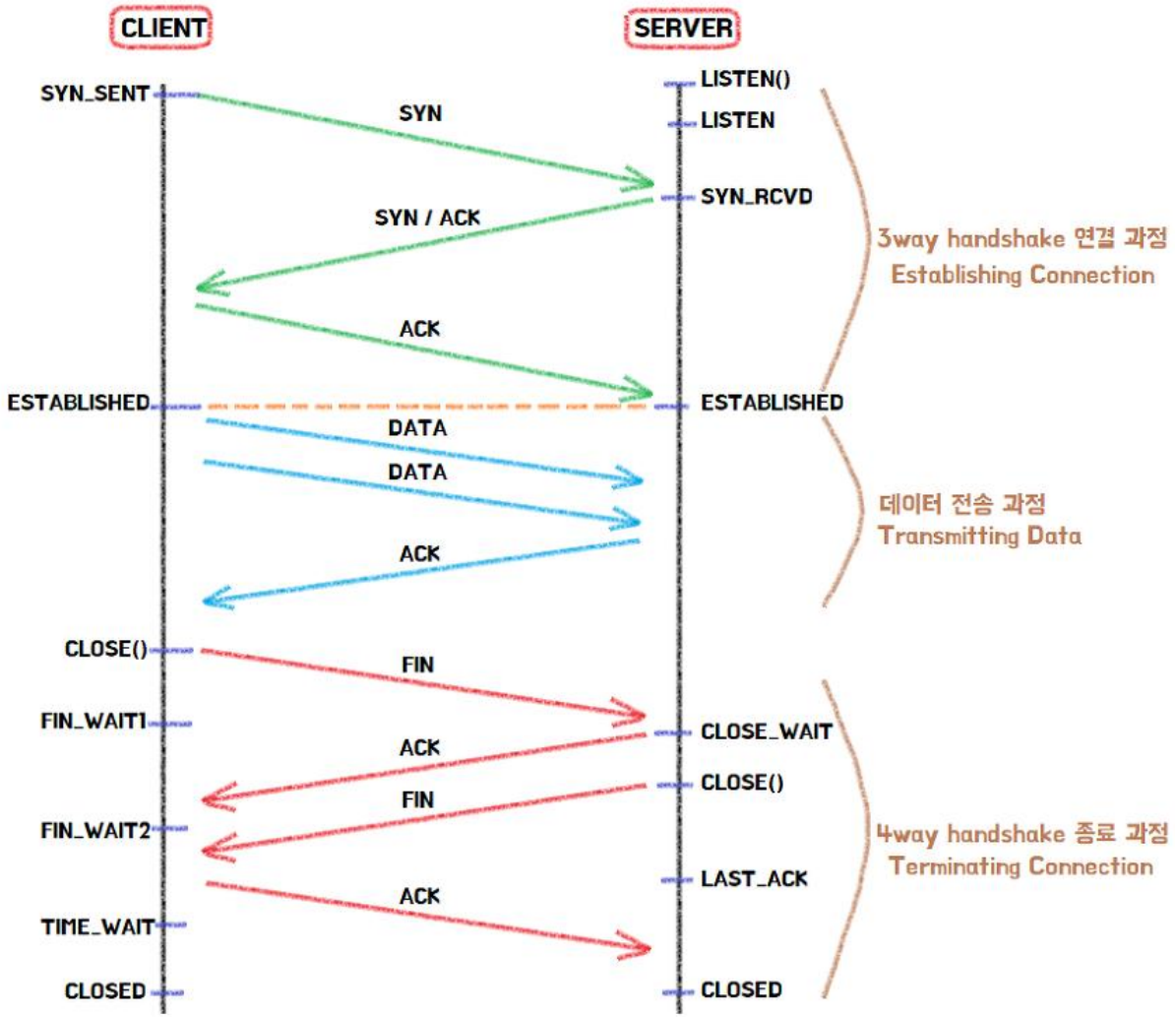
- **TCP (Transmission Control Protocol)**



TCP/IP

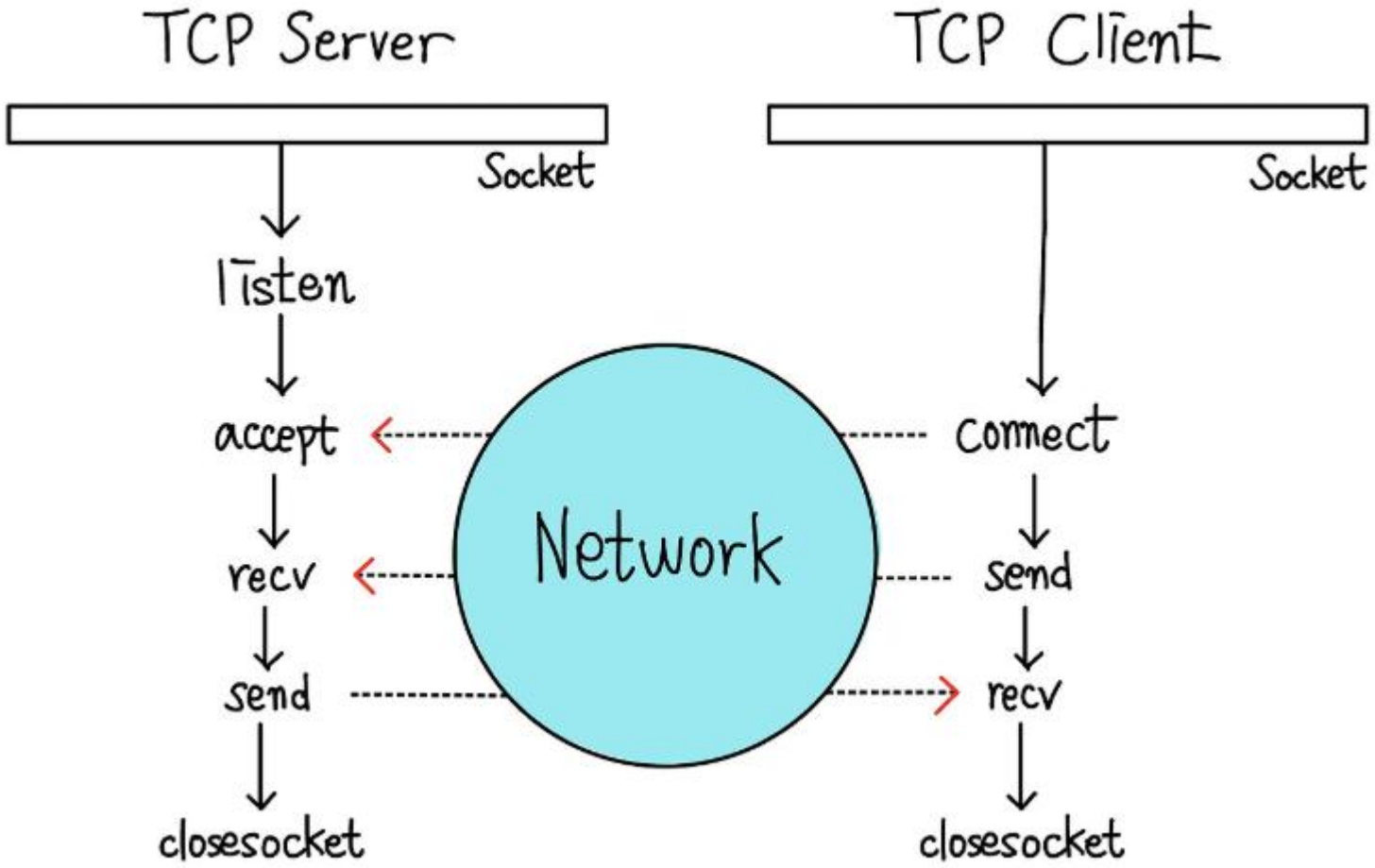
■ TCP (Transmission Control Protocol)

- **3-way handshake**
  - ✓ 클라이언트와 서버가 신뢰성 있는 연결을 맺기 위한 과정
- **4-way handshake**
  - ✓ 연결을 종료하기 위한 과정



출입관리 시스템 프로토타입

■ 클라이언트 - 서버 통신 절차



출입관리 시스템 프로토타입

■ 구현 유형

번호	기기	방식	프로토콜	서버
1	Arduino	Raw Socket	HTTP	FastAPI(HTTP 기반)
2	Arduino	Library	HTTP	FastAPI(HTTP 기반)
3	Arduino	Raw Socket	Custom	TCP 서버(Raw TCP 기반)
4	Arduino	Library	Custom	TCP 서버(Raw TCP 기반)
5	RPi	Raw Socket	HTTP	FastAPI(HTTP 기반)
6	RPi	Library	HTTP	FastAPI(HTTP 기반)
7	RPi	Raw Socket	Custom	TCP 서버(Raw TCP 기반)
8	RPi	Library	Custom	TCP 서버(Raw TCP 기반)

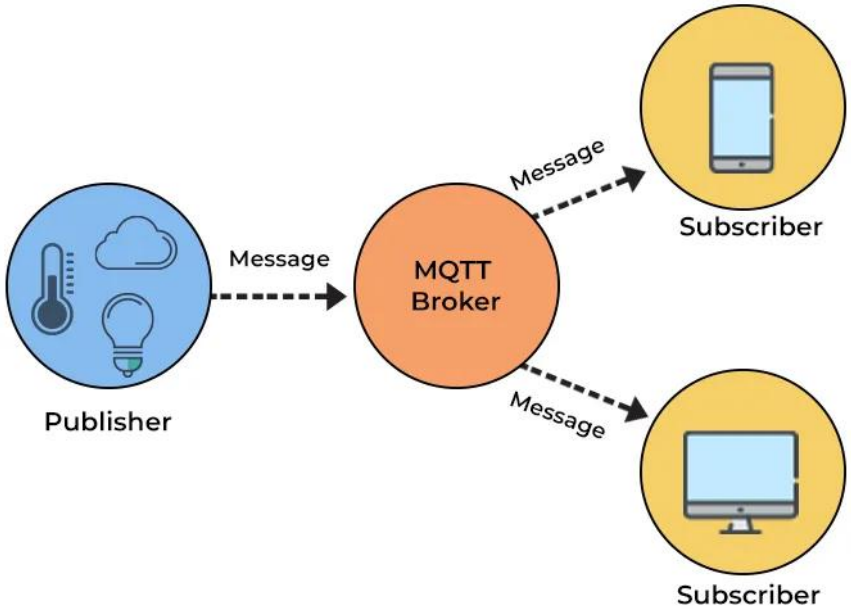
출입관리 시스템 프로토타입

https://colab.research.google.com/drive/1fO0wsXemOVvdb5P_1sTQ9oBOwvDhRTjY?usp=sharing

추가 고려 사항

■ MQTT 적용

- TCP를 기반으로 한 Publish-Subscribe 구조의 프로토콜
- 전통적인 클라이언트-서버 구조와는 달리, Publisher는 메시지를 발행하고 , Subscriber는 관심 있는 메시지를 구독하는 구조
- 메시지는 Broker를 통해 전달되며 Broker는 Publisher와 Subscriber 사이에서 메시지를 중계



추가 고려 사항

■ MQTT 적용

MQTT 요소	출입관리 시스템	설명
클라이언트	게이트 장치 (Arduino, Raspberry Pi 등)	NFC로 acc_id를 읽고 MQTT 메시지를 발행(Publish)
브로커	중앙 메시지 중계기 (Mosquitto, EMQX 등)	MQTT 프로토콜로 장치와 서버 사이 메시지 중계
서버 (구독자)	인증 서버 (FastAPI 등)	게이트 장치가 보낸 access/request 메시지를 구독하고, access/response로 결과를 발행

추가 고려 사항

■ MQTT 적용시 장점

항목	MQTT	HTTP
통신 방식	지속 연결 (TCP 기반의 구독/발행)	요청-응답 기반 (매번 연결/해제)
데이터 흐름	양방향 가능 (게이트 ↔ 서버)	일방향 위주 (게이트 → 서버)
속도/실시간성	푸시 기반, 실시간 반응 우수	클라이언트 polling 또는 재요청 필요
네트워크 부하	매우 낮음 (헤더 수 바이트 수준)	상대적으로 큼 (HTTP 헤더 포함)
지속 연결 유지	기본적으로 유지됨 (keep-alive)	요청마다 재연결
구현 복잡도	초반 학습 필요, 하지만 구조 단순	HTTP는 쉬우나 비동기 처리 어려움

MQTT 장점	**출입관리 시스템 적용**
실시간 출입 승인 응답	게이트에서 요청 → 서버 응답을 기다릴 수 있음 (push 구조)
네트워크 트래픽 최소화	단순한 센서 데이터 통신에서 효율적
수십 개 게이트 확장	브로커 기반이므로 수평 확장에 유리
게이트 장애 감지	브로커가 연결 끊김을 감지하고 서버에 알릴 수 있음

추가 고려 사항

■ MQTT 흐름

- 게이트 장치 (클라이언트)
  - ✓ acc_id, gate_id, time을 포함한 메시지를 MQTT 브로커의 access/request 토픽에 발행
- 브로커
  - ✓ 메시지를 자동 중계
  - ✓ access/request를 구독 중인 인증 서버에 전달
- 인증 서버 (구독자)
  - ✓ 메시지를 수신하고 allow 여부 판단
  - ✓ 결과를 access/response 토픽으로 발행
- 게이트 장치
  - ✓ access/response를 구독 중
  - ✓ 허용 여부에 따라 출입 제어

추가 고려 사항

■ 보안

- 출입관리 시스템은 인증정보와 출입 이력이라는 민감한 데이터를 다루기 때문에 보안이 중요함

항목	방식	설명
통신 보안	**TLS 적용**	- HTTP → HTTPS, MQTT → MQTTS로 전환 - 데이터를 암호화하여 탈취되더라도 해독 불가능하게 함
데이터 보안	**DB 보안**	- DB 저장 시 민감 정보(acc_id)를 암호화(AES256, SHA 256 등) - DB 접근 가능 주체를 API 서버로 한정
인증 시스템	JWT API 키 IP 화이트리스트	- 서바가 로그인 또는 등록 시 토큰 발급 - 게이트 장치나 외부 시스템에 고유 키 발급 - 신뢰된 IP 목록만 접근 허용

추가 고려 사항

■ 로컬 캐싱 및 오프라인 대응

- 서버와의 연결이 끊겼을 때도 출입을 허용할 수 있어야 함
- 게이트 장치에 최근 허용된 acc_id 목록을 캐싱해두고,
  - ✓ 오프라인일 때는 로컬 기준으로 임시 승인
  - ✓ 온라인 복수 시 출입 기록을 서버로 일괄 전송

■ AI 이상 감지

- 비정상적 출입 패턴 탐지
  - ✓ 새벽 시간 반복 출입, 다수 게이트 중복 시도 등

■ 모니터링 및 알림 시스템

- 출입 실패, 인증 에러, 서버 장애 등을 실시간으로 관리자에게 통보
  - ✓ Slack, Email, SMS 등 연동