QT 4일차 보고서

로빛 20기 인턴 김민조

목차

1. 통신

(1) 정의

- 정보를 송신자에서 수신자로 전달하는 과정
- 음성, 문자, 데이터, 이미지, 영상 등 다양한 형태의 정보를 포함

(2) 용어

- **송신자**
- 정보를 생성하고 보내는 주체
- 예: 사용자, 컴퓨터, IoT 기기

- **수신자**
- 정보를 받는 주체
- 예: 서버, 네트워크 장비

- **메시지**
- 전달하고자 하는 정보 자체

- **채널**
- 메시지를 전달하는 매체
- 예: 케이블, 무선 주파수 등

- **인코딩**
- 메시지를 전달하는 매체
- 예: 메시지를 전송 가능한 형식으로 변환

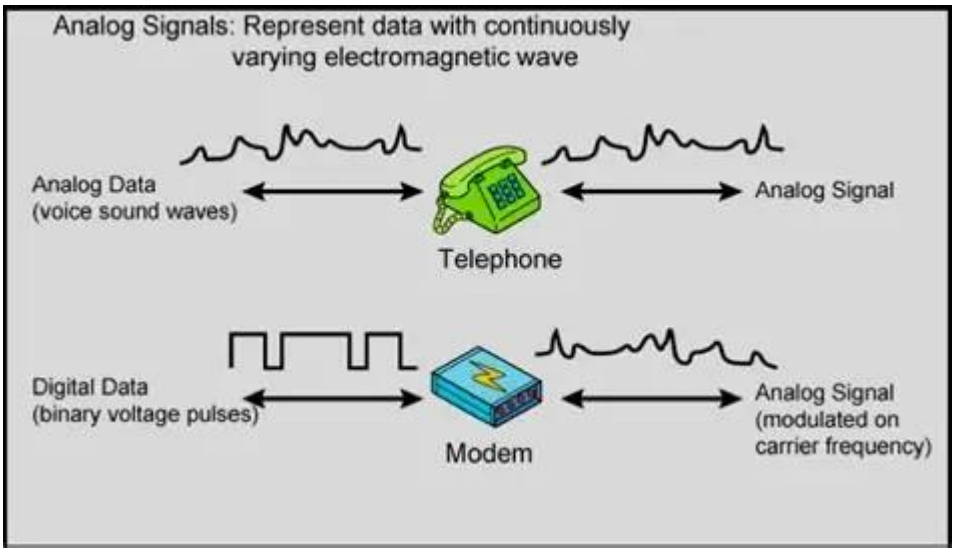
- **디코딩**
- 수신자가 수신한 메시지를 이해 가능한 형태로 복원

- **프로토콜**
- 통신 규칙 및 형식

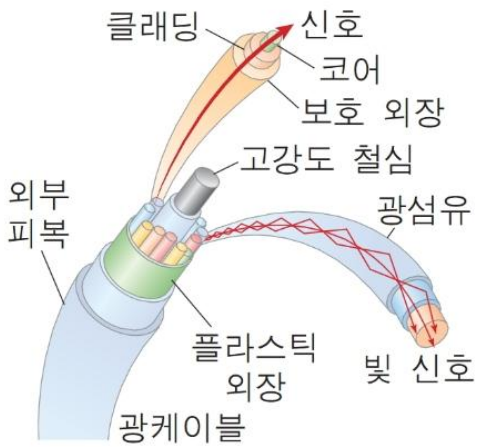
(3) 통신의 유형

- 아날로그 통신
- 연속적인 신호로 정보를 전달
- 예: 라디오 방송, 아날로그 전화기

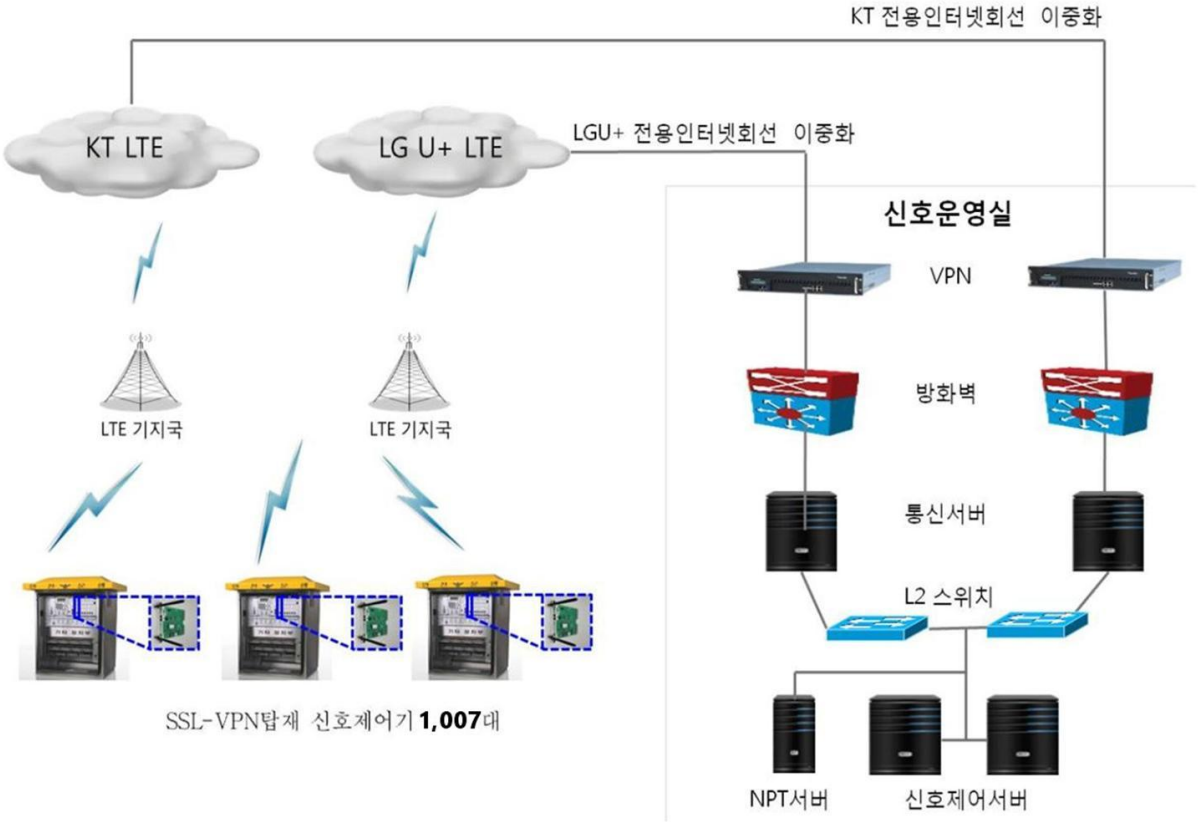
- 디지털 통신
- 이산적인 비트 형태로 데이터를 전달
- 예: 인터넷, 컴퓨터 간 통신



- 유선 통신
- 케이블이나 광섬유 등을 통해 데이터 전송
- 예: 구리선, 광케이블 등

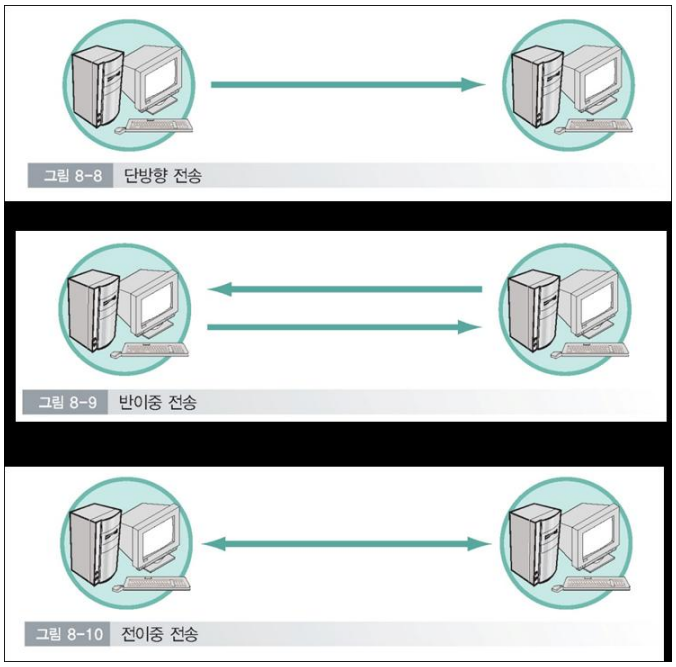


- 무선 통신
- 전파, 적외선, 위성 등을 통해 데이터 전송
- 예: 전파, 적외선, 위성 등



- 단방향
- 한 방향으로만 데이터 전송
- 예: TV 방송

- 반이중
- 양방향 가능 but 동시는 불가
- 예: 무전기
- 전이중
- 양방향 동시에 통신 가능
- 예: 휴대전화 통화



(4) 데이터 통신

- 현대 통신은 대부분 디지털 데이터 통신을 기반으로 이루어짐

- 네트워크를 통해 데이터를 주고받으며, 이 과정은 여러 통신프로토콜을 따름

- IP: 네트워크 상 주소 지정 및 데이터 전달
- TCP: 신뢰성 있는 데이터 전송 보장
- UDP: 빠른 전송을 위한 비연결성 통신
- HTTP/HTTPS: 웹 페이지 요청 및 응답을 위한 상위 프로그램
- FTP, SMTP, DNS 등: 다양한 용도의 전용 프로토콜 존재

2. OSI 7 계층

(1) 정의

- ISO 에서 제정한 통신 프로토콜 모델로, 네트워크 통신 과정을 7 개의 계층으로 나누어 구조화한 모델
- 각 계층은 서로 독립적으로 동작하며, 상하 계층과만 인터페이스를 가짐

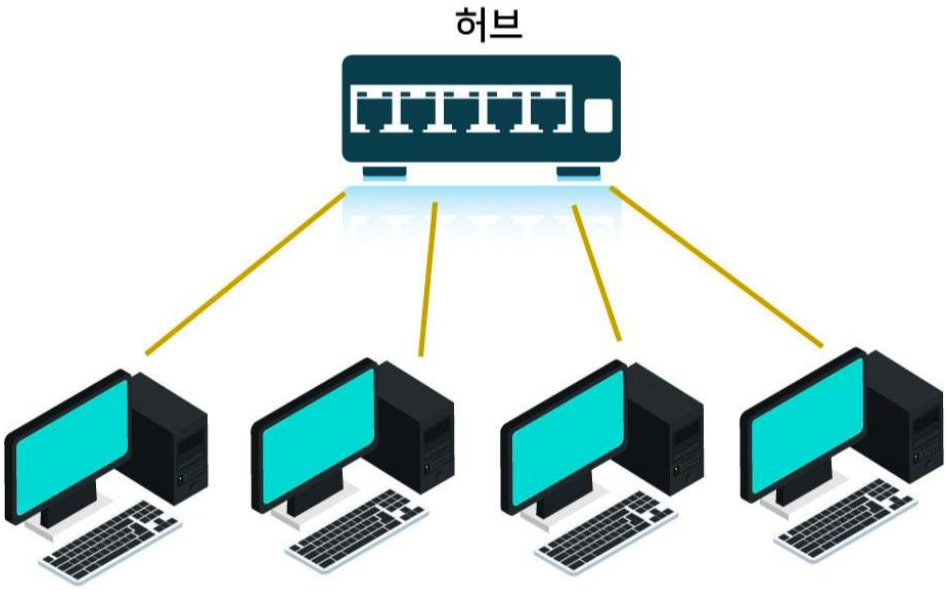
(2) 목적

- 네트워크 프로토콜 설계를 체계적으로 하기 위해
- 장비 간 호환성과 상호운용성 향상
- 문제 발생 시 디버깅을 쉽게 하기 위해

(3) 물리 계층

- **기능**
- 비트를 전기적 신호로 바꾸어 전송

- **장비/기술 예시**
- 허브, 리피터, 케이블, 광섬유, 무선 주파수



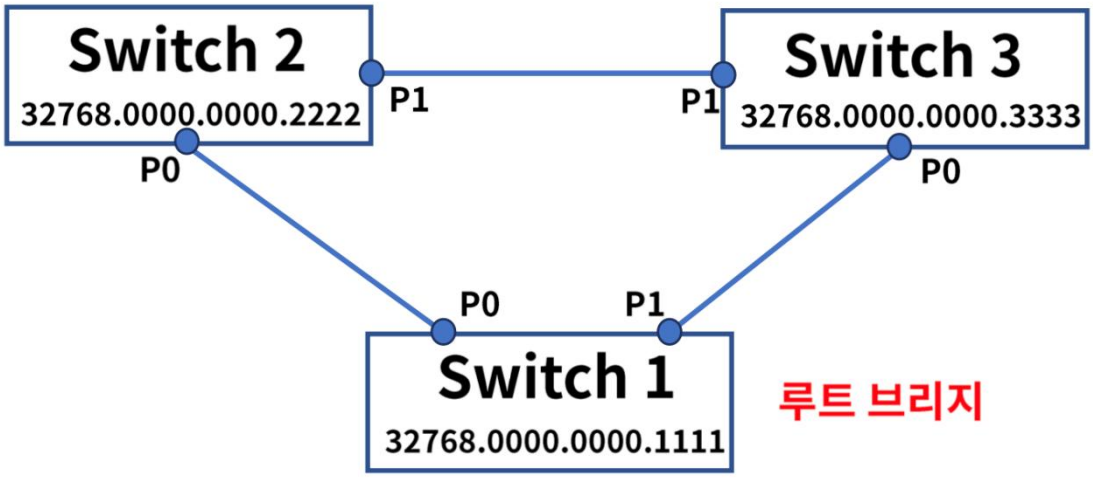
(4) 데이터 링크 계층

- **기능**
- MAC 주소를 기반으로 이더넷 통신, 프레임 단위로 오류
- 감지 및 흐름 제어

- **하위 계층**: LLC, MAC

- **장비 예시**
- 스위치

- **프로토콜**
- Ethernet, ARP, PPP

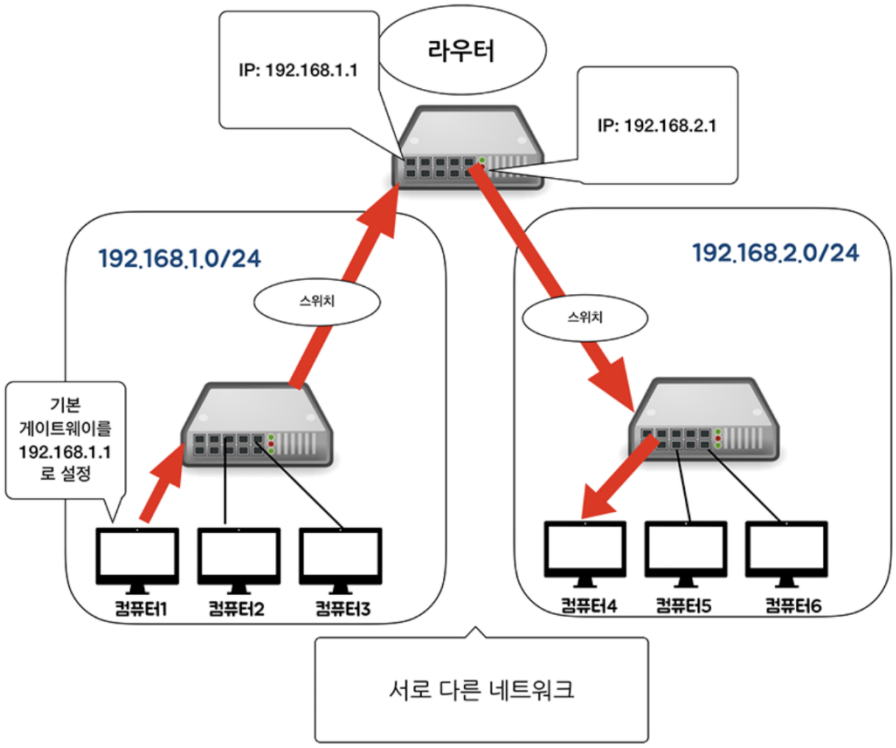


(5) 네트워크 계층

- **기능**
- 데이터를 목적지까지 전달하는 경로 설정

- **프로토콜**
- IP, ICMP, IGMP, OSPF, RIP

- **장비예시**
- 라우터

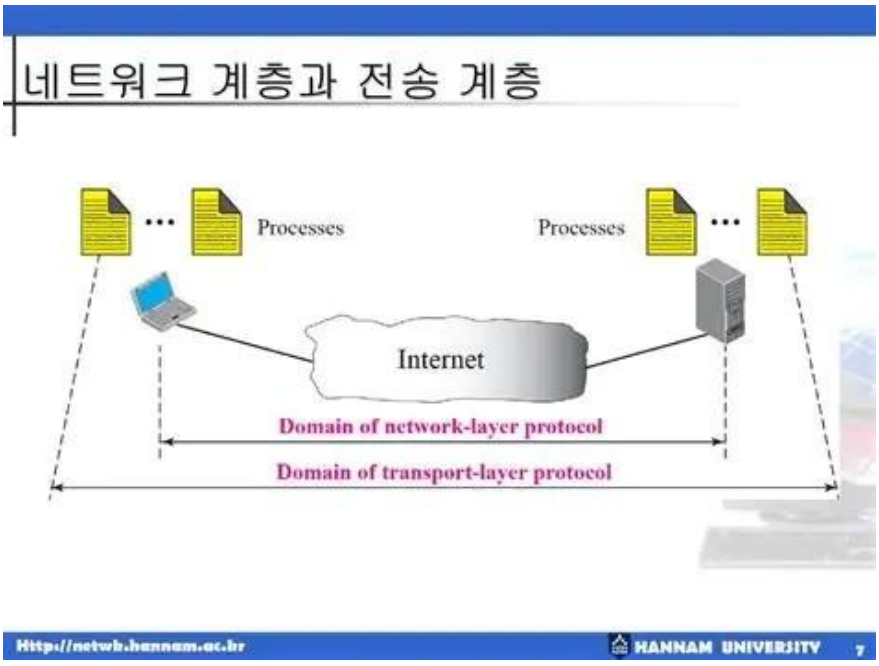


(6) 전송 계층

- **기능**
- 종단 간의 데이터 전송, 흐름 제어, 오류 제어
- 세그먼트 분할

- **프로토콜**
- TCP, UDP

- **역할**
- 여러 어플리케이션 간의 데이터 전송 구분 및 제어



(7) 세션 계층

- **기능**
- 통신 세션의 생성, 유지, 종료 / 연결 제어

- **사용 예**
- 원격 데스크톱, 화상회의, 스트리밍

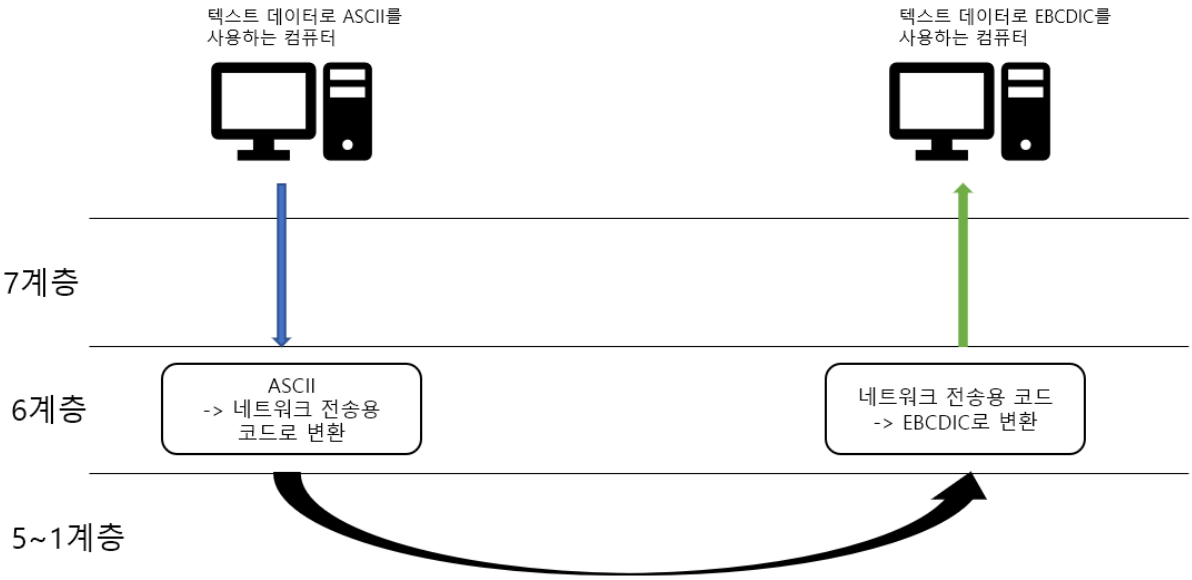
- **프로토콜**
- NetBIOS, RPC

(8) 표현 계층

- **기능**
- 데이터 포맷 변환, 암호화/복호화, 압축/해제

- **역할**
- 서로 다른 시스템 간 데이터 형식 일치

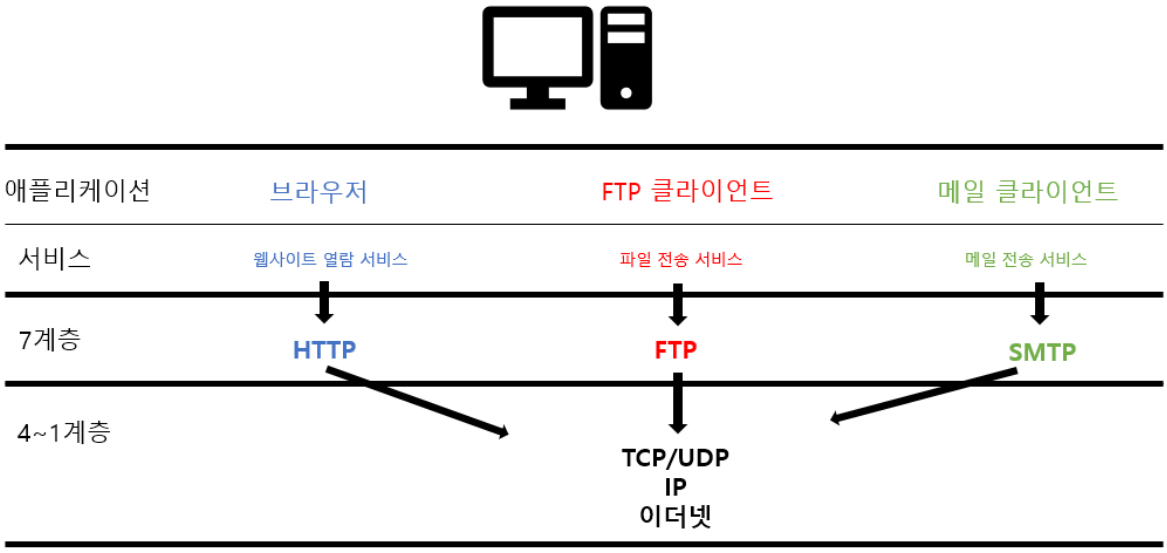
- **예시**
- JPEG, MP3, TLS/SSL



(9) 응용 계층

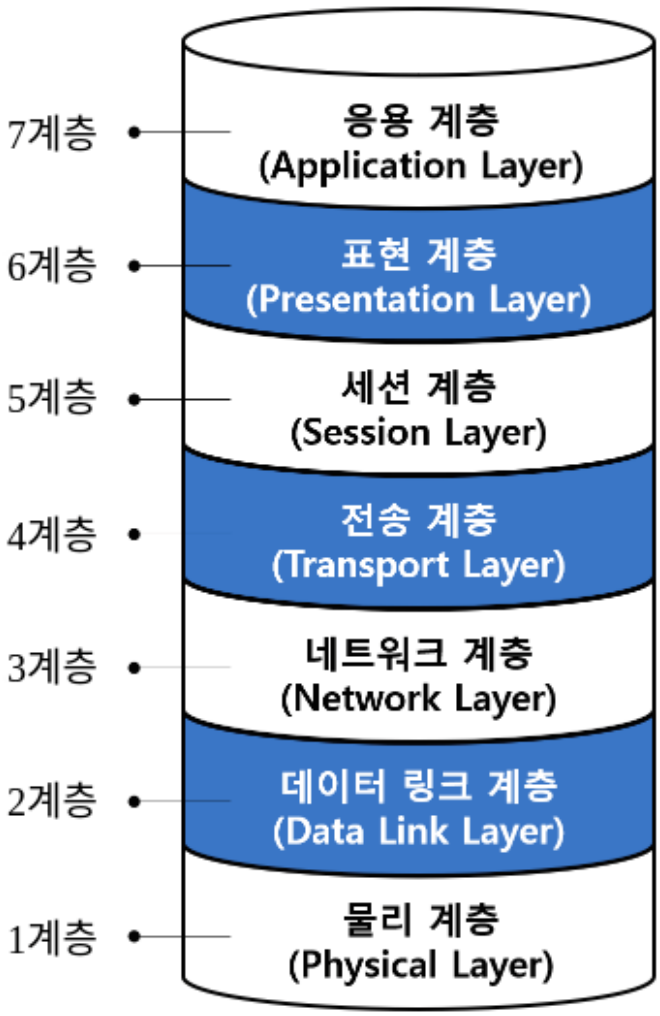
- **기능**
- 사용자가 네트워크에 접근할 수 있도록 지원

- **프로토콜**
- HTTP, FTP, SMTP, DNS, POP3, Telnet



(10) OSI 7 계층의 필요성

- 네트워크 기능을 모듈화하여 설계 및 관리가 쉬움
- 문제 발생 시 원인 추적이 쉬움
- 다양한 장비, 운영체제, 프로토콜 간의 호환성 확보



3. UDP

(1) UDP 란?

- 인터넷 프로토콜 위에서 동작하는 전송 계층의 통신 프로토콜로 빠른 전송 속도를 목표로 설계된 비연결형, 비신뢰성 데이터 전송 방식

(2) 개발 배경

- 1980 년대 초반, TCP 보다 간단하고 가벼운 통신 방식이 필요해지면서 만들어짐
- RFC 768 에 의해 정의 되었으며, TCP 의 복잡한 흐름 제어 / 오류 제어를 제거하고 단순화 추구

(3) 주요 특징

- **비연결성**
- 연결을 설정하지 않고 바로 데이터를 전송

- **비신뢰성**
- 데이터가 손실되거나 순서가 바뀔 수 있음

- **오버헤드 적음**
- 헤더가 간단하여 전송 속도가 빠름

- **순서 보장 X**
- 수신 순서와 전송 순서가 다를 수 있음

- **에러 제어 X**
- 수신자가 오류를 확인하지 않음

- **멀티캐스트/브로드캐스트 O**
- 하나의 송신자가 여러 수신자에게 전송 가능

(4) UDP 헤더 구조
- Source Port: 송신자의 포트 번호
- Destination Port: 수신자의 포트 번호
- Length: UDP 헤더 + 데이터 길이
- Checksum: 오류 검출용 값

(5) UDP 동작 방식
- 데이터를 데이터그램이라는 단위로 바로 전송
- 1. 애플리케이션이 데이터를 준비
- 2. 전송 계층이 헤더를 붙여 데이터그램 생성
- 3. IP 계층에 전달하여 목적지 IP 로 전송
- 4. 수신 측에서 도착한 데이터그램을 애플리케이션에 전달

(6) UDP 사용 사례
- 실시간 스트리밍: 유튜브, 넷플릭스 등 영상 스트리밍
- 온라인 게임: FPS, MOBA 등 빠른 반응속도 요구
- VoIP/화상통화: Skype, Zoom, Discord 등
- DNS 요청: Domain -> IP 주소 변환
- DHCP: IP 주소 자동 할당
- TFTP: 단순한 파일 전송 프로토콜

(7) UDP 의 장점
- 구조가 단순하여 구현이 쉬움
- 전송 속도가 매우 빠름
- 브로드캐스트/멀티캐스트 지원 가능

- 서버 부하가 적음

(8) UDP 의 단점

- 데이터 손실, 중복, 순서 뒤바뀜이 발생할 수 있음
- 전송의 성공 여부를 알 수 없음
- 보안에 취약

(9) TCP 와 UDP 비교

- **연결 방식**
- TCP: 연결지향형
- UDP: 비연결형

- **신뢰성**
- TCP: 높음(재전송, 흐름제어, 순서 보장)
- UDP: 낮음(오류 발생 시 무시)

- **데이터 전송**
- TCP: 스트림 기반(연속된 바이트)
- UDP: 데이터그램 기반(개별 패킷)

- **속도**
- TCP: 느림(신뢰성 처리로 인한 오버헤드)
- UDP: 빠름(오버헤드 없음)

- **헤더 크기**
- TCP: 20~60 바이트
- UDP: 8 바이트

- **순서 보장**
- TCP: 있음
- UDP: 없음

- **흐름 제어**
- TCP: 있음(윈도우 크기, ACK 등)
- UDP: 없음

- **오류 제어**
- TCP: 있음(체크섬, 재전송)
- UDP: 체크섬만 있음

- **멀티캐스트/브로드캐스트**
- TCP: 지원 X
- UDP: 지원 O

- **대표 사용 사례**
- TCP: 웹, 이메일, 파일 전송
- UDP: 실시간 영상, 게임, VoIP, DNS

- **TCP 가 적합한 상황**
- 정확성과 신뢰성이 중요한 서비스
- 웹 브라우징
- 이메일
- 파일 전송
- 데이터베이스 통신
- 데이터 누락/중복이 발생해서는 안되는 상황

- **UDP 가 적합한 상황**
- 속도와 실시간성이 중요한 서비스
- 실시간 게임
- 음성 통화, 화상 회의
- 스트리밍
- DNS 조회, DHCP 요청
- 일부 데이터 손실이 발생해도 문제 없는 상황

4. UDP 구현

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // UDP 소켓 생성 및 바인드
    udpSocket = new QUdpSocket(this);
    udpSocket->bind(localPort, QUdpSocket::ShareAddress | QUdpSocket::ReuseAddressHint);
    // 로컬포트를 열어줌, 12345로 설정해놓음
    // ShareADDress: 여러 프로세스가 같은 UDP 포트 공유할 수 있게 허용
    // ReuseADDressHint: 여러 소켓이 동일한 IP/포트 조합을 사용할 수 있게 허용하는 힌트

    connect(udpSocket, &QUdpSocket::readyRead,
            this, &MainWindow::readPendingDatagrams);
    // 송신되면 함수 호출해서 동작수행

    // 엔터 눌렀을 때도 전송되도록 설정
    connect(ui->lineEditMessage, &QLineEdit::returnPressed,
            this, &MainWindow::on_sendButton_clicked);
}
```

- Mainwindow 에서 UDP 소켓을 생성해주고 지정 포트에 바인드
- 데이터 수신 시 자동 처리와 엔터 입력 시 전송 가능하도록 해줌

```
MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_sendButton_clicked() // 수신 처리하는 함수
{
    QByteArray data = ui->lineEditMessage->text().toUtf8();
    QHostAddress destIP(ui->lineEditIP->text());
    quint16 destPort = ui->lineEditPort->text().toUShort();

    if (data.isEmpty() || destIP.isNull() || destPort == 0) {
        return; // 빈 메시지나 잘못된 주소는 무시
    }

    // UDP 메시지 전송
    udpSocket->writeDatagram(data, destIP, destPort);

    // 내 메시지 출력
    QTextCursor cursor(ui->textBrowser->textCursor());
    cursor.movePosition(QTextCursor::End);// 커서 마지막으로 유지

    QTextBlockFormat format;
    format.setAlignment(Qt::AlignRight); // 오른쪽 정렬
    cursor.insertBlock(format);

    QTextCharFormat charFormat;
    charFormat.setForeground(QBrush(Qt::black)); // 글자색
    cursor.insertText(QString("나: %1").arg(ui->lineEditMessage->text()), charFormat); // %1자리에 문장 삽입

    // 입력창 비우기
    ui->lineEditMessage->clear();
}
```

- 사용자가 입력한 메시지를 UDP 로 지정 IP 와 포트를 읽어 전송
- 사용자가 전송한 메시지는 오른쪽으로 정렬 해주고 메시지를 보내고 난뒤 입력창을 비워줌

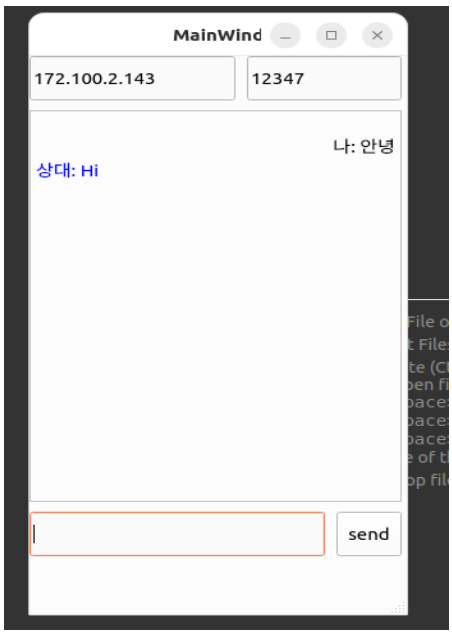
```
void MainWindow::readPendingDatagrams() // 송신 처리하는 함수
{
    while (udpSocket->hasPendingDatagrams()) { // 여러 패킷을 받을 수 있게
        QByteArray datagram;
        datagram.resize(int(udpSocket->pendingDatagramSize()));
        udpSocket->readDatagram(datagram.data(), datagram.size()); // 실제 데이터 읽어옴

        // QTextCursor 가져오기
        QTextCursor cursor(ui->textBrowser->textCursor());
        cursor.movePosition(QTextCursor::End); // 문서 맨 끝으로 이동해 새 메시지가 계속 아래에 출력되게

        // 왼쪽 정렬
        QTextBlockFormat format;
        format.setAlignment(Qt::AlignLeft);
        cursor.insertBlock(format);

        // 텍스트 출력
        QTextCharFormat charFormat;
        charFormat.setForeground(QBrush(Qt::blue));
        cursor.insertText(QString("상대: %1").arg(QString::fromUtf8(datagram)), charFormat);
    }
}
```

- 수신된 UDP 메시지를 읽어 왼쪽 정렬로 출력해줌

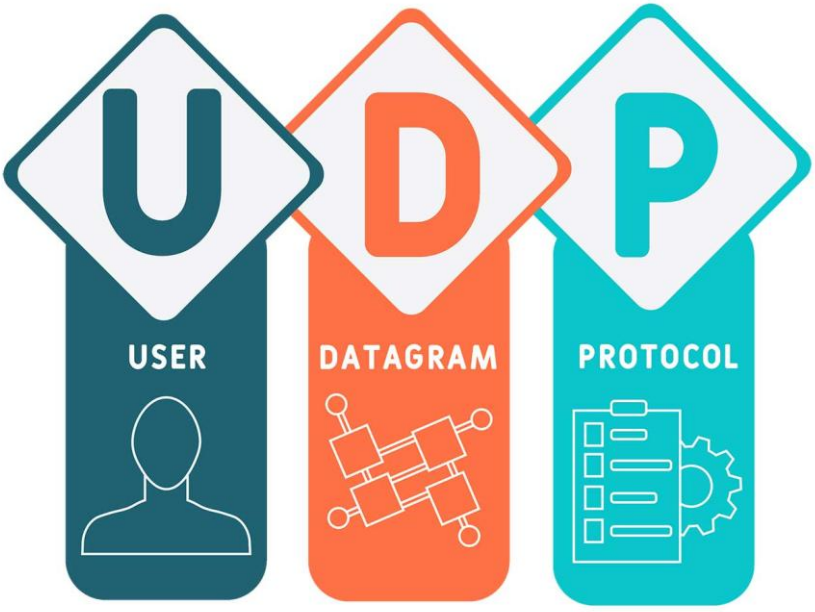


- 사진처럼 출력

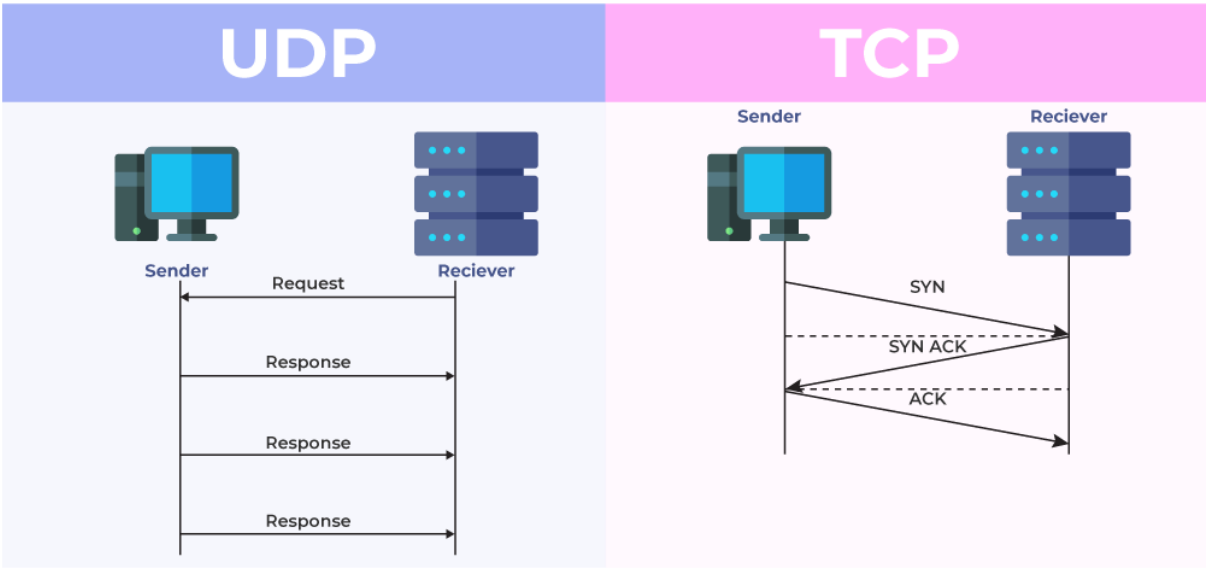
5. 결론 및 요약

- (a) 통신의 기본 요소와 용어
- 송신자: 정보를 생성하고 전송하는 주체
- 수신자: 정보를 받아 처리하는 주체
- 메시지: 전송되는 데이터 자체
- 채널: 메시지를 전달하는 매체
- 인코딩/디코딩: 메시지 변환 과정
- 프로토콜: 통신 규칙과 형식

- (b) UDP 특징
- (1)비연결형 통신
- 연결 설정 없이 데이터 전송 가능
- (2) 비신뢰성
- 데이터 손실, 순서 뒤바뀜 가능
- (3) 낮은 오버헤드
- 단순한 헤더 구조로 빠른 전송
- (4) 순서 보장 및 오류 제어 없음



- (c) TCP 와 비교
- (1) TCP
- 연결지향, 신뢰성 높음,순서 보장, 흐름/오류 제어 있음
- (2) UDP
- 비연결형, 속도 빠름, 신뢰성 낮음
- 멀티캐스트/브로드캐스트 지원



- (d) UDP 활용 분야
- 실시간 게임, VoIP, 화상 회의, 영상 스트리밍
- DNS/DHCP, TFTP 등
- 속도와 실시간성이 중요한 서비스에 적합
- 일부 데이터 손실이 발생해도 문제 없는 환경에서 활용

- (e) 결론
- UDP 는 속도와 실시간성이 중요한 환경에서 적합
- 신뢰성이 필요한 경우에는 어플리케이션 레벨에서 오류 처리 추가 필요