

HW2

20 기 인턴 강은구

목차

- 1 서론
 - 1.1 보고서 작성 목적
- 2 통신의 기본 개념
 - 2.1 ㄹ통신의 정의
 - 2.2 통신에 필요한 기본 요소
 - 2.3 통신 방식의 분류
- 3 통신 프로토콜
 - 3.1 프로토콜의 정의
 - 3.2 주요 기능
 - 3.3 프로토콜이 필요한 이유
- 4 UDP 개요
 - 4.1 UDP 의 정의
 - 4.2 TCP 와 UDP
 - 4.3 UDP 특징
 - 4.4 데이터그램 구조
- 5 UDP 동작 원리
 - 5.1 송신측
 - 5.2 수신측
 - 5.3 포트 번호의 종류와 사용 규칙
- 6 UDP 프로그래밍
 - 6.1 QUdpSocket
 - 6.2 데이터 보내기
 - 6.3 데이터 받기
 - 6.4 QByteArray 데이터 처리 방식
- 7 UDP 장단점
 - 7.1 장점
 - 7.2 단점
- 8 결론

1.1서론

우리가 사용하는 대부분의 전자기기와 서비스는 통신기술을 기반으로 이루어집니다. 휴대폰으로 메시지를 주고받거나, 컴퓨터로 온라인 게임을 하고, 로봇이 센서 데이터를 서버에 전송하는 모든 과정이 통신을 통해 이루어집니다

해당 보고서에서는 통신 프로토콜 중 하나인 UDP통신을 중심으로 학습한 내용을 정리하였습니다. UDP는 인터넷에서 자주 쓰이는 전송 방식 중 하나로, 빠른 속도가 필요한 영상 스트리밍, 온라인 게임, 센서 데이터 교환 등에 활용됩니다.

2. 통신의 기본 개념

2.1통신의 정의

통신(Communication)이란 두 개 이상의 개체가 데이터를 주고받는 과정을 말합니다.

단순히 신호만 전달하는 것이 아니라, 정보 전달과 의미 해석이 동시에 이루어져야 합니다.

2.2 통신의 기본 요소

- 송신기(Transmitter): 데이터를 보내는 장치-> 스마트폰, 서버
- 수신기(Receiver): 데이터를 받는 장치-> PC, IoT 기기
- 전송 매체(Channel): 데이터를 전달하는 경로->유선(LAN), 무선(Wi-Fi)
- 프로토콜(Protocol): 원활한 통신을 위해 정해진 규칙과 약속-> TCP/IP

2.3 통신 방식의 분류

1. 단방향(Simplex): 한쪽에서 송신만 가능, 다른 쪽은 수신만 가능
2. 반이중(Half Duplex): 양쪽 송수신이 가능하지만 동시에 불가능
3. 전이중(Full Duplex): 양쪽에서 동시에 송수신 가능

2.4 유선/무선 통신

1. 유선 통신(Wired Communication).

- ① 정의: 물리적 케이블을 통해 데이터를 전송하는 방식

② 특징

- 1) 안정적이고 외부 간섭에 강하다
- 2) 고속 데이터 전송이 가능하다
- 3) 설치 시 물리적 제약이 크며, 이동성이 떨어진다

③ 활용 사례

- 1) 기업 내 유선 네트워크
- 2) 데이터 센터 및 서버 간 고속 전송

2. 무선 통신(Wireless Communication)

① 정의: 전파나 적외선, 위성을 통해 데이터를 전송하는 방식

② 특징

- 1) 이동성과 편의성이 뛰어나다
- 2) 설치 비용이 낮고 사용자 접근성이 좋다
- 3) 전파 간섭에 취약하고, 보안이 중요하다

③ 활용 사례

- 1) 스마트폰 및 태블릿 인터넷 접속
- 2) IoT 센서 네트워크
- 3) 로봇 원격 제어 및 모니터링

3. 통신 프로토콜

3.1 정의

:통신 프로토콜(Protocol)이란, 두 개체가 데이터를 주고 받을 때 지켜야 할 규칙과 약속을 의미한다. 하드웨어적으로는 신호가 잘 전달되더라도, 이를 어떤 순서와 형식으로 해석해야 하는지 정해져 있지 않으면 올바른 의사 전달이 불가능하다. 따라서 모든 네트워크 시스템은 프로토콜을 기반으로 동작한다.

3.2 주요 기능

: 프로토콜은 단순한 데이터 전달을 넘어, 통신의 신뢰성과 효율성을 높이기 위해 다음과 같은 기능을 수행한다.

① 주소 지장(Addressing)

- 데이터가 정확히 누구에게 전달되어야 하는지 지정
- 네트워크에서는 IP 주소와 포트 번호를 통해 통신 대상 구분

② 에러 제어(Error Control)

- 전송 도중 발생할 수 있는 오류를 검출 및 복구
- TCP 통신은 오류 시 재전송이 가능하지만, UDP 는 오류를 무시하거나 최소한의 체크섬만 제공

③ 동기화(Synchronization)

- 데이터가 정해진 순서대로, 적절한 시점에 도착할 수 있도록 보장
- 동기화가 맞지 ksgdmaus 데이터가 뒤섞이거나 손상될 위험이 존재

3.3 프로토콜 종류

① 응용 계층 프로토콜

- HTTP: 웹 브라우저와 서버 간 통신
- FTP: 파일 전송
- SMTP/IMAP: 이메일 송수신

② 전송 계층 프로토콜

- TCP: 신뢰성 있는 연결 기향 프로토콜
- UDP: 비연결성, 빠른 전송이 필요한 경우 사용

③ 네트워크 계층 프로토콜

- IP(Internet Protocool): 목적지까지 데이터 패킷을 전달

④ 데이터 링크 계층 프로토콜

- Ethernet: 근거리 유선 통신
- Wi-Fi: 무선 근거리 통신

⑤ 프로토콜 계층 구조 및 역할

계층	역할	주요 프로토콜
응용 계층	사용자 서비스 제공	HTTP, FTP, SMTP
표현 계층	데이터 변환, 암호화	JPEG, MPEG
세션 계층	연결 관리	NetBIOS
전송 계층	종단 간 데이터 전송	TCP, UDP
네트워크 계층	라우팅, 주소 지정	IP
데이터 링크 계층	물리적 주소 기반 통신	Ethernet, Wi-Fi
물리 계층	신호 전송	RS-232, 광섬유

4. UDP 의 개요

4.1 정의

:UDP(User Datagram Protocol)는 OSI 7 계층 중 전송 계층(4 계층)에 속하는 비연결형 프로토콜이다

4.2 주요 특징

- ① 비연결성(Connectionless)
 - 데이터를 전송하기 전에 연결을 설정하지 않는다.
 - 송신 측은 단순히 수신 측 주소(IP, Port)를 지정하고 데이터를 전송한다.
 - 수신 측이 준비되어 있지 않아도 송신이 가능하다.
- ② 비신뢰성(Unreliable)
 - 데이터가 손실되거나 순서가 바뀌더라도 이를 보장하지 않는다.
 - 오류 검출은 헤더의 Checksum 을 통해 최소한만 제공된다.
- ③ 빠른 전송 속도
 - TCP 보다 훨씬 단순한 구조이기 때문에 지연이 적고 처리 속도가 빠르다.
 - 실시간성이 중요한 서비스에 적합하다.

4.3 OSI 7 계층 내 위치

계층	역할	UDP 와의 관계
응용 계층	서비스 제공(HTTP, 게임, 스트리밍)	응용 계층에서 내려온 데이터를 UDP 에 전달
전송 계층	종단 간 데이터 전달	UDP 가 속하는 계층
네트워크 계층	IP 주소 기반 라우팅	UDP 는 IP 계층을 이용해 목적지까지 데이터 전송
데이터 링크/물리 계층	프레임 전송, 신호 전송	실제 전송 매체를 통해 전달

4.4 UDP 데이터그램 구조

: UDP 는 데이터를 데이터그램(Datagram)단위로 전송한다. 헤더는 8 바이트로 매우 단순하며, 다음과 같은 필드로 구성된다.

필드	크기	설명
출발지 포트(Source Port)	2 바이트	송신 측 포트 번호

목적지 포트(Destination Port)	2 바이트	수신 측 포트 번호
길이(Length)	2 바이트	헤더 + 데이터 전체 길이
체크섬(Checksum)	2 바이트	오류 검출용

4.5 TCP/UDP 비교

구분	TCP	UDP
연결 방식	연결형(3-way Handshake)	비연결형
신뢰성	신뢰성 보장(재전송, 순서 제어)	신뢰성 없음
속도	상대적으로 느림	빠름
데이터 단위	세그먼트(Segment)	데이터 그램(Datagram)
활용 분야	파일 전송, 웹 서비스	스트리밍, 게임, 센서 데이터

4.6 활용 예시

- ① 영상 스트리밍 서비스: 프레임 손실이 발생해도 다음 프레임으로 대체 가능
- ② 온라인 게임: 지연 최소화가 중요, 일부 데이터 손실 허용 가능
- ③ 센서 데이터 전송: 주기적으로 데이터 전송, 일부 손실 발생해도 전체 서비스는 큰 영향 x

5. UDP 의 동작 원리

5.1 데이터 전송 단위: 데이터 그램

: UDP 에서 데이터 전송은 데이터그램(Datagram)단위로 이루어진다. 데이터그램은 독립적으로 처리되며, 각각이 출발지와 목적지 정보를 포함한다. 따라서 전송된 데이터그램은 서로 다른 경로를 통해 도착할 수도 있고, 순서가 바뀌거나 일부가 손실될 수 있다.

5.2 송신 과정

- ① 응용 계층에서 데이터 생성

- ② 전송 계층에서 UDP 헤더를 붙여 데이터 그램 생성
- ③ 네트워크 계층(IP)을 거쳐 목적지 주소로 전달
- ④ 데이터 링크/물리 계층을 통해 실제 신호로 전송

→UDP 는 연결 설정 없이 바로 송신 가능하며, 목적지 IP 와 포트만 지정하면 된다

5.3 수신 과정

- ① 수신 측 장치는 특정 포트 번호를 통해 데이터 그램을 기다린다
- ② 네트워크 계층(IP)에서 데이터그램을 수신
- ③ 전송 계층(UDP)이 헤더를 확인하고 오류 검출을 수행
- ④ 응용 계층으로 데이터 전달

→Qt 에서는 readDatagram()함수를 통해 소켓에서 수신한 데이터를 읽을 수 있다.

5.4 포트 번호와 역할

UDP 통신에서 포트 번호는 프로세스를 구분하는 데 사용된다. 하나의 장치(IP) 여러 응용 프로그램이 동시에 UDP 를 사용할 수 있기 때문에, 포트 번호를 통해 구분해야 한다.

- ① 0~1023: 예약된 시스템 서비스용
- ② 1024~49151: 일반 애플리케이션 등록용
- ③ 49152~65535: 임시 연결 및 클라이언트 측 자동 할당
- ④ 실제 개발에서는 1024 이상의 포트 번호 사용이 권장된다.

5.5 동작 특성 요약

- ① 연결 과정: 없음, 송신 즉시 전송
- ② 데이터 단위: 데이터그램(Datagram)
- ③ 신뢰성: 보장되지 않음(순서 보장 x, 재전송 없음)
- ④ 오류 제어: 체크섬을 통한 최소한의 검출
- ⑤ 포트 번호: 프로세스 구분 및 데이터 수신 구명 역할
- ⑥ 적합한 활용 분야: 실시간성 요구 서비스(게임, 스트리밍, 센서 데이터)

→ UDP 는 데이터그램 단위로 동작하며, 송신 측은 연결 절차 없이 데이터를 전송하고 수신 측은 지정된 포트를 통해 데이터를 수신한다. 포트 번호를 통해

여러 응용 프로그램을 구분할 수 있으며, 신뢰성보다는 빠른 전송속도가
강점이다.

6. UDP 프로그래밍(카카오톡 구현 예제)

6.1 QUdpSocket 클래스

: Qt 에서 UDP 통신을 구현할 때는 QUdpSocket 클래스를 사용한다.

```
55 //메시지 수신 함수
56 void MainWindow::readPendingDatagrams()
57 {
58     while (udpSocket->hasPendingDatagrams()) {
59         QByteArray datagram;
60         datagram.resize(int(udpSocket->pendingDatagramSize()));
61         udpSocket->readDatagram(datagram.data(), datagram.size());
62
63         // QTextCursor 가져오기
64         QTextCursor cursor(ui->textBrowser->textCursor());
65         cursor.movePosition(QTextCursor::End);
66
67         // 문단 포맷: 왼쪽 정렬
68         QTextBlockFormat format;
69         format.setAlignment(Qt::AlignLeft);
70         cursor.insertBlock(format);
71
72         // 텍스트 출력 (파란색으로)
73         QTextCharFormat charFormat;
74         charFormat.setForeground(QBrush(Qt::blue));
75         cursor.insertText(QString("Friend: %1").arg(QString::fromUtf8(datagram)), charFormat);
76     }
77 }
```

6.2 데이터 수신

- 데이터는 QByteArray 형태로 처리된다.
- 수신 시에는 소켓에서 readDatagram()함수로 사용하여 데이터를 읽는다

```

23 //메시지 전송 함수
24 void MainWindow::on_sendButton_clicked()
25 {
26     QString message = ui->lineEditMessage->text().trimmed(); // 공백 제거 후 문자열 가져오기
27     if (message.isEmpty()) {
28         // 입력이 없으면 전송하지x
29         return;
30     }
31
32     QByteArray data = ui->lineEditMessage->text().toUtf8();
33     QHostAddress destIP(ui->lineEditIP->text());
34     quint16 destPort = ui->lineEditPort->text().toUShort();
35
36     // UDP 메시지 전송
37
38     udpSocket->writeDatagram(data, destIP, destPort);
39
40     // 내 메시지 출력: 오른쪽 정렬
41     QTextCursor cursor(ui->textBrowser->textCursor());
42     cursor.movePosition(QTextCursor::End);
43
44     QTextBlockFormat format;
45     format.setAlignment(Qt::AlignRight); // 오른쪽 정렬
46     cursor.insertBlock(format);
47     //상대 메시지 출력: 왼쪽 정렬
48     QTextCharFormat charFormat;
49     charFormat.setForeground(QBrush(Qt::black)); // 글자색
50     cursor.insertText(QString("Me: %1").arg(ui->lineEditMessage->text()), charFormat);
51
52     // 입력창 비우기
53     ui->lineEditMessage->clear();
54 }

```

6.3 데이터 송신

- 송신할 데이터를 QByteArray 에 저장한다.
- 대상 IP 와 포트 번호를 지정하여 writeDatagram() 함수로 전송한다.

7. UDP 의 장단점

7.1 장점

- 연결 과정이 없어 속도가 빠르다
- 단순한 구조로 구현이 용이하다
- 실시간 데이터 전송에 적합하다

7.2 단점

- 데이터 전송의 신뢰성이 보장되지 않는다
- 데이터 순서 보장이 되지 않는다
- 오류 발생 시 자동 재전송 기능이 없다

7.3 활용 사례

- 영상 스트리밍: 초당 수십 프레임 전송, 손실 시 다음 프레임으로 대체 가능
- 온라인 게임: 빠른 반응이 중요, 일부 데이터 손실 허용
- 센서 데이터 전송: 실시간 상태 전송에 활용
- 로봇 제어 신호: 지연보다 즉각 반응이 중요한 경우 적합