2011021764
강병욱
2017년 6월 4일

Project 3

Reliable UDP Chatting (Stop-&-Wait ARQ)

1. 클래스 설명

a) LLC.java

a) 설명

a) 추상 클래스

b) SFrame, IFrame, UFrame 의 부모 클래스

b) 변수

a) 정적 변수

a) 기본적인 LLC프레임의 각 정보 위치(PDU_INDEX 등)

b) 프레임을 판단하기 위한 변수(IS_ERROR 등)

c) 패킷의 최대 사이즈

b) 인스턴스 변수

a) 패킷에 정보를 넣기 위한 버퍼(buffer)

b) 패킷 정보를 담을 바이트 배열(data)

c) 메소드

a) 인스턴스 메소드

a) 패킷을 가져옴 : byte[] getData()

b) 패킷을 인스턴스에 할당 : void setData(byte[] bytes)

b) 정적 메소드

a) 패킷의 전체 크기를 알아냄 : int getDataSize(byte[] byte)

b) CRC32 값을 알아냄: long getCRC32Value(byte[] bytes)

c) CRC 값 검증: boolean checkCRC(byte[] bytes)

d) 패킷 타입 판단: int whichFrame(byte[] bytes)

c) 추상 메소드

a) 패킷을 시각화(디버깅): String byteArrayToHex()

b) SFrame.java

a) 설명

a) S-format 패킷 클래스

b) LLC 클래스를 상속

c) ACK/NACK 신호를 담당

b) 변수

a) 정적 변수

a) 패킷의 코드를 나타내는 변수(RR, RNR, REJECT)

b) 인스턴스 변수

a) 시퀀스 번호: int sequence

c) 메소드

a) 생성자

a) 코드, 시퀀스 번호를 인자로 받아 패킷을 구성.

b) Source, Destination address, SAP 등은 고려하지 않음

b) 인스턴스 메소드

a) 패킷으로부터 코드를 추출 : int extractCode()

b) 패킷으로부터 시퀀스를 추출: int extractSequence()

c) IFrame.java

a) 설명

a) I-format 패킷 클래스

b) LLC 클래스 상속

c) 메시지 전달을 담당

b) 변수

a) 정적변수

a) 패킷에서 정보가 들어갈 위치(INFORMATION_INDEX)

b) 인스턴스 변수

a) 시퀀스 번호: int sequence

c) 메소드

a) 생성자

a) 메시지와 시퀀스를 받아 패킷을 구성

b) 인스턴스 메소드

a) 패킷으로부터 메시지를 추출 : String extractMessage()

b) 패킷으로부터 시퀀스를 추출 : int extarctSequence()

d) UFrame.java

a) 설명

a) U-foramt 패킷 클래스

b) LLC 클래스 상속

c) Connection 담당

b) 변수

a) 정적변수

a) 코드값을 나타내는 변수(SABME, UA 등)

b) 메소드

a) 생성자

a) 코드값을 받아 패킷을 구성

b) 인스턴스 메소드

a) 패킷으로부터 코드값을 추출 : int extractCode()

e) UDPStopAndWaitServer.java

a) 설명

a) UDP 서버 실행

b) 무한 루프를 돌면서 클라이언트로부터의 메시지를 기다린다

b) 과정

a) 소켓을 생성한다

b) 해당 소켓으로부터 메시지를 기다린다

c) 메시지가 도착하면 우선 CRC 체크를 한다

a) CRC 체크 통과한 경우

a) 패킷의 format을 확인한다

a) I-frame의 경우

a) 해당 메시지로부터 메시지를 추출한다

b) 콘솔에 메시지 출력

c) U-frame ACK 패킷을 생성한 후 클라이언트에게 응답

a) 에러 발생한 경우

a) NACK 패킷을 생성한 후 클라이언트에게 응답

b) 1/2 확률로 타임아웃 발생 시킨다

a) 이때는 아무것도 응답해주지 않는다.

b) U-frame의 경우

a) 클라이언트의 연결 요청

b) SABME 패킷의 경우

a) UA 패킷을 생성하여 클라이언트에게 응답해준다.

c) S-frame의 경우

a) 단방향 통신이므로 고려하지 않는다.

b) CRC 체크를 통과하지 못한 경우

a) 클라이언트에게 S-format RNR 패킷을 보낸다

f) UDPstopAndWaitClinet.java

a) 설명

a) UDP 클라이언트 실행

b) 타임아웃 1초로 설정

c) 시퀀스를 0부터 1씩 올리면서 “Hello world!” 메시지를 보낸다

d) 타임아웃 또는 NACK가 온 경우 메시지를 재전송한다

b) 과정

a) 소켓 생성

b) 소켓을 통해 SABME 패킷을 보낸다

c) 응답을 기다린다

a) UA 패킷이 온 경우 I-Frame 전송을 시작한다

d) I-frame 패킷을 전송한다

e) 응답을 기다린다

a) 직전에 보낸 패킷의 시퀀스번호와 같은 ACK 가 온 경우

a) 시퀀스 번호를 1 증가시키고 d)로 돌아간다

b) 타임아웃에러 또는 NACK가 온 경우

a) 패킷을 그대로 다시 보낸다

2. 실행

a) **서버 실행(java -jar Server.jar <포트번호>)**

```
java -jar Server.jar 8080
```

b) **클라이언트 실행(java -jar Client.jar <호스트명> <포트번호>)**

```
java -jar Client.jar localhost 8080
```

c) **실행화면**

```
서버에게 SABME U-frame을 전송합니다.
서버로부터 UA 도착!! 연결되었습니다.
메시지: Hello world!(시퀀스 No.1)
서버로부터 ACK 도착(시퀀스 No.1)
메시지: Hello world!(시퀀스 No.2)
에러 발생ㅠㅠ(시퀀스 No.2 재전송)
메시지: Hello world!(시퀀스 No.2)
에러 발생ㅠㅠ(시퀀스 No.2 재전송)
메시지: Hello world!(시퀀스 No.2)
에러 발생ㅠㅠ(시퀀스 No.2 재전송)
메시지: Hello world!(시퀀스 No.2)
서버로부터 ACK 도착(시퀀스 No.2)
메시지: Hello world!(시퀀스 No.3)
```

<클라이언트>

1. SABME를 보내 서버와 연결
2. 서버로부터 UA가 도착하면 메시지 전송 시작
3. 서버로부터 ACK가 도착하지 않거나 (타임아웃: 1초) NACK가 도착하면 해당 메시지를 다시 보낸다.

```
CRC 통과.
U Frame입니다.
U-format SABME 패킷 도착.
CRC 통과.
I Frame입니다.
클라이언트로부터의 메시지: Hello world!
CRC 통과.
I Frame입니다.
타임아웃을 발생시키기 위해 아무것도 하지 않습니다.
CRC 통과.
I Frame입니다.
타임아웃을 발생시키기 위해 아무것도 하지 않습니다.
```

<서버>

1. U-frame의 SABME 패킷이 도착하면 클라이언트에게 UA 패킷 전송
2. 이후에 I-frame이 도착하면 ACK 패킷으로 응답해주거나 의도적으로 보내지 않아 타임아웃 에러를 발생시킨다.