Kerschel James : 814004296

Gerard Rique:814002944

Date: 3/11/2018

Advanced networks

To compile use javac Client.java and javac Server.java

To run the program use java Client and java Server

The server and client defaults to the local host network.

Structure of Frame

This was the structure of how each frame was packaged and sent over the network to the Server. We decided to put the End Flag before the End of Packet since we used a variable size payload, we need to know where the payload ends. We used the End Flag to determine this.

8	*8*	*8*	*160-480*	*8*	*8*
Start flag	Sequence #	Error byte	Payload	End Flag	End of packet

Input File

The input.raw file contains as outlined in the document the p number of packets followed by a number indicating the size of the packet and the packet. In our code, we only catered for hexstrings that do not have spaces in them. E.g FF EF has to be FFEF. This was a result of how we read the data from the file. We used https://www.rapidtables.com/convert/number/ascii-to-hex.html to generate hex from ascii. Then we removed all the spaces.

Error simulation

Client: This piece of code was used to simulate the error on the client side for every 5th transmissions. The FlipBit() function changes a bit in the CRC code. Therefore, when the server recalculates it an error will be seen.

```
clientLog(j, i, state: 1);

DataLinkLayer dataLinkLayer = new DataLinkLayer( start: "7E", end: "7E", String.valueOf(i),payload, packetend);
if(error %5 == 0) {dataLinkLayer.FlipBit();}

outToServer.writeBytes( s: dataLinkLayer.returnFrame() + "\n");
```

In addition, the timer is started after each packet is send by the client and waits 5 seconds for an ACK. If ACK is received before the timeout, it is canceled.

```
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        System.out.println("timeout");
        resend previous frame because no ACK received
        try {
            clientLog(finalJ, frameNo, state: 6);
            clientLog(finalJ, frameNo, state: 2);
            outToServer.writeBytes( s: new DataLinkLayer( start: "7E",
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}, delay: 10 *  1000); // wait for 10 secs

          Get ACK from server
String ACK = inFromServer.readLine();
received good cancel the resend of frame
if (Integer.parseInt(ACK, radix: 2) == i) {
    timer.cancel();
```

The assignment did not specify when the client should send a duplicate frame, therefore we decided to send a duplicate at every 3rd transmission of a packet with more than 1 frame. Lines 43-45

```
if(error %3 == 0 && i ==2){
    i--;// to simulate sending a duplicate packet/frame. It was not specified when
this should be done
}
```

Server: The server waits every 8th transmissions to send an error. This is where the simulation of error occurs. The comments explains it. Lines 64-108

```
                    if(new CRC().validateData(inputData ) && ((error%8) !=0)){ // if
it is true
                        if(endOfPacket.equals("0") &&
sequenceList.indexOf(data.getSequence()) < 0){ // so if that sequence was never sent
before
                            message += payload;
                            sequenceList.add(data.getSequence());
                        }
                        else if(endOfPacket.equals("1") &&
sequenceList.indexOf(data.getSequence()) <0){ // So here we have the entire packet
together
//                              Packet received
                            serverLog(packetNo,frameNo,3);
                            message += payload;
                            // so if that sequence was never sent before
                            sequenceList.add(data.getSequence());
//                        end of packet

                            hex = new BigInteger(message, 2).toString(16);//Convert
the binary message received from the client to hexadecimal
                            outputData.println(hex);//output the hexadecimal message
to the output file.
                            System.out.println(message + " Sequence " +
data.getSequence());
                            message ="";
                            sequenceList.clear();

                            packetNo++;
                            frameNo=1;

                        }
                        else if(sequenceList.indexOf(data.getSequence()) >
0){//duplicate packet or frame received
                            serverLog(packetNo,frameNo,4);
                            frameNo --;
                        }
//                        ACK sent to get next packet
                        serverLog(packetNo,frameNo,5);
                        out.writeBytes(data.getSequence() + "\n");
                    }else {
                        if (!new CRC().validateData(inputData )) { // wrong crc
calculated so want them to resend
                            serverLog(packetNo, frameNo, 2);
                            out.writeBytes("00010011" + "\n");
                        }
```

```
                        else{//everything is well just want to simulate a error by
sending nothing back

                        }
```