An-Najah National University

Faculty of Engineering

جامعة النجاح الوطنية

كلية الهندسة

Computer Engineering Department
Computer Networks 1(10636454)
Homework 3
Spring 2021
Dr. Raed Alqadi & Dr. Abdullah Rashed

Mini Project: Network Programming

In this project, you will write software to transfer a file between a client and server (or peer-to-peer) by using UDP. In the first part, you will use UDP only, hence no reliability. In the second part, you will use UDP, but you will add reliability to the transfer by implementing the simple **stop and wait protocol**. You will use Java to write the software. The file will be chopped into packets and transmitted packet by packet, we will restrict the packet data part to 1K or 512 bytes. The File can be any type of file, so we will assume any binary file.

Project: Two Parts

1. Part1: Convert the chatting Program you wrote in the previous homework to Send/Receive files y using UDP. You will Read Chunks of data with Max Size 1024 or 512 (Your choice) and send each Chunk in a UDP Datagram. Since you are using UDP you may have errors. The file transfer may not work.

2. **Part 2**: In this part, you will make the code in part 1 Reliable and we will use UDP**.**

    - But this time you have to do more. Define the packet as a Class, but this time you will have to add more fields like sequence-number, checksum, and type, one of the types should be ACK. Add any additional fields that you need. One of the fields will be the data (chunk of data you read from the file).
    - You will need to read a Chunk of data (e.g., 1K byte), create a Packet, then fill the header field and data and Send the Packet. You will need to use byte arrays or ByteBuffer. See the attached code that I wrote to help you. I created a simple packet; you will need to define actual fields. Follow what I did and it should be straight forward. Modify the Class Packet and you should be good to go.
    - Use Stop-and -go to guarantee reliability.
    - Also, we will need to use a function to inject error to force re-transmission. You can do that by creating a probability of error say 5% as follows. Add a text box to select the error rate. Generate a Random Number say from 0 –1000 if the umber is between 0 and 5% of the 1000 which means between 0 – 50, then send the packet with error. You can just change the Checksum to an error value.
    - To Compute Checksum, you can just add the Bytes in the packet and store the result in Checksum (int = 32 bit number
    - You will need a timer to retransmit the Packet if you do not receive an ACK within a certain amount of Time.
    - The Maximum data size should be set to 1K or 512 bytes.
    - Use a Nice GUI, you will also need to show the count of Retransmissions at the Sender Side and the count of Errors at the Receiver side.

- Work in Groups of 2

Some Help:

1. We are showing how to define a packet C. In C it is straight forward, all you need to do is typecast the structure to a char array.
2. To help you in Java, you will have to use either byte array and encode integers, floats ...etc. yourself. Or you can use ByteBuffer. To help you with that, see the attached jBuf.rar and try it. You will need to use the Packet class defined there, but you need first to change the fields to SeqNo, checksum, ...etc. Add whatever you like, but you need to adjust the header offsets accordingly. Everything is explained there.

1. Example of Packet in C (we will use Java) see #2 in page 2. The C Packet structure shows fields we need, adjust the code in java shown after it and also in the jBuf.rar accordingly.

```
#define MAX_DATA_SIZE  1024
enum PACKET_TYPE         // Do not use Enumerate in Java it Stinks , use constant values
{
    START,
    ACK,
    DONE,
    DATA,

     //all your types
    MAX_TYPE=0x0FFFF //forces to 16bit

};

struct PACKET {
    PACKET_TYPE type; // Chang to int in Java
    unsigned short length; // number of data bytes not all
    unsigned short Checksum; // Change to Intadd this in the UDP code, to do a check sum
    // other fields
    // --
    unsigned char data[MAX_DATA_SIZE]; // Change t Javawhen you compute checksum, add only
the bytes specified by length
    // or initialize the data to zeros before you fill it and add all
};
```

2. Packet Class taken from the jBuf example. I am showing general fields, you need to modifiy these and change them as explained above.

```
package jbuf;

import java.nio.*;

public class Packet {

   static final int MAX_DATA_LENGTH = 20;// Change this to 1024 or 512
   /*
     * Similar to structure, I will make them public but that is a Very Bad idea
```

```
  * These should be made private and use Accessors, Successors
  * These are not actual Packet fields, but will be similar
 */
public char charValue = 0; // 2 bytes = b.getChar(0); // index 0, 2 bytes
public int intValue = 0; //4 bytes  b.getInt(0 + 2 + 4);// stored at index 2
public float floatValue = 0; // 4 bytes = ; //4byte b.getFloat(0 + 2 + 4);//offset 6
public int dataLength = 0; //4 bytes; = b.getInt(0 + 2 + 4+4);

public ByteBuffer buffer = null;
byte[] data = new byte[MAX_DATA_LENGTH];
static final int HEADER_LENGTH = (int) (2 + 4 + 4 + 4);// Constant, adjust to your case

// You can add constructor to allocate the same as the data length instead of max
public Packet() {
    init('A', 0, 0);
}

public Packet(char cv, int iv, float fv) {
    this.init(cv, iv, fv);
}

private void init(char cv, int iv, float fv) {
    charValue = cv; // 2 bytes = b.getChar(0); // index 0, 2 bytes
    intValue = iv; //4 bytes  b.getInt(0 + 2 + 4);// stored at index 2
    floatValue = fv; // 4 bytes = ; //4byte b.getFloat(0 + 2 + 4);//offset 6
    dataLength = 0; //4 bytes; = b.getInt(0 + 2 + 4+4);
    buffer = null;

}
// You can read a chunck of data from a file and add it by using this functio
public boolean addData(byte[] data, int n) {// n is the number of bytes to add
    // n should be more the data.length
    dataLength = n;
    if (n > MAX_DATA_LENGTH) {
        System.out.println("Data too big");
        return false;
    }
    for (int i = 0; i < n; i++) {
        this.data[i] = data[i];
    }
    return true;
}

public ByteBuffer toByteBuffer() {
    buffer = ByteBuffer.allocate(HEADER_LENGTH + MAX_DATA_LENGTH);
    buffer.clear();
    buffer.putChar(charValue); // 2 bytes;
    buffer.putInt(intValue);//4 bytes

    buffer.putFloat(floatValue);//4 bytes

    buffer.putInt(dataLength);  // Save this, it is important to save it
    buffer.put(data, 0, dataLength);//copy ony the vailable bytes
    return buffer;

}
```

```
public void printPacketAsArray() {
    if (buffer != null) {
        Packet.printBufferHex(buffer, this.getPacketLength());
    } else {
        System.out.println("Buffer is , Execute to Buffer");
    }

}

static void printBufferHex(ByteBuffer b, int limit) {
    //int limit = b.limit();
    String S = "[";
    for (int i = 0; i < limit; i++) {
        S += String.format("%02X", b.array()[i] & 0x00ff);
        if (i != (limit - 1)) {
            S += ", ";
        }
    }
    S += "]\n";
    S += "Position: " + b.position()
            + "\nLimit: " + b.limit();
    System.out.println(S);
}

public void printPacket() {
    String S = "";
    S += ("Packet Cotents= { ");
    S += ("\n\tcharValue: " + this.charValue);
    S += ("\n\tintValue: " + this.intValue);
    S += ("\n\tfloatValue: " + this.floatValue);
    S += ("\n\tdataLength: " + this.dataLength);

    int limit = dataLength;
    S += "\n\tdata =[\n\t";
    for (int i = 0; i < limit; i++) {
        S += String.format("%02X", (byte) data[i] & 0x00ff);
        if (i != (limit - 1)) {
            S += ", ";
        }
    }
    S += "]";
    S += "\n}\n";
    System.out.println(S); // instead of this use next
    // return S; // use this to return a String to print it in GUI
    // change function type to String
}

public int getPacketLength() {
    return HEADER_LENGTH + dataLength;
}

public void extractPacketfromByteBuffer(ByteBuffer buf) {
    try {
        char cV = buf.getChar(0); // index 0, 2 bytes
        int iV = buf.getInt(0 + 2 + 4);// stored at index 2
```

```
            float fV = buf.getFloat(0 + 2 + 4);//offset 6
            int dataLen = buf.getInt(0 + 2 + 4 + 4);
            //  System.out.println(">> BaLength:= "+dataLen);
            byte[] ba = new byte[dataLen];
            for (int i = 0; i < dataLen; i++) {
                byte bt = (byte) buf.get(0 + 2 + 4 + 4 + 4 + i);
                ba[i] = bt;
            }
            this.charValue = cV;
            this.data = ba;
            this.dataLength = dataLen;
            this.floatValue = fV;
            this.intValue = iV;
            //    this.printPacket(); // to show it works

        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }

}
```