An-Najah National University

Faculty of Engineering

جامعة النجاح الوطنية

كلية الهندسة

Computer Engineering Department

Computer Networks 1(10636454)

Homework2

Student

Instructor Name: Dr. Raed Alqadi
Academic Year:2018/2019
Semester: Summer
Credit Hours: 3
Date: Part1: July 28

Dr. Raed Alqadi
Student Name: Ahmad
Registration Number:
Section: 2
Total Project Mark:
Project +HW Weight: 15%

Grades

Part	Description	Points	ILO's		Part Grade
Part 1	TCP File Transfer		2		
Part 2	UDP File Transfer		2		
Student Grade (of 30)					

Project Notes:

1- Submit every part of the Program (Software on time)
2- Use good programming practices and style.
3- Read the specs of the program on the next page carefully.

Project: Network Programming

In this project, you will write software to transfer a file between a client and server by using two methods. In the first part, you will use TCP, hence reliability is automatic. 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 QT 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 byte.
The File can be any type of file, so we will assume any binary file

Project: Two Parts

1. **Part 1:** Write Clients and Server code to transfer the file by using TCP.
   Define a packet as a structure, with fields: type, length(data length), and data (char array of max size 1k). Add additional fields if you want. We do not need checksum or sequence numbers because TCP guarantees the reliability. In the Type, you may use: START, DONE, DATA, ..etc. whatever you want. When you establish the connection. You can put the name of the file, length of the file in the data part. After that the data should contain actual data from the file. File

2. **Part 2:** Write Clients and Server code to transfer the file by using UDP. Define the packet as a structure, but this time you will have to add more fields like sequence-number, checksum, and of the types should be ACK. Add any additional fields that you need.
   Use Stop-and -go to guarantee reliability. You will have to use a QT timer for retransmission. Also, we will need to use a function to inject error to force retransmission. I will explain more about this in class.

   Some Extra help data will be provided below

Work in Groups of 2.

TCP Server Header file

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtNetwork/QTcpSocket>
#include <QtNetwork/QTcpServer>
#include <QMainWindow>

#define MAX_DATA_SIZE  1024
enum PACKET_TYPES
{
    START,
    ACK,
    DONE,
    DATA,

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

};

struct PACKET {
    PACKET_TYPE type;
    unsigned short length; // number of data bytes not all
    //unsigned long checksum; // add this in the UDP code, to do a check sum
    // other fields
```

```
    // --
    unsigned char data[MAX_DATA_SIZE]; // when you compute checksum, add only the bytes
specified by length
    // or initialize the data to zeros before you fill it and add all
};

namespace Ui {
class MainWindow;
}
//server
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
    QTcpServer *server;
    QTcpSocket* clientSocket;
    //QList <QTcpSocket*> clientList; // if you need multiple connections
    void startListening();
    sendData(PACKET *pkt);

public slots:
    void acceptConnection();
    void readClientData();

};

#endif // MAINWINDOW_H
```

Part of TCP server cpp file

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

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

    server = new QTcpServer(0);
    connect(server, SIGNAL(newConnection()),
       this, SLOT(acceptConnection()));

    //start listing here or put a button on which we start to listen

   //
}

MainWindow::~MainWindow()
{
```

```
    delete ui;
}

void MainWindow::acceptConnection()
{
    clientSocket = server->nextPendingConnection();
   // clientList.append(clientSocket);// append it to list if you want to accept other
sockets

    connect(clientSocket, SIGNAL(readyRead()),
            this, SLOT(startReadreadClientData()));
    //other code if needed
    // if you need multiple connections add to List of type sockets
}
void MainWindow::readClientData()
{
    char buffer[2048] = {0}; // make it bigger than your packet
    qDebug() << "receiving ..";

    //QTcpSocket* cl = qobject_cast<QTcpSocket*>(QObject::sender());
    // cl should be same client if only one connection f

    QTcpSocket* cl = clientSocket; // search the list for multiple connection, here on
    int bytesReceived = cl->bytesAvailable();
    cl->read(buffer, bytesReceived);

    //PACKET  *pkt = (PACKET *)buffer; // use tis to access the fields of your PACKET
    // for example pkt->type ... packet->data ..etc


}
// add a button to start listing and call this function

 MainWindow::startListening()
 {
     // fill the address and port number from your edit boxes

     server->listen(QHostAddress(ui->ipEdit->text()), ui->portEdit->text().toInt());
 }
 // you need to read the data from a file of max 1k bytes and send it

  MainWindow::sendData(PACKET *pkt)
  {
      clientSocket->write((const char *)pkt, sizeof(pkt));
  }

To write any data use any of the CLientSocket  clientSocket->write ( …
```

TCP Client Side :
Just Create a TCP Socket and Bind it to local IP and addess

Here is a snippet of code that should help

TO start a connection:

```
clientSocket = new QTcpSocket(0);
clientSocket->connectToHost(QHostAddress::LocalHost,8888);
if(clientSocket->waitForConnected(1000))
{
        connect(clientSocket, SIGNAL(readyRead()),
              this, SLOT(onReceiveFromServer()));empty strings*/
}
```

Do not forget to add the slot above in the header.
It is similar to UDP

To Send Data: you can use the code above o

```
MainWindow::sendData(PACKET *pkt)
{
    clientSocket->write((const char *)pkt, sizeof(pkt));
}
```

To receive data .. it is the same as the code in the server MainWindow::readClientData()

```
void MainWindow:: onReceiveFromServer ()
{
    char buffer[2048] = {0}; // make it bigger than your packet
    qDebug() << "receiving ..";

    int bytesReceived = clientSocket ->bytesAvailable();
    cl->read(buffer, bytesReceived);

    //PACKET  *pkt = (PACKET *)buffer; // use tis to access the fields of your PACKET
    // for example pkt->type ... packet->data ..etc


}
```

Timer

Here is an example of using timers, as shown in this simple application:

// Header file for timer example

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}
```

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:  // these are auto generated when u add the buttons, this is the example
shown In class
    void on_pushButton_clicked();

    void on_recvB_clicked();

    void on_pushStartTimer_clicked();

    void on_pushStartTimer_2_clicked();

private:
    Ui::MainWindow *ui;
    void timerEvent(QTimerEvent *event);  // need this
    int timerId;                          // and this
    int count;
};
#endif // MAINWINDOW_H
```

And here is the CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

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

}

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

void MainWindow::on_pushButton_clicked()
{
    ui->lineEdit->setText("hello");
}

void MainWindow::on_recvB_clicked()
{
    ui->recvB->setText("bb");
}

void MainWindow::on_pushStartTimer_clicked()
{
```

```
    timerId = startTimer(1000); //every 1000ms => 1sec
    count =0;

   // you can run several timers with different Ids
//    id1 = startTimer(50);     // 50-millisecond timer
//    id2 = startTimer(1000);   // 1-second timer
//    id2 = startTimer(60000);  // 1-minute timer
}

void MainWindow::timerEvent(QTimerEvent *event)
{
    //qDebug() << "Timer ID:" << event->timerId();
    QString qs;
    qs.sprintf("Timer Id %d, event->timerId =%d, count =%d", timerId,event->timerId(),
count++);

     ui->lineEdit->setText(qs);
}

void MainWindow::on_pushStartTimer_2_clicked()
{
    killTimer(timerId);
}
```

The UI for the simple timer example is