姓名：**陈华豪**

学号：**6130116238**

邮箱地址：**6130116238@email.ncu.edu.cn**

专业班级：**网络工程161班**

实验日期：**2018.11.18**

课程名称：**网络协议分析与实现**

实验项目名称

Home Work 4

实验目的

- UDP拥塞控制

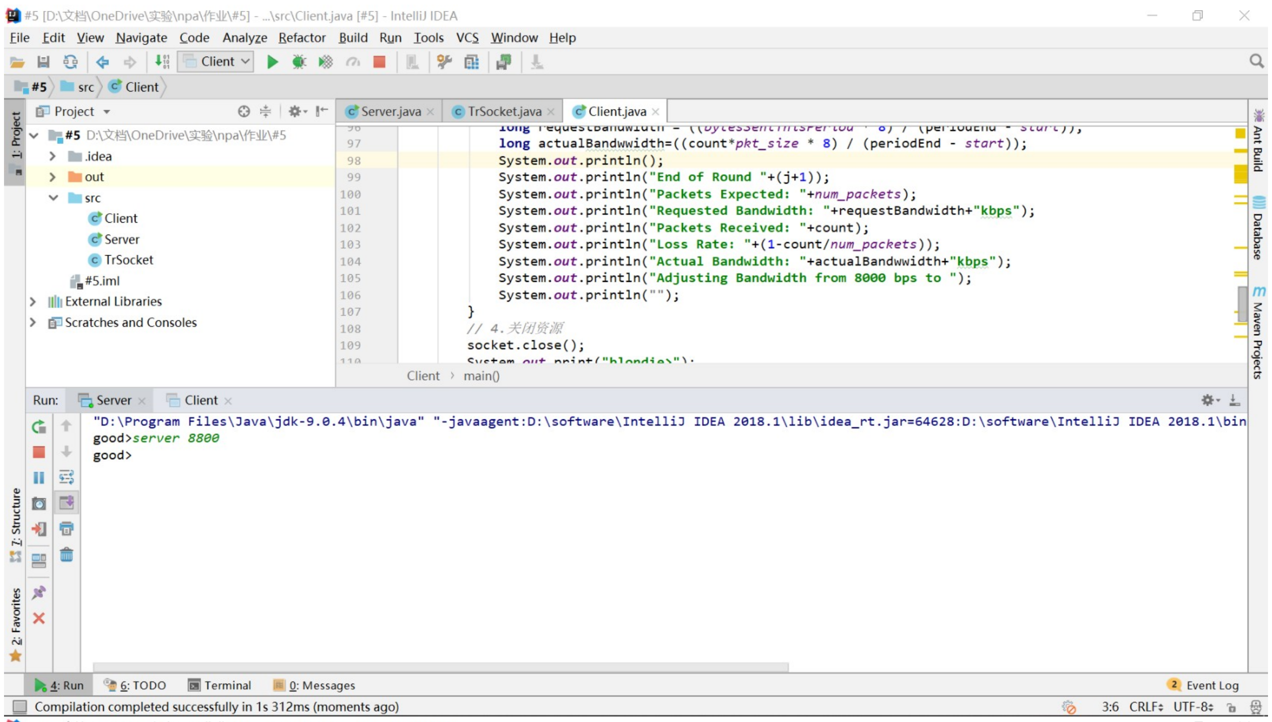
实验基础

- http://good.ncu.edu.cn/doc/h4.pdf

实验步骤

1. Server command line parameters.
2. Client command line parameters.
3. Format of the message from the client to the server.
4. Format of the data packets sent from the server to the client.
5. Operation and output of the client.
6. Ending the test and closing the programs.

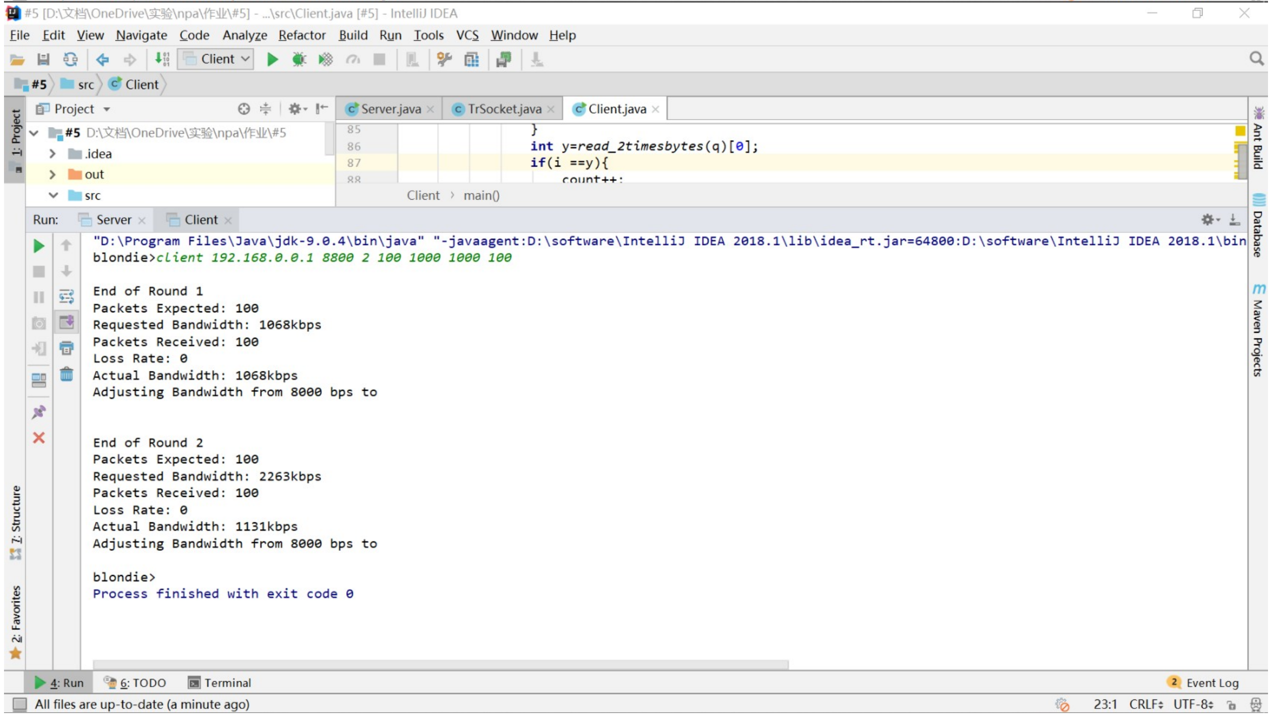
实验数据或结果


#5 [D:\文档\OneDrive\实验\npa\作业\#5] - ...\src\Client.java [#5] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Server.java
TrSocket.java
Client.java
Run: Server Client
4: Run
6: TODO
Terminal
0: Messages
Event Log

```java
        long requestBandwidth = ((bytesSentThisPeriod * 8) / (periodEnd - start));
        long actualBandwwidth=((count*pkt_size * 8) / (periodEnd - start));
        System.out.println();
        System.out.println("End of Round "+(j+1));
        System.out.println("Packets Expected: "+num_packets);
        System.out.println("Requested Bandwidth: "+requestBandwidth+"kbps");
        System.out.println("Packets Received: "+count);
        System.out.println("Loss Rate: "+(1-count/num_packets));
        System.out.println("Actual Bandwidth: "+actualBandwwidth+"kbps");
        System.out.println("Adjusting Bandwidth from 8000 bps to ");
        System.out.println("");
    }
    // 4.关闭资源
    socket.close();
    System.out.print("blondie>");
```

```
"D:\Program Files\Java\jdk-9.0.4\bin\java" "-javaagent:D:\software\IntelliJ IDEA 2018.1\lib\idea_rt.jar=64628:D:\software\IntelliJ IDEA 2018.1\bin
good>server 8800
good>
```

Compilation completed successfully in 1s 312ms (moments ago)


#5 [D:\文档\OneDrive\实验\npa\作业\#5] - ...\src\Client.java [#5] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Server.java
TrSocket.java
Client.java
Run: Server Client
4: Run
6: TODO
Terminal
Event Log

```java
        }
        int y=read_2timesbytes(q)[0];
        if(i ==y){
            count++;
```

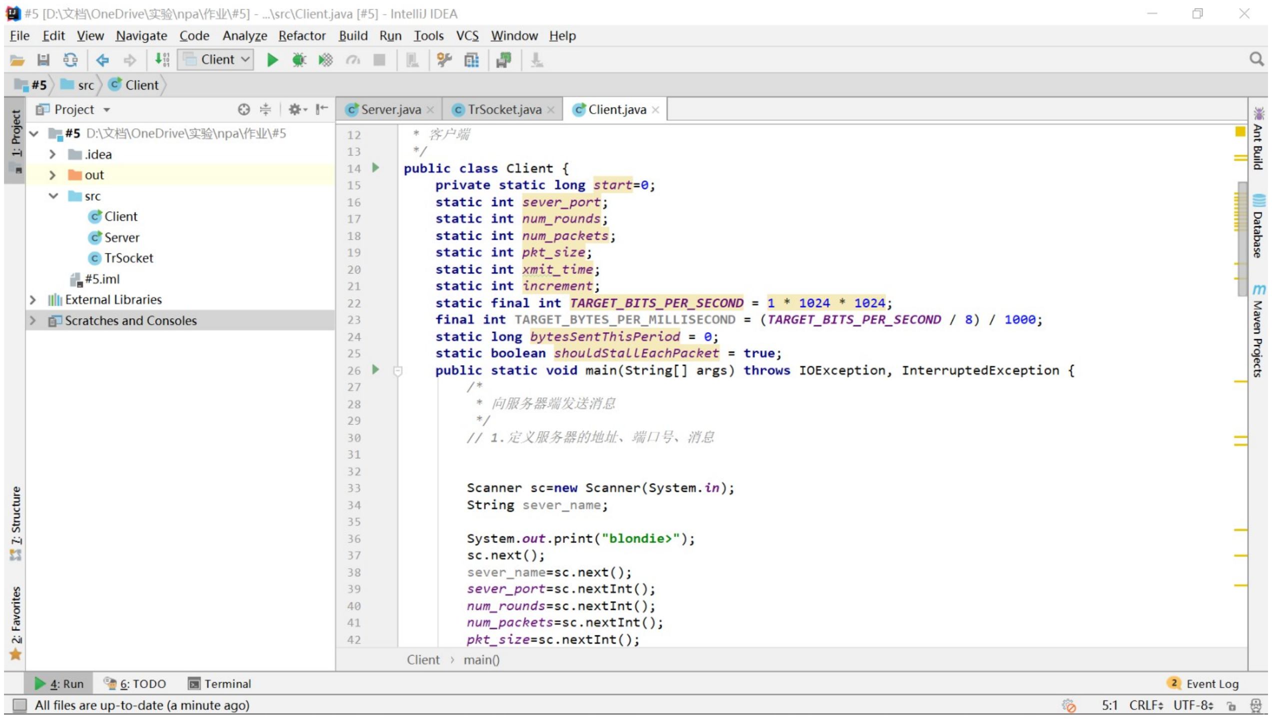
```
"D:\Program Files\Java\jdk-9.0.4\bin\java" "-javaagent:D:\software\IntelliJ IDEA 2018.1\lib\idea_rt.jar=64800:D:\software\IntelliJ IDEA 2018.1\bin
blondie>client 192.168.0.0.1 8800 2 100 1000 1000 100

End of Round 1
Packets Expected: 100
Requested Bandwidth: 1068kbps
Packets Received: 100
Loss Rate: 0
Actual Bandwidth: 1068kbps
Adjusting Bandwidth from 8000 bps to


End of Round 2
Packets Expected: 100
Requested Bandwidth: 2263kbps
Packets Received: 100
Loss Rate: 0
Actual Bandwidth: 1131kbps
Adjusting Bandwidth from 8000 bps to

blondie>
Process finished with exit code 0
```

All files are up-to-date (a minute ago)



```java
     * 客户端
     */
public class Client {
    private static long start=0;
    static int sever_port;
    static int num_rounds;
    static int num_packets;
    static int pkt_size;
    static int xmit_time;
    static int increment;
    static final int TARGET_BITS_PER_SECOND = 1 * 1024 * 1024;
    final int TARGET_BYTES_PER_MILLISECOND = (TARGET_BITS_PER_SECOND / 8) / 1000;
    static long bytesSentThisPeriod = 0;
    static boolean shouldStallEachPacket = true;
    public static void main(String[] args) throws IOException, InterruptedException {
        /*
         * 向服务器端发送消息
         */
        // 1.定义服务器的地址、端口号、消息


        Scanner sc=new Scanner(System.in);
        String sever_name;

        System.out.print("blondie>");
        sc.next();
        sever_name=sc.next();
        sever_port=sc.nextInt();
        num_rounds=sc.nextInt();
        num_packets=sc.nextInt();
        pkt_size=sc.nextInt();
```

代码：

Sever.java

```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Random;
import java.util.Scanner;

public class Server {
    static int num_rounds;
    static int num_packets;
    static int pkt_size=1000;
    static int xmit_time;
    static final int TARGET_BITS_PER_SECOND = 1 * 1024 * 1024;
    final static int TARGET_BYTES_PER_MILLISECOND = (TARGET_BITS_PER_SECOND / 8) / 1000;
    public static void main(String[] args) throws IOException, InterruptedException {
        Scanner sc=new Scanner(System.in);

        System.out.print("good>");
        sc.next();
        int listenport =sc.nextInt();
        System.out.print("good>");
        DatagramSocket socket = new DatagramSocket(listenport);

        byte[] data = new byte[8];
        DatagramPacket packet = new DatagramPacket(data, data.length);
        socket.receive(packet);

        int[] dataInt= read_2timesbytes(data);

        num_rounds=dataInt[0];
        num_packets=dataInt[1];
        pkt_size=dataInt[2];
        xmit_time=dataInt[3];

TrSocket trSocket =new TrSocket(socket);
        //向客户端响应数据
        //定义客户端的地址、端口号、数据
        for (int j = 0; j <num_rounds ; j++) {
            InetAddress address = packet.getAddress();
            int port = packet.getPort();
            int sequenseNum =0;
            byte[] data2;
            DatagramPacket packet2;
            for (int i = 0; i <num_packets ; i++) {
                data2=new byte[pkt_size];
                new Random().nextBytes(data2);
                int[] q={sequenseNum};
                for (int k = 0; k < 2; k++) {
                    data2[k]= (bytetimes2(q)[k]);
                }
                // 创建数据报，包含响应的数据信息
                packet2 = new DatagramPacket(data2, data2.length, address, port);
                // 响应客户端
                trSocket.send(packet2);
                sequenseNum++;
            }
        }
        // 关闭资源
        socket.close();



    }
    public static int[] read_2timesbytes(byte[] bytes){
        int[] x=new int[bytes.length/2];
        for (int i = 0; i <x.length; i++) {
            int t=bytes[2*i]&0xFF;
            t=(t<0?t+256:t);
            t+=(bytes[2*i+1]&0xFF)*256;
            x[i]=t;
        }
        return x;
    }
    public static byte[] bytetimes2(int[] x) {
        byte[] bytes2= new byte[x.length*2];
        for (int i = 0; i < x.length; i++) {
            bytes2[i*2]=(byte)(x[i]);
            bytes2[i*2+1]=(byte)(x[i]>>8);
        }
        return bytes2;
    }
}
```

Client.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Scanner;

import static java.lang.System.currentTimeMillis;

public class Client {
    private static long start=0;
    static int sever_port;
    static int num_rounds;
    static int num_packets;
    static int pkt_size;
    static int xmit_time;
    static int increment;
    static final int TARGET_BITS_PER_SECOND = 1 * 1024 * 1024;
    final int TARGET_BYTES_PER_MILLISECOND = (TARGET_BITS_PER_SECOND / 8) / 1000;
    static long bytesSentThisPeriod = 0;
    static boolean shouldStallEachPacket = true;
    public static void main(String[] args) throws IOException, InterruptedException {


        Scanner sc=new Scanner(System.in);
        String sever_name;

        System.out.print("blondie>");
        sc.next();
        sever_name=sc.next();
        sever_port=sc.nextInt();
        num_rounds=sc.nextInt();
        num_packets=sc.nextInt();
        pkt_size=sc.nextInt();
        xmit_time=sc.nextInt();
        increment=sc.nextInt();


        InetAddress address = InetAddress.getByName("localhost");
        int[] dataInt={num_rounds,num_packets,pkt_size,xmit_time};
        byte[] data = bytetimes2(dataInt);
        // 2.创建数据报，包含发送的数据信息
        DatagramPacket packet = new DatagramPacket(data, data.length, address, sever_port);
        // 3.创建DatagramSocket对象
        DatagramSocket socket = new DatagramSocket();
        // 4.向服务器端发送数据报
        socket.send(packet);

        /*
         * 接收服务器端响应的数据
         */
        //创建数据报，用于接收服务器端响应的数据
        for (int j = 0; j < num_rounds; j++) {
            int count=0;
            start=0;
            byte[] data2;
            DatagramPacket packet2;
            int[] count1 =new int[num_packets];
            for (int i = 0; i <num_packets ; i++) {
                if (start == 0) {
                    start = currentTimeMillis();
                }
                data2 = new byte[pkt_size];
                if (shouldStallEachPacket) {
                    Thread.sleep(1);
                }
                // 2.接收服务器响应的数据
                packet2 = new DatagramPacket(data2, data2.length);

                // 3.读取数据
                socket.receive(packet2);
                bytesSentThisPeriod += packet2.getLength();
                String reply = new String(data2, 0, packet2.getLength());
                byte[] q = new byte[2];
                for (int k = 0; k <2 ; k++) {
                    q[k]=data2[k];
                }
                int y=read_2timesbytes(q)[0];
                if(i ==y){
                    count++;
                }




            }

            long periodEnd = currentTimeMillis();
            long requestBandwidth = ((bytesSentThisPeriod * 8) / (periodEnd - start));
            long actualBandwwidth=((count*pkt_size * 8) / (periodEnd - start));
            System.out.println();
            System.out.println("End of Round "+(j+1));
            System.out.println("Packets Expected: "+num_packets);
            System.out.println("Requested Bandwidth: "+requestBandwidth+"kbps");
            System.out.println("Packets Received: "+count);
            System.out.println("Loss Rate: "+(1-count/num_packets));
            System.out.println("Actual Bandwidth: "+actualBandwwidth+"kbps");
            System.out.println("Adjusting Bandwidth from 8000 bps to ");
            System.out.println("");
        }
        // 4.关闭资源
        socket.close();
        System.out.print("blondie>");

    }
    public static byte[] bytetimes2(int[] x) {
        byte[] bytes2= new byte[x.length*2];
        for (int i = 0; i < x.length; i++) {
            bytes2[i*2]=(byte)(x[i]);
            bytes2[i*2+1]=(byte)(x[i]>>8);
        }
        return bytes2;
    }
    public static int[] read_2timesbytes(byte[] bytes){
        int[] x=new int[bytes.length/2];
        for (int i = 0; i <x.length; i++) {
            int t=bytes[2*i]&0xFF;
            t=(t<0?t+256:t);
            t+=(bytes[2*i+1]&0xFF)*256;
            x[i]=t;
        }
        return x;
    }
}
```

TrSocket.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

import static java.lang.System.currentTimeMillis;


class TrSocket {

    static final int TARGET_BITS_PER_SECOND = 1 * 1024 * 1024;         // 1 M bit

    final static int TARGET_BYTES_PER_MILLISECOND = (TARGET_BITS_PER_SECOND / 8) / 1000;


    final static int BYTES_BETWEEN_DELAY_CHECKS = TARGET_BYTES_PER_MILLISECOND * 100;

    static DatagramSocket socket;
    static long start = 0;
    static long bytesSentThisPeriod = 0;
    static boolean shouldStallEachPacket = true;

    public TrSocket(DatagramSocket socket) {
        this.socket = socket;
    }

    TrSocket(int port) throws SocketException {
        this.socket = new DatagramSocket(port);
    }

    void send(byte[] bytes, InetAddress addr, int port) throws IOException, InterruptedExcept
        DatagramPacket packet = new DatagramPacket(bytes, bytes.length, addr, port);
        send(packet);
    }


    static void send(DatagramPacket packet) throws IOException, InterruptedException {

        if (start == 0) {
            start = currentTimeMillis();
        }
        if (shouldStallEachPacket) {
            Thread.sleep(1);
        }

        socket.send(packet);
        bytesSentThisPeriod += packet.getLength();

        if (bytesSentThisPeriod >= BYTES_BETWEEN_DELAY_CHECKS) {
            long end = currentTimeMillis();
            long thisPeriodMs = end - start;
            if (thisPeriodMs == 0) {
                thisPeriodMs++;
            }


            long sleepTime = (bytesSentThisPeriod / TARGET_BYTES_PER_MILLISECOND) - thisPerio

            shouldStallEachPacket = (sleepTime > thisPeriodMs);

            if (sleepTime > 0) {
                Thread.sleep(sleepTime);
            }

            long periodEnd = currentTimeMillis();
            long periodBandwidth = ((bytesSentThisPeriod * 8) / (periodEnd - start)); // kilo
            start = periodEnd;
            bytesSentThisPeriod = 0;
        }

    }


}
```

实验思考

参考资料

- http://good.ncu.edu.cn/doc/h4.pdf