Министерство образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2  
по курсу «Разработка клиент-серверных приложений»

Выполнили студенты группы 19ВВ3.1:

Дорофеев А.А.,

Павлов А.А.,

Петров О.М.

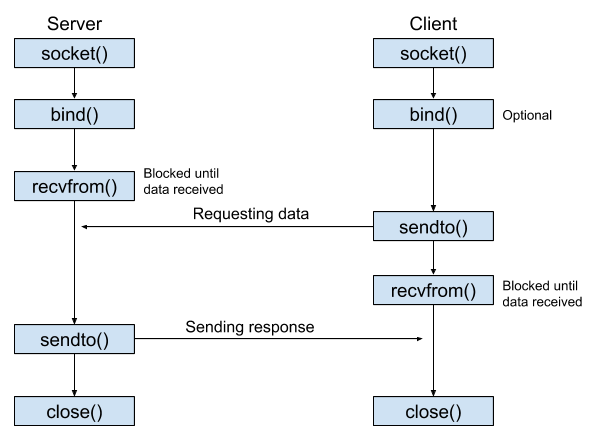
Принял:

Попов К.В.

Пенза, 2022

**Цель** **работы:**

Разработка программы-сервера и программы-клиента в архитектуре взаимодействия “клиент-сервер” с использованием протокола UDP. Обеспечить закрытие данных (шифрование), при передаче их по сети между программами клиента и сервера.



**Алгоритм работы программы:**

Программа делится на два потока: основной, в котором выполняется ожидание ввода сообщения с клавиатуры, последующего его шифрования методом гаммирования и отправки получателю при помощи DatagramPacket и метода <socketName>.send(packet) и потока приема сообщения, в котором производится прием сообщения (создается DatagramPacket и применяется метод .receive, аналогично .send), дешифрация сообщения и вывод в консоль получателю.

Работа будет выполняться до тех пор, пока сервер или клиент не введут ключевое слово «exit», которое завершит работу программы.

По окончании работы процесс-клиент и процесс-сервер закрывают свои копии сокетов (системный вызов closesocket) и уничтожаются.

**Результат работы программы:**

После запуска программы, пользователю предлагается ввести свой IP-адрес и адрес собеседника, которому необходимо отсылать сообщения. Далее можно обмениваться сообщениями, просто написав их в консоль и нажав Enter, либо наблюдать как поступают сообщения от собеседника (или собеседников) [рисунок 1, 2]. Чтобы завершить работу, необходимо ввести ключевое слово exit (рисунок 3). Если возникает такая ситуация, что пользователь пытается создать клиент по IP-адресу, который уже участвует в работе, то программа выдаст соответствующую ошибку (рисунок 4).

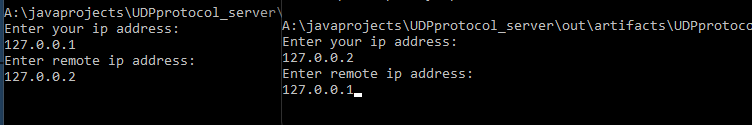


Рисунок 1 – Ввод IP-адресов

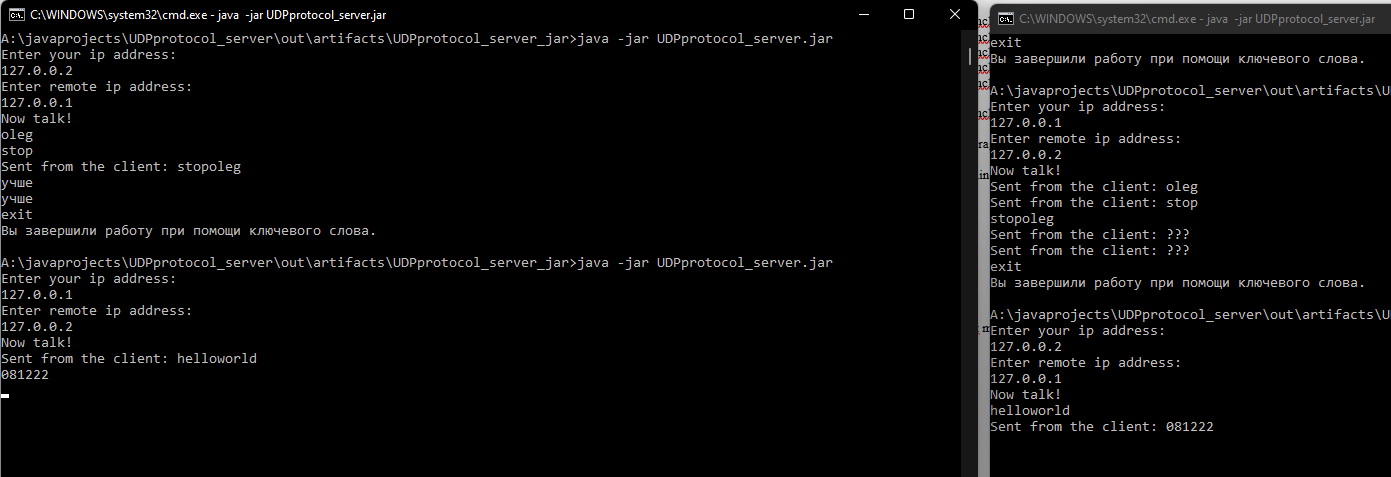


Рисунок 2 – Обмен сообщениями

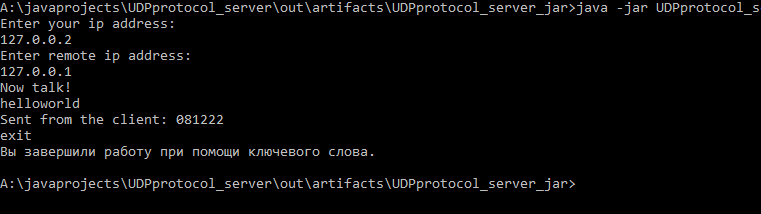


Рисунок 3 – Завершение работы программы

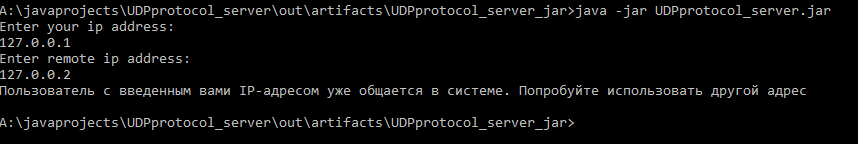


Рисунок 4 – Сообщение об ошибке

**Вывод:**

При выполнении данной лабораторной работы мы научились разрабатывать программы-сервера и программы-клиента в архитектуре взаимодействия “клиент-сервер” с использованием протокола UDP, а также обеспечивать закрытие данных (шифрование), при передаче их по сети между программами клиента и сервера.

**Приложение А** *–* **Файл Main.java**

package org.example;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Scanner;

public class Main {

/\* Порт сервера, к которому собирается

подключиться клиентский сокет \*/

public final static int SERVICE\_PORT = 50001;

public final static String exitWord = "exit";

public static void main(String[] args) throws IOException {

try {

System.out.println("Enter your ip address: ");

InetAddress IPAddressSender;

Scanner scanner = new Scanner(System.in);

IPAddressSender = InetAddress.getByName(scanner.nextLine());

System.out.println("Enter remote ip address: ");

InetAddress IPAddressReciever;

IPAddressReciever = InetAddress.getByName(scanner.nextLine());

DatagramSocket serverSocket = new DatagramSocket(SERVICE\_PORT, IPAddressSender);

byte[] keyArray = new byte[1024];

Thread receiverThread = new Thread(new ReceiverUDP(serverSocket));

receiverThread.start();

System.out.println("Now talk!");

while (true)

{

byte[] sendingDataBuffer = new byte[1024];

for (short i = 0; i < keyArray.length; i++)

keyArray[i] = (byte) (1 + (byte) (Math.random() \* 9)); //генерация ключа

String message = scanner.nextLine();

sendingDataBuffer = message.getBytes();

if (message.equals(exitWord)) {

System.out.println("Вы завершили работу при помощи ключевого слова. ");

break;

}

for (short i = 0; i < sendingDataBuffer.length; i++) //тип шифруем

sendingDataBuffer[i] += keyArray[i];

DatagramPacket senderPacket = new DatagramPacket(sendingDataBuffer, sendingDataBuffer.length, IPAddressReciever, SERVICE\_PORT);

serverSocket.send(senderPacket);

DatagramPacket keyPacket = new DatagramPacket(keyArray, keyArray.length, IPAddressReciever, SERVICE\_PORT);

serverSocket.send(keyPacket);

}

System.exit(0);

} catch (SocketException e) {

System.out.println("Пользователь с введенным вами IP-адресом уже общается в системе. Попробуйте использовать другой адрес");

}

}

public static class ReceiverUDP implements Runnable{

private DatagramSocket serverSocket;

public ReceiverUDP(DatagramSocket serverSocket){

this.serverSocket = serverSocket;

}

public void run(){

try {

while (true) {

byte[] receivingDataBuffer = new byte[1024];

byte[] safePlace = new byte[1024];

byte[] keyArray = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receivingDataBuffer, receivingDataBuffer.length);

DatagramPacket keyPacket = new DatagramPacket(keyArray, keyArray.length);

serverSocket.receive(receivePacket);

serverSocket.receive(keyPacket);

for (int i = 0; i < receivingDataBuffer.length; i++) {

safePlace[i] = receivingDataBuffer[i];

if (receivingDataBuffer[i] == 0) {

receivingDataBuffer = new byte[i];

break;

}

}

for (int i = 0; i < receivingDataBuffer.length; i++){

receivingDataBuffer[i] = safePlace[i];

}

for (short i = 0; i < receivingDataBuffer.length; i++) //тип дешифруем

receivingDataBuffer[i] -= keyArray[i];//KEY;

String receivedData = new String(receivingDataBuffer, StandardCharsets.UTF\_8);

System.out.println("Sent from the client: " + receivedData);

}

} catch (SocketException e) {

e.printStackTrace();

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

}