Пермский филиал федерального государственного автономного образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

**СОКЕТЫ**

*отчет по практическому заданию №3*

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

**Руководитель:**

доцент кафедры

информационных

технологий в бизнесе

Е. Б. Замятина

Пермь, 2021 год

**Содержание**

[Задание 1 3](#_Toc94786411)

[Задание 2 4](#_Toc94786412)

[Задание 3 5](#_Toc94786413)

[Задание 4 7](#_Toc94786414)

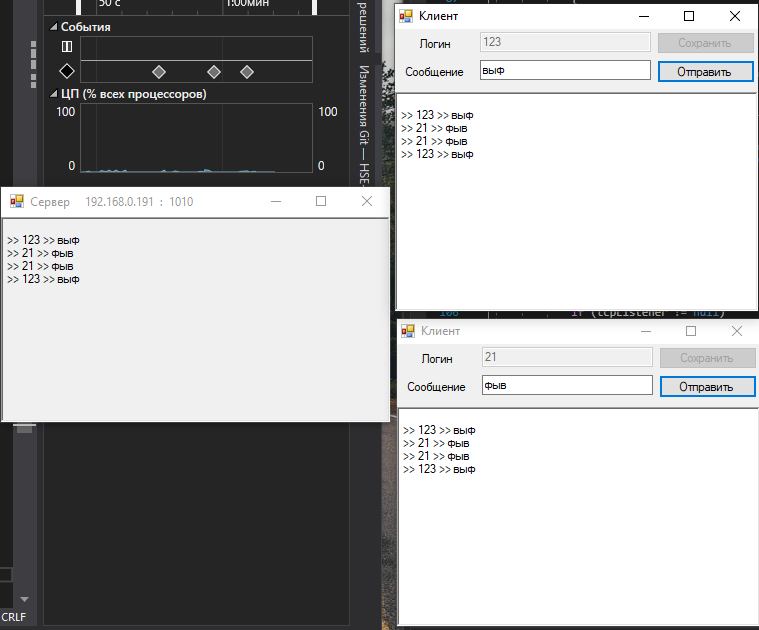
[Задание 5 11](#_Toc94786415)

# Задание 1

Изучите разработанное приложение. Запустите исполняемый файл сервера и клиента на одной машине, затем перенесите один из компонентов распределенного приложения на другой вычислительный узел (лучше на ноутбук, т.к. администратор ЛВС вуза может запретить обмен сообщения между различными узлами ЛВС) и попробуйте его запустить. Запустите несколько клиентов и попробуйте отправить сообщения одному серверу.

**Решение**

После запуска одного сервера и двух клиентов, ошибок при отправке сообщений не возникло. Ошибок не возникло как при передаче сообщений с помощью одного узла, так и с помощью нескольких (рис. 1.1).



**Рисунок 1.1 Передача сообщений с двух клиентов одному серверу**

# Задание 2

Запустите несколько серверов на одной машине. Отправьте им сообщения от нескольких клиентов. Объясните, почему приложение перестало работать.

**Решение**

Запуск нескольких серверов невозможен. Так как разрешается только одно использование адреса сокета (протокол/сетевой адрес/порт).

# Задание 3

Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по имени вычислительного узла, а по нику/логину пользователя.

**Решение**

Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по IP-адресу вычислительного узла, а по нику/логину пользователя.

private void btnSave\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(tbLogin.Text))

{

login = tbLogin.Text;

tbLogin.Enabled = false;

btnSave.Enabled = false;

SendMessage("AddClient:" + clientPort.ToString());

btnSend.Enabled = true;

}

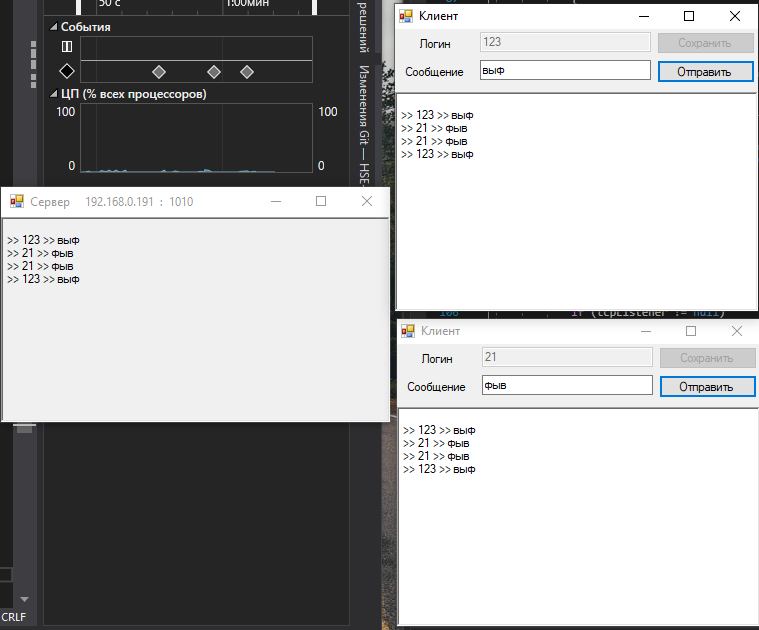
}

// отправка сообщения

private void btnSend\_Click(object sender, EventArgs e)

{

SendMessage(login + " >> " + tbMessage.Text);

 }

***Рисунок 3.3 Работа приложения с никнеймами***

# Задание 4

Модифицируйте приложение так, чтобы получился полноценный чат. Клиент может отправлять сообщения всем клиентам, участвующим в беседе. Для этого каждый клиент должен иметь возможность просмотра всех сообщений от всех клиентов, а сервер должен содержать список клиентов, которые хотят участвовать в беседе, чтобы каждый раз выполнять им рассылку сообщений.

**Решение**

Необходимо добавить возможность клиенту получать сообщения.

Листинг сервера:

// работа с клиентскими сокетами

private void ReceiveMessage()

{

IPEndPoint RemoteIpEndPoint = null;

bool write = true;

// входим в бесконечный цикл для работы с клиентскими сокетом

while (\_continue)

{

byte[] receiveBytes = receivingUdpClient.Receive(

ref RemoteIpEndPoint);

string msg = Encoding.Unicode.GetString(receiveBytes);

//Добавление клиента

write = true;

if (msg.Contains("AddClient:"))

{

//Получаем порт

int portNumber = int.Parse(msg.Split(':')[1]);

if (portsAndTCPClients.ContainsKey(portNumber))

continue;

TcpClient tcpClient = new TcpClient();//Создаем tcp лиснер и конектимся к отправителю по Ip и порту

tcpClient.Connect(ip, portNumber);

portsAndTCPClients.Add(portNumber,tcpClient);//Сохраняю, чтобы потом отключить его

write = false;

}

//Удаление клиента из рассылки

if (msg.Contains("DeleteClient:"))

{

int portNumber = int.Parse(msg.Split(':')[1]);

portsAndTCPClients.Remove(portNumber);

write = false;

}

//Отправка сообщения по клиентам

if (write)

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg.Replace("\0", "") != "")

rtbMessages.Text += "\n >> " + msg;

byte[] newbuff = Encoding.Unicode.GetBytes(msg);

//Отправление по клиентам, тоесть клиент, который отправил сообщение увидит его только, если сервер отправит ему

//(Клиент не записывает сообщение сам себе)

foreach (TcpClient tcpClient in portsAndTCPClients.Values)

{

Stream stm = tcpClient.GetStream();// получаем файловый поток сокета клиента

stm.Write(newbuff, 0, receiveBytes.Length);

}

});

}

}

Листинг клиента:

private void ReadMessages()

{

string msg = "";

clientSocket = tcpListener.AcceptSocket();

try

{

while (\_continue)

{

byte[] buff = new byte[1024];

clientSocket.Receive(buff);

msg = Encoding.Unicode.GetString(buff);

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg.Replace("\0", "") != "")

rtbMessages.Text += "\n >> " + msg;

// выводим полученное сообщение на форму

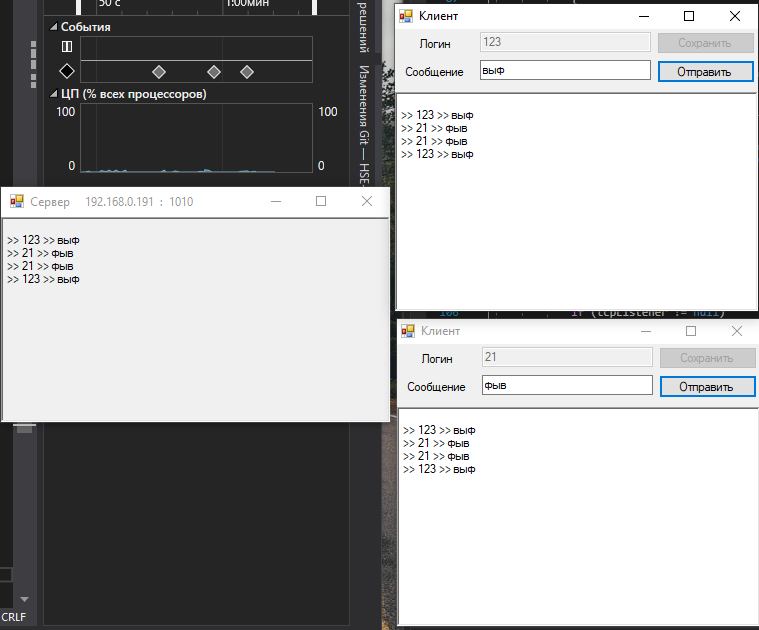
});

Thread.Sleep(500);

}

}

catch (Exception ex) { }

 }

**Рисунок 4.1 Пример работы программы**

# Задание 5

Изучите самостоятельно, как работать с UDP-сокетами. Модифицируйте приложение так, чтобы клиенты, желающие принять участие в беседе, находили сервер в сети с помощью широковещательного запроса (UDP-пакета), а не через ввод IP-адреса машины, на которой расположен сервер, в форме клиента.

Изменение кода в клиенте:

private void SendMessage(string text)

{

UdpClient udpClient = new UdpClient();

IPEndPoint endPoint = new IPEndPoint(ip, PORT);

byte[] buff = Encoding.Unicode.GetBytes(text);

try

{

udpClient.Send(buff, buff.Length, endPoint);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

udpClient.Close();

}

}

Изменение кода в сервере:

UdpClient receivingUdpClient = new UdpClient(PORT);

byte[] receiveBytes = receivingUdpClient.Receive(

ref RemoteIpEndPoint);

receivingUdpClient.Close();