Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК5 «Системы обработки информации»***

ДОМАШНЯЯ РАБОТА

«Разработка сетевых приложений»

ДИСЦИПЛИНА: «Сети и телекоммуникации»

Выполнил: студент гр. ИУК4-52Б _____ (____Губин Е.В.____)
(Подпись) (Ф.И.О.)

Проверил: _____ (___Смирнов М.Е.___)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга , 2024

Цель: получение практических навыков по разработке клиент-серверных приложений, использующих для связи механизм сокетов.

Задачи:

1. Создать приложение-сервер, предназначенное для параллельной обработки запросов и работающее в ОС Windows или *NIX.
2. Создать приложение-клиент, которое будет подключаться к серверу с удаленных компьютеров. Рекомендуется использовать интерфейс сокетов и протокол TCP.
3. Разработать и реализовать протокол прикладного уровня для взаимодействия приложений.
4. Обеспечить обмен произвольными данными между клиентом и сервером.

Схема функционирования сервера:

Схема 1: Функционирование сервера

Схема функционирования клиента:

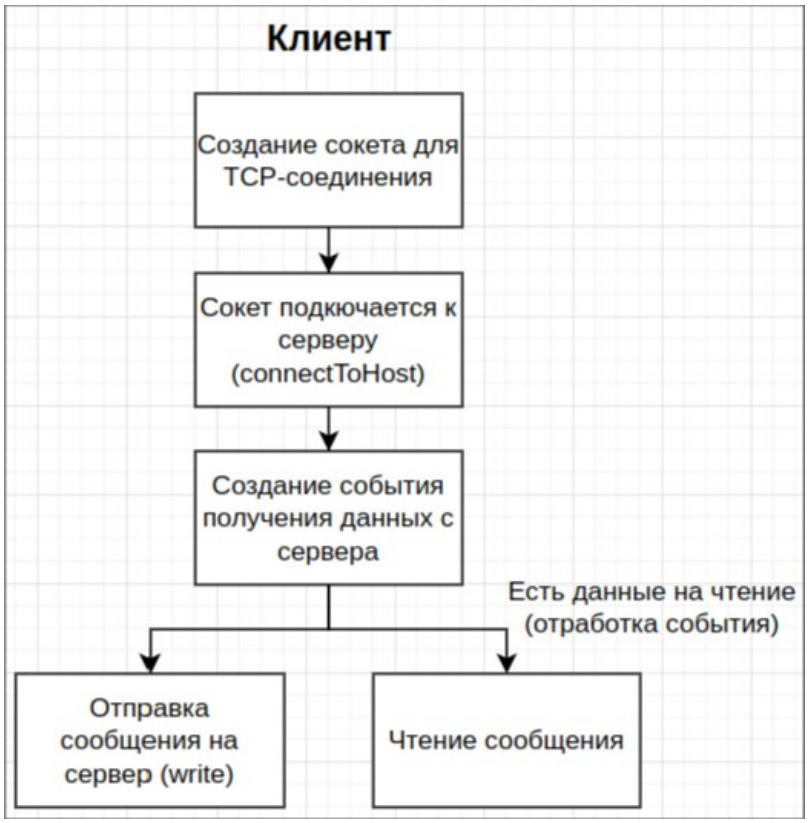

Схема 2: Функционирование клиента

Схема взаимодействия клиента и сервера:

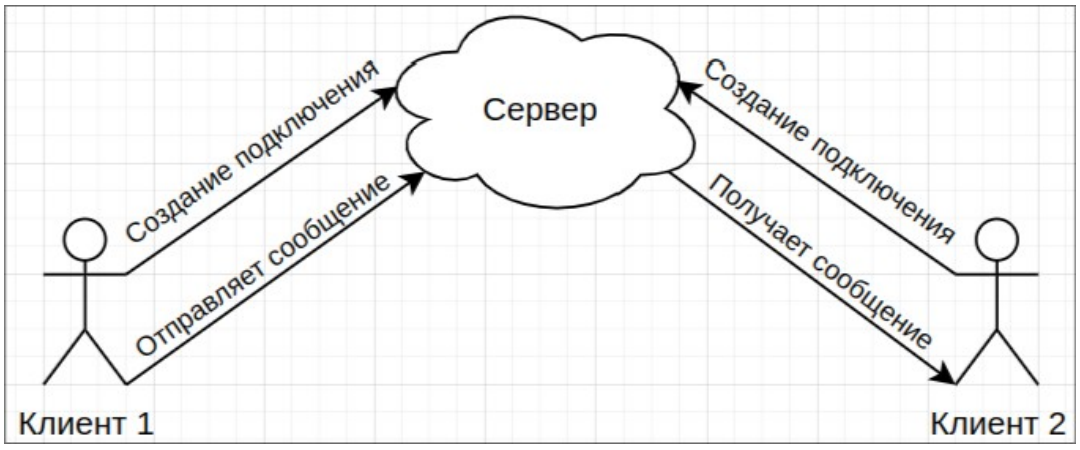

Схема 3: Взаимодействие клиентов и сервера

Листинг программы сервера:

```
#include <iostream>
#include <thread>
#include <vector>
#include <string>
#include <cstring>
#include <netinet/in.h>
#include <unistd.h>
#include <mutex>
#include <algorithm>

constexpr int PORT = 5000;
constexpr int BUFFER_SIZE = 1024;

std::vector<int> clients;
std::mutex clients_mutex;

void broadcast_message(const std::string &, int);
void handle_client(int);

int main()
{
    int server = socket(AF_INET, SOCK_STREAM, 0);

    if (server == -1)
    {
        perror("Socket creation failed");
        return 1;
    }

    sockaddr_in socket_settings{};
    socket_settings.sin_family = AF_INET;
    socket_settings.sin_addr.s_addr = INADDR_ANY;
    socket_settings.sin_port = htons(PORT);

    if (bind(server, reinterpret_cast<sockaddr *>(&socket_settings),
sizeof(socket_settings)) < 0)
    {
        perror("Bind failed");
        return 1;
    }

    if (listen(server, 3) == -1)
    {
        perror("Listen failed");
        return 1;
    }

    std::cout << "Server listening on port " << PORT << "..." << std::endl;

    while (true)
    {
        sockaddr_in client_addr{};
        socklen_t client_len = sizeof(client_addr);
        int client = accept(server, reinterpret_cast<sockaddr *>(&client_addr),
&client_len);
        if (client == -1)
        {
            perror("Client accept failed");
```

```
            continue;
        }

        std::lock_guard<std::mutex> lock(clients_mutex);
        clients.push_back(client);

        std::thread(handle_client, client).detach();
    }

    close(server);
    return 0;
}

void broadcast_message(const std::string &message, int sender)
{
    std::lock_guard<std::mutex> lock(clients_mutex);
    std::string message_with_newline = message + '\n';
    for (int client : clients)
    {
        if (client != sender)
        {
            send(client, message_with_newline.c_str(), message_with_newline.size(),
0);
        }
    }
}

void handle_client(int client)
{
    char buffer[BUFFER_SIZE];
    while (true)
    {
        std::memset(buffer, 0, BUFFER_SIZE);
        ssize_t bytes_received = recv(client, buffer, BUFFER_SIZE, 0);

        if (bytes_received <= 0)
        {
            break;
        }

        std::string message(buffer, bytes_received);
        std::cout << "Received: " << message << std::endl;
        broadcast_message(message, client);
    }

    close(client);

    std::lock_guard<std::mutex> lock(clients_mutex);
    clients.erase(std::remove(clients.begin(), clients.end(), client), clients.end());
}
```

Листинг программы клиента:

main.cpp:

```
#include "Chat.h"

#include <QApplication>
#include <QScreen>
```

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QScreen *screen = QApplication::primaryScreen();
    QRect screenRect = screen→availableGeometry();

    Chat client1;
    client1.setWindowTitle("Chat Client");
    client1.resize(screenRect.width(), screenRect.height());
    client1.show();
    qDebug() << screenRect.height();
    qDebug() << screenRect.height();

    Chat client2;
    client2.setWindowTitle("Chat Client");
    client2.resize(screenRect.width(), screenRect.height());
    client2.show();

    return a.exec();
}
```

Chat.h:

```
#ifndef CHAT_H
#define CHAT_H

#include <QMainWindow>
#include <QTcpSocket>
#include <QKeyEvent>

QT_BEGIN_NAMESPACE
namespace Ui {
class Chat;
}
QT_END_NAMESPACE

class Chat : public QMainWindow
{
    Q_OBJECT

public:
    Chat(QWidget *parent = nullptr);
    ~Chat();

private slots:
    void sendMessage();
    void receiveMessage();

private:
    Ui::Chat *ui;
    QTcpSocket *socket;

protected:
    bool eventFilter(QObject*, QEvent*) override;
};

#endif // CHAT_H
```

Chat.cpp:

```
#include "Chat.h"
#include "./ui_Chat.h"

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

    socket = new QTcpSocket(this);
    socket->connectToHost("127.0.0.1", 5000);

    if (!socket->waitForConnected(5000))
    {
        qDebug() << "Connection failed!";
        return;
    }

    connect(ui->sendNewMsgBtn, &QPushButton::clicked, this, &Chat::sendMessage);
    connect(socket, &QTcpSocket::readyRead, this, &Chat::receiveMessage);

    ui->newMsgLineEdit->installEventFilter(this);
}

void Chat::sendMessage()
{
    if (!ui->newMsgLineEdit->text().isEmpty())
    {
        QString message = ui->newMsgLineEdit->text();
        socket->write(message.toUtf8());
        ui->newMsgLineEdit->clear();
        ui->msgsBox->append("You: " + message);
    }
}

void Chat::receiveMessage()
{
    while (socket->canReadLine())
    {
        QString message = socket->readLine();
        ui->msgsBox->append("Companion: " + message.trimmed());
    }
}

bool Chat::eventFilter(QObject *watched, QEvent *event)
{
    if (watched->isWidgetType() && event->type() == QEvent::KeyPress)
    {
        QKeyEvent *keyEvent = static_cast<QKeyEvent *>(event);
        if (keyEvent->key() == Qt::Key_Return || keyEvent->key() == Qt::Key_Enter)
        {
            sendMessage();
            return true;
        }
    }
    return QWidget::eventFilter(watched, event);
}

Chat::~Chat()
{
```

```
    delete ui;
}
```

Chat.ui:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Chat</class>
 <widget class="QMainWindow" name="Chat">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>800</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Chat</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="sendNewMsgBtn">
    <property name="geometry">
     <rect>
      <x>625</x>
      <y>740</y>
      <width>110</width>
      <height>30</height>
     </rect>
    </property>
    <property name="text">
     <string>Send</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="newMsgLineEdit">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>730</y>
      <width>560</width>
      <height>50</height>
     </rect>
    </property>
    <property name="placeholderText">
     <string>Type your message here...</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="msgsBox">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>20</y>
      <width>720</width>
      <height>700</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <pointsize>14</pointsize>
     </font>
```

```
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>800</width>
     <height>20</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Результаты тестирования:

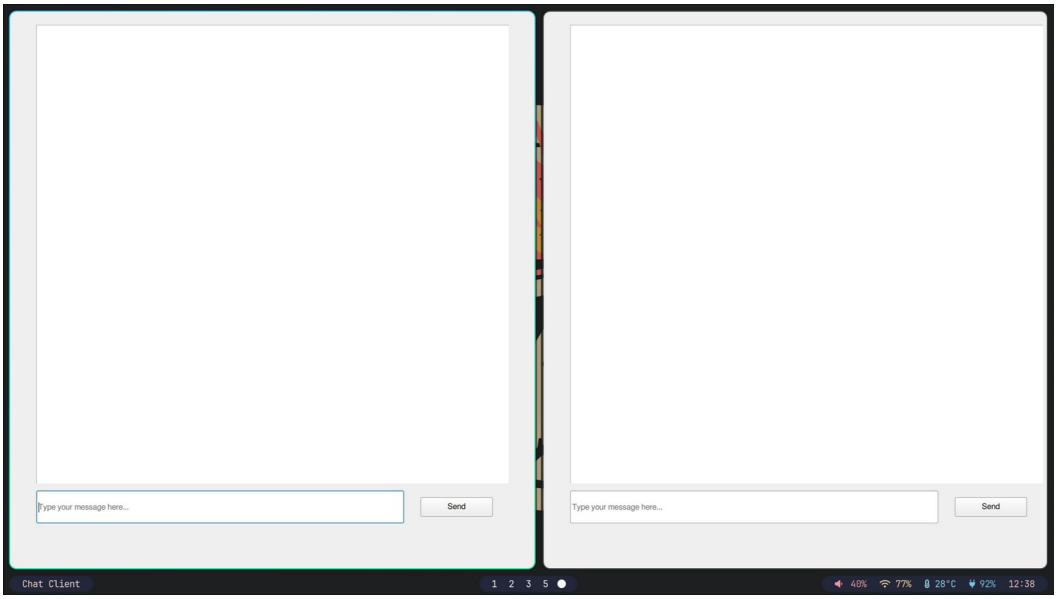

Рис. 1: Окна чата с двух клиентов

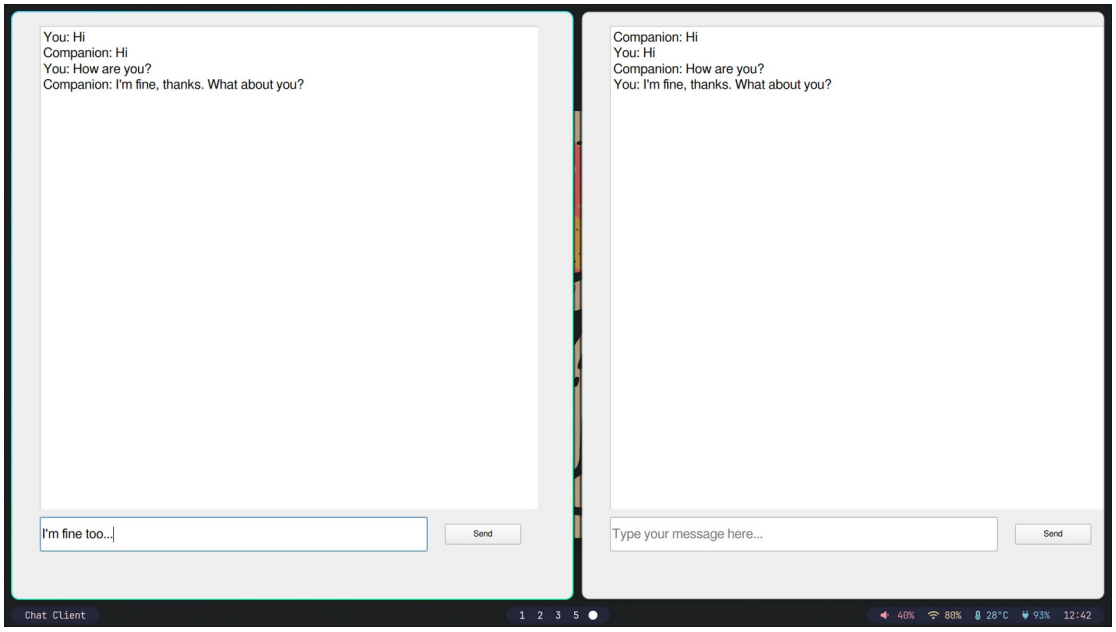

Рис. 2: Тестировани сокетов на двух клиентах

Вывод: в ходе лабораторной работы был изучен и реализован на примере чата механизм сокетов.