Министерство связи и цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра вычислительных систем

**Курсовая работа  
 по дисциплине  
 «Сетевое программирование»**

Разработка сетевого приложения «Чат».

Сервер без установления соединения (протокол UDP).

Выполнил:

студент гр.ИП-213

Терновский Д. Р.

Проверил:

Профессор кафедры ВС

Павский К. В.

Новосибирск 2025 г.

**Содержание**

[Постановка задачи 3](#_Toc195813153)

[Описание протокола UDP 4](#_Toc195813154)

[Описание реализации приложения 5](#_Toc195813155)

[Реализация серверной части 5](#_Toc195813156)

[Реализация клиентской части 6](#_Toc195813157)

[Скан экрана работы программы 7](#_Toc195813158)

[Текст программы 8](#_Toc195813159)

[Структура проекта 8](#_Toc195813160)

[Текст программы (серверная часть) 9](#_Toc195813161)

[Текст программы (клиентская часть) 14](#_Toc195813162)

[Текст программы (файл сборки проекта) 19](#_Toc195813163)

[Список источников 20](#_Toc195813164)

# Постановка задачи

Целью данной курсовой работы является разработка сетевого приложения «Чат» без установления соединения (протокол UDP). Приложение должно обеспечивать возможность обмена текстовыми сообщениями между пользователями в режиме реального времени посредством протокола UDP.

В рамках работы требуется реализовать два компонента:

* Сервер, отвечающий за приём сообщений от клиентов и их последующую рассылку другим участникам.
* Клиентское приложение, позволяющее отправлять сообщения на сервер и получать сообщения от других пользователей чат

Требования к серверу:

* Использование **UDP-сокетов** для обмена данными.
* Поддержка работы с несколькими клиентами без создания отдельного потока или соединения для каждого клиента (что возможно благодаря особенностям протокола UDP).
* Приём сообщений от клиентов и их **рассылка всем подключённым участникам**, кроме отправителя.
* Обеспечение **обработки входящих датаграмм в реальном времени**, без задержек и очередей.

Требования к клиенту:

* Возможность **подключения к серверу по заданным IP-адресу и порту**.
* Отправка текстовых сообщений на сервер.
* Приём сообщений, рассылаемых сервером от других клиентов.

# Описание протокола UDP

UDP (User Datagram Protocol) — это транспортный протокол, работающий поверх IP и обеспечивающий максимально быструю передачу данных за счёт минимальных накладных расходов. В отличие от TCP, протокол UDP не устанавливает соединение между клиентом и сервером перед началом обмена информацией. Каждый пакет, называемый датаграммой, передаётся независимо от других, без предварительного «рукопожатия» и без сохранения состояния сессии.

Главной особенностью UDP является отсутствие гарантии доставки. Пакеты могут теряться в процессе передачи, приходить в неверном порядке или дублироваться. При этом протокол не обеспечивает контроль перегрузки и не управляет потоком данных — отправка происходит без проверки готовности получателя, что позволяет минимизировать задержки и увеличить производительность в ситуациях, где скорость важнее надёжности.

Для обмена сообщениями сервер и клиент используют функции sendto() для отправки и recvfrom() для приёма данных. При этом в каждом вызове указывается адрес получателя или отправителя, поскольку постоянное подключение между сторонами не требуется. UDP особенно эффективен для широковещательных и multicast-рассылок, когда одни и те же данные необходимо быстро отправить нескольким адресатам.

На сетевом уровне UDP использует протокол IP, применяя для адресации как IPv4 (AF\_INET), так и IPv6 (AF\_INET6). Пакеты маршрутизируются на основе IP-адресов и номеров портов, что позволяет доставлять их конечным получателям без установления соединения и дополнительных подтверждений.

# ****Описание реализации приложения****

## Реализация серверной части

Серверная часть реализована с использованием UDP сокетов на языке C. Основная структура udp\_server\_t хранит необходимые данные для работы сервера: дескриптор сокета, структуру адреса сервера, массив адресов клиентов и их количество.

Основные функции:

* udp\_server\_init — инициализирует серверный сокет, привязывает его к заданному IP-адресу и произвольному порту (порт выбирается автоматически системой). Выполняется проверка успешности создания и привязки сокета.
* udp\_server\_run — основной цикл работы сервера. Ожидает сообщения от клиентов, принимает их с помощью recvfrom, добавляет новые адреса клиентов в список и пересылает сообщение всем подключённым клиентам, кроме отправителя.
* udp\_server\_sendto — отправляет сообщение конкретному клиенту.
* udp\_server\_has\_client — проверяет наличие клиента в списке зарегистрированных.
* udp\_server\_rm\_client — удаляет клиента из списка по его адресу.
* udp\_server\_destroy — закрывает серверный сокет по завершении работы.

Работа сервера осуществляется бесконечно в цикле до принудительного завершения. Клиент может отправить команду /quit, после чего его адрес удаляется из списка активных клиентов.

## Реализация клиентской части

Клиентская часть также реализована с использованием UDP сокетов и потоков (POSIX Threads) для обеспечения параллельной отправки и получения сообщений. Основная структура udp\_client\_t содержит дескриптор сокета, структуру адреса сервера и семафор для синхронизации вывода сообщений.

Основные функции:

* udp\_client\_init — создаёт сокет клиента, задаёт параметры подключения (IP-адрес и порт сервера) и инициализирует семафор.
* udp\_client\_run — запускает два параллельных потока:
  + поток для получения сообщений от сервера;
  + поток для отправки сообщений на сервер. Основной поток ожидает завершения потока отправки, после чего завершает поток получения.
* udp\_client\_recv\_thread — в бесконечном цикле ожидает сообщения от сервера, получая их с помощью recvfrom, и выводит их на экран. Вывод синхронизируется с потоком отправки сообщений с помощью семафора.
* udp\_client\_send\_thread — в цикле запрашивает ввод сообщения пользователя, отправляет его серверу с помощью sendto. Если введена команда /quit, завершает выполнение потока.
* udp\_client\_recv\_msg и udp\_client\_send\_msg — выполняют приём и отправку сообщений соответственно.
* udp\_client\_destroy — закрывает сокет и уничтожает семафор.

Таким образом, каждый клиент может одновременно получать и отправлять сообщения, а сервер осуществляет их пересылку всем другим подключённым клиентам.

## Скан экрана работы программы

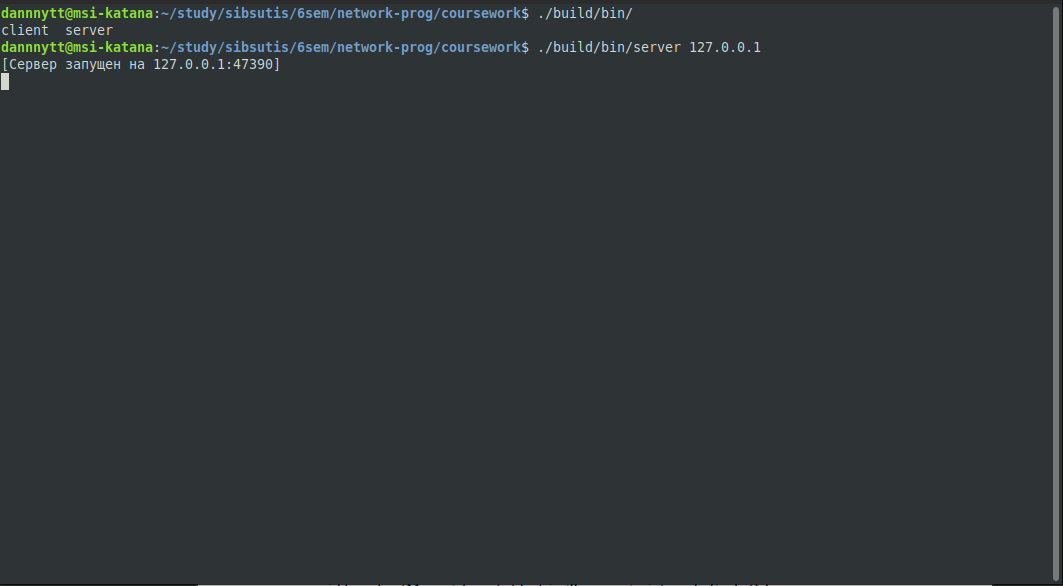


Рис. 1 – Запуск сервера.

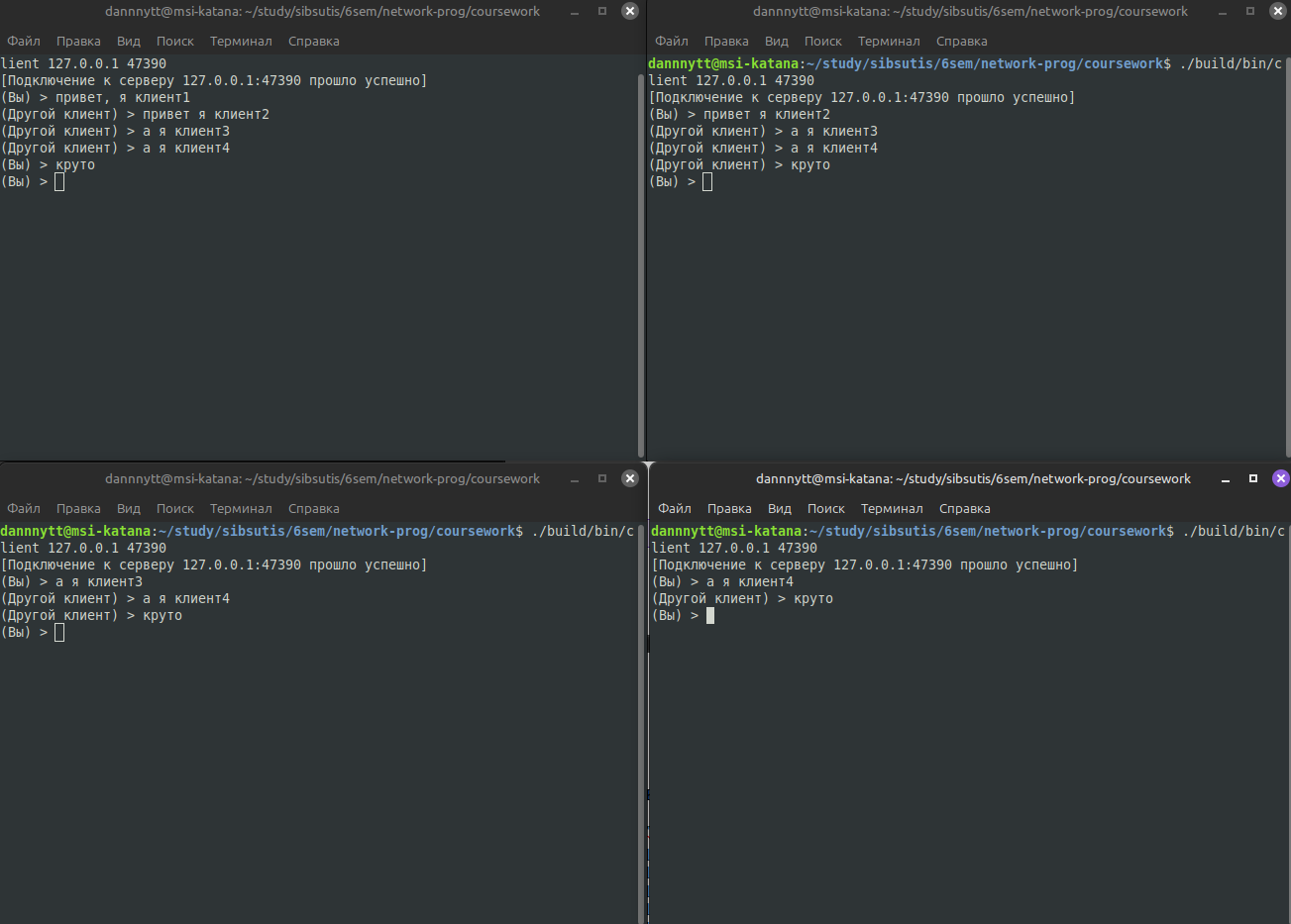
****

Рис. 2 – Запуск клиентов и демонстрация общения между клиентами.

# Текст программы

## Структура проекта



Рис. 3 – Структура курсовой работы

**Текст программы (серверная часть)**

**Файл server/include/udp\_server.h :**

#ifndef UDPSERVER\_H

#define UDPSERVER\_H

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define MAX\_CLIENTS 10

typedef struct udp\_server {

int sockfd;

struct sockaddr\_in server\_addr;

struct sockaddr\_in client\_addr;

struct sockaddr\_in clients[MAX\_CLIENTS];

int client\_count;

} udp\_server\_t;

void udp\_server\_init(udp\_server\_t \*server, const char\* ip\_addr);

void udp\_server\_destroy(udp\_server\_t \*server);

void udp\_server\_run(udp\_server\_t \*server);

void udp\_server\_sendto(

udp\_server\_t \*server,

const struct sockaddr\_in \*client\_addr,

const struct sockaddr\_in \*sender\_addr,

const char \*msg

);

int udp\_server\_has\_client(udp\_server\_t \*server, struct sockaddr\_in \*addr);

void udp\_server\_rm\_client(udp\_server\_t \*server, struct sockaddr\_in \*addr);

#endif // UDPSERVER\_H

**Файл server/src/udp\_server\_init.c :**

#include "udp\_server.h"

void udp\_server\_init(udp\_server\_t \*server, const char \*ip\_addr) {

server->sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if (server->sockfd < 0) {

perror("Ошибка: сокет сервера не создан.\n");

exit(EXIT\_FAILURE);

}

memset(&server->server\_addr, 0, sizeof(server->server\_addr));

server->server\_addr.sin\_family = AF\_INET;

server->server\_addr.sin\_port = 0;

if (inet\_pton(AF\_INET, ip\_addr, &server->server\_addr.sin\_addr) <= 0) {

perror("Ошибка: IP-адрес сервера не преобразован.\n");

close(server->sockfd);

exit(EXIT\_FAILURE);

}

socklen\_t server\_addr\_len = sizeof(server->server\_addr);

if (bind(server->sockfd, (struct sockaddr\*) &server->server\_addr, server\_addr\_len) < 0) {

perror("Ошибка: привязка сокета прошла неудачно.\n");

close(server->sockfd);

exit(EXIT\_FAILURE);

}

if (getsockname(server->sockfd, (struct sockaddr\*) &server->server\_addr, &server\_addr\_len) == -1) {

perror("Ошибка: порт не получен.\n");

close(server->sockfd);

exit(EXIT\_FAILURE);

}

server->client\_count = 0;

printf("[Сервер запущен на %s:%d]\n", ip\_addr, ntohs(server->server\_addr.sin\_port));

}

**Файл server/src/udp\_server\_run.c :**

#include "udp\_server.h"

void udp\_server\_run(udp\_server\_t \*server) {

socklen\_t client\_addr\_len = sizeof(server->client\_addr);

char buffer[1024];

while (true) {

ssize\_t recvd\_bytes = recvfrom(

server->sockfd,

buffer,

sizeof(buffer) - 1,

0,

(struct sockaddr\*)&server->client\_addr,

&client\_addr\_len

);

buffer[recvd\_bytes] = '\0';

if (strcmp(buffer, "/quit") == 0) {

udp\_server\_rm\_client(server, &server->client\_addr);

continue;

}

if (!udp\_server\_has\_client(server, &server->client\_addr))

server->clients[server->client\_count++] = server->client\_addr;

for (int i = 0; i < server->client\_count; i++) {

udp\_server\_sendto(

server,

&server->clients[i],

&server->client\_addr,

buffer

);

}

}

}

**Файл server/src/udp\_server\_rm\_client.c :**

#include "udp\_server.h"

void udp\_server\_rm\_client(udp\_server\_t \*server, struct sockaddr\_in \*addr) {

for (int i = 0; i < server->client\_count; i++) {

if (memcmp(&server->clients[i], addr, sizeof(struct sockaddr\_in)) == 0) {

for (int j = i; j < server->client\_count - 1; j++) {

server->clients[j] = server->clients[j + 1];

}

server->client\_count--;

break;

}

}

}

**Файл server/src/udp\_server\_has\_client.c :**

#include <udp\_server.h>

int udp\_server\_has\_client(udp\_server\_t \*server, struct sockaddr\_in \*addr) {

for (int i = 0; i < server->client\_count; i++) {

if (memcmp(&server->clients[i], addr, sizeof(struct sockaddr\_in)) == 0) {

return true;

}

}

return false;

}

**Файл server/src/udp\_server\_sendto.c :**

#include <udp\_server.h>

void udp\_server\_sendto(udp\_server\_t \*server,

const struct sockaddr\_in \*client\_addr,

const struct sockaddr\_in \*sender\_addr,

const char \*msg)

{

if (memcmp(client\_addr, sender\_addr, sizeof(struct sockaddr\_in)) != 0) {

sendto(

server->sockfd,

msg,

strlen(msg),

0,

(struct sockaddr\*)client\_addr,

sizeof(\*client\_addr)

);

}

}

**Файл server/udp\_server\_destroy.c :**

#include "udp\_server.h"

void udp\_server\_destroy(udp\_server\_t \*server) {

close(server->sockfd);

}

**Файл server/main.c :**

#include "udp\_server.h"

int main(int argc, char\* argv[]) {

udp\_server\_t server;

udp\_server\_init(&server, argv[1]);

udp\_server\_run(&server);

udp\_server\_destroy(&server);

return 0;

}

## Текст программы (клиентская часть)

**Файл client/include/udp\_client.h :**

#ifndef UDPCLIENT\_H

#define UDPCLIENT\_H

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

#include <arpa/inet.h>

#include <netinet/in.h>

typedef struct udp\_client {

int sockfd;

struct sockaddr\_in server\_addr;

sem\_t print\_semaphore;

} udp\_client\_t;

void udp\_client\_init(udp\_client\_t \*client, const char \*server\_addr, uint16\_t server\_port);

void udp\_client\_destroy(udp\_client\_t \*client);

void udp\_client\_run(udp\_client\_t \*client);

void udp\_client\_send\_msg( udp\_client\_t \*client, const char \*message);

void udp\_client\_recv\_msg(udp\_client\_t \*client, char \*buffer, size\_t buffer\_size);

void\* udp\_client\_recv\_thread(void\* arg);

void\* udp\_client\_send\_thread(void\* arg);

#endif // UDPCLIENT\_H

**Файл client/src/udp\_client\_init.c :**

#include "udp\_client.h"

void udp\_client\_init(udp\_client\_t \*client, const char \*server\_addr, uint16\_t server\_port) {

client->sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

if (client->sockfd < 0) {

perror("Ошибка: сокет клиента не создан.\n");

exit(EXIT\_FAILURE);

}

memset(&client->server\_addr, 0, sizeof(client->server\_addr));

client->server\_addr.sin\_family = AF\_INET;

client->server\_addr.sin\_port = htons(server\_port);

if (inet\_pton(AF\_INET, server\_addr, &client->server\_addr.sin\_addr) != 1) {

perror("Ошибка: ip-адрес сервера не преобразован.\n");

close(client->sockfd);

exit(EXIT\_FAILURE);

}

printf("[Подключение к серверу %s:%d прошло успешно]\n",

server\_addr, server\_port);

sem\_init(&client->print\_semaphore, 0, 1);

}

**Файл client/src/udp\_client\_run.c :**

#include "udp\_client.h"

void udp\_client\_run(udp\_client\_t \*client) {

pthread\_t recv\_thread, send\_thread;

pthread\_create(&recv\_thread, NULL, udp\_client\_recv\_thread, (void\*)client);

pthread\_create(&send\_thread, NULL, udp\_client\_send\_thread, (void\*)client);

pthread\_join(send\_thread, NULL);

pthread\_cancel(recv\_thread);

pthread\_join(recv\_thread, NULL);

}

**Файл client/src/threads/udp\_client\_recv\_thread.c :**

#include "udp\_client.h"

void\* udp\_client\_recv\_thread(void \*arg) {

udp\_client\_t \*client = (udp\_client\_t\*) arg;

char buffer[1024];

while (true) {

udp\_client\_recv\_msg(client, buffer, sizeof(buffer));

sem\_wait(&client->print\_semaphore);

printf("\r(Другой клиент) > %s\n", buffer);

printf("(Вы) > ");

fflush(stdout);

sem\_post(&client->print\_semaphore);

}

return NULL;

}

**Файл client/src/threads/udp\_client\_send\_thread.c :**

#include "udp\_client.h"

void\* udp\_client\_send\_thread(void\* arg) {

udp\_client\_t\* client = (udp\_client\_t\*) arg;

char message[1024];

while (true) {

sem\_wait(&client->print\_semaphore);

printf("(Вы) > ");

fflush(stdout);

sem\_post(&client->print\_semaphore);

fgets(message, sizeof(message), stdin);

message[strcspn(message, "\n")] = '\0';

if (strcmp("/quit", message) == 0) {

udp\_client\_send\_msg(client, message);

break;

}

udp\_client\_send\_msg(client, message);

}

return NULL;

}

**Файл client/src/udp\_client\_recv\_msg.c :**

#include "udp\_client.h"

void udp\_client\_recv\_msg(udp\_client\_t \*client, char \*buffer, size\_t buffer\_size) {

socklen\_t server\_addr\_len = sizeof(client->server\_addr);

ssize\_t recvd\_bytes = recvfrom(

client->sockfd,

buffer,

buffer\_size - 1,

0,

(struct sockaddr\*)&client->server\_addr,

&server\_addr\_len

);

if (recvd\_bytes > 0) {

buffer[recvd\_bytes] = '\0';

}

else perror("Ошибка при вызове 'recvfrom'");

}

**Файл client/src/udp\_client\_send\_msg.c :**

#include "udp\_client.h"

void udp\_client\_send\_msg(udp\_client\_t \*client, const char\* message) {

char buffer[1024];

snprintf(buffer, sizeof(buffer), "%s", message);

socklen\_t server\_addr\_len = sizeof(client->server\_addr);

ssize\_t sent\_bytes = sendto(client->sockfd, buffer, strlen(buffer), 0,

(struct sockaddr\*) &client->server\_addr,

server\_addr\_len);

if (sent\_bytes < 0)

perror("Ошибка: сообщение не отправлено.\n");

}

**Файл client/src/udp\_client\_destroy.c :**

void udp\_client\_destroy(udp\_client\_t \*client) {

close(client->sockfd);

sem\_destroy(&client->print\_semaphore);

}

**Файл client/main.c :**

#include "udp\_client.h"

int main(int argc, char\* argv[]) {

const char\* server\_ip = argv[1];

u\_int16\_t server\_port = atoi(argv[2]);

udp\_client\_t client;

udp\_client\_init(&client, server\_ip, server\_port);

udp\_client\_run(&client);

udp\_client\_destroy(&client);

return 0;

}

## Текст программы (файл сборки проекта)

**Файл CmakeLists.txt :**

cmake\_minimum\_required(VERSION 3.28)

project(udp\_chat C)

set(CMAKE\_C\_STANDARD 17)

set(CMAKE\_C\_STANDARD\_REQUIRED True)

set(CMAKE\_BINARY\_DIR "${CMAKE\_SOURCE\_DIR}/build")

set(CMAKE\_RUNTIME\_OUTPUT\_DIRECTORY "${CMAKE\_BINARY\_DIR}/bin")

set(client\_sources

${CMAKE\_SOURCE\_DIR}/client/src/udp\_client\_init.c

${CMAKE\_SOURCE\_DIR}/client/src/udp\_client\_run.c

${CMAKE\_SOURCE\_DIR}/client/src/udp\_client\_send\_msg.c

${CMAKE\_SOURCE\_DIR}/client/src/udp\_client\_recv\_msg.c

${CMAKE\_SOURCE\_DIR}/client/src/udp\_client\_destroy.c

${CMAKE\_SOURCE\_DIR}/client/src/threads/udp\_client\_send\_thread.c

${CMAKE\_SOURCE\_DIR}/client/src/threads/udp\_client\_recv\_thread.c

${CMAKE\_SOURCE\_DIR}/client/main.c

)

set(server\_sources

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_init.c

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_run.c

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_sendto.c

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_destroy.c

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_has\_client.c

${CMAKE\_SOURCE\_DIR}/server/src/udp\_server\_rm\_client.c

${CMAKE\_SOURCE\_DIR}/server/main.c

)

add\_executable(client ${client\_sources})

add\_executable(server ${server\_sources})

target\_include\_directories(client PRIVATE ${CMAKE\_SOURCE\_DIR}/client/include)

target\_include\_directories(server PRIVATE ${CMAKE\_SOURCE\_DIR}/server/include)

# Список источников

1. Павский К. В Введение в разработку сетевых приложений (протоколы TCP/IP, клиент-сервер, PCAP): Учебное пособие / Сибирский государственный университет телекоммуникаций и информатики. – Новосибирск, 2020. – 91 с.
2. Павский К. В., Ефимов А. В. Разработка сетевых приложений (протоколы TCP/IP, клиент-сервер, PCAP, Boost.ASIO) : Учебное пособие / Сибирский государственный университет телекоммуникаций и информатики. – Новосибирск, 2018. – 80 с.
3. Протоколы TCP/IP и разработка сетевых приложений : учеб. пособие / К.В. Павский ; Сиб. гос. ун-т телекоммуникаций и информатики. - Новосибирск : СибГУТИ, 2013. – 130c.
4. Дубаков, А. А. Сетевое программирование [Электронный ресурс] : учебное пособие / А. А. Дубаков. — Электрон. текстовые данные. — СПб. : Университет ИТМО, 2013. — 249 c. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/68118.html> *Лицензия: до 01.10.2022*
5. Олифер, В. Г. Основы сетей передачи данных [Электронный ресурс] / В. Г. Олифер, Н. А. Олифер. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 219 c. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/73702.html> *Лицензия: до 23.01.2021*
6. Семенов, Ю. А. Алгоритмы телекоммуникационных сетей. Часть 1. Алгоритмы и протоколы каналов и сетей передачи данных [Электронный ресурс] / Ю. А. Семенов. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 757 c. — 978-5-94774-706-5. — Режим доступа: <http://www.iprbookshop.ru/62806.html> *Лицензия: до 31.03.2020*