

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)**

Лабораторная работа №3
дисциплина «Сети ЭВМ и телекоммуникации»
по теме «Программирование протокола IP с использованием библиотеки
Winsock»

Выполнил: студент группы ВТ-31
Проверил:

Макаров Д.С.
Федотов Е.А.

Белгород 2020

Лабораторная работа №3

«Программирование протокола IP с использованием библиотеки Winsock»

Цель работы: Цель работы: изучить принципы и характеристику протокола IP и разработать программу для приема/передачи пакетов с использованием библиотеки Winsock.

Вариант 6

Содержание отчета

1. Краткие теоретические сведения.
2. Основные функции API, использованные в данной работе.
3. Разработка программы. Блок-схемы программы.
4. Анализ функционирования разработанных программ.
5. Выводы.
6. Тексты программ. Скриншоты программ.

Ход работы

1. Краткие теоретические сведения.

IP - маршрутизируемый сетевой протокол сетевого уровня семейства TCP/IP. Протокол IP используется для негарантированной доставки данных, разделяемых на так называемые пакеты от одного узла сети к другому. Это означает, что на уровне этого протокола (*третий уровень сетевой модели OSI*) не даётся гарантий надёжной доставки пакета до адресата. В частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (когда приходят две копии одного пакета - в реальности это бывает крайне редко), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прибыть вовсе. Гарантии безошибочной доставки пакетов дают протоколы более высокого (транспортного) уровня сетевой модели OSI - например, TCP - который использует IP в качестве транспорта. Обычно в сетях используется IP четвёртой версии, также известный как IPv4.

В протоколе IP этой версии каждому узлу сети ставится в соответствие IP-адрес длиной 4 октета (1 октет состоит из 8 бит). При этом компьютеры в подсетях объединяются общими начальными битами адреса. Количество этих бит, общее для данной подсети, называется маской подсети (*ранее использовалось деление пространства адресов по классам — A, B, C; класс сети определяется диапазоном значений старшего октета и определяет число адресуемых узлов в данной сети*). IP-пакет представляет собой форматированный блок информации, передаваемый по вычислительной сети. Соединения вычислительных

сетей, которые не поддерживают пакеты, такие как традиционные соединения типа «точка-точка» в телекоммуникациях, просто передают данные в виде последовательности байтов, символов или битов. При использовании пакетного форматирования сеть может передавать длинные сообщения более надежно и эффективно.

1. Основные функции API, использованные в данной работе.

- `WSAStartup(WORD wVersionRequested, LPWSADATA lpWSADATA)`
- `WSACleanup (void)`
- `WSAGetLastError (void)`
- `u_short htons (u_short hostshort)`
- `socket (int af, int type, int protocol)`
- `bind (SOCKET s, const struct sockaddr FAR* name, int namelen)`
- `recvfrom (SOCKET s, char FAR* buf, int len, int flags, struct sockaddr FAR* from, int FAR* fromlen)`
- `sendto (SOCKET s, const char FAR * buf, int len, int flags, const struct sockaddr FAR * to, int tolen)`

3. Разработка программы. Блок-схемы программы.

Программа-сервер ждет широковещательные пакеты по заданному порту, при получении выделяется поток в котором открывается отдельный сокет для возвращения содержимого пакета клиенту, после завершения работы сокет и поток удаляются, тем самым обеспечивая многопоточную работу сервера.

Программа-клиент генерирует при запуске случайное 4 байтовое число и отправляет его по заданному адресу, после отправки она ждет пакет и выводит содержимое этого пакета на экран.

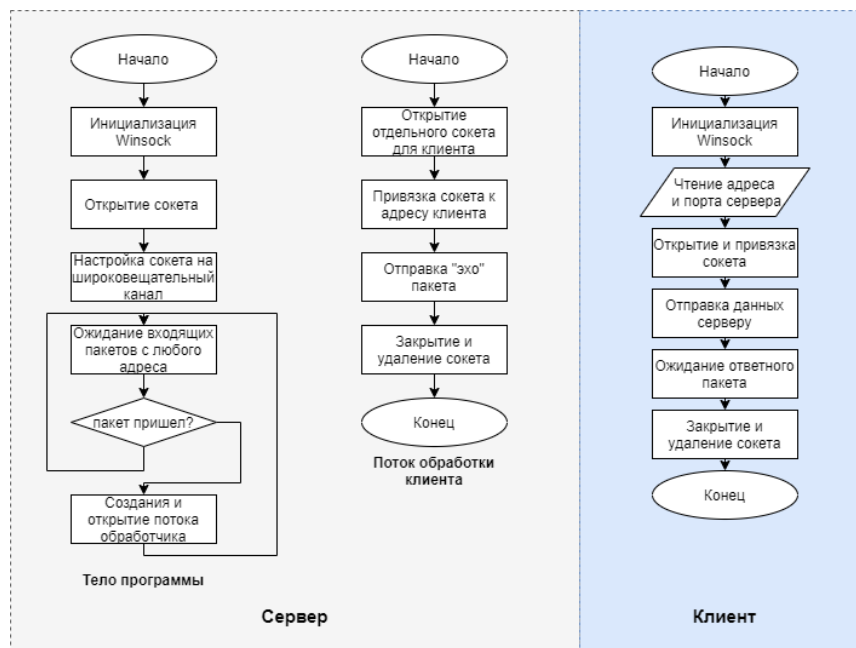


Рис. 1: Блок схемы программы сервер/клиент

4. Анализ функционирования разработанных программ.

Анализ функционирования программ проводился в пределах одного ПК (за неимением возможности протестировать передачу данных между несколькими машинами). Для одновременного запуска нескольких клиентов был написан скрипт, запускающий несколько экземпляров программы-клиент. Текст скрипта приведен ниже:

```
@echo off
set loopcount=1000
:loop
start /b client < input.txt
set /a loopcount=loopcount-1
if %loopcount%==0 goto exitloop
goto loop
:exitloop
pause
```

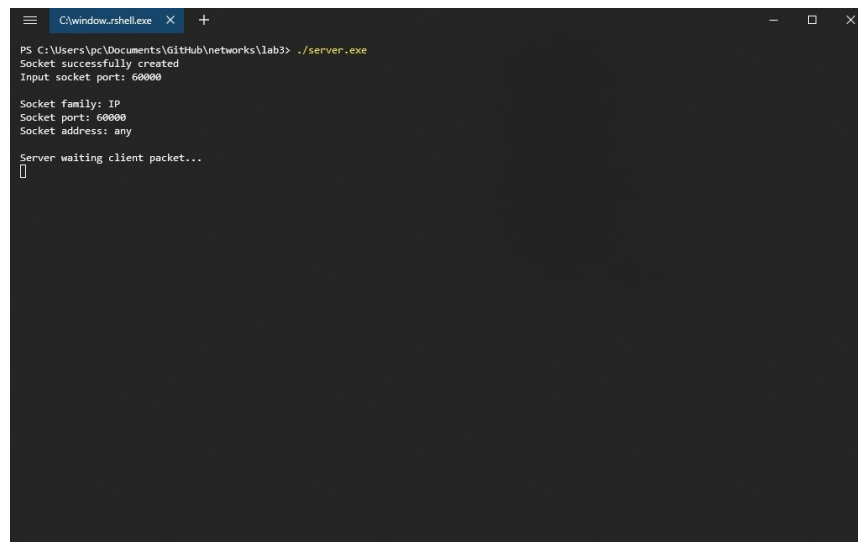
При запуске более 1000 экземпляров программы клиента не было обнаружено ошибок и все программы клиенты корректно получили обратно отправленный ими пакет.

5. Выводы.

Библиотека Winsock имеет удобную API для написания многопоточного сетевого программного обеспечения для операционной системы Windows, с использованием протокола IP. Протокол IP не гарантирует доставку пакетов и порядок их доставки.

6. Тексты программ. Скриншоты программ.

Тексты программ см. в приложении.

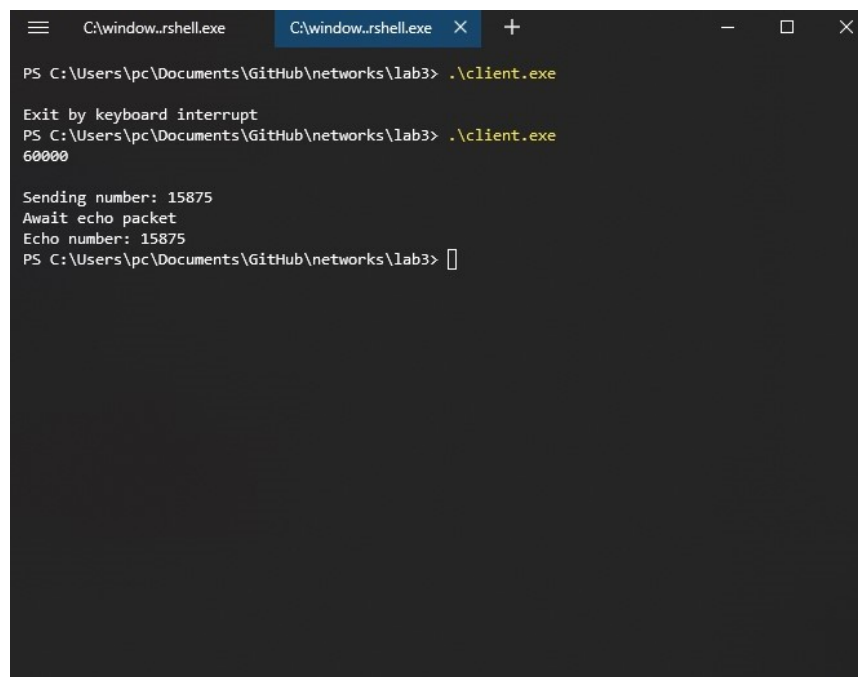


```
PS C:\Users\pc\Documents\GitHub\networks\lab3> ./server.exe
Socket successfully created
Input socket port: 60000

Socket family: IP
Socket port: 60000
Socket address: any

Server waiting client packet...
█
```

Рис. 2: Программа-сервер



```
PS C:\Users\pc\Documents\GitHub\networks\lab3> .\client.exe

Exit by keyboard interrupt
PS C:\Users\pc\Documents\GitHub\networks\lab3> .\client.exe
60000

Sending number: 15875
Await echo packet
Echo number: 15875
PS C:\Users\pc\Documents\GitHub\networks\lab3> █
```

Рис. 3: Программа-клинет

Приложение

Содержимое файла client.c

```
#include <windows.h>
#include <winsock.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>

void intHandler(int dummy) {
    printf("Exit by keyboard interrupt\n");
    exit(0);
}

int main(){
    signal(SIGINT, intHandler);

    time_t t;
    srand((unsigned) time(&t));
    int data = rand();

    WSADATA wsaData = {0};
    int iResult = WSStartup(MAKEWORD(2, 2), &wsaData);

    //создаём сокет
    int s = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
    if(s < 0){
        perror("Socket creating error\n");
        return 1;
    };
    printf("Socket successfully created\n");

    struct sockaddr_in addr;
    //семейство IP
    addr.sin_family = AF_INET;
    //порт
    unsigned short port;
    printf("Input socket port: ");
    scanf("%hu",&port);
    addr.sin_port = htons(port);

    addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    printf("\nSocket family: IP\n");
    printf("Socket port: %hu\n",port);
    printf("Socket address: 127.0.0.1\n\n");

    printf("Press Enter to start transsmition...\n");
    getchar();
    getchar();
    printf("Sending identification packet to server\n");
    printf("Sending number: %d\n",data);
    int ret_val = sendto(s,&data, sizeof(int), 0, (SOCKADDR *) & addr, sizeof (addr));
    if (ret_val == SOCKET_ERROR) {
        wprintf(L"\nsendto failed with error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 1;
    }
}
```

```

printf("Await echo packet\n");
int echo_data;
struct sockaddr_in sender_addr;
int sender_addr_size = sizeof(sender_addr);

ret_val = recvfrom(s,&echo_data,sizeof(int),0,(SOCKADDR*)&sender_addr, &sender_addr_size);
if (ret_val == SOCKET_ERROR) {
    wprintf(L"\nrecvfrom failed with error: %d\n", WSAGetLastError());
    closesocket(s);
    WSACleanup();
    return 1;
};
printf("Echo number: %d\n",echo_data);
return 0;
}

```

Содержимое файла input.txt

60000

Содержимое файла makefile

```

all: build-all

build-all: build-server build-client

build-client:
    gcc client.c -o client.exe -lws2_32

build-server:
    gcc server.c -o server.exe -lws2_32

```

Содержимое файла server.c

```

#include <windows.h>
#include <winsock.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>

typedef struct _thread_arg{
    struct sockaddr_in adr;
    int data;
} thread_arg;

void send_echo_adr(thread_arg *arg){
    printf("\nReceived packet from client\n");
    printf("Creating thread for response\n");
    printf("Received number: %d\n",arg->data);
    printf("Send echo packet\n");
    int s = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
    if(s < 0){
        perror("Socket creating error\n");
        return 1;
    };
    int ret_val = sendto(s,&arg->data, sizeof(int), 0, (SOCKADDR *) &arg->adr, sizeof
↵ (arg->adr));
    if (ret_val == SOCKET_ERROR) {
        wprintf(L"sendto failed with error: %d\n", WSAGetLastError());
        closesocket(s);
    }
}

```

```

        WSACleanup();
        return 1;
    }
    printf("Close thread\n\n");
    closesocket(s);
    return;
}

void intHandler(int dummy) {
    printf("Exit by keyboard interrupt\n");
    exit(0);
}

int main(){
    signal(SIGINT, intHandler);
    WSADATA wsaData = {0};
    int iResult = WSASStartup(MAKEWORD(2, 2), &wsaData);

    //создаём сокет
    int s = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
    if(s < 0){
        perror("Socket creating error\n");
        return 1;
    };
    printf("Socket successfully created\n");

    struct sockaddr_in addr;
    //семейство IP
    addr.sin_family = AF_INET;
    //порт
    unsigned short port;
    printf("Input socket port: ");
    scanf("%hu",&port);
    addr.sin_port = htons(port);
    //слушаем всю сеть
    addr.sin_addr.s_addr = htonl(INADDR_ANY);
    printf("\nSocket family: IP\n");
    printf("Socket port: %hu\n",port);
    printf("Socket address: any\n\n");

    //назначаем сокету адрес
    if(bind(s, (SOCKADDR *)&addr, sizeof(addr)) < 0){
        printf(L"bind failed with error %d\n", WSAGetLastError());
        return 2;
    };

    printf("Server waiting client packet...\n");
    int rec_data;
    while(1){
        struct sockaddr_in sender_addr;
        int sender_addr_size = sizeof(sender_addr);
        int ret_val =
        ↪ recvfrom(s,&rec_data,sizeof(int),0,(SOCKADDR*)&sender_addr,&sender_addr_size);
        if (ret_val == SOCKET_ERROR) {
            wprintf(L"recvfrom failed with error: %d\n", WSAGetLastError());
            closesocket(s);
            WSACleanup();
            return 1;
        }else{
            thread_arg arg;

```



```

        arg.data = rec_data;
        arg.adr = sender_addr;
        HANDLE thread = CreateThread(NULL,0,send_echo_adr,&arg,0,NULL);
    };
};

return 0;
}

```

Содержимое файла test.bat

```

@echo off
set loopcount=10000
:loop
start /b client < input.txt
set /a loopcount=loopcount-1
if %loopcount%==0 goto exitloop
goto loop
:exitloop
pause

```