**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3

дисциплина: Сети ЭВМ и телекоммуникации

тема: «Программирование протокола IP с использованием

библиотеки Winsock»

Выполнил: ст. группы ВТ-32

Воскобойников И. С.

Проверил: Федотов Е.А.

Белгород 2021

**Цель работы:** изучить принципы и характеристику протокола IP и составить программу для приема/передачи пакетов.

**Задание**

1. Разработать программу “Сервер”, которая принимает запросы от клиентов и посылает им в качестве ответа некоторое сообщение.
2. Разработать программу “Клиент”, которая посылает запрос серверу и “ждет” от него ответного сообщения.
3. Провести анализ функционирования разработанных программ (одновременная работа 2-х, 3-х и т.д. приложений на 2-х, 3-х и т.д. компьютерах ЛВС), сделать выводы.

**Теоретические сведения**

Internet Protocol или IP (англ. internet protocol – межсетевой протокол) – 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 сетиставится 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).

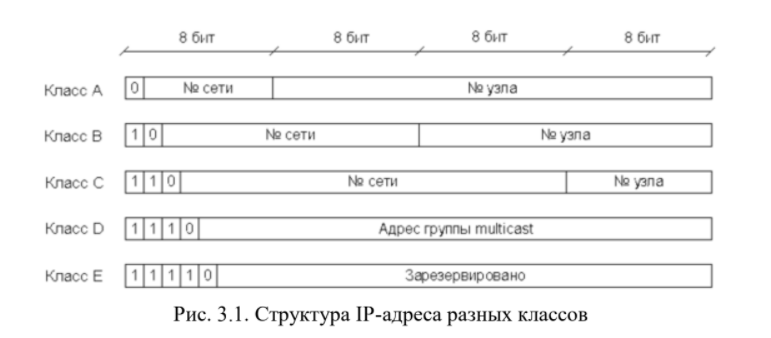
IP-пакет представляет собой форматированный блок информации, передаваемый по вычислительной сети. Соединения вычислительных сетей, которые не поддерживают пакеты, такие как традиционные соединения типа «точка-точка» в телекоммуникациях, просто передают данные в виде последовательности байтов, символов или битов. При использовании пакетного форматирования сеть может передавать длинные сообщения более надежно и эффективно.

**Структура IP адреса**

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например, 128.10.2.30 – традиционная десятично-точечная форма представления адреса, 10000000 00001010 00000010 00011110 - двоичная форма представления этого же адреса.

**Классы сетей IP**

IP-адреса разделяются на 5 классов: A, B, C, D, E. Адреса классов A, B и C делятся на две логические части: номер сети и номер узла. На рис. 3.1 показана структура IP-адреса разных классов.



Идентификатор сети, также называемый адресом сети, обозначает один сетевой сегмент в более крупной объединенной сети, использующей протокол TCP/IP. IP-адреса всех систем, подключенных к одной сети, имеют один и тот же идентификатор сети. Этот

идентификатор также используется для уникального обозначения каждой сети в более крупной объединенной сети.

Идентификатор узла, также называемый адресом узла, определяет узел TCP/IP (рабочую станцию, сервер, маршрутизатор или другое устройство) в пределах каждой сети. Идентификатор узла уникальным образом обозначает систему в том сегменте сети, к которой она подключена.

У адресов класса A старший бит установлен в 0. Длина сетевого префикса 8 бит. Для номера узла выделяется 3 байта (24 бита). Таким образом, в классе A может быть 126 сетей (27 - 2, два номера сети имеют специальное значение). Каждая сеть этого класса может поддерживать максимум 16777214 узлов (224 - 2). Адресный блок класса A может содержать максимум 231 уникальных адресов, в то время как в протоколе IP версии 4 возможно существование 232 адресов. Таким образом, адресное пространство класса A занимает 50% всего адресного пространства протокола IP версии 4. Адреса класса A предназначены для использования в больших сетях, с большим количеством узлов. На данный момент все адреса класса A

распределены.

У адресов класса B два старших бита установлены в 1 и 0 соответственно. Длина сетевого префикса – 16 бит. Поле номера узла тоже имеет длину 16 бит. Таким образом, число сетей класса B равно 16384 (214); каждая сеть класса B может поддерживать до 65534 узлов (216 - 2). Адресный блок сетей класса Класс B предназначен для применения в сетях среднего размера. У адресов класса C три старших бита установлены в 1, 1 и 0

соответственно. Префикс сети имеет длину 24 бита, номер узла - 8 бит. Максимально возможное количество сетей класса C составляет 2097152 (221). Каждая сеть может поддерживать максимум 254 узла (28 - 2). Класс C предназначен для сетей с небольшим количеством узлов. Адреса класса D представляют собой специальные адреса, не относящиеся к отдельным сетям. Первые 4 бита этих адресов равны 1110. Таким образом, значение первого октета этого диапазона адресов находится в пределах от 224 до 239.

Адреса класса D используются для многоадресных пакетов, с помощью которых во многих разных протоколах данные передаются многочисленным группам узлов. Эти адреса можно рассматривать как заранее запрограммированные в логической структуре большинства сетевых устройств. Это означает, что при обнаружении в пакете адреса получателя такого типа устройство на него обязательно отвечает. Например, если один из хостов передает пакет с IP-адресом получателя 224.0.0.5, на него отвечают все маршрутизаторы (использующие протокол OSPF), которые находятся в сегменте сети с этим адресом Ethernet.

Адреса в диапазоне 240.0.0.0 - 255.255.255.255 называются адресами класса E. Первый октет этих адресов начинается с битов 1111. Эти адреса зарезервированы для будущих дополнений в схеме адресации IP. Но возможность того, что эти дополнения когда-либо будут приняты, находится под вопросом, поскольку уже появилась версия 6 протокола IP (IPv6).

**Служебные IP-адреса**

Некоторые IP-адреса являются зарезервированными. Для таких адресов существуют следующие соглашения об их особой интерпретации:

1. Если все биты IP-адреса установлены в нуль, то он обозначаетадрес данного устройства.

2. Если в поле номера сети стоят нули, то считается, что

получатель принадлежит той же самой сети, что и отправитель.

3. Если все биты IP-адреса установлены в единицу, то пакет стаким адресом должен рассылаться всем узлам, находящимся в той же сети, что и отправитель. Такая рассылка называется ограниченным широковещательным сообщением.

4. Если все биты номера узла установлены в нуль, то пакет предназначен для данной сети.

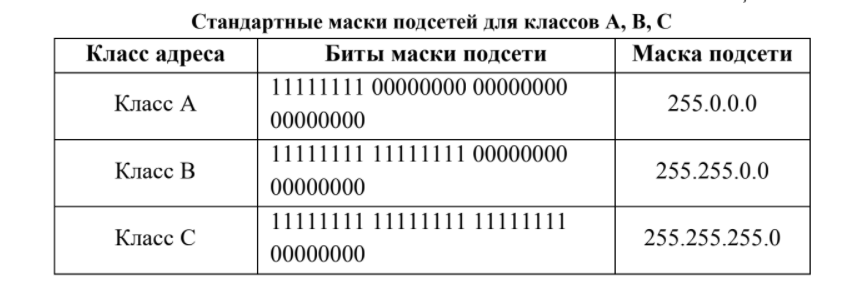
5. Если все биты в поле номера узла установлены в единицу, то пакет рассылается всем узлам сети с данным номером сети. Такая рассылка называется широковещательным сообщением.

6. Если первый октет адреса равен 127, то адрес обозначает тот же самый узел. Такой адрес используется для взаимодействия процессов на одной и той же машине (например, для целей тестирования). Этот адрес имеет название возвратного.

**Маска подсети**

Поля номеров сети и подсети образуют расширенный сетевой префикс. Для выделения расширенного сетевого префикса используется маска подсети (subnet mask). Маска подсети – это 32-разрядное двоичное число, в разрядах расширенного префикса содержащая единицу; в остальных разрядах находится ноль. Расширенный сетевой префикс получается побитным сложением по модулю два (операция XOR) IP-адреса и маски подсети. При таком построении очевидно, что число подсетей представляет собой степень двойки - 2^n , где n - длина поля номера подсети. Таким образом, характеристики IP-адреса полностью задаются собственно IP- адресом и маской подсети.

Стандартные маски подсетей для классов А, В, С приведены в табл. 3.1.



Для упрощения записи применяют следующую нотацию (так называемая CIDR-нотация): IP-адрес/длина расширенного сетевого префикса. Например, адрес 192.168.0.1 с маской 255.255.255.0 будет в данной нотации выглядеть как 192.168.0.1/24 (24 – это число единиц, содержащихся в маске подсети).

Для разбития сети на подсети необходимо найти минимальную степень двойки, большую или равную числу требуемых подсетей. Затем эту степень прибавить к префиксу сети. Количество IP-адресов в каждой подсети будет на 2 меньше теоретически возможного, потому что сеть должна будет вместить адрес сети и броадкастовый адрес.

**Функции библиотеки Winsock для работы с протоколом IP**

Функция **WSAStartup (WORD wVersionRequested, LPWSADATA lpWSAData)** необходима для инициализациибиблиотеки Winsock. Здесь wVersionRequested – запрашиваемая версияwinsock; lpWSAData – структура, в которую возвращается информацияпо инициализированной библиотеке (статус, версия и пр.). В случаеуспеха возвращает 0, иначе возвращает код возникшей ошибки.

Функция **WSAGetLastError (void)** возвращает код ошибки возникшей при выполнении последней операции.

Функция **WSACleanup (void)** очищает память, занимаемую библиотекой Winsock. Возвращает 0, если операция была выполнена успешно, иначе возвращает SOCKET\_ERROR и номер ошибки можно получить при помощи функции WSAGetLastError.

Функция **u\_short htons (u\_short hostshort)** осуществляет перевод целого короткого числа из порядка байт, принятого на компьютере, в сетевой порядок байт. hostshort – число, которое необходимо преобразовать. Возвращает преобразованное число [11].

Функция **socket (int af, int type, int protocol)** нужна для создания и инициализации сокета. Здесь af – сведения о семействе адресов. Для интернет-протоколов указывается константа AF\_INET; type – тип передаваемых данных (поток или дейтаграммы). В данной лабораторной работе используется константа SOCK\_DGRAM; protocol – протокол передачи данных. Для протокола IP используется константа IPPROTO\_IP. Функция возвращает дескриптор созданного сокета.

Функция **bind (SOCKET s, const struct sockaddr FAR\* name, int namelen)** привязывает адрес и порт к ранее созданному сокету Здесь s– дескриптор сокета; name – структура, содержащая нужный адрес ипорт; namelen – размер, в байтах, структуры name.

Функция **unsigned long inet\_addr (const char FAR \*cp)** конвертирует строку в значение, которое можно использовать вструктуре sockaddr\_in. Здесь cp – строка, которая содержит IP адрес вдесятично-точечном формате (например, 123.23.45.89). Возвращает IPадрес в виде целого числа, либо если произошла ошибка возвращаетконстанту INADDR\_NONE.

Для конвертации адреса в стандартный формат используетсяфункция **char (FAR \* inet\_ntoa(struct in\_addr in)**; in – IP-адрес,заданный в сетевом порядке расположения байт. Она возвращаетстроку, содержащую IP-адрес в стандартном строчном виде, с числамии точками.

Для определения IP адреса по имени используется **функция struct hostent FAR \* gethostbyname (const char FAR \* name)**. В качестве результата, функция возвращает структуру hostent. В случае автоматического распределения адресов и портов узнать

какой адрес и порт присвоен сокету можно при помощи функции **getsockname (SOCKET s, struct sockaddr FAR\* name, int FAR\* namelen)**. Если операция выполнена успешно, возвращает 0, иначе возвращает SOCKET\_ERROR и номер ошибки можно получить при помощи функции WSAGetLastError.

Функция **gethostname (char FAR \* name, int namelen)** записывает в буфер name длинной namelen стандартное имя хоста для данного компьютера. В случае успеха возвращает 0, иначе возвращает SOCKET\_ERROR.

Функция **sendto (SOCKET s, const char FAR \* buf, int len, int flags, const struct sockaddr FAR \* to, int tolen)** служит для передачиданных. Здесь s - дескриптор сокета; buf - указатель на буфер сданными, которые необходимо переслать; len - размер (в байтах)данных, которые содержатся по указателю buf; flags - совокупностьфлагов, определяющих, каким образом будет произведена передачаданных; to - указатель на структуру sockaddr, которая содержит адрессокета-приёмника; tolen - размер структуры to. Если операциявыполнена успешно, возвращает количество переданных байт, иначе

возвращает SOCKET\_ERROR и номер ошибки можно получить при помощи функции WSAGetLastError.

Функция **recvfrom (SOCKET s, char FAR\* buf, int len, int flags, struct sockaddr FAR\* from, int FAR\* fromlen)** служит для приемаданных. Если операция выполнена успешно, возвращает количествополученных байт, иначе возвращает SOCKET\_ERROR и номерошибки можно получить при помощи функции WSAGetLastError.

Функция **closesocket (SOCKET s)** нужна для закрытия сокета. Она возвращает 0, если операция была выполнена успешно, иначе возвращает SOCKET\_ERROR и номер ошибки можно получить при помощи функции **WSAGetLastError**.

**Выполнение работы**

**Задание 1**

Разработать программу “Сервер”, которая принимает запросы от клиентов и посылает им в качестве ответа некоторое сообщение.

**ServerIP.с**

#include <WinSock2.h>

#pragma comment(lib, "WS2\_32.lib")

#include <time.h>

#include <iostream>

using namespace std;

#define MAX\_DATA\_LEN 512

#pragma warning(disable: 4996)

void IPServer(){

char Buffer[MAX\_DATA\_LEN];

//Далее для инициализации Winsock вызываем функцию WSAStartup

WSADATA WSAData;

if (WSAStartup(0x0202, &WSAData)){

cout << "WSAStartup() faild: " << WSAGetLastError() << endl;

return;

}

//создание и инициализация сокета

SOCKET ServerSocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_IP);

if (ServerSocket == INVALID\_SOCKET){

cout << "Socet() error: " << WSAGetLastError() << endl;

WSACleanup();

return;

}

//необходимо привязать адрес и порт к сокету. При помощи gethostbyname получаем локальный адрес компьютера.

struct hostent \*HostName = gethostbyname("");

SOCKADDR\_IN ServerAddress;

u\_short Port = 0;

cout << "Enter port: ";

cin >> Port;

memset(&ServerAddress, 0, sizeof(ServerAddress));

ServerAddress.sin\_family = AF\_INET;

ServerAddress.sin\_addr.S\_un.S\_addr = \*(DWORD \*)HostName->h\_addr\_list[0];

ServerAddress.sin\_port = htons(Port);

cout << "Server IP: " << inet\_ntoa(ServerAddress.sin\_addr) << endl;

//привязывает адрес и порт к ранее созданному сокету

if (bind(ServerSocket, (PSOCKADDR)&ServerAddress, sizeof(ServerAddress)) == SOCKET\_ERROR){

cout << "Bind() error: " << WSAGetLastError() << endl;

}

cout << "Waiting connect ..." << endl;

int SizeFile = 0;

SOCKADDR\_IN ClientAddress;

int AddressLength = sizeof(ClientAddress);

// принимаем размер файла

int Result = recvfrom(ServerSocket, (char \*)&SizeFile, sizeof(SizeFile), 0, (PSOCKADDR)&ClientAddress, &AddressLength);

if (Result == SOCKET\_ERROR){

cout << "Recvfrom() error: " << WSAGetLastError() << endl;

WSACleanup();

return;

}

int CountPacket = SizeFile / MAX\_DATA\_LEN;//считаем пакеты

if (SizeFile % MAX\_DATA\_LEN)

CountPacket++;

char NameFile[50];

//передача данных

Result = sendto(ServerSocket, "test", strlen("test"), 0, (PSOCKADDR)&ClientAddress, sizeof(ClientAddress));

if (Result == SOCKET\_ERROR){

cout << "Sendto() error: " << WSAGetLastError() << endl;

WSACleanup();

return;

}

// принимаем имя файла

Result = recvfrom(ServerSocket, NameFile, 200, 0,(PSOCKADDR)&ClientAddress, &AddressLength);

if (Result == SOCKET\_ERROR){

cout << "Recvfrom() error: " << WSAGetLastError() << endl;

WSACleanup();

return;

}

NameFile[Result] = '\0';

int CountRecvPacket = 0;

int Value = 3000;

//устанавливаем опцию сокета.

setsockopt(ServerSocket, SOL\_SOCKET, SO\_RCVTIMEO, (char \*)&Value, sizeof(Value));

char newName[50];

strcpy(newName, "Copy\_");

strcat(newName, NameFile);

FILE \*File = fopen(newName, "wb");

cout << "Count packet: " << CountPacket << endl;

clock\_t Time1 = clock();

while (true){

//принимаем данные от клиента. пишем в буфер

Result = recvfrom(ServerSocket, Buffer, MAX\_DATA\_LEN, 0,(PSOCKADDR)&ClientAddress, &AddressLength);

if (Result == SOCKET\_ERROR)

break;

CountRecvPacket++;

fwrite(Buffer, 1, Result, File);//пишем в файл

}

cout << "Time: " << (double)(clock() - Time1) / CLOCKS\_PER\_SEC << " sec" << endl;

cout << "Count lost packets: " << CountPacket - CountRecvPacket << endl;

fclose(File);

closesocket(ServerSocket);

//очистка памяти, занимаемой библиотекой Winsock

if (WSACleanup())

cout << "WSACleanup() error: " << WSAGetLastError() << endl;

}

int main(){

IPServer();

system("pause");

return 0;

}

**Задание 2**

Разработать программу “Клиент”, которая посылает запрос серверу и “ждет” от него ответного сообщения.

**ClientIP.с**

#include <WinSock2.h>

#pragma comment(lib, "WS2\_32.lib")

#include <sys/stat.h>

#include <iostream>

using namespace std;

#pragma warning(disable: 4996)

#define MAX\_DATA\_LEN 512

int GetFileSize(const char \*fileName){

struct stat FileStat;

stat(fileName, &FileStat);

return FileStat.st\_size;

}

void IPClient(){

WSADATA WSAData;

if (WSAStartup(0x0202, &WSAData)){

cout << "WSAStartup() faild: " << WSAGetLastError() << endl;

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*создание сокета\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SOCKET ClientSocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_IP);

if (ClientSocket == INVALID\_SOCKET){

cout << "Socket error: " << WSAGetLastError() << endl;

WSACleanup();

return;

}

//выбор файла для передачи

char NameFile[50];

cout << "Enter file name: "; cin >> NameFile;

FILE \*File = fopen(NameFile, "rb");

if (File == NULL){

cout << "Open file error!" << endl;

WSACleanup();

return;

}

int SizeFile = GetFileSize(NameFile);

//отправление данных на сервер и прием ответа от сервера

SOCKADDR\_IN ServerAddress;

u\_short Port = 0;

cout << "Enter port: "; cin >> Port;

char IPAddress[50];

cout << "Enter IP address server: "; cin >> IPAddress;

memset(&ServerAddress, 0, sizeof(ServerAddress));

ServerAddress.sin\_family = AF\_INET;

ServerAddress.sin\_addr.S\_un.S\_addr = inet\_addr(IPAddress);

ServerAddress.sin\_port = htons(Port);

sendto(ClientSocket, (char \*)&SizeFile, sizeof(SizeFile), 0, (PSOCKADDR)&ServerAddress, sizeof(ServerAddress));

char Buffer[MAX\_DATA\_LEN];

recvfrom(ClientSocket, Buffer, MAX\_DATA\_LEN, 0, (PSOCKADDR)&ServerAddress, nullptr);

//отправляем имя файла и сам файл

sendto(ClientSocket, NameFile, strlen(NameFile), 0, (PSOCKADDR)&ServerAddress, sizeof(ServerAddress));

int CountPacket = SizeFile / MAX\_DATA\_LEN;

if (SizeFile % MAX\_DATA\_LEN)

CountPacket++;

cout << "Count packets = " << CountPacket << endl;

while (!feof(File)){

int Result = fread((void \*)Buffer, 1, MAX\_DATA\_LEN, File);

sendto(ClientSocket, Buffer, Result, 0, (PSOCKADDR)&ServerAddress, sizeof(ServerAddress));

}

fclose(File);

cout << "File " << NameFile << " sended!" << endl;

closesocket(ClientSocket);

if (WSACleanup())

cout << "WSACleanup() error: " << WSAGetLastError() << endl;

}

int main(){

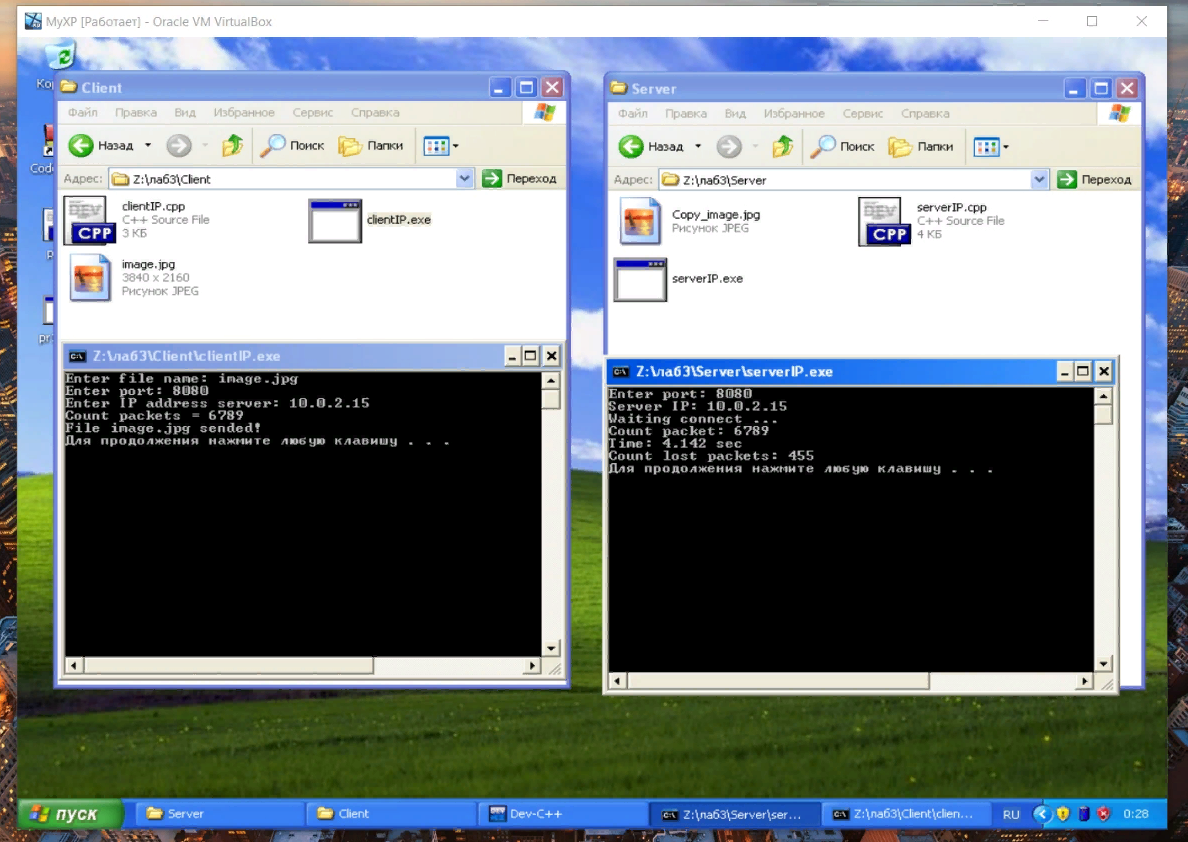
IPClient();

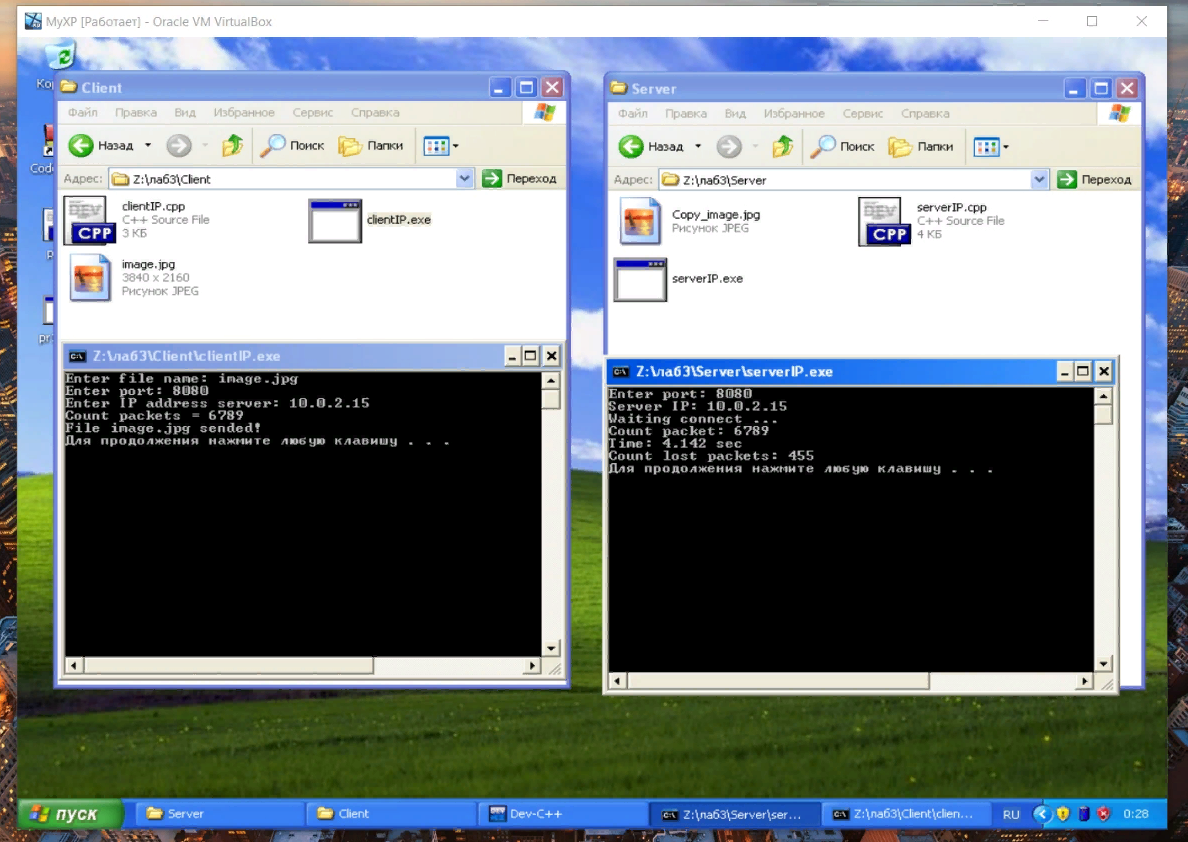
system("pause");

return 0;

}

Клиент и сервер IP:





**Задание 3**

Провести анализ функционирования разработанных программ (одновременная работа 2-х, 3-х и т.д. приложений на 2-х, 3-х и т.д. компьютерах ЛВС), сделать выводы.

**Анализ функционирования разработанных программ**

|  |  |
| --- | --- |
| 6.2 мб | https://pinterest.ru.com/images/2018/11/24/MORE-KAMENISTYI-BEREG.jpg |

Размер файла: 6213147 байт

Количество пакетов: 12136

IPX:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Время передачи, мс | Число утерянных пакетов, шт | Процент потерь |
| 1 сервер 1 клиент | 3531 | 16 | 0.132% |
| 2 сервера 2 клиента | 3547  3532 | 1  14 | 0.001%  0.115% |
| 3 сервера 3 клиента | 3547  3547  3563 | 97  145  129 | 0.799%  1.195%  1.063% |

**Вывод:** в процессе выполнения лабораторной работы были изучены принципы и характеристики протокола IP, а также была разработана программа для приема/передачи пакетов с использованием библиотеки Winsock. В процессе тестирования разработанных программ, было выявлено, что протокол IP не гарантирует передачу данных без потерь. Также, количество клиентов/серверов влияет на время передачи данных и процент потерь.

**Ответы на вопросы**

**1. Сущность протокола IP.**

Internet Protocol или IP (англ. internet protocol - межсетевой протокол) - 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.

**2. Опишите структуру IP-адресов в классах A, B, C, D, E.**

IP-адреса разделяются на 5 классов: A, B, C, D, E. Адреса классов A, B и C делятся на две логические части: номер сети и номер узла.

Структура IP-адреса разных классов.



**3. Какие IP-адреса являются зарезервированными для специального использования?**

Некоторые IP-адреса являются зарезервированными. Для таких адресов существуют следующие соглашения об их особой интерпретации:

1. Если все биты IP-адреса установлены в нуль, то он обозначает адрес данного устройства.

2. Если в поле номера сети стоят нули, то считается, что получатель принадлежит той же самой сети, что и отправитель.

3. Если все биты IP-адреса установлены в единицу, то пакет с таким адресом должен рассылаться всем узлам, находящимся в той же сети, что и отправитель. Такая рассылка называется ограниченным широковещательным сообщением.

4. Если все биты номера узла установлены в нуль, то пакет предназначен для данной сети.

5. Если все биты в поле номера узла установлены в единицу, то пакет рассылается всем узлам сети с данным номером сети. Такая рассылка называется широковещательным сообщением.

6. Если первый октет адреса равен 127, то адрес обозначает тот же самый узел. Такой адрес используется для взаимодействия процессов на одной и той же машине (например, для целей тестирования). Этот адрес имеет название возвратного.

**4. Что представляет собой маска подсети?**

Поля номеров сети и подсети образуют расширенный сетевой префикс. Для выделения расширенного сетевого префикса используется маска подсети (subnet mask).

**5. Как создать сокет для работы с протоколом IP?**

Функция socket (int af, int type, int protocol) нужна для создания и инициализации сокета. Здесь af – сведения о семействе адресов. Для интернет-протоколов указывается константа AF\_INET; type – тип передаваемых данных (поток или дейтаграммы). В данной лабораторной работе используется константа SOCK\_DGRAM; protocol – протокол передачи данных. Для протокола IP используется константа IPPROTO\_IP. Функция возвращает дескриптор созданного сокета.

**6. Какие функции Winsocket необходимо вызвать для вывода на экран IP-адреса данного компьютера?**

В случае автоматического распределения адресов и портов узнать какой адрес и порт присвоен сокету можно при помощи функции getsockname (SOCKET s, struct sockaddr FAR\* name, int FAR\* namelen). Если операция выполнена успешно, возвращает 0, иначе возвращает SOCKET\_ERROR и номер ошибки можно получить при помощи функции WSAGetLastError.

**7. Разбейте сеть 10.10.0.0 / 15 на 8 частей. Запишите диапазоны доступных адресов в каждой из получившихся сетей.**

Запишем адрес сети и маску в двоичном виде:  
01000110 01000110 00000000 00000000 (10.10.0.0)  
11111111 11111110 00000000 00000000 (255.254.0.0)

Увеличим длину маски на 3 бита:  
11111111 11111111 11000000 00000000   
Маска в десятичной системе счисления:  
255.255.192.0

Наложим новую маску на сеть:  
01000110 01000110 00000000 00000000 (10.10.0.0)  
11111111 11111111 11000000 00000000 (255.255.192.0)

Полученные 8 сетей:  
00001010 00001010 00000000 00000000 (10.10.0.0 / 18)  
00001010 00001010 01000000 00000000 (10.10.64.0 / 18)  
00001010 00001010 10000000 00000000 (10.10.128.0 / 18)  
00001010 00001010 11000000 00000000 (10.10.192.0 / 18)  
00001010 00001011 00000000 00000000 (10.71.0.0 / 18)  
00001010 00001011 01000000 00000000 (10.71.64.0 / 18)  
00001010 00001011 10000000 00000000 (10.71.128.0 / 18)  
00001010 00001011 11000000 00000000 (10.71.192.0 / 18)

Доступные в каждой из сетей адреса будут следующими:  
10.10.0.0 / 18:  10.10.0.1 - 10.10.63.254  
10.10.64.0 / 18:  10.10.64.0 - 10.10.127.254  
10.10.128.0 / 18:  10.10.128.1 – 10.10.191.254  
10.10.192.0 / 18:  10.10.192.1 - 10.10.255.254  
10.11.0.0 / 18:  10.11.0.1 - 10.11.63.254  
10.11.64.0 / 18:  10.11.64.1 - 10.11.127.254  
10.11.128.0 / 18:  10.11.128.1 - 10.11.191.254  
10.11.192.0 / 18:  10.11.192.1 – 10.11.255.254