**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ 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-31

Подкопаев Антон Валерьевич

Проверил: ст. пр. ПО и ВТАС

Федотов Евгений Александрович

**Белгород 2020**

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

**Краткие 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 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.

**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 рисунке ниже показана структура IP-адреса разных классов.

**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 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.

**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 - 2n, 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 таблице ниже.

**Стандартные маски подсетей для классов А, В, С**

|  |  |  |
| --- | --- | --- |
| **Класс адреса** | **Биты маски подсети** | **Маска подсети** |
| Класс А | 11111111 00000000 00000000 00000000 | 255.0.0.0 |
| Класс B | 11111111 11111111 00000000 00000000 | 255.255.0.0 |
| Класс С | 11111111 11111111 11111111 00000000 | 255.255.255.0 |

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).

**Задание к работе**

1. Разработать программу “Сервер” (на языке программирования Pascal или C), которая принимает запросы от клиентов и посылает им в качестве ответа некоторое сообщение.

2. Разработать программу “Клиент” (на языке программирования Pascal или C), 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.

**Анализ функционирования разработанной программы.**

При передаче без задержек, IP протокол, так же, как и SPX в предыдущей лабораторной работе обеспечивает передачу абсолютно всех пакетов без потерь, но протокол IP в отличие от SPX имеет гораздо быструю скорость передачи данных, например, для передачи 46,6 МБ (48 943 200 байт) по SPX протоколу требуется ~40 секунд, в то время как протокол IP справляется менее чем за секунду. При сравнении скоростей передачи, следует отметить тот факт, что в предыдущей лабораторной работе мы передавали между компьютерами (виртуальными машинами, причём работающим на ОС DOS), а в этой лабораторной работе мы в нашей программе отправляем на localhost, т.е. на наш же компьютер, который причём работает не на DOS, а на более совершенной ОС Windows 10.

**Вывод:** в данной работе мы разобрались изучили принципы и характеристики протокола IP, а также закрепили свои теоретические знания практикой, разработав программу для приема/передачи пакетов с использованием библиотеки Winsock.

*Приложение*

**Server.c**

#include <stdio.h>

#include <string.h>

#include <stdio.h>

#include <windows.h>

#include <windowsx.h>

#include <commctrl.h>

#include <winsock.h>

#include <cstdlib>

#include <algorithm>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <iostream>

#define MAX 65000

int main()

{

WSADATA wsaData = {0};

int iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

unsigned char buffer[MAX+1];

//создаём сокет

int s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_IP);

if(s < 0)

{

perror("Error calling socket"); return 0;

}

//определяем прослушиваемый порт и адрес

struct sockaddr\_in addr;

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(4161);

addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if( bind(s, (struct sockaddr \*)&addr, sizeof(addr)) < 0 )

{

perror("Error calling bind"); return 0;

}

//помечаем сокет, как пассивный - он будет слушать порт

if( listen(s, 5) ) //5 - максимальная длина очереди ожидающих соединений

{

perror("Error calling listen"); return 0;

}

//начинаем слушать, для соединения создаём другой сокет, в котором можем общаться.

int s1 = accept(s, NULL, NULL);

if( s1 < 0 )

{

perror("Error calling accept"); return 0;

}

//читаем данные из сокета и записываем в файл

FILE \*stream;

int logging = 0, Err, i, nulls = 0, val, batchsize = MAX;

stream = fopen("c:\\a\\image.jpg", "wb");

while (nulls != batchsize)

{

nulls = 0;

if(logging) printf("\nWait for packet...\nWAIT...\n ");

int rc = recv(s1, (char\*)&buffer, MAX, 0);

if (Err != SOCKET\_ERROR)

{

for (i = 0; i < batchsize; i++)

{

if (logging) printf("(#%i)%i ", i + 1, buffer[i]);

if (buffer[i] == 0) nulls++;

}

if (nulls != batchsize) fwrite(buffer, batchsize, 1, stream);

}else{

printf("Error %i!\n",Err);

}

if(logging) printf("\n");

}

printf("\nDownload finished!");

fclose(stream);

return 0;

}

**Client.c**

#include <stdio.h>

#include <string.h>

#include <stdio.h>

#include <windows.h>

#include <windowsx.h>

#include <commctrl.h>

#include <winsock.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <errno.h>

#define MAX 65000

void func(int sockfd)

{

unsigned char buffer[MAX];

int n, i, logging = 0, timedelay = 0;

FILE \*ptrFile;

char filename[256] = "c:\\a\\a.jpg";

long last = 0, lSize, sizesended=0;

size\_t result = 1;

if ((ptrFile = fopen(filename, "rb+")) == NULL) printf("Cannot open file\n"); printf("\n");

fseek(ptrFile, 0L, SEEK\_END);

lSize = ftell(ptrFile);

printf("%i", lSize);

rewind(ptrFile);

while (last < lSize)

{

for (i = 0; i <= sizeof(buffer); i++) buffer[i] = 0;

result = fread(buffer, sizeof(buffer), 1, ptrFile);

send(sockfd, buffer, sizeof(buffer),0);

if(logging) {

for (i = 0; i < sizeof(buffer); i++) printf("(#%i)%i ", i + 1, buffer[i]);

printf("\nMessage has been sended\n");

}

sizesended+=sizeof(buffer);

last = ftell(ptrFile);

}

for (i = 0; i <= sizeof(buffer); i++) buffer[i] = 0;

send(sockfd, buffer, sizeof(buffer),0);

if(logging) for (i = 0; i < sizeof(buffer); i++) printf("(#%i)%i ", i + 1, buffer[i]);

printf("\nMessage has been sended\n");

}

int main()

{

WSADATA wsaData = {0};

int iResult = WSAStartup(MAKEWORD(2, 2), &wsaData), sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// Создание сокета и проверка, что при создании нет ошибок

sockfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_IP);

if (sockfd == -1) {

printf("socket creation failed\n"); exit(0);

}

else

printf("Socket successfully created\n");

bzero(&servaddr, sizeof(servaddr));

// Заполнение структуры для "Привязки" по IP и порту

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(4161);

// Подключение к серверу

if (connect(sockfd, (struct sockaddr\_in\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed\n"); exit(0);

}

else

printf("connected to the server\n");

// Читаем данные из сокета и записываем в файл

func(sockfd);

// Закрытие сокета

close(sockfd);

return 0;

}

**Результат работы программы**

Сервер:



Клиент:

