**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 №2**

**Дисциплина: Сети ЭВМ и телекоммуникации**

**Тема: Протоколы IPX/SPX в библиотеке Winsock**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: ст. пр. ПО и ВТАС

Федотов Евгений Александрович

**Белгород 2020**

**Цель работы:** изучить протоколы IPX/SPX, основные функции библиотеки Winsock и разработать программу для приема/передачи пакетов.

**Ход работы**

**Краткие теоретические сведения.**

**Протокол IPX** (*Internetwork Packet Exchange*) является оригинальным протоколом сетевого уровня стека Novell, разработанным в начале 80-х годов на основе протокола Internetwork Datagram Protocol (IDM) компании Xerox. Протокол IPX соответствует сетевому уровню модели OSI и поддерживает только дейтаграмный (без установления соединений) способ обмена сообщениями. В сети NetWare самая быстрая передача данных при наиболее экономном расходовании памяти реализуется именно протоколом IPX.

Для надежной передачи пакетов используется протокол транспортного уровня SPX (*Sequenced Packet Exchange*), который работает с установлением соединения и восстанавливает пакеты при их потере или повреждении. Если по каким-то причинам пакет не дошел до получателя, выполняется его повторная передача. Следовательно, последовательность отправления совпадает с последовательностью получения пакетов. Обмен пакетами на уровне сеанса связи реализован с помощью протокола SPX, который построен на базе IPX. **SPX** – протокол последовательного обмена пакетами (*Sequenced Packet Exchange Protocol*), разработанный Novell. Система адресов протокола SPX аналогична системе адресов протокола IPX и также состоит из 3 частей: номера сети, адреса станции и сокета. Протокол SPX использует такой же блок ECB для передачи и приёма пакетов, что и протокол IPX. Однако, пакет, передаваемый при помощи протокола SPX, имеет более длинный заголовок. Дополнительно к 30 байтам стандартного заголовка пакета IPX добавляется еще 12 байт

**Основные функции API, использованные в данной работе.**

* WSAStartup (WORD wVersionRequested, LPWSADATA lpWSAData)
* SAGetLastError (void)
* WSACleanup (void)
* socket (int af, int type, int protocol)
* bind (SOCKET s, const struct sockaddr FAR\* name, int namelen)
* listen (SOCKET s, int backlog)
* connect (SOCKET s, const struct sockaddr FAR\* name,int namelen)
* accept (SOCKET s, struct sockaddr FAR\* addr, int FAR\* addrlen)
* getsockname (SOCKET s, struct sockaddr FAR\* name, int FAR\* namelen)
* sendto (SOCKET s, const char FAR \* buf, int len, int flags, const struct sockaddr FAR \* to, int tolen)
* send (SOCKET s, const char FAR \* buf, int len, int flags)
* recvfrom (SOCKET s, char FAR\* buf, int len, int flags, struct sockaddr FAR\* from, int FAR\* fromlen)
* recv (SOCKET s, char FAR\* buf, int len, int flags)
* closesocket(SOCKET s)

**Задание к работе**

1. Разработать программу “Сервер”, которая посылает клиентам сети файл с использованием протокола IPX в среде DOS на языке программирования Pascal или C.

2. Разработать программу “Клиент”, которая принимает от сервера файл на языке программирования Pascal или C.

3. Провести анализ функционирования разработанных программ при передаче файла в формате \*.jpg размером не менее 20 Мб (одновременная работа 2-х, 3-х и т.д. приложений на 2-х, 3-х и т.д. компьютерах ЛВС), сделать выводы.

4. Провести сравнительный анализ протоколов IPX и SPX. Сделать выводы.

**Анализ функционирования разработанной программы.**

При задержке в 0 мс можно увидеть, что SPX действительно восстанавливает пакеты при их потере или повреждении и поэтому при передаче изображения 46,6 МБ (48 943 200 байт) у нас нету ни одного искажения и самое, что главное, время передачи заняло менее минуты.

**Выводы:**

Вывод: в данной работе мы разобрались с библиотекой Winsock и разработали программу “Сервер”, которая посылает клиентам сети файл с использованием протокола IPX/SPX на языке программирования C, а также разработали программу “Клиент”, которая принимает от сервера большой файл. В конце работы мы провели анализ функционирования наших двух программ.

*Приложение*

**Server.c**

#include <stdio.h>

#include <string.h>

#include <dos.h>

#include <stdio.h>

#include <windows.h>

#include <windowsx.h>

#include <commctrl.h>

#include <winsock.h> // ????????? ??????? ??????????

#include <wsipx.h> // IPX/SPX ?????????

#include <wsnwlink.h>

#define IPX\_NET\_SIZE 4

#define IPX\_NODE\_SIZE 6

#define IPX\_SOCKET\_SIZE 2

SOCKET spx\_skt = NULL;

SOCKADDR\_IPX addrIPX = { 0 };

SOCKADDR\_IPX sendIPX = { 0 };

char buff[80] = { 0 };

char localNetNum[IPX\_NET\_SIZE];

char localNodeNum[IPX\_NODE\_SIZE];

WORD SPX\_SOCKET = 0x5647 ;

int Err;

void SPX\_open()

{

SOCKADDR\_IPX addr\_ipx;

SOCKADDR\_IPX adr = { 0 };

WSADATA wsaData = { 0 };

int res = 0x00, i;

// aa?ney winsock.dll aeaeeioaee

WORD wVersionRequested = MAKEWORD(1, 1);

// eieoeaeecaoey winsock.dll

Err = WSAStartup(wVersionRequested, &wsaData);

//

SetConsoleTitle("SPX Server");

if (Err != 0)

{

printf("\n Tell the user that we could not find a usable WinSock DLL.\n\n");

printf("Press any key to exit ...");

}

else

{

spx\_skt = socket(AF\_IPX, SOCK\_DGRAM, NSPROTO\_IPX);

if (spx\_skt == INVALID\_SOCKET)

printf("\n Can not open IPX/SPX supported socket.\n\n");

else

{

int sz = sizeof(addr\_ipx);

// iaioeei aa

memset(&addr\_ipx, 0, sz);

addr\_ipx.sa\_family = AF\_IPX; // oei i?ioieiea

addr\_ipx.sa\_socket = htons(SPX\_SOCKET); // iiia? nieaoa

// aeiaei nieao (i?eaycuaaai aai e iiia?o nieaoa)

Err = bind(spx\_skt, &addr\_ipx, sz);

// ociaai iao aa?ann

getsockname(spx\_skt, &addr\_ipx, &sz);

// iao iiia? naoe

memcpy(localNetNum, addr\_ipx.sa\_netnum, IPX\_NET\_SIZE);

// iao iiia? ocea

memcpy(localNodeNum, addr\_ipx.sa\_nodenum, IPX\_NODE\_SIZE);

if (Err == SOCKET\_ERROR)

printf("\n Can not bind IPX/SPX supported socket.\n\n");

else

{ printf("\n Socket opening success!\n\n");

}

// cae?uaaao nieao

}

}

}

int main()

{

FILE \*stream;

int logging = 0;

int i, nulls = 0, val;

int size = sizeof(SOCKADDR\_IPX);

printf("Enable logging(0-false, 1-true) : "); scanf("%i", &logging);

//open socket

SPX\_open();

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

printf("%i\n\n", stream);

while (nulls != 80)

{

nulls = 0;

size = sizeof(SOCKADDR\_IPX);

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

sendIPX.sa\_family = AF\_IPX;

sendIPX.sa\_socket = htons(0x5648); // nieao ia?aaao?eea

memset(sendIPX.sa\_nodenum, 0xFF, sizeof(sendIPX.sa\_nodenum));

memcpy(sendIPX.sa\_netnum, addrIPX.sa\_netnum, 4);

sendIPX.sa\_nodenum[6] = '\0';

printf("WAIT...\n");

Err = recvfrom(

spx\_skt,

buff,

80,

0, &sendIPX, &size);

//for(i=0; i < size; i++){

// printf("%c ",buff[i]);

//}

//printf("\n");

if (Err != SOCKET\_ERROR)

{

for (i = 0; i < 80; i++)

{

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

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

}

if (nulls != 80)

{

fwrite(buff, sizeof(buff), 1, stream);

fflush(stream);

}

}else{

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

}

printf("\n");

Sleep(1);

}

printf("\nDownload finished!");

fclose(stream);

//closing socket

closesocket(spx\_skt);

}

**Client.c**

//---------------------------------------------------------------------------

#pragma hdrstop

//---------------------------------------------------------------------------

# pragma argsused

#include <stdio.h>

#include <string.h>

#include <windows.h>

#include <windowsx.h>

#include <commctrl.h>

#include <winsock.h > // ????????? ??????? ??????????

#include <wsipx.h > // IPX/SPX ?????????

#include <wsnwlink.h>

#define IPX\_NET\_SIZE 4

#define IPX\_NODE\_SIZE 6

#define IPX\_SOCKET\_SIZE 2

#include<stdlib.h>

#include <dos.h>

SOCKET spx\_skt = NULL;

SOCKADDR\_IPX addrIPX = { 0 };

SOCKADDR\_IPX sendIPX = { 0 };

char buff[80] = { 0 };

char localNetNum[IPX\_NET\_SIZE];

char localNodeNum[IPX\_NODE\_SIZE];

int set\_broadcast;

WORD SPX\_SOCKET = 0x5647;

SOCKADDR\_IPX addr\_ipx;

SOCKET ipx\_skt;

void openSocket()

{

WSADATA wsaData = { 0 };

// aa?ney winsock.dll aeaeeioaee

WORD wVersionRequested = MAKEWORD(1, 1);

// eieoeaeecaoey winsock.dll

int Err = WSAStartup(wVersionRequested, &wsaData);

//

SetConsoleTitle("SPX Client");

if (Err != 0)

{

printf("\n Tell the user that we could not find a usable WinSock DLL.\n\n");

printf("Press any key to exit ...");

}

else

{

// ?aaioa n nieaoaie

//char \*broadcast\_msg = "Some broadcast stuff";

ipx\_skt = socket(AF\_IPX, SOCK\_DGRAM, NSPROTO\_IPX);

addr\_ipx.sa\_family = AF\_IPX;

// iiia? nieaoa, aaiiue iiia? eniieucoaony iinueee SAP iaeaoia a iaoaa?a naoyo

addr\_ipx.sa\_socket = htons(0x5647); // = IPXSKT\_SAP

// aey oe?ieiaauaoaeuiuo iaeaoia

memset(addr\_ipx.sa\_netnum, 0, IPX\_NET\_SIZE);

memset(addr\_ipx.sa\_nodenum, 0xff, IPX\_NODE\_SIZE);

// onoaiaaeeaaai oeaa aey iinueee oe?ieiaauaoaeuiuo iaeaoia

set\_broadcast = 1;

setsockopt(ipx\_skt, SOL\_SOCKET, SO\_BROADCAST, (char\*) &set\_broadcast, sizeof(set\_broadcast));

}

}

void main()

{

int batchsize = 400;

int val1, val2,i,length;

int logging = 0;

unsigned char buffer[400];

FILE \* ptrFile;

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

int timedelay = 0;

long last = 0, offset = 0, lSize;

size\_t result = 1;

openSocket();

//printf("Input src file name : "); scanf("%s", &filename);

//printf("Input send delay : "); scanf("%i", &timedelay);

//printf("Enable logging(0-false, 1-true) : "); scanf("%i", &logging);

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);

length = batchsize;

while (last < lSize)

{

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

result = fread(buffer, batchsize, 1, ptrFile); // read 10 bytes to our buffer

sendto(ipx\_skt, buffer, batchsize, MSG\_DONTROUTE, &addr\_ipx, sizeof(addr\_ipx));

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

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

}

Sleep(timedelay);

last = ftell(ptrFile);

}

Sleep(timedelay);

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

sendto(ipx\_skt, buffer, batchsize, MSG\_DONTROUTE, &addr\_ipx, sizeof(addr\_ipx));

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

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

}

**Результат работы программы**



