**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 №1

дисциплина: Сети ЭВМ и телекоммуникации

тема: «Протокол сетевого уровня IPX»

Выполнил: ст. группы ВТ-32

Воскобойников И. С.

Проверил: Федотов Е. А.

Белгород 2021

**Цель работы:** изучить протокол сетевого уровня IPX, основные функции API драйвера IPX и разработать программу для приема/передачи данных.

**Краткие теоретические сведения**

**Протокол IPX** – это протокол сетевого уровня модели  взаимодействия открытых систем (OSI) реализующий передачу  пакетов (сообщений) между станциями сети на уровне датаграмм.  Датаграмма – это сообщение, доставка которого получателю не  гарантируется. Следовательно, для обеспечения надежной работы  нужно предусмотреть схему уведомления других станций о том, что  переданные ими пакеты успешно приняты и обработаны. Более того,  последовательность отправления пакетов передающим узлом может  отличаться от последовательности приема этих пакетов, что также  необходимо учитывать [1, 3, 7].

В процессе обмена сообщениями на уровне сеанса связи  участвуют только две станции сети. На уровне датаграмм есть  возможность посылать сообщение одновременно всем станциям сети.

Система адресов, используемая в протоколе IPX, представлена  несколькими компонентами: это номер сети, адрес станции в сети и  идентификатор программы на рабочей станции.

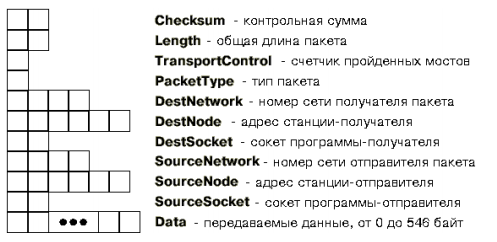
**Номер сети** - это номер сегмента сети, определяемого системным  администратором. Если в общей сети есть мосты, каждая отдельная  сеть, подключенная через мост, должна иметь свой, уникальный номер  сети.

**Адрес станции** - это число, которое является уникальным для  каждой рабочей станции. При использовании адаптеров Ethernet  уникальность обеспечивается изготовителем сетевого адаптера.  Специальный адрес FFFFFFFFFFFFh используется для рассылки

данных всем станциям данной сети одновременно.

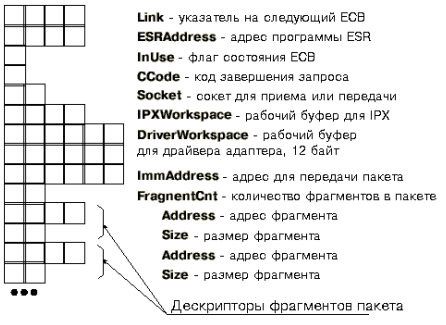
**Идентификатор программы на рабочей станции (сокет)** - число, которое используется для адресации определенной программы, работающей на компьютере. В среде мультизадачных операционных систем, на каждой рабочей станции в сети одновременно может быть запущено несколько приложений. Для того, чтобы послать данные конкретной программе, используется идентификация программ при помощи сокетов. Каждая программа, желающая принимать или передавать данные по сети, должна получить свой, уникальный для данной рабочей станции, идентификатор - сокет.

**Структура пакета**



Прикладное ПО взаимодействует с сетевым драйвером через драйвер IPX. Управление драйвером происходит с помощью блока ECB(Блок контроля события).

**Структура блока ECB**



Для осуществления взаимодействия с драйвером IPX нужно получить его адрес с помощью прерывания 2f поместив в ax 7A00h. А затем через эту точку входа в API вызывать функции для работы с IPX. Функциям передаются различные параметры(Блок ЕСВ, номер сокета, тип сокета).

Номер сокета передается в перевернутом виде, поэтому используем IntSwap

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

Для работы с IPX используются следующие основные функции.

Функция **IPXOpenSocket** предназначена для получения сокетов. На входе ВХ = 00h, AL = Тип сокета: 00h - короткоживущий; FFh - долгоживущий, DX = Запрашиваемый номер сокета или 0000h, если требуется получить динамический номер сокета. Байты номера сокета находятся в перевернутом виде. На выходе AL = Код завершения (00h - сокет открыт; FFh - этот  сокет уже был открыт раньше; FEh - переполнилась таблица сокетов),  DX = Присвоенный номер сокета.

Функция **IPXCloseSocket** предназначена для освобождения сокетов, т.к. сокеты являются ограниченным ресурсом. На входе ВХ =  0lh, DX = Номер закрываемого сокета. На выходе регистры не  используются.

Функция **IPXGetLocalTaget** применяется для вычисления значения непосредственного адреса, помещаемого в поле ImmAddress  блока ECB перед передачей пакета. На входе ВХ = 09h, ES:SI = Указатель на буфер длиной 10 байт, в который будет записан адрес станции, на которой работает данная

**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.

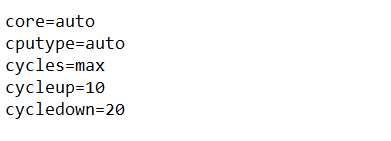
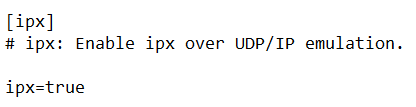
2. Разработать программу “Клиент”, которая принимает от сервера файл на языке программирования Pascal или C.

3. Провести анализ функционирования разработанных программ при передаче файла в формате \*.jpg размером не менее 1 Мб (одновременная работа 2-х, 3-х и т.д. приложений на 2-х, 3-х и т.д. компьютерах ЛВС), сделать выводы.

**Ход выполнения работы**

В данной лабораторной работе мы будем использовать **DOSbox** т.к в нём есть поддержка протокола **ipx.**

Для начала настраиваем DOS-BOX. В C:\Users\500a5\AppData\Local\DOSBox\dosbox-0.74-3.conf включаем максимальное значение для cycles. Включаем поддержку IPX, выключенную

****

В качестве языка программирования в данной лабораторной работе будем использовать Си и компилировать с помощью borland.

**Листинг файла ipx.h**

#define OPEN\_SOCKET 0x00  
#define CLOSE\_SOCKET 0x01  
#define SEND\_PACKET 0x03  
#define LISTEN\_FOR\_PACKET 0x04  
#define RELINQUISH\_CONTROL 0x0a  
  
#define SHORT\_LIVED 0  
#define LONG\_LIVED 0xff  
  
  
*// Внешние процедуры для инициализации и вызова драйвера IPX/SPX*void far ipxspx\_entry(void far \*ptr);  
int ipx\_init(void);  
  
  
  
struct IPXSPX\_REGS {  
 unsigned int ax;  
 unsigned int bx;  
 unsigned int cx;  
 unsigned int dx;  
 unsigned int si;  
 unsigned int di;  
 unsigned int es;  
};  
  
  
struct IPX\_HEADER {  
 unsigned int Checksum;  
 unsigned int Length;  
 unsigned char TransportControl;  
 unsigned char PacketType;  
 unsigned char DestNetwork[4];  
 unsigned char DestNode[6];  
 unsigned int DestSocket;  
 unsigned char SourceNetwork[4];  
 unsigned char SourceNode[6];  
 unsigned int SourceSocket;  
};  
  
  
struct ECB {  
 void far \*Link;  
 void far (\*ESRAddress)(void);  
 unsigned char InUse;  
 unsigned char CCode;  
 unsigned int Socket;  
 unsigned int ConnectionId;  
 unsigned int RrestOfWorkspace;  
 unsigned char DriverWorkspace[12];  
 unsigned char ImmAddress[6];  
 unsigned int FragmentCnt;  
 struct {  
 void far \*Address;  
 unsigned int Size;  
 } Packet[3];  
};  
  
unsigned IntSwap(unsigned i);  
int IPXOpenSocket(int SocketType, unsigned \*Socket);  
void IPXCloseSocket(unsigned \*Socket);  
void IPXListenForPacket(struct ECB \*RxECB);  
void IPXRelinquishControl(void);  
void IPXSendPacket(struct ECB \*TxECB);

**Листинг файла ipx.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <dos.h>  
#include "ipx.h"  
  
  
unsigned IntSwap(unsigned i) {  
 return((i>>8) | (i & 0xff)<<8);  
}  
  
  
int IPXOpenSocket(int SocketType, unsigned \*Socket) {  
  
 struct IPXSPX\_REGS iregs;  
  
 iregs.bx = OPEN\_SOCKET;  
 iregs.dx = IntSwap(\*Socket);  
 iregs.ax = SocketType;  
 ipxspx\_entry( (void far \*)&iregs );  
  
 \*Socket = IntSwap(iregs.dx);  
 return(iregs.ax);  
}  
  
  
  
void IPXCloseSocket(unsigned \*Socket) {  
  
 struct IPXSPX\_REGS iregs;  
  
 iregs.bx = CLOSE\_SOCKET;  
 iregs.dx = IntSwap(\*Socket);  
 ipxspx\_entry( (void far \*)&iregs );  
}  
  
  
void IPXListenForPacket(struct ECB \*RxECB) {  
  
 struct IPXSPX\_REGS iregs;  
  
 iregs.es = FP\_SEG((void far\*)RxECB);  
 iregs.si = FP\_OFF((void far\*)RxECB);  
 iregs.bx = LISTEN\_FOR\_PACKET;  
 ipxspx\_entry( (void far \*)&iregs );  
}  
  
  
  
void IPXSendPacket(struct ECB \*TxECB) {  
  
 struct IPXSPX\_REGS iregs;  
  
 iregs.es = FP\_SEG((void far\*)TxECB);  
 iregs.si = FP\_OFF((void far\*)TxECB);  
 iregs.bx = SEND\_PACKET;  
 ipxspx\_entry( (void far \*)&iregs );  
}  
  
  
void IPXRelinquishControl(void) {  
  
 struct IPXSPX\_REGS iregs;  
  
 iregs.bx = RELINQUISH\_CONTROL;  
 ipxspx\_entry( (void far \*)&iregs );  
}

**Листинг файла server.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <conio.h>  
#include <mem.h>  
#include <string.h>  
#include "ipx.h"  
  
#define BUFFER\_SIZE 512  
  
void main(void) {  
  
 int i;  
 char eof\_flag = 0;  
 FILE \*transfer\_file;  
  
*// Используем сокет 0x4567* static unsigned Socket = 0x4567;  
  
*// Этот ECB мы будем использовать и для приема  
// пакетов, и для их передачи.* struct ECB ECB;  
  
*// Заголовки принимаемых и передаваемых пакетов* struct IPX\_HEADER request\_header, response\_header;  
  
*// Буферы для принимаемых и передаваемых пакетов* unsigned char RxBuffer[BUFFER\_SIZE];  
 unsigned char TxBuffer[BUFFER\_SIZE];  
  
  
  
*// Проверяем наличие драйвера IPX и определяем  
// адрес точки входа его API* if(ipx\_init() != 0xff) {  
 printf("IPX driver not load!\n");  
 exit(-1);  
 }  
  
*// Открываем сокет, на котором мы будем принимать пакеты* if(IPXOpenSocket(SHORT\_LIVED, &Socket)) {  
 printf("Socket not open\n");  
 exit(-1);  
 };  
  
*// Подготавливаем ECB для приема пакета* memset(&ECB, 0, sizeof(ECB));  
 ECB.Socket = IntSwap(Socket);  
 ECB.FragmentCnt = 3;  
 ECB.Packet[0].Address = &request\_header;  
 ECB.Packet[0].Size = sizeof(request\_header);  
 ECB.Packet[1].Address = RxBuffer;  
 ECB.Packet[1].Size = BUFFER\_SIZE;  
 ECB.Packet[2].Addres = &eof\_flag;  
 ECB.Packet[2].Size = 1;  
  
 IPXListenForPacket(&ECB);  
  
  
 while(ECB.InUse) {  
 IPXRelinquishControl();  
 if(kbhit()) {  
 getch();  
 ECB.CCode = 0xfe;  
 break;  
 }  
 }  
 if(ECB.CCode == 0) {  
  
 printf("Transfer %s", RxBuffer);  
 transfer\_file = fopen(RxBuffer, "rb");  
  
  
 ECB.Socket = IntSwap(Socket);  
 ECB.FragmentCnt = 3;  
 ECB.Packet[0].Address = &response\_header;  
 ECB.Packet[0].Size = sizeof(response\_header);  
 ECB.Packet[1].Address = TxBuffer;  
 ECB.Packet[1].Size = BUFFER\_SIZE;  
 ECB.Packet[2].Address = &eof\_flag;  
 ECB.Packet[2].Size = 1;  
*// Подготавливаем заголовок пакета* response\_header.PacketType = 4;  
 memset(response\_header.DestNetwork, 0, 4);  
 memcpy(response\_header.DestNode, ECB.ImmAddress, 6);  
 response\_header.DestSocket = IntSwap(Socket);  
  
 for(;!eof\_flag;){  
 if (feof(transfer\_file)){  
 if(!transfer\_file)  
 eof\_flag = 2;  
 else  
 eof\_flag = 1;  
 }else  
 fread( RxBuffer, 1, BUFFER\_SIZE, transfer\_file);  
 IPXSendPacket(&ECB);  
 delay(30);  
 }  
 }  
  
*// Закрываем сокет* IPXCloseSocket(&Socket);  
 fclose(transfer\_file);  
 exit(0);  
}

**Листинг файла client.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <conio.h>  
#include <mem.h>  
#include <string.h>  
#include "ipx.h"  
  
  
  
#define BUFFER\_SIZE 512  
  
void main(void) {  
  
  
 int i;  
 char eof\_flag = 0;  
 FILE \*transfer\_file;  
  
 static unsigned Socket = 0x4567;  
  
*// ECB для приема и передачи пакетов* struct ECB RxECB, TxECB;  
  
*// Заголовки принимаемых и передаваемых пакетов* struct IPX\_HEADER RxHeader, TxHeader;  
  
*// Буфера для принимаемых и передаваемых данных* unsigned char RxBuffer[BUFFER\_SIZE];  
 unsigned char TxBuffer[BUFFER\_SIZE];  
  
  
  
 if(ipx\_init() != 0xff) {  
 printf("IPX driver not load!\n");  
 exit(-1);  
 }  
  
*// Открываем сокет, на котором будем принимать и передавать пакеты* if(IPXOpenSocket(SHORT\_LIVED, &Socket)) {  
 printf("Socket don't open\n");  
 exit(-1);  
 };  
  
*// Подготавливаем ECB для передачи пакета* memset(&TxECB, 0, sizeof(TxECB));  
  
 TxECB.Socket = IntSwap(Socket);  
 TxECB.FragmentCnt = 3;  
 TxECB.Packet[0].Address = &TxHeader;  
 TxECB.Packet[0].Size = sizeof(TxHeader);  
 TxECB.Packet[1].Address = TxBuffer;  
 TxECB.Packet[1].Size = BUFFER\_SIZE;  
 TxECB.Packet[2].Address = &eof\_flag;  
 TxECB.Packet[2].Size = 1;  
*// Пакет предназначен всем станциям данной сети* memset(TxECB.ImmAddress, 0xff, 6);  
  
*// Подготавливаем заголовок пакета* TxHeader.PacketType = 4;  
 memset(TxHeader.DestNetwork, 0, 4);  
 memset(TxHeader.DestNode, 0xff, 6);  
 TxHeader.DestSocket = IntSwap(Socket);  
  
 printf("Input download path: ");  
 scanf("%s", TxBuffer);  
 transfer\_file = fopen(TxBuffer, "wb");  
  
 printf("Input file path to server: ");  
 scanf("%s", TxBuffer);  
  
  
 IPXSendPacket(&TxECB);  
  
*// Подготавливаем ECB для приема пакета от сервера* memset(&RxECB, 0, sizeof(RxECB));  
 RxECB.Socket = IntSwap(Socket);  
 RxECB.FragmentCnt = 2;  
 RxECB.Packet[0].Address = &RxHeader;  
 RxECB.Packet[0].Size = sizeof(RxHeader);  
 RxECB.Packet[1].Address = RxBuffer;  
 RxECB.Packet[1].Size = BUFFER\_SIZE;  
 RxECB.Packet[2].Address = &eof\_flag;  
 RxECB.Packet[2].Size = 1;  
  
 for(;!eof\_flag;){  
 IPXListenForPacket(&RxECB);  
  
  
 while(RxECB.InUse) {  
 IPXRelinquishControl();  
 if(kbhit()) {  
 getch();  
 RxECB.CCode = 0xfe;  
 break;  
 }  
 }  
  
  
 if(eof\_flag == 1){  
 printf("END TRANSFER\n"); break;  
 }  
 if(eof\_flag == 2){  
 printf("FILE NOT EXIST"); break;  
 }  
 if(RxECB.CCode == 0){  
 fwrite(RxBuffer, BUFFER\_SIZE, 1, transfer\_file);  
 }  
 }  
  
  
  
*// Закрываем сокет* IPXCloseSocket(&Socket);  
 fclose(transfer\_file);  
 exit(0);  
}

**Листинг файла ipxdrv.asm**

.286

.MODEL SMALL

IPXSPX\_REGS struc

rax dw ?

rbx dw ?

rcx dw ?

rdx dw ?

rsi dw ?

rdi dw ?

res dw ?

IPXSPX\_REGS ends

.DATA

\_ipxspx\_drv\_entry dd ?

.CODE

PUBLIC \_ipxspx\_entry, \_ipx\_init

PUBLIC \_ipxspx\_drv\_entry

\_ipxspx\_entry PROC FAR

push bp

mov bp,sp

push es

push di

push si

push dx

push cx

push bx

push ax

push ds

mov bx, [bp+6]

mov ds, [bp+8]

mov es, ds:[bx].res

mov di, ds:[bx].rdi

mov si, ds:[bx].rsi

mov dx, ds:[bx].rdx

mov cx, ds:[bx].rcx

mov ax, ds:[bx].rax

mov bx, ds:[bx].rbx

pop ds

call [dword ptr \_ipxspx\_drv\_entry]

push ds

push dx

mov dx, bx

mov bx, [bp+6]

mov ds, [bp+8]

mov ds:[bx].rax, ax

mov ds:[bx].rcx, cx

mov ds:[bx].rbx, dx

pop dx

mov ds:[bx].rdx, dx

pop ds

pop ax

pop bx

pop cx

pop dx

pop si

pop di

pop es

pop bp

retf

\_ipxspx\_entry ENDP

\_ipx\_init PROC NEAR

push bp

mov bp,sp

mov ax, 7a00h

int 2fh

cmp al, 0ffh

jne \_ipx\_init\_exit

mov word ptr \_ipxspx\_drv\_entry+2, es

mov word ptr \_ipxspx\_drv\_entry, di

\_ipx\_init\_exit:

mov ah, 0

pop bp

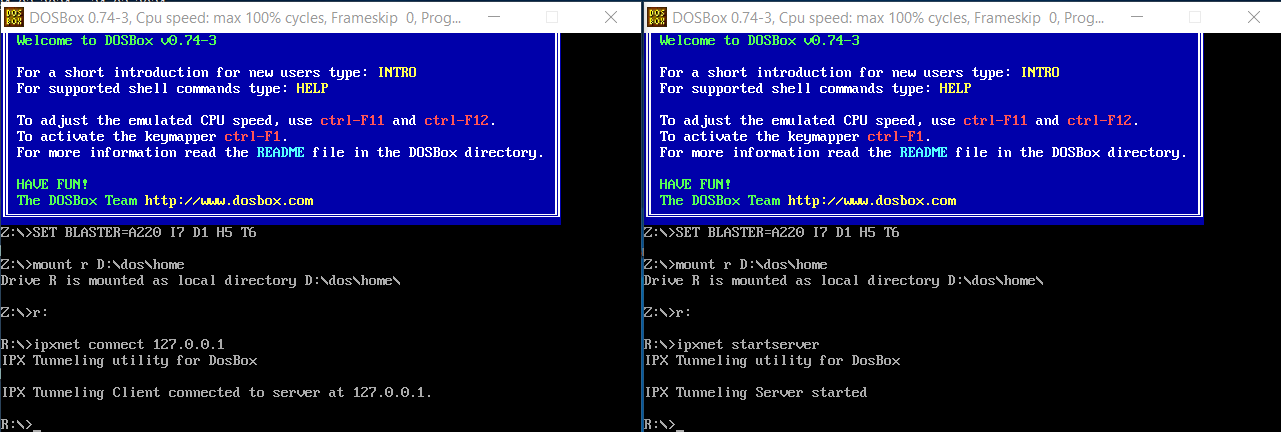
ret

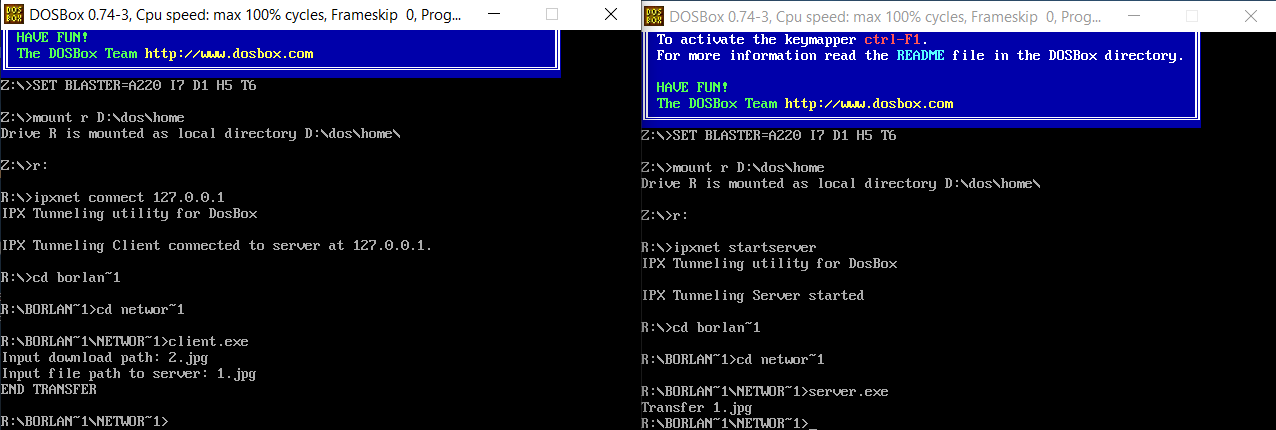
\_ipx\_init ENDP

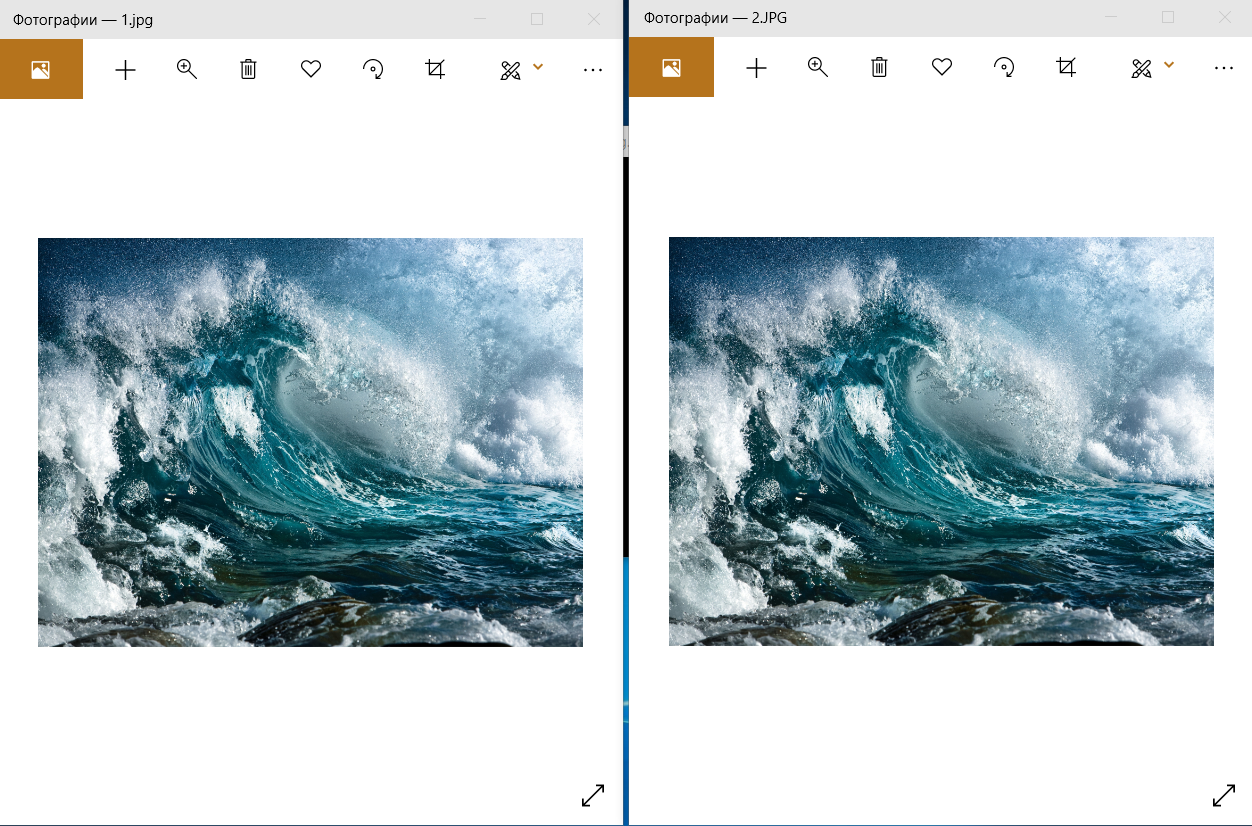
end

**Результат выполнения программ**

Для работы программ нам понадобится запустить два образа DOSBox, один из них будет сервером, а другой клиентом.

****

****

****

**Анализ функционирования разработанных программ**

Анализ широковещательной рассылки всем пользователям

Размер файла: 2927191 байт

Количество пакетов: 5895

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер файла | Число пакетов | Число получателей | Число утерянных пакетов | Процент потерь | Время |
| 2.79 мб | 5895 | 1 | 19 | 0.31% | 3.124 |
| 11790 | 2 | 27 | 0.30% | 4.032 |
| 23580 | 4 | 73 | 0.32% | 4.387 |
| 35370 | 6 | 86 | 0.36% | 4.742 |

**По мере увеличения кол-ва клиентов кол-во потерянных пакетов растет**

**Вывод:** в ходе работы были изучены и применены на практике основные функции API драйвера для работы с протоколом сетевого уровня IPX. Получены навыки использования IPX. Проведены замеры, показывающие, что при использовании IPX возможны потери пакетов. Чем больше клиентов – тем больше потерь пакетов, несмотря на погрешность, при которой иногда были результаты, не соответствующие этому. И чем больше клиентов, тем дольше передача.

**Контрольные вопросы:**

1. **Система адресации в протоколе IPX.**

Система адресов, используемая в протоколе IPX, представлена несколькими компонентами: это номер сети, адрес станции в сети и идентификатор программы на рабочей станции.

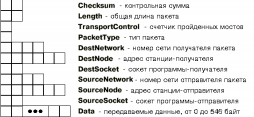
**Номер сети** - это номер сегмента сети, определяемого системным администратором. Если в общей сети есть мосты, каждая отдельная сеть, подключенная через мост, должна иметь свой, уникальный номер сети.

**Адрес станции** - это число, которое является уникальным для каждой рабочей станции. При использовании адаптеров Ethernet уникальность обеспечивается изготовителем сетевого адаптера.  Специальный адрес FFFFFFFFFFFFh используется для рассылки

данных всем станциям данной сети одновременно.

**Идентификатор программы на рабочей станции (сокет)** - число, которое используется для адресации определенной программы, работающей на компьютере. В среде мультизадачных операционных систем, на каждой рабочей станции в сети одновременно может быть запущено несколько приложений. Для того, чтобы послать данные конкретной программе, используется идентификация программ при помощи сокетов. Каждая программа, желающая принимать или передавать данные по сети, должна получить свой, уникальный для данной рабочей станции, идентификатор - сокет.

1. **Формат заголовка пакета, передаваемого через протокол IPX.**

****

Все поля, кроме поля Data, представляют собой заголовок пакета.

1. **Что представляет собой блок управления событием ECB? С какой целью он используется?**

Для приема или передачи пакета прикладная программа должна подготовить пакет данных, сформировав его заголовок, и построить так называемый блок управления событием ECB (Event Control Block). В блоке ECB задается адресная информация для передачи пакета, адрес передаваемого пакета в оперативной памяти и некоторая другая информация.  Подготовив блок ECB, прикладная программа передает его адрес соответствующей функции IPX для выполнения операции приема или передачи пакета.

1. **Опишите 2 способа, с помощью которых в программе можно узнать о завершении операций приёма или передачи.**

Индикатором завершения операции приема или передачи пакета может служить поле INUSE. Перед тем как вызвать функцию IPX, программа записывает в поле INUSE нулевое значение. Пока операция передачи данных, связанная с данным ECB, не завершилась, поле INUSEсодержит определенные коды. Второй идентификатор - ECB. С ним можно указать callback функцию, которая вызывается при завершении приема передачи.

**5. Какие функции API и в каком порядке нужно вызвать для начала операции отправки пакетов?**

1.IPXOpenSocket

2. IPXSendPacket

3. IPXCloseSocket

**6. Недостатки протокола IPX.**

1.В данный момент ipx считается устаревшим.

2. Протокол ipx – является протоколом сетевого уровня, что значит, что он не гарантирует доставку пакет.

3. По данным меркам максимальный размер пакета маловат.

**7. Сколько байт данных максимально можно передать в одном пакете?**

Максимальный размер пакета 546 байт.

**8. Каким образом можно выполнить рассылку пакета всем рабочим станциям сети?**

В качестве адреса рабочей станции необходимо указать FFFFFFFFFFFFh это так называемый широковещательный адрес, при указании его пакет придет всем устройствам в сети. Такие рассылки полезны, когда программа знает только сокет другого устройства