СОДЕРЖАНИЕ

1. Введение

2. Сетевые и межсетевые протоколы

3. Протоколы TCP/IP

4. Обмен сообщениями в сети

5. Работа с файлами в ЛВС

6. Вирусы в сети

7. Список литературы

1. Введение.

Накопленный опыт эксплуатации больших вычислительных сетей, та-

ких как ARPANET и TELENET, показывает, что около 80% всей генери-

руемой в таких сетях информации используется только тем же офисом,

который ее порождает, т.е. значительная часть сетевой информации

предназначается лишь местным потребителям. Поэтому в последние

10-15 лет выделился специальный класс вычислительных сетей - ло-

кальные вычислительные сети (ЛВС), оптимально сочетающие в себе

простоту и надежность, высокую скорость передачи и большой набор

реализуемых функций.

Международный комитет IEEE 802 (институт инженеров по электро-

нике и электротехнике -IEEE, США), специализирующийся на стандар-

тизации в области ЛВС, дает следующее определение этим сетям: "...

Локальные вычислительные сети отличаются от других типов сетей

тем, что они обычно ограничены умеренной географической областью,

такой, как группа рядом стоящих зданий: склад, студенческий горо-

док, и в зависимости от каналов связи осуществляют передачу данных

в диапазонах скоростей от умеренных до высоких с низкой степенью

ошибок... Значения параметров области, общая протяженность, коли-

чество узлов, скорость передачи и топология ЛВС могут быть самыми

различными, однако комитет IEEE 802 основывает ЛВС на кабелях

вплоть до нескольких километров длины, поддержки нескольких сотен

станций разнообразной топологии при скоростях порядка 1-20 и более

Мбит/сек" .

Таким образом, отличительными признаками ЛВС можно считать: ох-

ват умеренной площади, высокую скорость передачи и низкую вероят-

ность возникновения ошибок в коммуникационном оборудовании.

2. СЕТЕВЫЕ И МЕЖСЕТЕВЫЕ ПРОТОКОЛЫ

Базовая Модель Открытых Систем (МОС) определяет семь уровней

(слоев) коммуникационной системы. Каждый слой определяет соот-

ветствующее подмножество функций, необходимых для взаимодействия.

Каждый уровень взаимодействует только со смежными уровнями.

Физический уровень (слой 1) предназначен для собственно переда-

чи данных по сети. Этот уровень определяет набор передающих сред,

используемых для соединения различных сетевых компонент (например,

оптическое волокно, витой телефонный кабель, коаксиальный кабель и

устройство цифрового мультиплексирования).

Уровень управления каналом (слой 2) предназначен для передачи

данных в каждый тип передающей среды. В локальных сетях на этом

уровне решается проблема коллективного использования передающейц

среды и обнаружения и исправления ошибок.

Сетевой уровень (слой 3), часто называемый уровнем коммуникаци-

онной подсети, предназначен для переадресации пакетов. На этом

уровне осуществляется прокладка маршрутов пакетов в сети. .

Транспортный уровень (слой 4) обеспечивает надежный транспорт

данных между абонентами сети, включая средства управления потоком

и выявления и исправления ошибок.

Сеансовый уровень (слой 5) предназначен для управления коммуни-

кационными связями между двумя точками уровня представления. Уста-

новление, поддержка и окончание сеанса (сессии) обеспечиваются

этим уровнем. Кроме того, здесь же обеспечиваются соглашения об

именах.

Уровень представления (слой 6) предназначен для преобразования

данных в процессе их прохождения по сети. Кодировка, шифрование,

преобразование ASCII/EBCDIC, а также интерпретация управляющих

символов - примеры задач этого уровня.

Прикладной уровень (слой 7) представляет собой полный прог-

раммный интерфейс к прикладным процессам. Этот слой обеспечивает

полный набор служб для управления связанными распределенными про-

цессами, включая доступ к файлам, управление базами данных и уп-

равление сетью.

Существует два различных способа организации обмена данными в

сети - без установления логического соединения и с установлением

соединения.

Метод связи без логического соединения один из самых старых и

простейших в коммуникационной технологии. В таких системах каждый

пакет рассматривают как индивидуальный объект: каждый пакет содер-

жит адрес доставки и освобождает систему от предварительного обме-

на служебной информацией между передающим и принимающим узлами.

Примерами таких протоколов являются:

1.Прикладной дейтаграммный протокол Министерства

обороны США.

Ihe Dept. of Defense's User Datagram Protocol (UDP).

2.Протокол обмена пакетов сети Интернет фирмы Ксерокс.

Xerox's Internet Pasket Exchange Protocol (IPX).

3.Дейтаграммный протокол фирмы Apple.

Apple's Datagram Delivery Protocol (DDP).

Пример взаимодействия между абонентами без

установления соединения

Пакет данных 1

------------------------------>

Клиент "A" Пакет данных 2 Клиент "B"

------------------------------>

Пакет данных 3

------------------------------>

Пример взаимодействия между клиентом "A" и клиентом "B" без ус-

тановления логического канала. Поскольку никакого предварительного

обмена служебной информацией не производится, при этом методе пе-

редаются только данные.

Связь без логического соединения характеризуется следующим:

а. Переполнения соединений в рабочих станциях, межсетевых

рутерах, мостах и серверах полностью исключены.

в. Имеется возможность одновременно посылать пакеты множеству

адресатов.

с. Синхронизация приемника и передатчика не является необходи-

мой. Используя систему очередей, сетевые компоненты буферизуют

запросы на передачу пакетов, увеличивая гибкость передачи.

Метод, ориентированный на логическое соединение, является более

поздней технологией. При этом методе устанавливается логический

канал между двумя клиентами перед собственно передачей данных. Па-

кеты запроса на соединение посылаются удаленному клиенту для уста-

новки логического канала. Если удаленный клиент "согласен", то

возвращается пакет подтверждения установления логического канала и

только после этого начинается обмен данными с управлением потоком,

сегментацией и исправлением ошибок. Когда обмен данными завершает-

ся, посылается пакет подтверждения клиенту - инициатору.

Примерами протоколов, ориентированных на соединение,(с установ-

кой логического канала являются):

1.Протокол управления передачей Министерства обороны США. Ihe

Dept. of Defenee's Transmission Control Protocol (TCP)

2.Протокол последовательных пакетов фирм Ксерокс Xerox's

Sequenced Packet Protocol (SPP). Примерами протоколов, ориентиро-

ванных на соединение,(с установкой логического канала являются):

1.Протокол управления передачей Министерства обороны США. Ihe

Dept. of Defenee's Transmission Control Protocol (TCP) 2.Протокол

последовательных пакетов фирм Ксерокс Xerox's Sequenced Packet

Protocol (SPP). 3.Протокол транзакции фирмы Эппл. Apple's

Appletalk Transaction Protocol (ATP)

Пример взаимодействия с предварительным

установлением логического канала

Пакет запроса "вызов"

----------------------------------->

Пакет подтверждения вызова

<----------------------------------

Пакет данных 1

----------------------------------->

Клиент "A" Пакет данных 2 Клиент "B"

----------------------------------->

Пакет данных 3

----------------------------------->

Пакет подтверждения данных

<----------------------------------

Пример взаимодействия между клиентом "A" и клиентом "B" с пред-

варительной установкой логического канала. Пакет запроса "вызов",

пакет подтверждения вызова и пакет подтверждения данных являются

служебной информацией.

Связь с установлением соединения характеризуется большими нак-

ладными расходами, но обеспечивает значительно более высокий уро-

вень сервиса по сравнению с дейтаграммной связью.

NetWare базируется на эффективном методе без установления логи-

ческого канала. Служба логических каналов базируется на системе,

использующей метод передачи без установления соединения, и пред-

назначается для приложений, нуждающихся в таком сервисе.

2.1. Переадресация в интерсети

Схема адресации NetWare в интерсети изначально была определена

группой по сетевым системам фирмы Xerox (Xerox Network Systems

XNS) как межсетевой дейтаграммный и межсетевой транспортный прото-

колы (Internet Datagram and Internet Transport Protocols).

Пакет межсетевого протокола разделен на две основные части: за-

головок (блок адресации) и блок данных. Блок адресации подразделя-

ется в свою очередь на три части: часть управления, адресная часть

получателя (доставки) и адресная часть источника.

Оба адреса - адрес доставки и адрес отправителя состоят из трех

полей: номера хост-системы, номера сети и номера порта (сокета).

Номер хост-системы имеет размер 48 битов. Он обеспечивает уни-

кальную идентификацию любых хост-систем различных производителей

безотносительно к составу конкретной интерсети.

Номер сети имеет разрядность 32 бита, что обеспечивает уни-

кальным идентификатором любую сеть в интерсети.

Номер порта (сокета) имеет разрядность 16 битов для идентифика-

ции запроса приема/передачи между процессами.

2.2. Маршрутизация в интерсети.

Алгоритм маршрутизации в NetWare является распределенным таким

образом, что процессы принятия решения могут приниматься во мно-

жестве узлов интерсети - в любом из маршрутизаторов или сетевых

серверов NetWare. Алгоритм маршрутизации NetWare является адаптив-

ным. Он позволяет реагировать на изменяющиеся условия внутри ин-

терсети почти мгновенно.

При отказе сетевого сервера или моста , адаптивный маршрутиза-

тор ищет альтернативный маршрут. Если находятся несколько маршру-

тов, маршрутизатор использует наиболее оптимальный.

2.3. Алгоритм маршрутизации.

Сетевая операционная система NetWare использует распределенный

адаптивный алгоритм передачи пакетов в интерсети. NetWare быстро и

эффективно реагирует на динамические изменения в топологии интер-

сети, предоставляя новые эффективные маршруты практически немед-

ленно. Управление потоком обеспечивается Протоколом Обмена После-

довательными Пакетами (Sequenced Packet Exange Protocol), рассмат-

риваемым далее.

Алгоритм маршрутизации NetWare включает в себя: 1) процесс из-

мерения и идентификации для отслеживания за топологией интерсети,

и 2) протокол рассеивания информации о сетевых характеристиках со-

ответствующего узла. Маршрутизатор выполняет измерения посредством

записи числа точек сети интернет, через которые должен пройти па-

кет от маршрутизатора (измерителя) по всем остальным маршрутизато-

рам интерсети с отметками интервала времени прохождения между ни-

ми. Таким образом отмечается количество серверов и маршрутизато-

ров, которые пакет возвращает измерителю. Серверы и маршрутизаторы

следят друг за другом посредством периодической посылки информации

о своем состоянии.

Во время инициализации каждый маршрутизатор запрашивает инфор-

мацию о маршрутах от всех других маршрутизаторов интерсети. Табли-

цы маршрутизаторов строятся во время инициализации и обновляются

по мере приема информации.

После инициализации маршрутизатор посылает широковещательно ин-

формацию о всех сетях и серверах, о которых он имеет сведения. Се-

тевая информация включает в себя номер сети, удаленность и пример-

ное время передачи 576-байтового пакета от данного маршрутизаторра

до целевой сети.

В дополнение к данной информации, маршрутизатор передает обнов-

ляющую информацию, как только изменится информация в таблице. Эти

изменения включают в себя все удаления и включения других маршру-

тизаторов и серверов в таблицу, а также изменения в расстоянии

между серверами и маршрутизаторами. Как только изменение в таблице

обнаружено, сервер/маршрутизатор немедленно отсылает информацию об

изменении во все сети, к которым он подсоединен.

Когда теряется информация о сервере или сети, маршрутизатор пы-

тается выровнять ситуацию при помощи нового маршрута, который он

вычисляет на основе своих сведений. Как только такой маршрут нахо-

дится, маршрутизатор немедленно сообщает остальным маршрутизаторам

о новом маршруте. Если альтернативный маршрут не найден, маршрути-

затор также оповещает об этом.

2.4. Протокол обмена пакетов интерсети NetWare (IPX)

IPX обеспечивает сетевой уровень Advanced NetWare дейтаграммным

интерфейсом. IPX является реализацией Xerox's Interuetwork

Datagram PacKet Protocol (IDP). Назначение IPX - дать прикладным

программам рабочей станции NetWare доступ к сетевым драйверам и

взаимодействовать напрямую с другими рабочими станциями, серверами

или устройствами интерсети.

IPX позволяет прикладной программе присылать и принимать от-

дельные пакеты интерсети. Пакеты интерсети структурированы в соот-

ветствии с определением Xerox Network Systems (XNS). В среде ин-

терсети NetWare каждый узел имеет уникальный межсетевой адрес. Ис-

пользуя IPX, рабочая станция NetWare может посылать и принимать

пакет от любой станции интерсети. Маршрутизация пакетов между уз-

лами, физически находящимися в сетях различной архитектуры, явля-

ется автоматической и прозрачной. Эта прозрачность обеспечивается

средствами маршрутизации в серверах и маршрутизаторах NetWare.

IPX-пакеты структурированы в точности как пакеты Xerox's XNS

Internet Datagram Protocol (IDP). Пакеты разбиваются на две логи-

ческие части: на заголовок и блок данных. Заголовок в свою очередь

разделен на блок управления, на блок адреса получателя и на блок

адреса отправителя.

Каждый пакет содержит длину полного пакета интерсети, который

является суммой длины блока заголовка и длока данных. Минимальной

длиной пакета принято считать 30 байтов.

Каждый пакет имеет индикатор типа сервиса, предоставляемого или

запрашиваемого данным пакетом. Xerox определяет следующие величи-

ны:

0 : неопределенный тип пакета;

1 : пакет информации о маршруте;

2 : эхо-пакет;

3 : пакет объявления об ошибке;

4 : пакет обмена пакетами;

5 : пакет протокола последовательных пакетов;

16-31 : экспериментальные протоколы.

Формат пакета обмена Интерсети (IPX)

-¬

0 1 15 ¦

г====================¬¦

Управление ¦---------T----------¦¦

¦---------+----------¦¦

¦- сеть доставки -¦¦

¦--------------------¦¦

Сетевой адрес ¦- -¦¦

доставки ¦- хост доставки -¦¦

¦--------------------¦¦Заголовок

¦ порт доставки ¦+---------

¦--------------------¦¦

¦ сеть отправителя ¦¦

¦--------------------¦¦

Сетевой адрес ¦ -¦¦

отправителя ¦- хост отправителя -¦¦

¦- -¦¦

¦--------------------¦¦

¦ порт отправителя ¦¦

L====================-¦

--

г====================¬¬

¦ ¦¦

¦ 0 -546 байтов ¦¦

¦ ¦¦Данные

¦ прозрачных данных ¦+------

¦ ¦¦

¦ ---------¦¦

¦ ¦доп.байт¦¦

L===========¦========-¦

--

Рис. 2.3.

2.5. Программный интерфейс IPX

IPX использует блок управления событиями для координации и акти-

визации определенных операций. Могут возникать два типа событий:

события, связанные с приемопередачей и события специального назна-

чения, определяемые прикладной программой. Услуги IPX по приему и

передаче включают в себя следующее:

a. открыть порт;

b. закрыть порт;

c. получить локальную цель;

d. послать пакет;

e. получить межсетевой адрес;

f. сбросить управление;

g. отсоединиться от цели.

События специального назначения управляются посредством

Asynchronons Event Sheduler (AES), встроенного в IPX. AES является

дополнительным сервисом, обеспечивающим также средства измерения

затраченного времени и переключения событий в соответствии с отме-

ренными интервалами времени. Программный интерфейс AES включает в

себя следующие услуги:

a. планирование события IPX;

b. подавление события;

c. планирование специального события;

d. получение маркера интервала.

Протокол IPX предназначен для использования в качестве фунда-

мента для построения сложных прикладных систем, включая серверы

связи, шлюзы или системы прямого взаимодействия.

Тест наличия IPX в памяти использует мультиплексное прерывание

2F. Ниже приводится текст функции, возвращающей 1, если IPX уста-

новлен.

void far (\*ipx\_spx)(void)

int ipx\_installed(void)

{

union REGS regs;

struct SREGS sregs;

regs.x.ax =0x7a00;

int86x(0x2f,&regs,&regs,&sregs);

if (regs.h.al != 0xff)

return -1;

ipx\_spx = MK\_FP(sregs.es,regs.x.di);

return 1;

}

/\*

\*

\*

\*/

struct IPXHEADER

{

unsigned int checksum;

unsigned int length;

unsigned char transport\_control;

unsigned char packet\_type;

unsigned char dest\_network\_number [4];

unsigned char dest\_network\_node [6];

unsigned int dest\_network\_socket;

unsigned char source\_network\_number [4];

unsigned char source\_network\_node [6];

unsigned int source\_network\_socket;

};

struct ECB

{

void far \*link\_address;

void far (\*event\_service\_routine)(void);

unsigned char in\_use;

unsigned char completion\_code;

unsigned int socket\_number;

unsigned int connection\_id; /\* returned by Listen \*/

unsigned int rest\_of\_workspace;

unsigned char driver\_workspace [12];

unsigned char immediate\_address [ 6];

unsigned int packet\_count;

struct {

void far \*address;

unsigned int length;

} packet [2];

};

int ipx\_spx\_installed(void);

int ipx\_cancel\_event(struct ECB \*ecb\_ptr);

void close\_socket(unsigned int socket);

int open\_socket(unsigned int socket);

int get\_local\_target(unsigned char \*dest\_network,

unsigned char \*dest\_node,

unsigned int dest\_socket,

unsigned char \*bridge\_address);

void let\_ipx\_breath(void);

void ipx\_listen\_for\_packet(struct ECB \*ecb\_ptr);

void ipx\_send\_packet(struct ECB \*ecb\_ptr);

int get\_internet\_address(unsigned char connection\_number,

unsigned char \*network\_number,

unsigned char \*physical\_node);

unsigned int get\_1st\_connection\_num (char \*who);

unsigned char get\_connection\_number(void);

void get\_user\_id(unsigned char connection\_number,

unsigned char \*user\_id);

/\*

\* IPX.C -- helper routines for accessing IPX services

\* from Turbo C.

\*/

#include <stdlib.h>

#include <dos.h>

#include <mem.h>

#include <string.h>

#include <ipx.h>

void far (\*ipx\_spx)(void);

int ipx\_spx\_installed(void)

{

union REGS regs;

struct SREGS sregs;

regs.x.ax = 0x7a00;

int86x(0x2f, &regs, &regs, &sregs);

if (regs.h.al != 0xff) return -1;

ipx\_spx = MK\_FP(sregs.es, regs.x.di);

\_BX = 0x0010;

\_AL = 0x00;

ipx\_spx();

if (\_AL == 0x00) return 0;

return 1;

}

int ipx\_cancel\_event(struct ECB \*ecb\_ptr)

{

\_ES = FP\_SEG( (void far \*) ecb\_ptr);

\_SI = FP\_OFF( (void far \*) ecb\_ptr);

\_BX = 0x0006;

ipx\_spx();

\_AH = 0;

return \_AX;

}

void close\_socket(unsigned int socket)

{

if (ipx\_spx\_installed() < 1) return;

\_BX = 0x0001;

\_DX = socket;

ipx\_spx();

}

int open\_socket(unsigned int socket)

{

if (ipx\_spx\_installed() < 1) return -1;

\_DX = socket;

\_BX = 0x0000;

\_AL = 0xFF;

ipx\_spx();

\_AH = 0;

return \_AX;

}

int get\_local\_target(unsigned char \*dest\_network,

unsigned char \*dest\_node,

unsigned int dest\_socket,

unsigned char \*bridge\_address)

{

unsigned int temp\_ax;

struct {

unsigned char network\_number [4];

unsigned char physical\_node [6];

unsigned int socket;

} request\_buffer;

struct {

unsigned char local\_target [6];

} reply\_buffer;

memcpy(request\_buffer.network\_number, dest\_network, 4);

memcpy(request\_buffer.physical\_node, dest\_node, 6);

request\_buffer.socket = dest\_socket;

\_ES = FP\_SEG( (void far \*) &request\_buffer);

\_SI = FP\_OFF( (void far \*) &request\_buffer);

\_DI = FP\_OFF( (void far \*) &reply\_buffer);

\_BX = 0x0002;

ipx\_spx();

\_AH = 0;

temp\_ax = \_AX;

memcpy(bridge\_address, reply\_buffer.local\_target, 6);

return temp\_ax;

}

void let\_ipx\_breath(void)

{

\_BX = 0x000A;

ipx\_spx();

}

void ipx\_listen\_for\_packet(struct ECB \*ecb\_ptr)

{

\_ES = FP\_SEG( (void far \*) ecb\_ptr);

\_SI = FP\_OFF( (void far \*) ecb\_ptr);

\_BX = 0x0004;

ipx\_spx();

}

void ipx\_send\_packet(struct ECB \*ecb\_ptr)

{

\_ES = FP\_SEG( (void far \*) ecb\_ptr);

\_SI = FP\_OFF( (void far \*) ecb\_ptr);

\_BX = 0x0003;

ipx\_spx();

}

int get\_internet\_address(unsigned char connection\_number,

unsigned char \*network\_number,

unsigned char \*physical\_node)

{

union REGS regs;

struct SREGS sregs;

struct {

unsigned int len;

unsigned char buffer\_type;

unsigned char connection\_number;

} request\_buffer;

struct {

unsigned int len;

unsigned char network\_number [4];

unsigned char physical\_node [6];

unsigned int server\_socket;

} reply\_buffer;

regs.h.ah = 0xe3;

request\_buffer.len = 2;

request\_buffer.buffer\_type = 0x13;

request\_buffer.connection\_number = connection\_number;

reply\_buffer.len = 12;

regs.x.si = FP\_OFF( (void far \*) &request\_buffer);

sregs.ds = FP\_SEG( (void far \*) &request\_buffer);

regs.x.di = FP\_OFF( (void far \*) &reply\_buffer);

sregs.es = FP\_SEG( (void far \*) &reply\_buffer);

int86x(0x21, &regs, &regs, &sregs);

memcpy(network\_number, reply\_buffer.network\_number, 4);

memcpy(physical\_node, reply\_buffer.physical\_node, 6);

regs.h.ah = 0;

return regs.x.ax;

}

unsigned int get\_1st\_connection\_num (char \*who)

{

union REGS regs;

struct SREGS sregs;

struct {

unsigned int len;

unsigned char buffer\_type;

unsigned int object\_type;

unsigned char name\_len;

unsigned char name [47];

} request\_buffer;

struct {

unsigned int len;

unsigned char number\_connections;

unsigned char connection\_num [100];

} reply\_buffer;

regs.h.ah = 0xe3;

request\_buffer.len = 51;

request\_buffer.buffer\_type = 0x15;

request\_buffer.object\_type = 0x0100;

request\_buffer.name\_len = (unsigned char) strlen(who);

strcpy(request\_buffer.name, who);

reply\_buffer.len = 101;

regs.x.si = FP\_OFF( (void far \*) &request\_buffer);

sregs.ds = FP\_SEG( (void far \*) &request\_buffer);

regs.x.di = FP\_OFF( (void far \*) &reply\_buffer);

sregs.es = FP\_SEG( (void far \*) &reply\_buffer);

int86x(0x21, &regs, &regs, &sregs);

if (regs.h.al != 0) return 0;

if (reply\_buffer.number\_connections == 0) return 0;

regs.h.ah = 0;

regs.h.al = reply\_buffer.connection\_num[0];

return regs.x.ax;

}

unsigned char get\_connection\_number(void)

{

\_AH = 0xDC;

geninterrupt(0x21);

return \_AL;

}

void get\_user\_id(unsigned char connection\_number,

unsigned char \*user\_id)

{

union REGS regs;

struct SREGS sregs;

struct {

unsigned int len;

unsigned char buffer\_type;

unsigned char connection\_number;

} request\_buffer;

struct {

unsigned int len;

unsigned char object\_id[4];

unsigned char object\_type[2];

char object\_name[48];

char login\_time[7];

} reply\_buffer;

regs.h.ah = 0xe3;

request\_buffer.len = 2;

request\_buffer.buffer\_type = 0x16;

request\_buffer.connection\_number = connection\_number;

reply\_buffer.len = 61;

regs.x.si = FP\_OFF( (void far \*) &request\_buffer);

sregs.ds = FP\_SEG( (void far \*) &request\_buffer);

regs.x.di = FP\_OFF( (void far \*) &reply\_buffer);

sregs.es = FP\_SEG( (void far \*) &reply\_buffer);

int86x(0x21, &regs, &regs, &sregs);

strncpy(user\_id, reply\_buffer.object\_name, 48);

}

2.6. Протокол последовательного обмена пакетами NetWare (SPX)

Протокол обмена последовательными пакетами (SPX) строится на

основе IPX и предлагает дополнительные услуги Xerox's Sequenced

Packet Protocol (SPP). SPX дает возможность прикладным программам

рабочей станции NetWare получать некоторые преимущества при ис-

пользовании сетевых драйверов при прямых коммуникациях с другими

рабочими станциями, серверами и устройствами интерсети с дополни-

тельной гарантией достоверности и последовательности пакетов.

Внутренне SPX построен на дейтаграммных примитивах IPX и дает

простой интерфейс, ориентированный на установление соединения.

В дополнение к структуре IPX, SPX включает 12 байтов блока уп-

равления соединения.

2.7. Программный интерфейс SPX

В дополнение к программному интерфейсу IPX, SPX предоставляет

следующие функции:

a. проверка установки SPX;

b. установка соединения;

c. прослеживание соединения;

d. окончание соединения;

e. разрыв соединения;

f. получение состояния соединения;

g. посылка последовательного пакета;

h. прослушивание (ожидание) последовательного пакета.

Эти функции управляют установкой, поддержанием, cбросом соеди-

нения. Прикладные системы, использующие SPX не обязаны организовы-

вать свои собственные схемы тайм-аутов для гарантирования воста-

новления по обрыву установленного соединения, посылки последова-

тельного пакета или по запросу об окончании соединения.

Формат пакета протокола SPX

0 1 15 ¦

г====================¬¦

¦ контрольная сумма ¦¦

¦--------------------¦¦

¦ длина ¦¦

¦---------T----------¦¦

¦упр.транс¦тип пакета¦¦

¦---------+----------¦¦

¦- сеть доставки -¦¦

¦--------------------¦¦

Сетевой адрес ¦- -¦¦

доставки ¦- хост доставки -¦¦

¦--------------------¦¦Уровень 1

¦ порт доставки ¦+---------

¦--------------------¦¦адресация

¦- сеть отправителя-¦¦

¦--------------------¦¦

Сетевой адрес ¦ -¦¦

отправителя ¦- хост отправителя -¦¦

¦- -¦¦

¦--------------------¦¦

¦ порт отправителя ¦¦

L====================-¦

-+

г==========T=========¬¦

¦упр.соедин¦тип поток¦¦

¦----------+---------¦¦

¦идент.источ.соединен¦¦

¦--------------------¦¦

¦идент. приемника ¦¦Уровень 2

¦--------------------¦+---------

¦ номер пакета ¦¦протокол

¦--------------------¦¦последова-

¦номер подтверждения ¦¦тельных

¦--------------------¦¦пакетов

¦ номер размещения ¦¦

L====================-¦

--

г====================¬¦

¦ 0 - 546 байтов ¦¦Уровень 3

¦ ¦+---------

¦ прозрачных данных ¦¦управление

¦ -----------¦¦

¦ ¦доп.байт ¦¦

L=========¦==========-¦

--

гTTT======T==========¬

¦¦¦¦резерв¦тип потока¦

L+++======¦==========-

¦¦¦¦конец сообщения

¦¦¦внимание

¦¦посылка подтверждений

¦системный пакет

Рис.2.4

2.8. Управление потоком SPX

Управление потоком является средством, позволяющим уменьшить

количество повторных передач пакетов. В таких случаях, когда про-

исходит тайм-аут без успешного подтверждения, SPX выполняет опти-

мальное количество повторов, прежде чем декларирует обрыв соедине-

ния. SPX автоматически выбирает оптимальный тайм-аут и значение

количества повторов, которые наиболее подходят к физическим харак-

тетистикам соответствующих сетевых устройств. SPX использует эв-

ристический алгоритм тайминга для вычисления оптимального коли-

чества повторов, адаптируясь под время задержки пакетов.

Протокол SPX предназначен для использования в качестве фунда-

мента для всевозможных сложных прикладных систем, включая коммуни-

кационные серверы, шлюзы и системы пересылки сообщений рабочих

станций интерсети.

2.9. Протокол объявления услуг

Создавая открытую архитектуру, фирма Novell открывает разработ-

чикам возможность добавлять к существующему набору сетевых услуг

новые сервисы и услуги. Для облегчения таких разработок среда ин-

терсети NetWare включает в себя протокол и механизм, который поз-

воляет сетевым сервисным программам объявлять свои услуги по имени

и типу интерсети NetWare.

Серверы, использующие этот протокол, имеют свое имя, тип серви-

са и адрес интерсети, та же самая информация существует в маршру-

тизаторах NetWare.

Данный механизм позволяет рабочей станции издать широковеща-

тельный пакет запроса в локальную сеть для получения идентификации

всех серверов любого типа, всех серверов специфического типа или

только ближайшего сервера конкретного типа услуг. Дополнительно

рабочая станция может запросить любой сервер для получения имен и

адресов всех серверов данного конкретного типа.

Весь этот механизм известен как протокол объявления услуг

(SAP). Каждый сервер определенного типа имеет заданное уникальное

имя, которое позволяет клиенту выбирать среди серверов определен-

ный тип и определять желаемый сервер по имени, а не по адресу ин-

терсети.

3. ПРОТОКОЛЫ TCP/IP

Соотношение пpотоколов TCP/IP и Модели Откpытых Систем

Уpовни МОС Пpотоколы TCP/IP

------------------¬ --------T----T-----T-----T---------¬

¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦

¦Пpикладной ¦ ¦Telnet ¦FTP ¦TFTP ¦SMTP ¦ DNS ¦

¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦

+-----------------+ ¦ ¦ ¦ ¦ +---------+

¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦

¦Пpедставительный ¦ ¦ ¦ ¦ ¦ ¦ Дpугие ¦

¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦

+-----------------+ +-------+----+---T-+-----+---------+

¦ ¦ ¦ ¦ ¦

¦Сеансовый ¦ ¦ TCP ¦ UDP ¦

¦ ¦ ¦ ¦ ¦

¦ ¦ ¦ ¦ ¦

¦ ¦ ¦ ¦ ¦

¦Тpанспоpтный ¦ ¦ ¦ ¦

¦ ¦ ¦ ¦ ¦

+-----------------+ +-------T--------+T----------------+

¦ ¦ ¦ IP ¦ ICMP ¦ ---------------+

¦Сетевой ¦ ¦ L---------- ¦ ARP RARP ¦

¦ ¦ ¦ ¦ ¦

+-----------------+ +----------T--------+--T-----------+

¦ ¦ ¦ ¦ ¦ ¦

¦Канальный ¦ ¦ ¦ ¦ ¦

¦ ¦ ¦Ethernet ¦Token Ring ¦ Дpугие ¦

+-----------------+ ¦ ¦ ¦ ¦

¦ ¦ ¦ ¦ ¦ ¦

¦Физический ¦ ¦ ¦ ¦ ¦

¦ ¦ ¦ ¦ ¦ ¦

L------------------ L----------+-----------+------------

Рис.3.1.

Уровневая структура модели позволяет разработчикам сетей сосре-

доточиться на функциях конкретного уровня. Другими словами, нет

необходимости создавать все механизмы для посылки информации по

сети. Разработчикам нужно знать сервисы, которые должны быть обес-

печены вышележащему уровню, какие сервисы нижележащего уровня дос-

тупны, и какие протоколы модели обеспечивают эти сервисы.

Сумму уровней протокольного набора часто называют протокольным

стеком (protocol stack) - см. pис .3.2.

Таблица иллюстрирует некоторые из наиболее общих протоколов

TCP/IP и сервисы, обеспечиваемые ими.

Каждый уровень протокольного стека исходного компьютера взаимо-

действует с соответствующим уровнем компьютера-адресата как равный

с равным. С точки зрения программного обеспечения или пользователя

передача данных имеет место так, словно равноправные

Пpотоколы TCP/IP

Хост-ЭВМ - источник Хост-ЭВМ - адpесат

---------------¬ --------------¬

¦ Пpикладной ¦ ¦ Пpикладной ¦

¦ ¦ ¦ ¦

L-------T------- сообщения или L------T-------

¦ ---------- потоки -----------¦

-------+------¬ -------+------¬

¦Тpанспоpтный ¦ ¦Тpанспоpтный ¦

¦ ¦ ¦ ¦

L------T------- дейтагpаммы (UDP) L------T-------

¦ --------- или сегменты (TCP) ------¦

-------+------¬ -------+------¬

¦Межсетевой ¦ ¦ Межсетевой ¦

¦(Интеpсеть) ¦ ¦ (Интеpсеть)¦

L------T------- L------T-------

¦ ---------- дейтагpаммы IP ------- ¦

-------+------¬ -------+------¬

¦Сетевой ¦ ¦ Сетевой ¦

¦интеpфеис ¦ ¦ интеpфейс ¦

L------T------- L------T-------

¦ ---------- кадpы сети ------------ ¦

¦ ¦

¦ ¦

¦ -----------------------¬ ¦

L-------+ Сетевая аппаpатуpа +------

L-----------------------

Рис.3.2.

уровни посылают свои пакеты непосредственно друг другу.

Например, прикладная программа, посылающая файлы с использова-

нием TCP, выполняет следующие операции:

- прикладной уровень передает поток байтов транспортному уровню

на исходном компьютере; - транспортный уровень делит поток на сег-

менты TCP, добавляя заголовок с последовательным номером сегмента

и передает сегмент межсетевому уровню (Internet-IP);

- уровень IP создает пакет с порцией данных, содержащий сегмент

TCP. Уровень добавляет заголовок пакета, содержащий адреса источ-

ника и получателя IP. Уровень определяет также физический адрес

компьютера - адресата или промежуточных компьютеров на пути к узлу

- адресату и передает пакет уровню логической связи;

- уровень звена данных передает пакет IP в порции данных кадра

уровня звена данных компьютеру - адресату. Здесь может иметь место

направление пакета IP промежуточными системами;

- на компьютере - адресате уровень звена данных удаляет заголо-

вок уровня звена данных и передает пакет IP уровню IP;

- уровень IP проверяет заголовок пакета IP. Если контрольная

сумма, содержащаяся в заголовке, не совпадает с контрольной сум-

мой, вычесленной уровнем IP, пакет уничтожается;

- если контрольные суммы совпадают, уровень IP удаляет

Набоp пpотоколов TCP/IP

------------------------T---------------------------------¬

¦ Пpотокол ¦ Сеpвис ¦

+-----------------------+---------------------------------+

¦ ¦ ¦

¦Internet Protocol (IP) ¦ Межсетевой пpотокол обеспечивает¦

¦ ¦ сеpвис доставки пакетов между ¦

¦ ¦ узлами ¦

¦ ¦ ¦

¦Internet Control ¦ Межсетевой пpотокол упpавления ¦

¦Message Protocol (ICMP)¦ сообщениями упpавляет пеpедачей ¦

¦ ¦ упpавляющих сообщений и сообще- ¦

¦ ¦ ний об ошибках между хост-ЭВМ и ¦

¦ ¦ шлюзами ¦

¦ ¦ ¦

¦Address Resolution ¦ Пpотокол pазpешения адpесов ¦

¦Protokol (ARP) ¦ мапиpует межсетевые адpеса в ¦

¦ ¦ физические ¦

¦ ¦ ¦

¦Reverse Address ¦ Обpатный пpотокол pазpешения ¦

¦Resolution Protocol ¦ адpесов мапиpует физические ¦

¦(RARP) ¦ адpеса в интеpсетевые ¦

¦ ¦ ¦

¦Transmission Control ¦ Пpотокол упpавления пеpедачей ¦

¦Protocol (TCP) ¦ обеспечивает сеpвис надежной ¦

¦ ¦ доставки потока между клиен- ¦

¦ ¦ тами ¦

¦ ¦ ¦

¦User Datagram ¦ Пользовательский дейтагpаммный ¦

¦Protocol (UDP) ¦ пpотокол обеспечивает ненадеж- ¦

¦ ¦ ный сеpвис доставки пакетов без ¦

¦ ¦ установления соединения между ¦

¦ ¦ клиентами ¦

¦ ¦ ¦

¦File Transfer ¦ Пpотокол тpанспоpта файлов ¦

¦Protocol (FTP) ¦ обеспечивает услуги тpанспоpта ¦

¦ ¦ файлов пользовательского уpовня ¦

¦ ¦ ¦

¦Telnet ¦ Эмуляция теpминала ¦

¦ ¦ ¦

L-----------------------+----------------------------------

заголовок пакета IP и передает сегмент TCP уровню TCP. Уровень

TCP проверяет последовательный номер для определения - является ли

данный сегмент корректным в последовательности;

- уровень TCP подсчитывает контрольную сумму для заголовка TCP

и данных. Если вычесленная и принятая в заголовке контрольные сум-

мы не совпадают, уровень TCP уничтожает сегмент. Если контрольная

сумма корректна и номер сегмента соответствует последовательности,

уровень TCP посылает положительное подтверждение на компьютер -

источник;

- на компьютере - адресате уровень TCP удаляет заголовок TCP и

передает полученные байта из сегмента прикладной программе;

- прикладная программа на компьютере - адресате получает поток

байтов так, словно она была связана непосредственно с прикладной

программой на компьютере - испточнике.

3.1. Физические адреса и межсетевые адреса.

На уровне звена данных узлы в сети взаимодействуют с другими

узлами сети, используя адреса, специфичные для данной сети. Каждый

узел имееет физический адрес для аппаратуры выхода в сеть. Физи-

ческие адреса имеют различные формы в различных сетях. Например,

физический адрес в Ethernet является 6-байтным числовым значением,

таким как 08-00014-57-69-69. Это значение назначается производите-

лем аппаратуры. Сети X.25 используют стандарт X.121 физических ад-

ресов длиной в 14 цифр. Сети LocalTalk используют 3байтовые адре-

са, состоящие из 2-байтового номера сети и 1-байтового номера уз-

ла. В сети LocalTalk номер сети статический, а номер узла назнача-

ется динамически при запуске узла.

Адрес Межсетевого Протокола (адрес IP) для узла является логи-

ческим адресом - он не зависит от аппаратуры или конфигурации сети

и имеет одну и туже форму независимо от типа сети. Это 4байтное

(32 бита) числовое значение, которое идентифицирует как сеть, так

и локальный узел (компьютер или другое устройство) в данной сети.

4-байтовый адрес IP обычно представляется десятичными числами

(каждый байт), разделяемыми точками, например, 129.47.6.17. Иногда

адреса представляются шестнадцатеричными цифрами.

Узлы, использующие протоколы TCP/IP, транслируют адреса назна-

чения IP в физические адреса аппаратуры подуровня доступа к пере-

дающей среде для того, чтобы посылать пакеты к другим узлам сети.

Каждая посылающая прикладная программа посылает свой адрес IP в

пакете. Принимающая программа может послать ответ источнику, ис-

пользуя адрес IP источника из пакета.

Поскольку адреса IP не зависят от конкретного типа сети, они

могут использоваться для посылки пакета из сети одного типа в дру-

гую сеть. В каждом типе сети программное обеспечение TCP/IP ставит

в соответствие физические адреса сети и адреса IP. Если пакет пе-

редается в другую сеть, адрес IP получателя транслируется в физи-

ческий адрес соответствующей сети.

Сетевой адрес может быть определен одним из следующих способов:

- если Вы хотите соединить Вашу сеть с Интерсетью DARPA, Вы

должны получить зарегистрированный адрес Интерсети в следующей

организации: DDN Network Information Center SRI International

333 Ravenswood Avenue, Room EJ291 Menlo Park, CA 94025 USA - если

Ваша сеть не является частью Интерсети DARPA, Вы можете

выбрать произвольный сетевой адрес. При этом для всех узлов в

сети должны быть выполнены следующие требования:

- сетевая часть каждого адреса должна соответствовать адресу

сети, например, все узлы в сети 129.47 должны использовать адреса

сети 129.47;

- адрес IP для каждого узла должен быть уникальным внутри Вашей

сети.

3.2. Трансляция межсетевых адресов в физические

Когда пакет IP передается по сети, он прежде всего включается в

физический кадр, используемый в данной сети, например, на рис.3.3.

показан пакет IP, включенный в кадр Ethernet. Пакет IP содержит

межсетевой адрес узла, но кадр

Ethernet должен иметь физический адрес узла, чтобы быть достав-

ленным по сети. Следовательно, посылающий узел должен быть в сост-

оянии определить какой физический адрес в сети соответствует адре-

су IP, содержащемуся в пакете IP.

Адрес IP отображается в физический адрес с использованием Про-

токола Разрешения Адресов (ARP) на широковещательных сетях, таких

как Ethernet, Token Ring, ARCnet. Когда узел должен послать пакет

IP он должен определить какой физический адрес в сети соответству-

ет адресу получателя IP, заданному в пакете IP. Для нахождения фи-

зического адреса узел посылает широковещательный пакет ARP, содер-

жащий адрес IP получателя. После этого пакета ожидается ответ от

узла с данным адресом IP получателя. Узел с этим адресом посылает

свой физический адрес назад запрашивающему узлу.

Для быстрой передачи пакетов и уменьшения числа широковеща-

тельных запросов, каждый узел поддерживает кеш разрешения адресов.

Каждый раз когда узел посылает широковещательный запрос ARP и по-

лучает ответ, он создает вход в кеш-памяти разрешения адресов.

Вход отображает адрес IP в физический адрес. Когда узел нуждается

в посылке следующего пакета IP, он ищет адрес IP в кешпамяти. Если

данный адрес IP находится, узел использует соответствующий физи-

ческий адрес для пакета. Широковещательный запрос IP посылается

только если соответствующий адрес IP не обнаруживается в кеш-памя-

ти.

Включение дейтагpаммы IP в кадp Ethernet

¦дейтаграмма ¦

¦ IP ¦

г=======T========T========T========T=============T========¬

¦ Пpеам-¦ адрес ¦ адрес ¦ тип ¦ данные ¦контр. ¦

¦ була ¦ получа-¦ источ-¦ пакета¦ пакета ¦сумма ¦

¦ ¦ теля ¦ ника ¦ ¦ ¦Ethernet¦

L=======¦========¦========¦========¦=============¦========-

¦ ¦

¦------- заголовок кадра-----------¦--- данные---¦ ¦

¦ ¦ кадра ¦ ¦

¦ ¦

¦-------------------- кадр Ethernet-----------------------¦

Рис.3.3.

3.3. Ненадежный сервис доставки пакетов

В наборе протоколов TCP/IP все пакеты доставляются ненадежным

сервисом доставки пакетов неориентированным на соединение Межсете-

вого Протокола. Сервис ненадежен, поскольку доставка пакета не га-

рантируется. Сервис не ориентирован на соединение, поскольку все

пакеты передаются независимо друг от друга. Приложения TCP/IP ис-

пользующеее этот сервис должны заботиться о состоянии достаки,

например, ожиданием ответов от узла получателя. Кроме того,

маршрутизаторы (routers) в Интерсети могут посылать сообщения об

ошибках (ICMP) для информирования узлов о проблемах. Термин

"маршрутизация" относится к передаче дейтаграммы от одного узла к

другому на той же или другой сети. Термин относится также к путям,

которые выбираются для передачи дейтаграммы IP от источника к по-

лучателю на базе адреса IP, содержащегося в дейтаграмме. Существу-

ет два способа маршрутизации: прямая и непрямая.

Прямая маршрутизация имеет место при передаче дейтаграммы в

рамках одной сети. Узел, посылающий дейтаграмму IP, может прямо

запросить другие узлы в сети о физическом адресе, соответствующем

адресу IP, включить дейтаграмму IP в физический кадр с данным фи-

зическим адресом и послать его непосредственно узлу назначения в

сети.

Непрямая маршрутизация связана спередачей дейтаграммы из одной

сети в другую через узел, называемый маршрутизатором. Когда дей-

таграмма посылается к узлу другой сети, сетевая часть адреса IP

источника и адреса IP получателя различны. Посылающий узел распоз-

нает эту разницу и посылает пакет к маршрутизатору, который соеди-

няет исходную сеть с другими сетями.

Посылающий узел имеет таблицу адресов IP для одного или больше

компьютеров в сети, которые функционируют в качестве маршрутизато-

ров к другим сетям. Узел ищет адрес IP маршрутизатора в своей таб-

лице и посылает широковещательный запрос ARP к маршрутизатору для

получения его физического адреса. Затем пакет, содержащий дейтаг-

рамму IP, посылается по физическому адресу маршрутизатора. Когда

маршрутизатор получает дейтаграмму IP, он использует адрес IP в

дейтаграмме для посылки получателю аналогичным образом. Если адрес

IP принадлежит к сети, подключенной непосредственно к маршрутиза-

тору, последний посылает дейтаграмму непосредственно узлу назначе-

ния. Для всех других адресов сетей маршрутизатор имеет только ад-

рес другого маршрутизатора, который может направить пакет к полу-

чателю.

3.4. Межсетевой Протокол IP

Межсетевой протокол определяет форму пакетов и способы под-

держки пакетов при передаче и приеме. Форма пакета называется дей-

таграммой IP. Дейтаграмма IP аналогична физическому кадру, переда-

ваемому по сети. Дейтаграмма имеет секцию заголовка, содержащую

адреса IP отправителя и получателя и секцию данных.

Структура дейтаграммы IP

¦ Заголовок дейтаграммы IP ¦ Данные дейтаграммы IP ¦

¦ ¦ ¦

г=============T=============+==============T==============¬

¦ Информация ¦ Адрес ¦ Адрес ¦ Данные ¦

¦ заголововка¦ источника IP¦ получателя IP¦ ¦

L=============¦=============¦==============¦==============-

Рис.3.4.

В отличие от кадра сети, имеющего физическую длину, установлен-

ную в соответствии с характеристиками физической сети, длина дей-

таграммы устанавливается сетевым программным обеспечением. Прог-

раммное обеспечение IP на узле создает дейтаграмму, которая поме-

щается внутрь кадра сети. Двигаясь к узлу назначения, однако, дей-

таграмма может пройти по многим сетям различных типов с различными

длинами физических кадров. Для поддержки передачи пакета протокол

IP задает метод разбиения дейтаграмм на фрагменты на каждом узле,

который должен передавать дейтаграммы и соответствующий метод ре-

ассемблирования пакетов на узле назначения. Так маршрутизатор,

связывающий две сети, будет должен фрагментировать пакеты IP, по-

лучаемые из одной сети, если вторая сеть имеет меньший размер фи-

зического кадра, чем первая. Будучи однажды фрагментированными,

пакеты не реассемблируются до достижения точки назначения.

3.5. Сообщения об ошибках и управляющие сообщения

Другой протокол набора TCP/IP это Межсетевой Протокол Управляю-

щих Сообщений (ICMP). Пакеты ICMP содержат информацию об авариях в

сети: нефункционирующих узлах и шлюзах, проблемах с пакетами в

шлюзах и т.д. Программное обеспечение IP интерпретирует сообщение

ICMP и затем предпринимает соответствующие действия в соответствии

с сообщением независимо от прикладной программы. Поскольку сообще-

ние ICMP может передаваться через несколько сетей для достижения

назначения, оно помещается в порцию данных дейтаграммы IP.

3.6. Протоколы транспортного уровня: UDP и TCP

Два протокола составляют транспортный уровень набора TCP/IP: м

Пользовательский Дейтаграммный Протокол (UDP) и Протокол Управле-

ния Передачей (TCP). UDP обеспечивает ненадежный сервис доставки

без соединений для посылки и получений сообщений. TCP добавляет

надежные потоковые сервисы доставки над ненадежным сервисом доста-

ки пакетов без соединений IP.

В наборе TCP/IP UDP позволяет приложениям обмениваться индиви-

дуальными пакетами сообщений через сеть. Протокол UDP определяет

набор назначений как протокольные порты. При этом определяются два

типа протокольных портов: хорошо известное назначение портов и ди-

намически связанные порты. В первом случае,

TCP/IP резервирует соответствующие номера портов для соответст-

вующих приложений. Порты сномерами от 1 до 255 являются хорошо из-

вестными номерами портов и назначаются соответствующим широко из-

вестным приложениям. В случае динамически связанных портов прило-

жение, запрашивающее сервисы у процесса, должны прежде всего зап-

росить узел для идентификации порта, который использует процесс.

Он может в дальнейшем направлять дейтаграммы UDP этому порту.

Дейтаграмма UDP включается в одну или более дейтаграмм IP, ко-

торые в свою очередь включаются в кадр сети, например, Ethernet

как показано на рис. 3.5. В этом примере адрес IP

Включение дейтаграммы UDP

¦ Заголовок ¦ Данные ¦

¦ UDP ¦ UDP ¦

г======T=====T=+==========¬

¦Порт ¦Порт ¦ ¦ Данные ¦

¦источн¦назн ¦ ¦ UDP ¦

L======¦=====¦=¦==========-

¦ ¦

¦--- Дейтаграмма UDP ---- ¦

¦

г=========+=========================¬

¦Заголовок¦ ¦

¦ IP ¦ ¦

L=========¦=========================-

¦ ¦

¦--------- Дейтаграмма IP ----------¦

¦ ¦

г==========+===================================+==========¬

¦Заголовок ¦ ¦ Контр. ¦

¦кадра ¦ ¦ сумма ¦

L==========¦===================================¦==========-

¦ ¦

¦------------------- Кадр Ethernet -----------------------¦

¦ ¦

Рис.3.5.

направляет дейтаграмму IP к соответствующему узлу, на котором

программное обеспечение IP извлекает дейтаграмму UDP и доставляет

ее программному обеспечению уровня UDP. Программное обеспечение

уровня UDP доставляет данные UDP и управляющую информацию к задан-

ному протокольному порту назначения. Процесс на этом порту ис-

пользует данные из дейтаграммы UDP. Дейтаграмма UDP также содержит

протокольный порт источника, позволяющий процессу назначения отве-

тить корректно.

Для приложений, которые нуждаются в посылке значительных

объемов данных, ненадежная передача может создать трудности. Прик-

ладные программисты будут должны разрабатывать дорогие процедуры и

модули обеспечения надежной передачи. Набор протоколов TCP/IP об-

ходит эти проблемы через использование Протокола Управления Пере-

дачей (TCP) - надежного протокола с потоковой доставкой. Он уста-

навливает виртуальный канал между двумя

приложениями и посылает поток байтов получателю в том же поряд-

ке, в котором он покидает отправителя. Перед началом передачи при-

ложения на обоих концах получают порты TCP от своих операционных

систем. Эти порты аналогичны тем, что использует протокол UDP.

Приложение, начинающее передачу, известно как активная сторона,

обычно получает порт динамически. Приложение, товечающее на запрос

передачи, известное как пассивная сторона, обычно использует широ-

ко известный порт TCP на пассивной стороне. Подобно дейтаграммам

UDP, сегменты TCPвключаются в дейтаграммы IP. TCP буферизует поток

ожиданием достаточного количества данных для заполнения большой

дейтаграммы перед ее посылкой. Поток неструктурирован, что означа-

ет, что посылающая и принимающая стороны (приложения) до передачи

должны установить соглашения относительно содержимого потока. TCP

использует полнодуплексную передачу.

Протокол TCP присваивает каждому сегменту последовательный но-

мер. На приемном конце виртуального канала приложение проверяеет

последовательность намеров для определения того, все ли сегменты

получены и обработаны в порядке последовательных номеров. Когда

приемный конец получает следующий сегмент последовательности, он

посылает квитанцию - подтверждение узлу - источнику. Когда узел -

источник получает подтверждение, он индицирует приложению то, что

сегмент успешно передан. Если узел - источник в течение определен-

ного таймаута не получает подтверждения, он повторно передает со-

ответствующий сегмент. Эта схема, называемая "положительное

подтверждение с повторной передачей", обеспечивает надежность пе-

редачи.

3.7. Транспортная система TCP/IP в NetWare версии 3.11

Транспортная система TCP/IP обеспечивает возможности подключе-

ния к сетям с этими протоколами для сетей фирмы Novell. На файло-

вом сервере TCP/IP включает в себя набор NLM для поддержки возмож-

ностей маршрутизации IP и построения интерсетей, поддержки функци-

онирования приложений, таких как NetWare NFS, возможность проклад-

ки туннелей IPX через интерсети IP. Дополнительно TCP/IP обеспечи-

вает транспортный интерфейс, используемый как NFS, так и другими

приложениями, написанными для интерфейсов типа сокет 4.3BSD UNIX

или AT&T Streams Transport Layer Interface (TLI). NetWare v3.11

TCP/IP включает следующие загружаемые модули и файлы базы данных:

NLM NetWare TCP/IP (TCPIP.NLM)

The Simple Network Management Protocol NLM

(SNMP.NLM)

The SNMP event logger NLM (SNMPLOG.NLM)

The TCP/IP Console NLM (TCPCON.NLM)

The IP configuration NLM (IPCONFIG.NLM)

The IPX/IP Tunnel module (IPTUNNEL.LAN)

Sample Internet database files (GATEWAYS, HOSTS,

NETWORKS, PROTOCOL, and SERVICES)

Архитектура TCP/IP приведена на рис.3.6.

Архитектура TCP/IP в NetWare

-----------------------------------------------------------¬

¦ ¦

+----------------------------------------------------------+

¦------------¬ ----------¬ -----------¬ --------------¬¦

¦¦ Другие ¦ ¦ TCPCON ¦ ¦ SNMPLOG ¦ ¦ Другие ¦¦

¦¦ приложения¦ ¦ ¦ ¦ ¦ ¦ приложения ¦¦

¦¦ TLI ¦ ¦ ¦ ¦ ¦ ¦ для сокетов ¦¦

¦L-----T------ L----T----- L-----T----- L------T-------¦

¦ ¦ L--------------+--------------+ ¦

¦ ¦ ¦ ¦

¦ ¦ г============¬ ¦ ¦

¦ \¦/ ¦ ¦ \¦/ ¦

¦--------------¬ ¦ ¦ -------+------¬¦

¦¦ Интерфейс ¦ ¦ NetWare ¦ ¦ Сокеты ¦¦

¦¦ транспортн ¦ ¦ NFS ¦ ¦ BSD ¦¦

¦¦ уровня TLI ¦ ¦ ¦ ¦ ¦¦

¦L-----T-------- L======T=====- L------T-------¦

¦ ¦ ¦ ¦ ¦

¦ \¦/ \¦/ \¦/ ¦

¦------+----------------------+--------------------+------¬¦

¦¦ Транспортная система TCP/IP NetWare ¦¦

¦L--------------------------T------------------------------¦

¦ ¦ ¦

¦ ----------------+----------------¬ ¦

¦ ¦Открытый интерфейс ODI ¦ ¦

¦ L----T----------T----------T------ ¦

¦ ----------- \¦/ L-----------¬ ¦

¦ -----+----¬ -------+------¬ ------+----¬ ¦

¦ ¦ Драйвер ¦ ¦ Драйвер ¦ ¦ Драйвер ¦ ¦

¦ ¦ Ethernet¦ ¦ Token Ring ¦ ¦ ARCnet ¦ ¦

¦ L----T----- L------T------- L-----T----- ¦

¦ \¦/ \¦/ \¦/ ¦

¦------+--------¬ --------+---------¬ -------+-----¬ ¦

¦¦ Адаптер ¦ ¦ Адаптер ¦ ¦ Адаптер ¦ ¦

¦¦ ¦ ¦ ¦ ¦ ¦ ¦

¦¦ Ethernet ¦ ¦ Token Ring ¦ ¦ Arcnet ¦ ¦

¦L--------------- L------------------ L------------- ¦

¦ ¦

L-----------------------------------------------------------

Рис.3.6.

Программное обеспечение TCP/IP позволяет реализовать различные

конфигурации сетей IP IPX.

Сеть IP (серверы и клиенты UNIX) могут взаимодействовать между

собой, используя протокол TCP/IP, в одной физической сети с серве-

рами и клиентами NetWare. Пример такой сети Ethernet с адресом се-

ти IP 129.1.0.0 и с адресом сети IPX 84404556 приведен на рисунке

3.7.

Совместная работа TCP/IP и SPX/IPX

г================¬

¦ NetWare Server ¦ ---------------¬

¦ ¦ ¦ UNIX Server ¦

L=========T======- L-------T-------

129.1.0.3 ¦ ¦ 129.1.0.1

¦ ¦

¦IPX Net=84404556 ¦ IP Net=129.1.0.0

------T---+------------T----+------------T-------------

¦ ¦ ¦

¦ ¦ ¦

------+--------¬ -----+---------¬ ¦ 129.1.0.2

¦NetWare Client¦ ¦NetWare Client¦ -----+-----------¬

¦ ¦ ¦ ¦ ¦UNIX Workstation¦

L--------------- L--------------- L-----------------

Рис.3.7.

Сети IP и IPX могут подключаться к одному серверу NetWare. Кли-

енты UNIX могут использовать файловое пространство сервера с по-

мощью Netware NFS (рис.3.8.).

Подключение сетей TCP/IP и SPX/IPX к серверу

---------------------¬ --------------¬

¦Рабочая станция UNIX¦ ¦ Сервер UNIX ¦

L---------T----------- L-----T--------

¦129.1.0.2 ¦129.1.0.1

¦ ¦

¦ ¦Сеть IP=129.1.0.0

-----------+----------------T------+-------------------

129.1.03 ¦

г========¦=========¬

¦ Сервер NetWare ¦

L=========T========-

¦

Сеть IPX = 84404556 ¦ Сеть Token Ring

------------------------+-----------------------¬

L------T------------------------------------T----

¦ ¦

-------+-------¬ -------+-------¬

¦Клиент NetWare¦ ¦Клиент NetWare¦

L--------------- L---------------

Рис.3.8.

Есть возможность маршрутизации пакетов IP через сервер NetWare.

На рис.3.9. TCP/IP в сервере Netware 1 обеспечивают маршрутизацию

пакетов IP, давая возможность клиентам UNIX из сети IP Ethernet

работать с сервером Netware 2.

Маршрутизация пакетов IP через сервер NetWare

---------------------¬ --------------¬

¦Рабочая станция UNIX¦ ¦ Сервер UNIX ¦

L---------T----------- L-----T--------

¦129.1.0.2 ¦129.1.0.1

¦ ¦

¦ ¦Сеть IP=129.1.0.0

-----------+----------------T------+-------------------

129.1.03 ¦

г========¦=========¬

¦ Сервер NetWare 1¦

L=========T========-

¦

Сеть IPX = 84404556 ¦ Сеть Token Ring

------------------------+-----------------------¬

L----T------------------T-----------------T------

¦ ¦ ¦

-------+-------¬ г=======¦========¬ -------+-------¬

¦Клиент NetWare¦ ¦Сервер NetWare 2¦ ¦Клиент NetWare¦

L--------------- L================- L---------------

Рис.3.9.

Включив в состав сети IP, изображенной на рис.3.10.-, маршрути-

затор IP можно обеспечить выход в глобальную интерсеть IP.

Средства прокладки туннелей IP обеспечивают прозрачное взаимо-

действие сетей IPX через интерсеть IP. Туннель IP представляет

стандартный интерфейс драйвера локальной сети для системы NetWare.

Сама интерсеть IP является средой передачи. В этом смысле адрес IP

выполняет в среде IP ту же функцию, что физический адрес в физи-

ческой среде. Пример использования туннеля IP для взаимодействия

сетей IPX через интерсеть IP приведен на рис.3.10.

Взаимодействие сетей NetWare через туннель IP

Туннель IP г============¬

¦ ¦ Сеть IPX ¦

¦ L======T=====-

¦ ¦

-----------------¬ ¦ г============¬ ----------+------¬

¦ Сервер NetWare +-+-¦Интерсеть IP¦---+ Сервер NetWare ¦

¦ +---¦ ¦-T-+ ¦

L--------T-------- L============- ¦ L-----------------

¦ ¦

¦ ¦

г========¦=======¬ Туннель IP

¦ Сеть IPX ¦

L================-

Рис.3.10.

4. ОБМЕН СООБЩЕНИЯМИ В СЕТИ.

Две программы, работающие на разных узлах сети, могут обмени-

ваться данными двумя способами: пересылка пакетов через низкоуров-

невый драйвер сети и пересылка сообщений через общий файл-сервер.

Пример программы, посылающей сообщение на консоль сервера :

Uses Dos;

var k : integer;

sendString : string;

Procedure SendToCon(Message : string);

var

reg : registers;

s : array [0..64] of byte;

i : integer;

begin

s[0] := 0;

s[1] := 4;

s[2] := $09;

s[3] := length(Message);

for i := 1 to Length(Message) do

s[i+3] := ord(Message[i]);

reg.ah := $0E1;

reg.bx := reg.ds;

reg.ds := seg(s);

reg.si := ofs(s);

intr($21,reg);

reg.ds := reg.bx;

end;

begin

If ParamCount = 0 then

begin

Writeln('Send message to console utility by S.Perevoznik, 1993');

Writeln('Usage : ssend <message>');

halt;

end;

for k := 1 to paramcount do

begin

SendString := SendString + Paramstr(k) + ' ';

end;

SendToCon(sendString);

Writeln('Message send to console ...');

end.

Для передачи сообщения конкретному пользователю можно использовать

следующие функции :

{$F+,O+}

Unit UICNet;

Interface

Uses Dos;

Function GetUserName(ConnectionNumber : byte;

Var ObjectFree : boolean) : string;

Function LocalDisk : boolean;

Procedure SendBroadcastMessage(Message:string;ConnectionNumber:byte);

Implementation

Function GetUserName(ConnectionNumber : byte;

Var ObjectFree : boolean) : string;

var

WordPtr:^Word;

r : registers;

SendPacket : array[0..4] of byte;

ReplyPacket : array[0..64] of byte;

UserName : string;

ObjectType : word;

function GetWord(P: pointer): word;

var

WordPtr :^word;

begin

WordPtr := P;

GetWord := swap(WordPtr^);

end;

begin

SendPacket[2] := $16;

SendPacket[3] := ConnectionNumber;

WordPtr := addr(SendPacket);

WordPtr^:=2;

WordPtr := addr(SendPacket);

WordPtr^ := 62;

r.ah := $e3;

r.ds := seg(SendPacket);

r.si := ofs(SendPacket);

r.es := seg(ReplyPacket);

r.di := ofs(ReplyPacket);

intr($21,r);

if r.al = 0

then

begin

ObjectType := GetWord(addr(ReplyPacket[6]));

if ObjectType = 0 then ObjectFree := true

else

ObjectFree := false;

move(ReplyPacket[8],UserName[1],48);

UserName[0] := chr(48);

GetUserName := Username;

end;

end;

Function LocalDisk : boolean;

var r : registers;

begin

r.ah := $19;

intr($21,r);

r.dl := r.al+1;

r.ah := $e9;

r.al := 0;

r.dh := 0;

intr($21,r);

if r.ah = 128 then localdisk := true

else localdisk := false;

end;

Procedure SendBroadcastMessage(Message:string; ConnectionNumber:byte);

var

r : registers;

WordPtr : ^word;

SendPacket : array [0..160] of byte;

ReplyPacket : array [0..103] of byte;

begin

SendPacket[2] := 0;

SendPacket[3] := 1;

SendPacket[4] := ConnectionNumber;

SendPacket[5] := length(Message);

move(Message[1],SendPacket[6],length(Message));

WordPtr := addr(SendPacket);

WordPtr^ := Length(Message) + 4;

r.ah := $e1;

r.ds := seg(SendPacket);

r.si := ofs(SendPacket);

r.es := seg(ReplyPacket);

r.di := ofs(ReplyPacket);

intr($21,r);

end;

end.

Следующая программа демонстрирует возможность получения списка поль-

зователей, подключенных к сети:

uses Dos;

function GetWord(P: pointer): word;

var

WordPtr :^word;

begin

WordPtr := P;

GetWord := swap(WordPtr^);

end;

Function GetLong(P:Pointer): longint;

type long= record

case integer of

1:(Long1: Longint);

2:(Word1,Word2:word);

end;

var LongPtr :^Long;

L : long;

begin

LongPtr := P;

L.Word1 := swap(LongPtr^.Word2);

L.Word2 := swap(LongPtr^.Word1);

GetLong:= L.Long1;

end;

Function GetConnectionInformation(ConnectionNumber:byte;

Var ObjectName : string; var ObjectType : word;

var ObjectID : longint; var LoginTime : string): byte;

var

WordPtr:^Word;

r:registers;

SendPacket : array[0..4] of byte;

ReplyPacket : array[0..64] of byte;

begin

SendPacket[2] := $16;

SendPacket[3] := ConnectionNumber;

WordPtr := addr(SendPacket);

WordPtr^:=2;

WordPtr := addr(SendPacket);

WordPtr^ := 62;

r.ah := $e3;

r.ds := seg(SendPacket);

r.si := ofs(SendPacket);

r.es := seg(ReplyPacket);

r.di := ofs(ReplyPacket);

intr($21,r);

if r.al = 0

then

begin

ObjectID := GetLong(addr(ReplyPacket[2]));

ObjectType := GetWord(addr(ReplyPacket[6]));

move(ReplyPacket[8],ObjectName[1],48);

ObjectName[0] := chr(48);

move(ReplyPacket[56],LoginTime[1],7);

LoginTime[0] := chr(7);

end;

GetConnectionInformation := r.al;

end;

var

ObjectName,LoginTime : string;

ObjectType : word;

ObjectID : longint;

ConnectionNumber, CCode : byte;

begin

ConnectionNumber := 1;

CCode := GetConnectionInformation(ConnectionNumber,

ObjectName,ObjectType,ObjectID,LoginTime);

while (CCode <> 253) and (ConnectionNumber<255)

do

begin

if CCode <> 0

then

writeln('Cod ',CCode,ConnectionNumber)

else if ObjectType <>0 then begin

Writeln(ConnectionNumber,' ',ObjectType,' ',ObjectID);

Writeln(ObjectName);

end;

inc(ConnectionNumber);

CCode := GetConnectionInformation(ConnectionNumber,

ObjectName,ObjectType,ObjectID,LoginTime);

end;

end.

5. РАБОТА С ФАЙЛАМИ В ЛВС.

Ниже приводятся тексты процедур, используемых при работе в ЛВС

для обслуживания файловой системы:

; Данная библиотека процедур разработана для использования в прог-

; раммах, написанных на Турбо-Паскале версии 4.0 и выше.

; Joe R. Doupnik and Sergey V. Perevoznik, 1988 - 1993

lprog equ 1 ; 0 for small memory, 1 for large memory model

; Modify lprog to match the memory model

if lprog

x equ 6 ; prologue overhead for large memory model

else

x equ 4 ; ditto, small memory model

endif

begin macro name ; begin a function, near or far

public name

if lprog

name proc far

else

name proc near

endif

endm

; define Borland Pascal segments

; use neither Group nor Class

data segment word public

data ends

cseg segment byte public

assume cs:cseg, ds:data, es:nothing

;--------------------------------

; Extended Open a File func(61) (3DH)

;

; var

; Mode, Handle, RetCode : Integer;

; Filename : String[n];

;

; RetCode := xtndopn(Mode, Handle, Filename);

;

begin xtndopn

push bp

mov bp,sp

push ds

lds si,[bp+x+8] ; put the mode in ax

mov ax,[si]

mov ah,03Dh ; put the function in ah

lds dx,[bp+x] ; ptr to ASCIIZ string

inc dx ; add 1 to get to the beginning of the string

int 21h

jc xtndopx ; carry set means failure

lds di,[bp+x+4] ; put the handle into the variable

mov [di],ax ; if there is no error and

xor ax,ax ; return a 0

xtndopx:pop ds

pop bp

ret 14

xtndopn endp

;--------------------------------

; Get or Set File Attributes(handle) func(67) (43H)

;

; var

; Func, Attribute, ErrCode : Integer;

; Filename : String[n];

;

;ErrCode := setattr(Func, Attribute, Filename);

;

begin setattr

push bp

mov bp,sp

push ds

lds si,[bp+x+8] ; put the subfunction in ax

mov ax,[si]

mov ah,043h ; put the function in ah

lds dx,[bp+x] ; ptr to var desc block for ASCIIZ string

inc dx ; add 1 to get the address of the string

lds si,[bp+x+4] ; put the attribute to set in cx

mov cx,[si]

int 21h

jc setattx ; carry set means failure

lds di,[bp+x+4] ; put the returned attributes in the

mov [di],cx ; variable if there is no error

xor ax,ax ; and return zero

setattx:mov ah,0

pop ds

pop bp

ret 14

setattr endp

;--------------------------------

; End of Job Flag Status func(187) (BBH)

;

; var

; Flag : Integer;

;

;ErrCode := eojstat(Flag);

;

begin eojstat

push bp

mov bp,sp

push ds

lds si,[bp+x] ; put the flag in ax

mov ax,[si]

mov ah,0BBh ; put the function in ah

int 21h

mov ah,0

lds di,[bp+x]

mov [di],ax

pop ds

pop bp

ret 6

eojstat endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Log Record BCH

;

; var

; ErrCode,FileHandle,HiByteOffset,LoByteOffset : Integer

; HiLockLen,LoLockLen,Flags,TimeOut : Integer

;

;ErrCode := PRLH\_Log(FileHandle,HiByteOffset,LoByteOffset,HiLockLen,

; LoLockLen,Flags,TimeOut);

;

begin PRLH\_Log

push bp

mov bp,sp

push ds

lds si,[bp+x+24] ; FileHandle

mov bx,[si]

lds si,[bp+x+20] ; ByteOffset

mov cx,[si]

lds si,[bp+x+16]

mov dx,[si]

lds si,[bp+x+4] ; AL = Log Flags

mov ax,[si]

lds si,[bp+x+12] ; LockLen

mov si,[si]

lds di,[bp+x+8]

mov di,[di]

lds bp,[bp+x] ; time out

mov bp,[bp]

mov ah,0BCH

int 21H

mov ah,0 ; clear the high byte of the return

pop ds

pop bp

ret 30

PRLH\_Log endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Release Rec BDH

;

; var

; FileHandle,HiByteOffset,LoByteOffset,ErrCode,

; HiWordLen,LoWordLen : integer;

; ErrCode := PRLH\_Rel(FileHandle,HiByteOffset,LoByteOffset,HiWordLen,

; LoWordLen:integer);

begin PRLH\_Rel

push bp

mov bp,sp

push ds

lds si,[bp+x+16] ; FileHandle

mov bx,[si]

lds si,[bp+x+12] ; HiByteOffset

mov cx,[si]

lds si,[bp+x+8] ; LoByte

mov dx,[si]

lds si,[bp+x+4] ; HiWordLen

mov si,[si]

lds di,[bp+x] ; LoWordLen

mov di,[di]

mov ah,0BDH

int 21H

mov ah,0

pop ds

pop bp

ret 22

PRLH\_Rel endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Release Clr BEH

;

; var

; FileHandle,HiByteOffset,LoByteOffset,ErrCode,

; HiWordLen,LoWordLen : integer;

; ErrCode := PRLH\_Rel(FileHandle,HiByteOffset,LoByteOffset,HiWordLen,

; LoWordLen:integer);

;

begin PRLH\_Clr

push bp

mov bp,sp

push ds

lds si,[bp+x+16] ; FileHandle

mov bx,[si]

lds si,[bp+x+12] ; ByteOffset

mov cx,[si]

lds si,[bp+x+8]

mov dx,[si]

lds si,[bp+x+4] ; HiWordLen

mov si,[si]

lds di,[bp+x] ; LoWordLen

mov di,[di]

mov ah,0BEH

int 21H

mov ah,0

pop ds

pop bp

ret 22

PRLH\_Clr endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Log Rec FCB BFH

;

; var

; ErrCode,fcb,HiByteOffset,LoByteOffset : Integer;

; HiLockLen,LoLockLen,Flags,TimeOut : Integer;

;

; ErrCode := PRLF\_Log(fcb,HiByteOffset,LoByteOffset,HiLockLen,LoLockLen,Flags,TimeOut);

;

begin PRLF\_Log

push bp

mov bp,sp

push ds

lds si,[bp+x+24] ; FCB

mov dx,[si]

lds si,[bp+x+20] ; ByteOffset

mov bx,[si]

lds si,[bp+x+16]

mov cx,[si]

lds si,[bp+x+4] ; AL = Log Flags

mov ax,[si]

lds si,[bp+x+8] ; low lock length

mov di,[si]

lds si,[bp+x+12] ; hi lock length

mov si,[si]

mov bp,[bp+x] ; timeout value

mov bp,[bp]

mov ah,0BFH

int 21H

mov ah,0

pop ds

pop bp

ret 28

PRLF\_Log endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Rel Rec FCB C0H

;

; var

; ErrCode,fcb,HiByteOffset,LoByteOffset : Integer;

;

; ErrCode := PRLF\_Rel(fcb,HiByteOffset,LoByteOffset);

;

begin PRLF\_Rel

push bp

mov bp,sp

push ds

lds dx,[bp+x+8] ; FCB

lds si,[bp+x+4] ; ByteOffset

mov bx,[si]

lds si,[bp+x]

mov cx,[si]

mov ah,0C0H

int 21H

mov ah,0

pop ds

pop bp

ret 12

PRLF\_Rel endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Physical Record Lock: Clr Rec FCB C1H

;

; var

; ErrCode,fcb,HiByteOffset,LoByteOffset : Integer;

;

; ErrCode := PRLF\_clr(fcb,HiByteOffset,LoByteOffset);

;

begin PRLF\_Clr

push bp

mov bp,sp

push ds

lds dx,[bp+x+8] ; FCB

lds si,[bp+x+4] ; ByteOffset

mov bx,[si]

lds si,[bp+x]

mov cx,[si]

mov ah,0C0H

int 21H

mov ah,0

pop ds

pop bp

ret 12

PRLF\_Clr endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Lock Record Set C2H

;

; var

; ErrCode, Flags, TimeOut : Integer;

;

; ErrCode := PRLS\_Lck(Flags,TimeOut);

;

begin PRLS\_Lck

push bp

mov bp,sp

push ds

lds si,[bp+x+4] ; AL = Lock flag bits

mov ax,[si]

lds si,[bp+x] ; TimeOut Value

mov bp,[si]

mov ah,0C2h

int 21h

mov ah,0

pop ds

pop bp

ret 10

PRLS\_Lck endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Release Record Set C3H

;

; var

; ErrCode : Integer;

;

; ErrCode := PRLS\_Rel;

;

begin PRLS\_Rel

mov ah,0C3h

int 21h

mov ah,0

ret

PRLS\_Rel endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Clear Record Set C4H

;

; var

; ErrCode : Integer;

;

; errcode := PRLS\_Clr;

;

begin PRLS\_Clr

mov ah,0C4h

int 21h

mov ah,0

ret

PRLS\_Clr endp

;--------------------------------

; Semaphores func(197) (C5H) subfunction 00h

;

; var

; RetCode,SemaValu,HiHandle,LoHandle,OpenCnt : Integer;

; Sema4 : String[n];

;

; RetCode := OpenSem(Sema4,SemaValu,HiHandle,LoHandle,OpenCnt);

;

begin OpenSem

push bp

mov bp,sp

push ds

lds dx,[bp+x+16] ; ptr to Semaphore string

; inc dx ; add 1 to get the address of the string

lds si,[bp+x+12] ; set semaphore value

mov cx,[si]

mov ax,0C500h ; set the function

int 21h

mov bh,00h

lds di,[bp+x] ; return the open count

mov [di],bx

lds di,[bp+x+4] ; return low handle

mov [di],cx

lds di,[bp+x+8] ; return high handle

mov [di],dx

mov ah,00h

pop ds

pop bp

ret 22

OpenSem endp

;--------------------------------

; Semaphores func(197) (C5H) subfunction 01h

;

; var

; RetCode,HiHandle,LoHandle,SemaValu,OpenCnt : Integer;

;

; RetCode := ExamSem(HiHandle,LoHandle,SemaValu,OpenCnt);

;

begin ExamSem

push bp

mov bp,sp

push ds

lds si,[bp+x+12] ; hi handle

mov dx,[si]

lds si,[bp+x+8] ; lo handle

mov cx,[si]

mov ax,0C501h ; set function

int 21h

lds di,[bp+x+4] ; return semaphore value

mov [di],cx

mov dh,00h

lds di,[bp+x]

mov [di],dx ; return open count

mov ah,00h

pop ds

pop bp

ret 16

ExamSem endp

;--------------------------------

; Semaphores func(197) (C5H) subfunction 02h

;

; var

; RetCode,HiHandle,LoHandle,TimeOut : Integer;

;

; RetCode := WaitSem(HiHandle,LoHandle,TimeOut);

;

begin WaitSem

push bp

mov bp,sp

push ds

lds si,[bp+x+8] ; hi handle

mov dx,[si]

lds si,[bp+x+4] ; lo handle

mov cx,[si]

lds si,[bp+x]

mov bp,[si] ; set time out

mov ax,0C502h ; set function

int 21h

mov ah,00h

pop ds

pop bp

ret 12

WaitSem endp

;--------------------------------

; Semaphores func(197) (C5H) subfunction 03h

;

; var

; RetCode,HiHandle,LoHandle : Integer;

;

; RetCode := SigSem(HiHandle,LoHandle);

;

begin SigSem

push bp

mov bp,sp

push ds

lds si,[bp+x+4] ; hi handle

mov dx,[si]

lds si,[bp+x] ; lo handle

mov cx,[si]

mov ax,0C503h ; set function

int 21h

mov ah,00h

pop ds

pop bp

ret 8

SigSem endp

;--------------------------------

; Semaphores func(197) (C5H) subfunction 04h

;

; var

; RetCode,HiHandle,LoHandle : Integer;

;

; RetCode := ClosSem(HiHandle,LoHandle);

;

begin ClosSem

push bp

mov bp,sp

push ds

lds si,[bp+x+4] ; hi handle

mov dx,[si]

lds si,[bp+x] ; lo handle

mov cx,[si]

mov ax,0C504h ; set function

int 21h

mov ah,00h

pop ds

pop bp

ret 8

ClosSem endp

;--------------------------------

; Get or Set Lock Mode func(198) (C6H)

;

; var

; Mode,Func : Integer;

;

; Mode := setlck(Func);

;

begin setlck

push bp

mov bp,sp

push ds

lds si,[bp+x]

mov ax,[si] ; set the subfunction

mov ah,0C6h ; set the function

int 21h

mov ah,0

pop ds

pop bp

ret 4

setlck endp

;--------------------------------

; Transactional Backout func(199) (C7H)

;

; var

; RetCode, Func : Integer;

;

; RetCode := BakOuts(Func);

;

begin BakOuts

push bp

mov bp,sp

push ds

lds si,[bp+x] ; set the subfunction

mov ax,[si]

mov ah,0C7h ; set the function

int 21h

mov ah,0

pop ds

pop bp

ret 4

BakOuts endp

;--------------------------------

; Begin transaction func(200) (C8H)

;

; var

; RetCode,Mode : Integer;

;

; RetCode := btrans(Mode);

;

begin btrans

mov ah,0C8h

call arg1t

ret 4

btrans endp

;--------------------------------

; End Transaction func(201) (C9H)

;

; var

; RetCode : Integer;

;

; RetCode := etrans;

;

begin etrans

mov ah,0C9H

int 021h

mov ah,00h

ret

etrans endp

;--------------------------------

; Exc.log File (202) (CA)

;

; var

; RetCode, FcbAddr : Integer;

;

; RetCode := exclog(FcbAddr);

;

begin exclog

mov ah,0CAH

call arg1f

ret 4

exclog endp

;--------------------------------

; Exc Lock Set func(203) (CBH)

;

; var

; RetCode, Mode : Integer;

;

; RetCode := exclcks(Mode);

;

begin exclcks

mov ah,0CBH

call arg1

ret 4

exclcks endp

;--------------------------------

; Exclusive file unlock (204) (CC)

;

; var

; RetCode, FcbAddr : Integer;

;

; RetCode := exculkf(FcbAddr);

;

begin exculkf

mov ah,0CCH

call arg1f

ret 4

exculkf endp

;--------------------------------

; Exclusive Unlock Set (205)(CD)

;

; var

; RetCode : Integer;

;

; RetCode := exculks;

;

begin exculks

mov ah,0CDH

int 21h

mov ah,00h

ret

exculks endp

;--------------------------------

; Exc Clear file func(206) (CEH)

;

; var

; RetCode, FcbAddr : Integer;

;

; RetCode := excclrf(FcbAddr);

;

begin excclrf

mov ah,0CEH

call arg1f

ret 4

excclrf endp

;--------------------------------

; Exc Clear Set func(207) (CFH)

;

; var

; RetCode : Integer;

;

; RetCode := excclrs;

;

begin excclrs

mov ah,0CFH

int 21h

mov ah,00h

ret

excclrs endp

;--------------------------------

; Log Record(s) func(208) (D0H)

;

; var

; RetCode, Flag, Timeout : Integer ;

; RecStr : String[n];

;

; RetCode := reclog(RecStr,Flag,Timeout);

;

begin reclog

push bp

mov bp,sp

push ds

lds si,[bp+x+4] ; load flags

mov ax,[si]

lds si,[bp+x] ; get timeout

mov si,[si]

lds dx,[bp+x+8] ; get ptr to string

; inc dx

mov bp,si ; move timeout to BP

mov ah,0D0h

int 21h

mov ah,0

pop ds

pop bp

ret 14

reclog endp

;--------------------------------

; Record Lock func(209) (D1H)

;

; var

; RetCode, Mode : Integer;

;

; RetCode := reclck(Mode);

;

begin reclck

push bp

mov bp,sp

push ds

lds si,[bp+x]

mov bp,[si]

mov ah,0D1h

int 21h

mov ah,0

pop ds

pop bp

ret 6

reclck endp

;--------------------------------

; Record Unlock func(210) (D2H)

;

; var

; RetCode : Integer;

; Semaphore : String[n];

;

; RetCode := reculk(Semaphore);

;

begin reculk

mov ah,0D2H

call arg1a

ret 4

reculk endp

;--------------------------------

; Unlock Record Set func(211) (D3H)

;

; var

; RetCode : Integer;

;

; RetCode := reculks;

;

begin reculks

mov ah,0D3H

int 21h

mov ah,00h

ret

reculks endp

;--------------------------------

; Record Clear func(212) (D4H)

;

; var

; RetCode : Integer;

; Semaphore : String[n];

;

; RetCode := recclr(Semaphore);

;

begin recclr

mov ah,0D4H

call arg1a

ret 4

recclr endp

;--------------------------------

; Clear record set func(213) (D5H)

;

; var

; RetCode : Integer;

;

; RetCode := recclrs;

;

begin recclrs

mov ah,0D5H

int 21h

mov ah,00h

ret 4

recclrs endp

;--------------------------------

; End of Job func(214) (D6H)

;

; var

; RetCode : Integer;

;

; RetCode := eoj;

;

begin eoj

mov ah,0D6H

int 21h

mov ah,00h

ret 4

eoj endp

;--------------------------------

; Logout System func(215) (D7H)

;

; var

; RetCode : Integer;

;

; RetCode := sysout;

;

begin sysout

mov ah,0D7H

int 21h

mov ah,00h

ret 4

sysout endp

;--------------------------------

; Vol Statistics func(218) (DAH)

;

; var

; RetCode,volume : Integer;

; reply : String[n];

;

; RetCode := volstat(volume, reply);

begin volstat

mov ah,0DAh

push bp

mov bp,sp

push ds

push es

lds si,[bp+x+4] ; addr for the volume number

mov dx,[si] ; get the volume number

les di,[bp+x] ; ptr to reply buffer

inc di ; add 1 to get the address of the string

int 21h

mov ah,00h

pop es

pop ds

pop bp

ret 8

volstat endp

;--------------------------------

; Local Disks func(219) (DBH)

;

; var

; NumDisks : Integer;

;

; NumDisks := locdrv;

;

begin locdrv

mov ah,0DBH

int 21h

mov ah,00h

ret

locdrv endp

;--------------------------------

; WorkStation ID func(220) (DCH)

;

; note: This routine returns the station number as an integer.

; It can also be returned in ASCII with this call.

;

; var

; StationNum : Integer;

;

; StationNum := wsid;

;

begin wsid

mov ah,0DCH

int 21h

mov ah,00h

ret

wsid endp

;--------------------------------

; Set Error mode func(221) (DDH)

;

; var

; mode : Integer;

;

; errmode(mode);

;

begin errmode

mov ah,0DDH

call arg1

ret 4

errmode endp

;--------------------------------

; Broadcast Mode func(222) (DEH)

;

; var

; mode : Integer;

;

; bcsmode(mode);

;

begin bcsmode

mov ah,0DEH

call arg1

ret 4

bcsmode endp

;--------------------------------

; Modify LST Device func(223) (DFH)

;

; var

; RetCode, Mode : Integer;

;

; RetCode := ctlspl(Mode);

;

begin ctlspl

mov ah,0DFh

call arg1

ret 4

ctlspl endp

;--------------------------------

; Spool Request func(224) (E0H)

;

; var

; ErrCode : Integer;

; RequestBlock, Reply : String[n];

;

; ErrCode := splreq(RequestBlock, Reply);

;

begin splreq

mov ah,0E0h

call req\_rep

ret 8

splreq endp

;--------------------------------

; Pipe Request func(225) (E1H)

;

; var

; ErrCode : Integer;

; RequestBlock, Reply : String[n];

;

; ErrCode := pipreq(RequestBlock, Reply);

;

begin pipreq

mov ah,0E1h

call req\_rep

ret 8

pipreq endp

;--------------------------------

; Directory Path func(226) (E2H)

;

; var

; ErrCode : Integer;

; RequestBlock, Reply : String[n];

;

; ErrCode := dpath(RequestBlock, Reply);

;

begin dpath

mov ah,0E2h

call req\_rep

ret 8

dpath endp

;--------------------------------

; Login system func(227) (E3H)

;

; var

; ErrCode : Integer;

; RequestBlock, Reply : String[n];

;

; ErrCode := syslog(RequestBlock, Reply);

;

begin syslog

mov ah,0E3h

call req\_rep

ret 8

syslog endp

;--------------------------------

; Set File Attribute func(228) (E4H)

;

; var

; RetCode, FcbAddr, Attribute : Integer;

;

; RetCode := fattr(FcbAddr, Attribute);

;

begin fattr

push bp

mov bp,sp

push ds

lds si,[bp+x]

mov cx,[si] ; cx = attribute

lds dx,[bp+x+4] ; dx = address of fcb

mov ah,0E4h

int 21h

mov ah,0

pop ds

pop bp

ret

fattr endp

;--------------------------------

; Update file size func(229) (E5H)

;

; var

; ErrCode,FcbAddr : Integer;

;

; ErrCode := updfcb(FcbAddr);

;

begin updfcb

mov ah,0E5H

call arg1f

ret 4

updfcb endp

;--------------------------------

; Copy File to File func(230) (E6H)

;

; var

; RetCode,FcbSource,FcbDest : Integer;

; CountLow, CountHigh : Integer;

;

; RetCode := cpyfile(FcbSource, FcbDest, CountLow, CountHigh)

;

;

begin cpyfile

mov ah,0E6h

push bp

mov bp,sp

push ds

push es

lds si,[bp+x] ; addr of the high byte cnt

mov dx,[si] ; dx=high word of byte cnt

lds si,[bp+x+4] ; addr low word of byte cnt

mov cx,[si] ; cx=low 16 bits of count

les di,[bp+x+8] ; address of dest fcb

lds si,[bp+x+12] ; address of the source fcb

int 21h

mov ah,0

pop es

pop ds

pop bp

ret 16

cpyfile endp

;--------------------------------

; get time / date string func(231) (E7H)

;

; var

; time : String[n];

;

; nettod(time);

;

; where time is a string of the form Y M D H M S

;

begin nettod

mov ah,0E7h

call arg1a

ret 4

nettod endp

;--------------------------------

; Set Close mode func(232) (E8H)

;

; var

; mode : Integer;

;

; clsmode(mode);

;

begin clsmode

mov ah,0E8H

call arg1wor

ret 4

clsmode endp

;--------------------------------

; Shell Base Status func(233) (E9H)

;

; var

; RetFlags, drive : Integer;

;

; RetFlags := drvmap(drive);

;

begin drvmap

mov ax,0E900h

call arg1

ret 4

drvmap endp

;--------------------------------

; Return Shell Version func(234) (EAH)

;

; var

; RetCode, Mode : Integer;

; EnvirStr : String[n];

;

; RetCode := retshl(EnvirStr, Mode);

;

begin retshl

push bp

mov bp,sp

push ds

push es

les di,[bp+x+4] ; set es:di to point at the buffer

inc di ; point di at the start of the data area

lds si,[bp+x] ; put the mode into al

mov ax,[si]

mov ah,0EAh

int 21h

mov ah,00h

pop es

pop ds

pop bp

ret 8

retshl endp

;--------------------------------

; ascii log File (235) (EB)

;

; var

; RetCode, Flags, TimeOut : Integer;

; Asciiz : String[n];

;

; RetCode := asclog(Flags, TimeOut, Asciiz);

;

begin asclog

push bp

mov bp,sp

push ds

lds si,[bp+x+8]

mov ax,[si] ; flags into al

lds dx,[bp+x] ; pointer to the asciiz string

lds si,[bp+x+4]

mov bp,[si] ; time out into bp

mov ah,0EBH

int 21h

mov ah,00h

pop ds

pop bp

ret 12

asclog endp

;--------------------------------

; ASCIIZ file unlock (236) (EC)

;

; var

; RetCode : Integer;

; Asciiz : String[n];

;

; RetCode := asculkf(Asciiz);

;

begin asculkf

mov ah,0ECH

call arg1a

ret 4

asculkf endp

;--------------------------------

; ASCIIZ Clear file func(237) (EDH)

;

; var

; RetCode : Integer;

; Asciiz : String[n];

;

; RetCode := ascclrf(Asciiz);

;

begin ascclrf

mov ah,0EDH

call arg1a

ret 4

ascclrf endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Get Physical Station Number EEH

;

; var

; StationNo : Integer;

;

; StationNo := Get\_PSN;

;

begin Get\_PSN

mov ah,0EEh

int 21h

mov ah,0

ret

Get\_PSN endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Get Shell Table Addresses (239) EFH

;

; var

; Mode,Segment,Offset : Integer;

;

; Get\_STA(Mode,Segment,Offset);

;

begin Get\_STA

push bp

mov bp,sp

push es

push ds

mov si,[bp+x+8] ; get the mode

mov ax,[si]

mov ah,0EFh ; set the function

int 21h

lds di,[bp+x+4] ; store the segment location

mov [di],es

lds di,[bp+x] ; store the offset location

mov [di],si

pop ds

pop es

pop bp

ret 12

Get\_STA endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Set Preferred File Server (240) F0H

;

; var

; RetServer, Mode, NewServ : Integer;

;

; RetServer := SetServ(Mode,NewServ);

;

begin SetServ

push bp

mov bp,sp

push ds

mov si,[bp+x+4] ; get the mode

mov ax,[si]

mov di,[bp+x] ; set the preferred server number

mov dx,[di]

mov ah,0F0h ; set the function

int 21h

mov ah,0

pop ds

pop bp

ret 6

SetServ endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Attach/Detach To File Server (241) F1H

;

; var

; RetCode, Mode, NewServ : Integer;

;

; RetCode := ModServ(Mode,NewServ);

;

begin ModServ

push bp

mov bp,sp

push ds

lds si,[bp+x+4] ; get the mode

mov ax,[si]

lds di,[bp+x] ; set the preferred server number

mov dx,[di]

mov ah,0F1h ; set the function

int 21h

mov ah,0

pop ds

pop bp

ret 6

ModServ endp

;-------------------------------

; Place a single integer argument into dx

; with a return code

arg1 proc near

push bp

mov bp,sp

push ds

lds si,[bp+x+2]

mov dx,[si] ; dx gets the argument

int 21h

mov ah,0 ; clear ah for al return code

pop ds

pop bp

ret

arg1 endp

;--------------------------------

; Place a single integer argument into bp

; for use with time outs - with a return code

arg1t proc near

push bp

mov bp,sp

push ds

lds si,[bp+x+2]

mov bp,[si] ; dx gets the argument

int 21h

mov ah,0 ; clear ah for al return code

pop ds

pop bp

ret

arg1t endp

;--------------------------------

; Process an fcb address in a

; function call with a return code

arg1f proc near

push bp

mov bp,sp

push ds

lds dx,[bp+x+2] ; dx gets the argument

int 21h

mov ah,0 ; clear ah for al return code

pop ds

pop bp

ret

arg1f endp

;--------------------------------

; Process a single string argument in a

; function call with a return code

arg1a proc near

push bp

mov bp,sp

push ds

lds dx,[bp+x+2] ; ptr to parameter string

; inc dx ; add 1 to get the address of the string

int 21h

mov ah,0 ; clear ah for al return code

pop ds

pop bp

ret

arg1a endp

;----------------------------

; Perform a single integer argument

; but without a return code

arg1wor proc near

push bp

mov bp,sp

push ds

lds si,[bp+x+2]

mov dx,[si] ; dx gets the arg (or ptr)

int 21h

pop ds

pop bp

ret

arg1wor endp

;----------------------------

; Perform a single string argument

; but without a return code

arg1awor proc near

push bp

mov bp,sp

push ds

lds dx,[bp+x+2]

inc dx ; add 1 to get the start of the string

int 21h

pop ds

pop bp

ret

arg1awor endp

;----------------------------

; Handle a request$ and reply$ function

; with error code

req\_rep proc near

push bp

mov bp,sp

push ds

push es

lds si,[bp+x+6] ; ptr to req. string

inc si ; add 1 to get the address of the string

les di,[bp+x+2] ; ptr to var desc block for reply buf

inc di ; add 1 to get the address of the string

pushf ; save the direction flag

int 21h

popf

mov ah,0

pop es

pop ds

pop bp

ret

req\_rep endp

cseg ends

end

6. В И Р У С Ы И N E T W A R E

При работе сети NetWare фирмы Novell для совместного использо-

вания сетевого дискового пространства используется "прозрачный ре-

жим". Пользователь за комьютером видит сетевые диски в виде допол-

нительных винчестеров (F: G: ...). Особенностью этих винчестеров

является то, что для них поддерживаются все операции над файлами и

не поддерживаются операции пямого доступа к секторам диска. Драй-

вера устойств для этих дополнительных букв не существуют и все

операции над ними иммитируются путем перехвата различных прерыва-

ний.

Проблема поведения вирусов на сетевых дисков сразу же распада-

ется на две в зависимости от ответа на вопрос: выполняется ли ви-

рус на машине управляющей сетью (и содержащей сетевые диски) или

нет.

Если вирус выполняется на управляющей машине, то он имеет дост-

уп в MBR, BOOT и к секторам диска. Пока нет подтвержденной инфор-

мации о существовании вирусов знакомых со структурой диска

NetWare. Попытка "не грамотного" вируса записаться на диск NetWare

и в MBR скорее всего окончится плачевно. И не сколько для вируса,

сколько для NetWare. Поэтому для защиты от BOOT-вирусов рекоменду-

ется сохранять 0-ю дорожку. На этой дорожке содержется не только

MBR, но и начало раздела NetWare. Восстановление 0-й дорожки из

копии не всегда, наверное, возможно, но в ряде случаев это помога-

ет. В случае неуспеха еще остается тонкий анализ содержимого ста-

рой и поврежденной 0-й дорожки. Если такой анализ удастся провес-

ти, то возможно ручное восстановление потеянной информации.

Если на управляющей машине вместе с сетью работает DOS, то воз-

можно нарушение работы NetWare в памяти и, как следствие, разнооб-

разные курьезы. Поэтому к чистоте от вирусов управляющей машины

локальной сети следует относится особенно серьезно.

Если вирус выполняется на другой машине, то он не имеет физи-

ческого доступа к диску. Разговор в этом случае может идти только

о файловых вирусах. При этом нужно учесть, что пользователь сети

обычно не имеет доступа на запись ко всем файлам (особенно систем-

ным). Возможность превысить права доступа практически не возможна.

Покрайней мере пока в системе защиты нет явных проколов админист-

ратора сети и не известны "люки" в защите NetWare.

Поведение вирусов в NetWare распадается на три группы: - те кто

успешно функционирует в сети; - те кто заражает файлы на локальных

дисках, но отказывается заpажать файлы в сети (среди них есть та-

кие которые самоочищаются при копировании в сеть зараженного файла

при активном вирусе); - и те кто теряет работоспособность при ак-

тивной сети.

Распределение вирусов по этим группам можно обьяснить следующи-

ми обстаятельствами. В первую группу попадают наиболее примитивные

файловые вирусы котоые законно используют функции DOS для своих

целей. Такие фирусы довольно легко отлавливаются различными сторо-

жевыми средствами. Во вторую группу попадают вирусы которые пыта-

ются бороться с антивирусными средствами. Сеть при этом они также

обходят и поэтому работа с сетью для них не видна. В эту группу

попадает наверное большинство "СТЕЛС"-вирусов. Третью группу сост-

авляют неудачно написанные вирусы из второй группы. Они перестают

работать из-за конфликтов с оболочкой NetWare.

Список литературы.

1. Aнгло-русский словарь терминов NetWare. (13 cтр.)

Чусов В.Е.

2. Англо-русский глоссарий терминов NetWare. (66 стр.)

Чусов В.Е.

3. Введение в локальные вычислительные сети. (45 стр.)

Груздев А.Ю.

4. Texничecкиe cpeдcтвa лoкaльныx ceтeй. (27 стр.)

Гaвpилoв A.B.

5. Введение в сетевые операционные системы NetWare

фирмы Novell. (55 стр.) Груздев А.Ю.

6. Концепции построения и основные особенности системы

NetWare 386. (49 стр.) Груздев А.Ю.

7. Установка NetWare 386. (86 стр.) Гордеев В.В.

8. Установка рабочей станции DOS ODI в сети NetWare 386.

(26 стр.) Гордеев В.В.

9. Cтpyктypa кaтaлoгa, cтpyктypa пoльзoвaтeлeй и cиcтeмa

зaщиты ceти NetWare (62 cтp.) Чycoв B.E.

10. Интерфейс пользователя с системой.

Утилиты меню и утилиты командной строки NetWare 386.

(152 стр.) Шабалин А.Р.

11. Печать в сетевой операционной системе NetWare 386.

(83 стр.) Милентьев А.Д

12. Оперативное управление работой сетевой ОС NetWare 386.

(51 стр.) Поваров А.В.

13. Средства построения гетерогенных комплексов

фирмы Novell. (42 стр.) Коровкин С.Д.

14. Коммуникационные средства фирмы Novell.

(47 стр.) Коровкин С.Д.

15. Проектирование, создание и использование баз данных,

реализованных с применением продуктов фирмы Novell.

(79 стр.) Зимин В.В.

16. Практическое сравнение различных средств управления

данными в среде сетевой ОС NetWare фирмы Novell.

(41 стр.) Зимин В.В.

17. Использование FoxBase+ и Clipper в локальной сети.

(17 стр.) Нестеренко Ю.П.

18. Барри Нанс, Программирование в локальных сетях