МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего
профессионального образования

“Оренбургский государственный университет”

Кафедра вычислительной техники

Ю.И. СИНИЦЫН, В.А. МЕДВЕДЕВ, Р.Р. ГАЛИМОВ

СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Рекомендовано к изданию редакционно-издательским советом государственного образовательного учреждения высшего профессионального образования “Оренбургский государственный университет”

Оренбург 2005

УДК 004.7(07)
ББК 32.973.202я7
С 38

Рецензент
кандидат технических наук, доцент Ю.Н.Пивоваров

Синицын Ю.И.
С 38 Методические указания к лабораторным работам по курсу “Сети ЭВМ и телекоммуникации” / Ю.И. Синицын, В.А. Медведев, Р.Р. Галимов. – Оренбург: ГОУ ОГУ, 2005. – 64 с.

Методические указания предназначены для выполнения лабораторных работ по курсу “Сети ЭВМ и телекоммуникации” для студентов четвертого курса по специальности 220100

Содержание

1 Лабораторная работа № 1. Протокол IPX/SPX

Цель работы: Получить основные теоретические сведения о протоколу IPX/SPX

Теоретическая справка.

Протокол IPX NetWare является реализацией протокола обмена межсетевымипакетами фирмы Xerox. Целью IPX является предоставление возможности прикладным системам, работающим в рабочих станциях NetWare, использовать драйверы NetWare для установления прямой связи с другими рабочими станциями, серверами или другими устройствами в объединенной сети (интерсети). IPX является сервисом, который обеспечивает прикладным системам возможность передавать и принимать пакеты информации по интерсети. Структура пакетов определена сетевым стандартом Xerox (XNS). В сетевом оборудовании NetWare (сеть включает одну или более сетей, использующих NetWare) каждый узел (устройство, подключенноу к сети) имеет уникальный интерсетевой адрес. Используя IPX, рабочая станция NetWare может передать/принять пакет к/от любого узла интерсети. Маршрутизация пакетов, источник и приемник которых расположены в физически разных локальных сетях, является в значительной степени автоматической и прозрачной. Эта прозрачность достигается средствами IPX в совокупности с сервисом, предоставляемым мостами NetWare.

Сетевые драйверы реализуют физическую доставку пакетов, но не гарантируют их доставку. Реализация надежной доставки, протоколов передачи данных и других протоколов взаимодействия может быть построена на основе протоколов IPX, в соответствии с требованиями различных прикладных программ /1/.

Протокол IPX предназначен для использования как основа для построения сложных прикладных систем, таких как коммуникационные серверы, концентраторы. IPX полностью поддерживается на всех топологиях локальных сетей, которые поддерживаются NetWare V2.0.

Асинхронный планировщик событий (AES) является вспомогательным сервисом, который обеспечивает средства для измерения прошедшего времени и/или запуска события по окончании отмеряемых временных интервалов.

Событиями являются: завершение требования передачи IPX, прием пакета при завершении требования просмотра IPX, окончание временного интервала, определенного прикладной программой. Каждому событию может соответствовать сервисная программа события, которая может вызываться, когда событие произойдет. При необходимости, сервисная программа события может перепланировать себя для выполнения по истечении определенного интервала времени.

Структура пакетов IPX полностью соответствует структуре пакетов протокола XNS Xerox.

Основная структура пакетов IPX.

Пакет IPX содержит 30 байтов заголовка, за которыми следуют от 0 до 546 байтов данных. Таким образом, минимальная длина пакета будет 30 байтов и максимальная длина пакета будет равна 576 байтов.

Содержимое и структура части пакета, в которой содержаться данные, находится полностью в введении прикладных программ, использующих IPX, и может быть любой требуемой структуры /5/.

Таблица 1 - Структура заголовков пакетов IPX

Начальный байт	Поле	Размер	Тип данных
0	Контрольная сумма	2 байта	Старшее-младшее целое
2	Длина	2 байта	Старшее-младшее целое
4	Транспортный контроль	1 байт	Целое без знака
5	Тип пакета	1 байт	Целое без знака
6	Сеть назначения	4 байта	Старшее-младшее целое без знака
10	Узел назначения	6 байт	Старшее-младшее целое без знака
16	Сокет назначения	2 байта	Старшее-младшее целое без знака
18	Сокет источника	4 байта	Старшее-младшее целое без знака
22	Узел источника	6 байт	Старшее-младшее целое без знака
28	Сокет источника	2 байта	Старшее-младшее целое без знака

Численные поля, состоящие больше чем из одного байта, могут размещаться двумя способами: старшее-младшее и младшее-старшее. При размещении старшее-младшее самый старший значащий байт размещается в первом байте поля, самый младший – в последнем байте поля. При размещении младший-старший байты хранятся в противоположном порядке.

Описание полей заголовка.

Контрольная сумма. Это поле содержит контрольную сумму 16-ти битовых слов заголовка пакета. Это поле будет содержать -0 (FFFFh), если контрольная сумма не требуется. Если контрольная сумма равна -0, тогда это поле должно содержать +0 (0000h).

Это поле содержит контрольную сумму только 30-ти байтового заголовка, если прикладной программе необходимо иметь контрольную сумму тогда она должна обеспечить контрольную сумму своих данных и разместить их в области данных.

В данной реализации оболочки NetWare контрольная сумма может как проверяться так и нет. Если проверка контрольной суммы необходима, она должна обеспечиваться прикладной программой, которой приходит пакет /1/.

Длина. Это поле содержит полную длину сетевого пакета, которую составляет длина заголовка плюс длина поля данных. Таким образом,

минимальное значение этого поля равно 30, максимальное значение равно 576.

Транспортный контроль. Это поле используется межсетевыми мостами NetWare и должно быть установлено в 0 перед передачей пакета.

Тип пакета. Это поле определяет тип услуг, которые предлагаются или требуются пакету. Xerox определяет следующие типы пакетов:

0 - неопределенный тип пакета;

1 - пакет информации маршрутизации;

2 - эхо пакет;

3 - пакет ошибок;

4 - пакет обмена;

5 - пакет последовательного протокола;

16-31 - экспериментальные протоколы.

Все пакеты пользователей должны быть либо типа 0, либо типа 4.

Сеть назначения. Это поле содержит номер сети, в которой может быть расположен узел, которому предназначается данный пакет.

В NetWare сетям, объединяемым в интерсеть администратором сети назначаются 4-х байтовые адреса. Каждая сеть в интерсети имеет уникальный адрес.

Если это поле устанавливается в 0, предполагается, что узел-приемник расположен в той же физической сети, что и узел источник; пакет в этом случае будет передаваться без обращения к межсетевому мосту NetWare.

Узел назначения. Это поле содержит 6-ти байтовый номер, который идентифицирует физический адрес (в сети назначения) узла, которому предназначается пакет.

Не все сети используют одинаковое адресное пространство. Так в сети EtherNet используются все 6 байтов под адрес, в то время как в сети OmniNet используется для адресации только один байт.

Если в данной физической сети для адресации требуется менее 6 байтов, то необходимая часть адреса должна занимать последние байты поля, первые байты поля должны быть установлены в 0.

Установка всех 6 байтов этого поля в FFh показывает что данный пакет является широковещательным для сети назначения.

Поддерживается или нет широковещательный пакет зависит от физических характеристик той сети, в которую пакет доставляется.

Сокет назначения. Это поле содержит адрес сокета системной программы, которой предназначается пакет.

Номера сокетов используются для маршрутизации пакетов к различным программам в данном узле /1/.

Xerox резервирует следующие сокеты для специальных целей:

1 - пакет информации маршрутизации;

2 - пакет эхо-протокола;

3 - пакет обработки ошибок;

32-63 - экспериментальные.

Сокеты с номерами меньшими 3.000 (десятичное значение) считаются статически назначаемыми сокетами, сокеты с номерами большими 3.000 считаются динамически назначаемыми сокетами.

Сеть источник. Это поле содержит номер сети, в которой находится узел -источник пакета. В этом поле может быть установлено значение 0. Если пакет передается через межсетевой мост NetWare, нулевое значение в этом поле будет заменено на реальный номер сети-источника. Только пакеты, передаваемые в рамках одной физической сети могут содержать нулевое значение в этом поле.

Узел источник. Это поле содержит физический адрес узла, из которого па-кеты передаются. Количество байт этого поля, используемых для адресации данного узла зависит от физической сети, в которую этот узел входит. Это поле формируется аналогично полю узел назначения, описанному выше.

Сокет источника. Это поле содержит адрес сокета, используемый прикладной программой, которая выдает пакет. Хотя это не требуется протоколами IPX, обычно для всех станций, взаимодействующих по отдельной задаче на равных правах, для передачи и приема используются одинаковые номера сокетов. В ситуации клиент/сервер, узел, работающий как сервер (возможно как коммуникационный шлюз), должен просматривать определенный сокет для обслуживания требований на прием. В этом случае номер сокета источника не обязательно будет таким же.

События и блоки управления событиями.

Блоки управления событиями (ECB) - это структура данных, которая содержит информацию, требуемую для координации планирования и/или активизации определенных операций. Почти все функции IPX или AES работают, используя ECB и/или информацию, которую они содержат. Существует два различных типа событий, которые могут произойти: события, относящиеся к операциям приема или передачи IPX и специальные события, определяемые прикладными процессами.

Например, когда прикладной процесс хочет передать пакет IPX, он должен подготовить ECB, в котором указывает, где содержаться данные для пакета. Завершенный процесс передачи пакета составляет "событие передачи IPX".

Аналогично, когда прикладному процессу требуется принимать пакеты, он должен создать один или более ECB, в которых будет указано, куда размещать принимаемые данные. Прием пакета вызовет "событие приема IPX".

Кроме того, прикладной процесс может определить и спланировать "специальное" событие для своих целей. Блок управления событием должен обеспечиваться планировщиком события в течении определенного интервала времени. AES будет уменьшать время в данном интервале, когда таймер достигнет нуля, произойдет планируемое событие.

Существует два способа, которыми прикладная система может определить и задействовать события: опрашивая флаг "в использовании" ("in

use") или обеспечивая сервисную программу события (ESR), которая может быть вызвана в режиме прерывания /1/.

Каждый ECB имеет флаг статуса "in use", который устанавливается в ненулевое состояние как только ECB передается функциям IPX или AES для обеспечения требования события (блок находится "в использовании"). Прикладная программа не должна изменять содержимое блока управления событием, пока флаг его статуса имеет ненулевое значение.

Когда IPX или AES заканчивают выполнение требуемой функции, включающей в себя данный ECB, флаг "in use" этого ECB будет автоматически устанавливаться в 0. После этого прикладная программа может изменять содержимое ECB и любую, связанную с ним информацию.

Второй способ заключается в том, что каждый блок управления событием определяет адрес сервисной программы события, которая будет вызываться в режиме прерывания, когда событие произойдет.

Структура блока управления событием для функций IPX несколько отличается от структуры блока управления событием для AES, однако, одинаковый формат первых нескольких полей обоих типов ECB позволяет использовать оба типа ECB асинхронным планировщиком событий. Таким образом, сервисная программа события, вызываемая событием IPX может использовать связанные с этим событием данные для перепланирования самой себя с помощью планировщика событий /1/.

ECB для событий IPX.

Блок управления событием для событий IPX ("ECB-IPX") имеет структуру, показанную в таблице 2.

Таблица 2 – Структура блока управлением событием

Начальный байт	Поле	Размер	Тип данных
0	Линия	4 байта	Длинный адрес 8086 (смещение, сегмент)
4	Адрес ESR	4 байта	Длинный адрес 8086 (смещение, сегмент)
8	"in use"	1 байт	Целое без знака
9	Код завершения	1 байт	Целое без знака
10	Номер сокета	2 байта	Старшее-младшее целое без знака
12	Рабочая область	4 байта	Не определено IPX
16	Рабочая область	12 байт	Не определено драйверов
28	Непосредственный	6 байт	Старшее-младшее целое без знака
34	Количество фрагментов	2 байта	Старшее-младшее целое без знака

Как видно из приведенной структуры, блок управления событием состоит из двух частей. Первая, фиксированная (36 байт длиной), содержит

поля статуса/информации, а также рабочую область для использования IPX и сетевыми драйверами. Вторая часть, переменной длины, является списком одного или более описателей фрагментов. Каждый описатель включает адрес и размер области памяти (фрагмента), из которой будет браться пакет для передачи или куда будет записываться пакет после приема /5/.

Описание полей IPX-ECB.

Линия. Это поле используется IPX, когда ECB имеет статус "in use". Если ECB не находится в режиме "in use", прикладная система может использовать это поле для своих целей. В наиболее общем случае, это поле может использоваться как связное для хранения ECB в списках и очередях.

Адрес ESR. Это поле содержит адрес определенной прикладной системой сервисной программы, которая будет вызываться IPX, когда ожидаемое событие (передача пакета или прием пакета) завершиться.

Поскольку IPX использует поля "in use" и "код завершения", прикладная программа может определять статус своих требований опросом в определенные моменты времени состояния этих полей и не использовать сервисную программу события. В этом случае поле "адрес ESR" должно быть установлено в ноль (четыре байта нулей).

"In use". Это поле имеет ненулевое значение в том случае, когда ECB используется IPX или AES. В этом случае это поле может иметь следующие значения:

FFh - ECB используется для передачи пакета;

FEh - ECB используется для просмотра определенного сокета при ожидании пакета;

FDh - ECB используется с AES и ожидает окончания временного интервала;

FBh - событие передачи или приема уже произошло, но ECB временно хранится в очереди, ожидая обработки.

IPX будет устанавливать это поле в ноль, когда обработка будет закончена.

Код завершения. Это поле устанавливается программами IPX и отражает результат обработки требования IPX. Это поле не может считаться действительным до тех пор пока IPX не установит флаг статуса "in use" в ноль.

Номер сокета. Это поле содержит номер сокета, с которым связан ECB. Если ECB используется для передачи, это поле содержит номер сокета, из которого передается пакет. Если ECB используется для приема, это поле содержит номер сокета, в который будет помещаться принятый пакет.

Рабочая область IPX. Это поле резервируется для использования стандартными программами IPX. Это поле может не устанавливаться в начальное состояние, но оно не должно изменяться пока блок управления событием используется программами IPX. Когда ECB не используется IPX, эта поле может использоваться для каких-либо других целей.

Рабочая область драйвера. Это 12-ти байтовое поле резервируется для использования сетевыми драйверами. Оно может не устанавливаться в какое-

то начальное значение, но оно не должно изменяться пока блок управления используется программами IPX. Когда ECB не используется IPX, это поле может использоваться для других целей.

Непосредственный адрес. Это поле содержит адрес узла, которому был только что передан пакет или от которого только что прибыл пакет (это будет адрес межсетевого моста, если пакет передавался или был принят от узла, расположенного вне локальной сети).

Количество фрагментов. Это поле содержит количество буферных фрагментов, из которых пакет будет формироваться для передачи или количество буферов, в которых принятый пакет должен быть размещен.

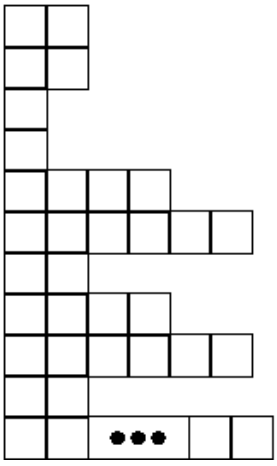
Значение этого поля должно быть больше 0. Значение 1 показывает, что пакет передается/принимается из/в смежную область памяти, то есть один фрагмент содержит целый пакет.

Список описателей фрагмента. Описатель фрагмента точно определяет, где размещена часть пакета, который должен быть передан или где расположена часть принятого пакета. Описатель фрагмента имеет два поля:

адрес - адрес буфера, из которого данные должны быть взяты при формировании пакета для передачи или в который данные должны быть помещены при приеме пакета;

размер - размер буферного фрагмента, указанного предшествующим адресом.

Любой блок управления событием может иметь произвольное число описателей фрагмента но не менее одного. Эти описатели составляют список описателей фрагмента. Необходимо, чтобы буферный фрагмент, определенный первым в списке описателей фрагментов должен быть по крайней мере 30 байт длиной и должен включать полный заголовок пакета IPX. Полный размер пакета (сумма размеров отдельных фрагментов) не должен превышать 576 байт /1/. Формат передаваемых по сети пакетов представлен на рисунке 1.



Checksum - контрольная сумма
Length - общая длина пакета
TransportControl - счетчик пройденных мостов
PacketType - тип пакета
DestNetwork - номер сети получателя пакета
DestNode - адрес станции-получателя
DestSocket - сокет программы-получателя
SourceNetwork - номер сети отправителя пакета
SourceNode - адрес станции-отправителя
SourceSocket - сокет программы-отправителя
Data - передаваемые данные, от 0 до 546 байт

Рисунок 1 - Формат передаваемых по сети пакетов

Пакет можно разделить на две части - заголовок и передаваемые данные. Все поля, кроме последнего (Data), представляют собой заголовок пакета.

Поле Checksum предназначено для хранения контрольной суммы передаваемых пакетов. При формировании собственных пакетов вам не придется заботиться о содержимом этого поля, так как проверка данных по контрольной сумме выполняется драйвером сетевого адаптера.

Поле Length определяет общий размер пакета вместе с заголовком. Длина заголовка фиксирована и составляет 30 байт. Размер передаваемых в поле Data данных может составлять от 0 до 546 байт, следовательно, в поле Length в зависимости от размера поля Data могут находиться значения от 30 до 576 байт. Если длина поля Data равна нулю, пакет состоит из одного заголовка. Как это ни странно, такие пакеты тоже нужны. При формировании собственных пакетов не надо проставлять длину пакета в поле Length, протокол IPX сделает это сам (вернее, программный модуль, отвечающий за реализацию протокола IPX, вычислит длину пакета на основании длины поля Data).

Поле TransportControl служит как бы счетчиком мостов, которые проходит пакет на своем пути от передающей станции к принимающей. Каждый раз, когда пакет проходит через мост, значение этого счетчика увеличивается на единицу. Перед передачей пакета IPX сбрасывает содержимое этого поля в нуль.

Поле PacketType определяет тип передаваемого пакета. Программа, которая передает пакеты средствами IPX, должна установить в поле PacketType значение 4. Протокол SPX, реализованный на базе IPX, использует в этом поле значение 5.

Поле DestNetwork определяет номер сети, в которую передается пакет. При формировании собственного пакета вам необходимо заполнить это четырехбайтовое поле.

Поле DestNode определяет адрес рабочей станции, которой предназначен пакет. Вам необходимо определить все шесть байт этого поля.

Поле DestSocket предназначено для адресации программы, запущенной на рабочей станции, которая должна принять пакет. При формировании пакета вам необходимо заполнить это поле.

Поля SourceNetwork, SourceNode и SourceSocket содержат соответственно номер сети, из которой посылается пакет, адрес передающей станции и сокет программы, передающей пакет.

Поле Data в пакете IPX содержит передаваемые данные. Как мы уже говорили, длина этого поля может быть от 0 до 546 байт. Если длина поля Data равна нулю, пакет состоит из одного заголовка. Такой пакет может использоваться программой, например, для подтверждения приема пакета с данными.

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 самого 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.

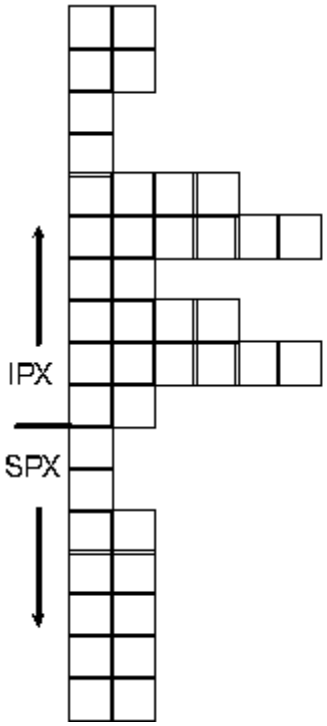
Если программе требуется измерять временные интервалы, она может воспользоваться асинхронным планировщиком событий AES, реализованным в рамках драйвера IPX.

Для функций AES можно использовать тот же формат ECB, что и для функций IPX. Однако поля используются немного по-другому.

Для некоторых приложений (например, для программ, передающих файлы между рабочими станциями) удобнее использовать сетевой протокол более высокого уровня, обеспечивающий гарантированную доставку пакетов в правильной последовательности /1/.

Протокол SPX - протокол последовательного обмена пакетами (Sequenced Packet Exchange Protocol), разработанный Novell.

Пакет, передаваемый при помощи протокола SPX, имеет более длинный заголовок. Дополнительно к 30 байтам стандартного заголовка пакета IPX добавляется еще 12 байт /4/.



Checksum - контрольная сумма
Length - общая длина пакета
TransportControl - счетчик пройденных мостов
PacketType - тип пакета
DestNetwork - номер сети получателя пакета
DestNode - адрес станции-получателя
DestSocket - сокет программы-получателя
SourceNetwork - номер сети отправителя пакета
SourceNode - адрес станции-отправителя
SourceSocket - сокет программы-отправителя
ConnControl - управление потоком данных
DataStreamType - тип данных в пакете
SourceConnID - идентификатор канала отправителя
DestConnID - идентификатор канала получателя
SeqNumber - счетчик переданных пакетов
AckNumber - номер следующего пакета
AllocNumber - количество буферов для приема

Рисунок 2 – Структура пакета протокола SPX

Выполнение работы:

- изучить назначение протокола IPX/SPX;
- изучить основную структура пакета IPX;
- изучить структуру заголовка пакета IPX;
- изучить описание полей заголовка IPX;

– изучить структуру блока управлением событием;
– изучить описание полей ECB;
– изучить формат передаваемых по сети пакетов;
– изучить основную структура пакета SPX;
– оформить отчет.

Отчет по лабораторной работе должен содержать следующее:
– название и цель работы;
– основные теоретические пункты общий сведений;
– выводы по выполненной работе;
– список использованных источников.

Контрольные вопросы:
– назначение протокола IPX/SPX?;
– назначение сетевых драйверов?;
– назначение асинхронного планировщика событий (AES)?;
– что является событиями?;
– что представляет собой основная структура пакетов IPX?;
– из каких полей состоит структура заголовка пакета IPX?;
– что такое сокет и его назначение?;
– какие сокеты используются в протоколе ?;
– назначение узла приемника и узла передатчика?;
– что представляют собой «События и блоки управления событиями»?;
– что представляет собой «Структура блока управлением событием»?;
– какие поля используются в «Структуре блока управлением событием»?;
– что такое «Рабочая область IPX» ?;
– что такое «Рабочая область драйвера» ?;
– что такое «Непосредственный адрес» ?;
– что такое «Количество фрагментов» ?;
– что такое «Список описателей фрагмента» ?;
– что представляет собой формат протокола IPX ?;
– что представляет собой формат протокола SPX ?.

2 Лабораторная работа № 2. Протокол NETBIOS

Цель работы: Получить основные теоретические сведения о протоколе NETBIOS.

Теоретическая справка.

В NetWare протокол NETBIOS является надстройкой над протоколом IPX и используется для организации обмена данными между рабочими станциями.

Протокол NETBIOS реализован в виде резидентной программы NETBIOS.EXE, входящей в комплект поставки NetWare. Для обмена данными между этими резидентными программами используются пакеты IPX с номером гнезда 0x0455 и типом пакета 20.

Для идентификации рабочей станции протоколы IPX и SPX используют номер сети, адрес станции в сети и номер гнезда. Адрес станции определяется на аппаратном уровне и представляет собой число длиной 6 байтов. Номер сети занимает 4 байта. Номер гнезда выделяе тся динамически протоколом IPX или может быть получен в фирме Novell. Номер гнезда занимает 2 байта /1/.

Протокол NETBIOS использует другой механизм адресации станций и программ. Для адресации станций используются имена размером 16 байтов. Каждая станция имеет одно постоянное имя (permanent name), которое образуется из аппаратного адреса добавлением к нему с лева десяти нулевых байтов. Кроме постоянного имени имеются обычные имена и групповые имена, которые протокол NETBIOS позволяет добавлять (и удалять). Обычные имена служат для идентификации рабочей станции, групповые имена могут служить для посылки пакетов одновременно нескольким станциям в сети. Постоянное имя удалять нельзя, так как оно полностью определяется аппаратным обеспечением станции.

При добавлении обычного имени протокол NETBIOS опрашивает всю сеть для проверки уникальности имени. Групповое имя может быть одинаковым для нескольких станций, поэтому при добавлении группового имени опрос сети не выполняется /1/.

После добавления новому имени присваивается так называемый номер имени (name number), который используется для передачи данных по сети.

Сравнивая методы адресации, используемые протоколами IPX/SPX и NETBIOS, можно заметить, что метод адресации протокола NETBIOS более удобен. Вы можете адресовать данные не только одной станции (как в IPX и SPX) или всем станциям сразу (как в IPX), но и гру ппе станций, имеющим одинаковое групповое имя /1/.

Чтобы выполнить функцию NETBIOS, в прикладной программе необходимо:

– заполнить поля блока NCB (Network Control Block);

– загрузить в регистр ES:BX дальний адрес блока NCB;

– вызвать программное прерывание 5C (INT 5C).

Назначение полей блока NCB.

Поле *Cmd* содержит код команды, которую необходимо выполнить.

Поле *CCode* содержит код ошибки, возвращаемый после проверки параметров до выполнения команды.

Поле *LocalSessionNumber* содержит номер канала, установленного с другой программой. Оно используется только при выдаче команд передачи данных через каналы.

Поле *NetworkNameNumber* содержит номер имени, который присваивается при добавлении обычного или группового имени. Это поле должно быть заполнено при приёме датаграмм.

Поле *Buffer* представляет собой дальний указатель на буфер, который должен содержать данные перед выполнением передачи, или на буфер, который будет использован для приёма данных.

Поле *Size* определяет размер буфера, используемого для приёма или передачи данных.

В поле *CallName* указывается имя станции-получателя.

Поле *OurName* содержит имя станции-отправителя. Обычно используется в командах создания имени станции или создания канала.

Поля *ReceiveTimeout* и *SendTimeout* содержат интервал времени (измеряемый в 1/2с), в течение которого ожидается завершение соответственно команд приёма и передачи.

Поле *PostRoutine* - ноль или указатель на программу (POST-программу), которая получает управление после завершения команды.

Поле *AdapterNumber* используется, если на рабочей станции установлено несколько сетевых адаптеров (в сетях Ethernet этого обычно не бывает). В этом поле указывается номер адаптера, для которого предназначена команда. Первый адаптер имеет номер 0, вт орой -1 и т. д.

Поле *FinalCCode* содержит во время выполнения команды значение 0xFF. После завершения выполнения команды в это поле записывается ноль или код ошибки, который относится к выполнению команды в целом (в отличие от кода в поле CCode).

Поле *Reserved* зарезервировано для использования протоколом NETBIOS.

Перед выполнением команды её код должен быть записан в поле Cmd блока NCB. Каждая команда NETBIOS реализована в двух вариантах: с ожиданием и без ожидания окончания выполнения команды /4/.

Все команды NETBIOS можно разделить на несколько групп:

1. Для работы с именами:

– 0x30, 0xB0 - добавить новое имя в таблицу имён станции (с ожиданием и без ожидания);

– 0x36, 0xB6 - добавить новое групповое имя в таблицу станции;

– 0x31, 0xB1 - удалить имя из таблицы имён станции.

2. Для приёма и передачи датаграмм:

– 0x20, 0xA0 - передать одной или группе станций блок данных в виде датаграммы;

– 0x22, 0xA2 - передать всем станциям блок данных в виде датаграммы;

– 0x21, 0xA1 - принять блок данных, переданный с помощью команды 0x20 или 0xA0;

– 0x23, 0xA3 - принять блок данных, переданный с помощью команды 0x22 или 0xA2.

3. Для работы с каналами (для приема и передачи данных):

– 0x10, 0x90 - установить канал между двумя именами, заданными в блоке NCB;

– 0x11, 0x91 - организовать канал с вызываемой стороны (работают в паре с командами 0x10 или 0x90);

– 0x12, 0x92 - закрыть канал;

– 0x34, 0xB4 - опросить состояние канала;

– 0x14, 0x94 - передать блок данных (до 64 Кб) по каналу;

– 0x71, 0xF1 - передать блок данных (до 64 Кб) по каналу без проверки доставки блока;

– 0x17, 0x97 - передать два буфера (каждый по 64 Кб) по каналу как один блок;

– 0x72, 0xF2 - передать два буфера (каждый по 64 Кб) по каналу как один блок без проверки доставки блока;

– 0x15, 0x95 - принять блок данных, переданный по каналу;

– 0x16, 0x96 - принять блок данных, переданный по любому каналу, который организовала принимающая станция.

5. Другие команды:

– 0x32 - удалить все имеющиеся каналы и имена;

– 0x35 - отменить ранее запущенную команду.

Выполнение работы:

- изучить назначение протокола NETBIOS;
- изучить основную структура пакета NETBIOS;
- изучить команды NETBIOS;
- изучить описание полей блока NCB (Network Control Block);
- оформить отчет.

Отчет по лабораторной работе должен содержать следующее:

- название и цель работы;
- основные теоретические пункты общий сведений;
- выводы по выполненной работе;
- список использованных источников.

Контрольные вопросы:

– назначение протокола NETBIOS?;
– как устроен механизм адресации в протоколе NETBIOS?;
– что делает прикладная программа для того, чтобы выполнить функцию NETBIOS?;
– какие поля содержит блок NCB (Network Control Block)?;
– как реализована каждая команда NETBIOS?;
– на какие группы делятся команды NETBIOS?.

3 Лабораторная работа № 3. Стек TCP/IP

Цель работы: Получить основные теоретические сведения по стеку TCP/IP.

Теоретическая справка.

Важнейшим направлением стандартизации в области вычислительных сетей является стандартизация коммуникационных протоколов. В настоящее время в сетях используется большое количество стеков коммуникационных протоколов. Наиболее популярны следующие стеки:

– TCP/IP;
– IPX/SPX;
– NetBIOS/SMB;
– DECnet;
– SNA;
– OSI.

Все эти стеки, кроме SNA на нижних уровнях — физическом и канальном, — используют одни и те же хорошо стандартизованные протоколы Ethernet, Token Ring, FDDI и ряд других, которые позволяют задействовать во всех сетях одну и ту же аппаратуру. Зато на верхних уровнях все стеки работают по своим протоколам. Эти протоколы часто не соответствуют рекомендуемому моделью OSI разбиению на уровни. В частности, функции сеансового и представительного уровня, как правило, объединены с прикладным уровнем. Такое несоответствие связано с тем, что модель OSI появилась как результат обобщения уже существующих и реально используемых стеков, а не наоборот.

Стек TCP/IP был разработан по инициативе Министерства обороны США более 20 лет назад для связи экспериментальной сети ARPAnet с другими сетями как набор общих протоколов для разнородной вычислительной среды. Стек TCP/IP на нижнем уровне поддерживает все популярные стандарты физического и канального уровней: для локальных сетей — это Ethernet, Token Ring, FDDI, для глобальных — протоколы работы на аналоговых коммутируемых и выделенных линиях SLIP, PPP, протоколы территориальных сетей X.25 и ISDN /1/.

Основными протоколами стека, давшими ему название, являются протоколы IP и TCP. Эти протоколы в терминологии модели OSI относятся к сетевому и транспортному уровням, соответственно. IP обеспечивает продвижение пакета по составной сети, а TCP гарантирует надежность его доставки. Стек TCP/IP вобрал в себя большое количество протоколов прикладного уровня. К ним относятся такие протоколы, как протокол пересылки файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы службы WWW и многие другие /1/.

Уровни.

Сетевые протоколы обычно разрабатываются по уровням, причем каждый уровень отвечает за собственную фазу коммуникаций. Семейства протоколов, такие как TCP/IP, это комбинации различных протоколов на различных уровнях. TCP/IP состоит из четырех уровней, как показано в таблице 3.

Таблица 3 – Уровни протокола TCP/IP

Прикладной	Telnet, FTP, e-mail и т.д.
Транспортный	TCP,UDP
Сетевой	IP, ICMP, IGMP
Канальный	драйвер устройства и интерфейсная плата

Каждый уровень несет собственную функциональную нагрузку:

1.Канальный уровень (link layer). Еще его называют уровнем сетевого интефейса. Обычно включает в себя драйвер устройства в операционной системе и соответствующую сетевую интерфейсную плату в компьютере. Вместе они обеспечивают аппаратную поддержку физического соединения с сетью (с кабелем или с другой используемой средой передачи).

2.Сетевой уровень (network layer), иногда называемый уровнем межсетевого взаимодействия, отвечает за передачу пакетов по сети. Маршрутизация пакетов осуществляется именно на этом уровне. IP (Internet Protocol - протокол Internet), ICMP (Internet Control Message Protocol - протокол управления сообщениями Internet) и IGMP (Internet Group Management Protocol - протокол управления группами Internet) обеспечивают сетевой уровень в семействе протоколов TCP/IP.

3.Транспортный уровень (transport layer) отвечает за передачу потока данных между двумя компьютерами и обеспечивает работу прикладного уровня, который находится выше. В семействе протоколов TCP/IP существует два транспортных протокола: TCP (Transmission Control Protocol) и UDP (User Datagram Protocol). TCP осуществляет надежную передачу данных между двумя компьютерами. Он обеспечивает деление данных, передающихся от одного приложения к другому, на пакеты подходящего для сетевого уровня размера, подтверждение принятых пакетов, установку тайм-аутов, в течение которых должно прийти подтверждение на пакет, и так далее. Так как надежность передачи данных гарантируется на транспортном уровне, на прикладном уровне эти детали игнорируются. UDP предоставляет более простой сервис для прикладного уровня. Он просто отсылает пакеты, которые называются датаграммами (datagram) от одного компьютера к другому. При этом нет никакой гарантии, что датаграмма дойдет до пункта назначения. За надежность передачи данных, при использовании датаграмм отвечает прикладной уровень. Для каждого транспортного протокола существуют различные приложения, которые их используют.

4.Прикладной уровень (application layer) определяет детали каждого конкретного приложения. Существует несколько распространенных приложений TCP/IP, которые присутствуют практически в каждой реализации:

– Telnet - удаленный терминал;

– FTP, File Transfer Protocol - протокол передачи файлов;

– SMTP, Simple Mail Transfer Protocol - простой протокол передачи электронной почты;

– SNMP, Simple Network Management Protocol - простой протокол управления сетью /1,4/.

Поскольку стек TCP/IP изначально создавался для глобальной сети Internet, он имеет много особенностей, которые обеспечивают ему преимущество перед другими протоколами, когда речь заходит о построении сетей, включающих глобальные связи. В частности, очень полезным свойством, благодаря которому этот протокол может применяться в больших сетях, является его способность фрагментировать пакеты. Сложная составная сеть часто состоит из сетей, построенных на совершенно разных принципах. В каждой из этих сетей может быть установлена собственная величина максимальной длины единицы передаваемых данных (кадра). В таком случае при переходе из одной сети, имеющей большую максимальную длину, в другую, с меньшей максимальной длиной, может возникнуть необходимость разделения передаваемого кадра на несколько частей. Протокол IP стека TCP/IP эффективно решает эту задачу /5/.

Другой особенностью технологии TCP/IP является гибкая система адресации, позволяющая более просто по сравнению с другими протоколами аналогичного назначения включать в интерсеть (объединенную или составную сеть) сети других технологий. Это свойство также способствует применению стека TCP/IP для построения больших гетерогенных сетей.

Однако платой за преимущества здесь оказываются высокие требования к ресурсам и сложность администрирования IP-сетей. Для реализации мощных функциональных возможностей протоколов стека TCP/IP требуются большие вычислительные затраты. Гибкая система адресации и отказ от широковещательных рассылок приводят к наличию в IP-сети различных централизованных служб типа DNS, DHCP и т. п. Каждая из этих служб упрощает администрирование сети и конфигурирование оборудования, но в то же время сама требует пристального внимания со стороны администраторов.

В стеке TCP/IP используются три типа адресов: локальные (называемые также аппаратными), IP-адреса и символьные доменные имена /1/.

В терминологии TCP/IP под локальным адресом понимается такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной интерсети. В разных подсетях допустимы разные сетевые технологии, разные стеки протоколов, поэтому при создании стека TCP/IP предполагалось наличие разных типов локальных адресов. Если подсетью интерсети является локальная сеть, то локальный адрес — это MAC-адрес. MAC-адрес назначается сетевым

адаптерам и сетевым интерфейсам маршрутизаторов. MAC-адреса назначаются производителями оборудования и являются уникальными, так как управляются централизованно. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байт, например 11-A0-17-3D-BC-01. Однако протокол IP может работать и над протоколами более высокого уровня, например над протоколом IPX или X.25. В этом случае локальными адресами для протокола IP соответственно будут адреса IPX и X.25. Следует учесть, что компьютер в локальной сети может иметь несколько локальных адресов даже при одном сетевом адаптере. Некоторые сетевые устройства не имеют локальных адресов. Например, к таким устройствам относятся глобальные порты маршрутизаторов, предназначенные для соединений типа «точка-точка» /1,4/.

Символьные доменные имена. Символьные имена в IP-сетях называются доменными и строятся по иерархическому признаку.

Составляющие полного символьного имени в IP-сетях разделяются точкой и перечисляются в следующем порядке: сначала простое имя конечного узла, затем имя группы узлов (например, имя организации), затем имя более крупной группы (поддомена) и так до имени домена самого высокого уровня (например, домена объединяющего организации по географическому принципу: RU — Россия, UK — Великобритания, SU — США). В сетях TCP/IP используется специальная распределенная служба Domain Name System (DNS), которая устанавливает это соответствие на основании создаваемых администраторами сети таблиц соответствия. Поэтому доменные имена называют также DNS-именами /3/.

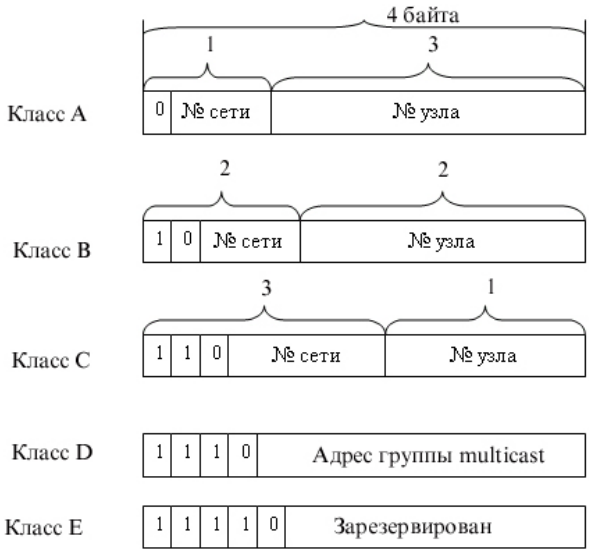


Рисунок 3 – Маски классов сетей

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками. Адрес состоит из двух логических частей — номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая — к номеру узла, определяется значениями первых бит адреса. Значения этих бит

являются также признаками того, к какому классу относится тот или иной IP-адрес.

Также для установки границы между номером сети и номером узла сейчас получили широкое распространение маски. Маска — это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Поскольку номер сети является цельной частью адреса, единицы в маске также должны представлять непрерывную последовательность. Для стандартных классов сетей маски имеют следующие значения /4/:

класс А - 11111111. 00000000. 00000000. 00000000 (255.0.0.0);

класс В - 11111111. 11111111. 00000000. 00000000 (255.255.0.0);

класс С-11111111.11111111.11111111.00000000(255.255.255.0).

В масках количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8, чтобы повторять деление адреса на байты. Пусть, например, для IP-адреса 129.64.134.5 указана маска 255.255.128.0, то есть в двоичном виде /4/:

IP-адрес 129.64.134.5 - 10000001. 01000000.10000110. 00000101;

Маска 255.255.128.0- 11111111.11111111.10000000.00000000.

Если игнорировать маску, то в соответствии с системой классов адрес 129.64.134.5 относится к классу В, а значит, номером сети являются первые 2 байта — 129.64.0.0, а номером узла — 0.0.134.5.

Если же использовать для определения границы номера сети маску, то 17 последовательных единиц в маске, «наложенные» на IP-адрес, определяют в качестве номера сети в двоичном выражении число: 10000001. 01000000. 10000000. 00000000 или в десятичной форме записи — номер сети 129.64.128.0, а номер узла 0.0.6.5. IP-пакет состоит из заголовка и поля данных. Заголовок, как правило, имеющий длину 20 байт, имеет следующую структуру /4/:

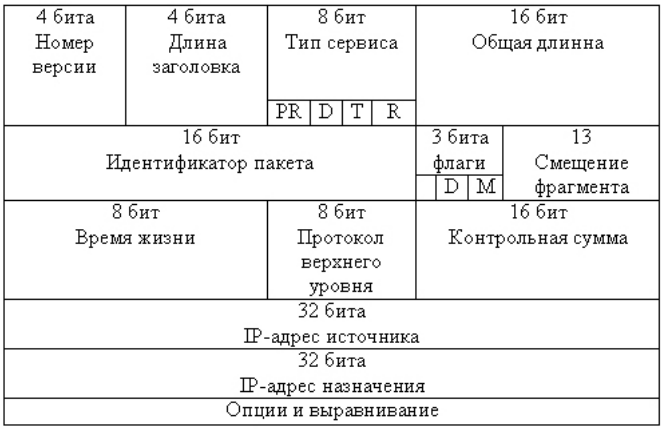


Рисунок 4 – Структура заголовка

Поле Номер версии (Version), занимающее 4 бит, указывает версию протокола IP. Сейчас повсеместно используется версия 4 (IPv4), и готовится переход на версию 6 (IPv6).

Поле Длина заголовка (IHL) IP-пакета занимает 4 бит и указывает значение длины заголовка, измеренное в 32-битовых словах.

Обычно заголовок имеет длину в 20 байт (пять 32-битовых слов), но при увеличении объема служебной информации эта длина может быть увеличена за счет использования дополнительных байт в поле Опции (IP Options).

Поле Тип сервиса (Type of Service) занимает один байт и задает приоритетность пакета и вид критерия выбора маршрута.

Первые три бита этого поля образуют подполе приоритета пакета (Precedence). Приоритет может иметь значения от самого низкого — 0 (нормальный пакет) до самого высокого — 7 (пакет управляющей информации). Маршрутизаторы и компьютеры могут принимать во внимание приоритет пакета и обрабатывать более важные пакеты в первую очередь. Поле Тип сервиса содержит также три бита, определяющие критерий выбора маршрута. Реально выбор осуществляется между тремя альтернативами: малой задержкой, высокой достоверностью и высокой пропускной способностью. Установленный бит D (delay) говорит о том, что маршрут должен выбираться для минимизации задержки доставки данного пакета, бит T — для максимизации пропускной способности, а бит R — для максимизации надежности доставки. Во многих сетях улучшение одного из этих параметров связано с ухудшением другого, кроме того, обработка каждого из них требует дополнительных в ычислительных затрат. Поэтому редко, когда имеет смысл устанавливать одновременно хотя бы два из этих трех критериев выбора маршрута. Зарезервированные биты имеют нулевое значение.

Поле Общая длина (Total Length) занимает 2 байта и означает общую длину пакета с учетом заголовка и поля данных. Максимальная длина пакета ограничена разрядностью поля, определяющего эту величину, и составляет 65 535 байт, однако в большинстве хост-компьютеров и сетей столь большие пакеты не используются. При передаче по сетям различного типа длина пакета выбирается с учетом максимальной длины пакета протокола нижнего уровня, несущего IP-пакеты. Если это кадры Ethernet, то выбираются пакеты с максимальной длиной в 1500 байт, умещающиеся в поле данных кадра Ethernet. В стандарте предусматривается, что все хосты должны быть готовы принимать пакеты вплоть до 576 байт длиной (приходят ли они целиком или по фрагментам). Хостам рекомендуется отправлять пакеты размером более чем 576 байт, только если они уверены, что принимающий хост или промежуточная сеть готовы обслуживать пакеты такого размера.

Поле Идентификатор пакета (Identification) занимает 2 байта и используется для распознавания пакетов, образовавшихся путем фрагментации исходного пакета. Все фрагменты должны иметь одинаковое значение этого поля.

Поле Флаги (Flags) занимает 3 бита и содержит признаки, связанные с фрагментацией. Установленный бит DF (Do not Fragment) запрещает маршрутизатору фрагментировать данный пакет, а установленный бит MF

(More Fragments) говорит о том, что данный пакет является промежуточным (не последний) фрагментом. Оставшийся бит зарезервирован.

Поле Смещение фрагмента (Fragment Offset) занимает 13 бит и задает смещение в байтах поля данных этого пакета от начала общего поля данных исходного пакета, подвергнутого фрагментации. Используется при сборке/разборке фрагментов пакетов при передачах их между сетями с различными величинами MTU. Смещение должно быть кратно 8 байт.

Поле Время жизни (Time to Live) занимает один байт и означает предельный срок, в течение которого пакет может перемещаться по сети. Время жизни данного пакета измеряется в секундах и задается источником передачи. На маршрутизаторах и в других узлах сети по истечении каждой секунды из текущего времени жизни вычитается единица; единица вычитается и в том случае, когда время задержки меньше секунды. Поскольку современные маршрутизаторы редко обрабатывают пакет дольше, чем за одну секунду, то время жизни можно считать равным максимальному числу узлов, которые разрешено пройти данному пакету до того, как он достигнет места назначения. Если параметр времени жизни станет нулевым до того, как пакет достигнет получателя, этот пакет будет уничтожен. Время жизни можно рассматривать как часовой механизм самоуничтожения. Значение этого поля изменяется при обработке заголовка IP-пакета /1,4,8/.

Идентификатор Протокол верхнего уровня (Protocol) занимает один байт и указывает, какому протоколу верхнего уровня принадлежит информация, размещенная в поле данных пакета (например, это могут быть сегменты протокола TCP, дейтаграммы UDP, пакеты ICMP или OSPF). Значения идентификаторов для различных протоколов приводятся в документе RFC «Assigned Numbers».

Контрольная сумма (Header Checksum) занимает 2 байта и рассчитывается только по заголовку. Поскольку некоторые поля заголовка меняют свое значение в процессе передачи пакета по сети (например, время жизни), контрольная сумма проверяется и повторно рассчитывается при каждой обработке IP-заголовка. Контрольная сумма — 16 бит — подсчитывается как дополнение к сумме всех 16-битовых слов заголовка. При вычислении контрольной суммы значение самого поля «контрольная сумма» устанавливается в нуль. Если контрольная сумма неверна, то пакет будет отброшен, как только ошибка будет обнаружена.

Поля IP-адрес источника (Source IP Address) и IP-адрес назначения (Destination IP Address) имеют одинаковую длину — 32 бита — и одинаковую структуру.

Поле Опции (IP Options) является необязательным и используется обычно только при отладке сети. Механизм опций предоставляет функции управления, которые необходимы или просто полезны при определенных ситуациях, однако он не нужен при обычных коммуникациях. Это поле состоит из нескольких подполей, каждое из которых может быть одного из восьми предопределенных типов. В этих подполях можно указывать точный маршрут

прохождения маршрутизаторов, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности, а также временные отметки. Так как число подполей может быть произвольным, то в конце поля Опции должно быть добавлено несколько байт для выравнивания заголовка пакета по 32-битной границе.

Поле Выравнивание (Padding) используется для того, чтобы убедиться в том, что IP-заголовок заканчивается на 32-битной границе. Выравнивание осуществляется нулями /1,4,9/.

Адресация Internet.

Каждый интерфейс в объединенной сети должен иметь уникальный IP адрес. Эти адреса представляют из себя тридцатидвухбитовые числа. Существует определенная структура адреса Internet. На рисунке 4 показано 5 классов адресов Internet.

Эти 32-битные адреса обычно записываются как 4 десятичных числа, по одному на каждый байт адреса. Такая форма записи называется "десятичной записью с точками" (dotted-decimal). Например, адрес сети класса B может быть записан как 140.252.13.33.

Определить класс адреса, или класс сети, можно по первому числу в адресе. В таблице 4 показаны различные классы /1/.

Таблица 4 - Пять классов адресов Internet.

Класс	Диапазон IP адресов в разных классах сетей
A	0.0.0.0 - 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255
D	224.0.0.0 - 239.255.255.255
E	240.0.0.0 - 247.255.255.255

Так как каждый интерфейс, подключенный к сети, должен иметь уникальный адрес, встает вопрос распределения IP адресов в глобальной сети Internet. Этим занимается сетевой информационный центр (Internet Network Information Center или InterNIC). InterNIC назначает только сетевые идентификаторы (ID). Назначением идентификаторов хостов в сети занимаются системные администраторы.

Регистрация сервисов Internet (IP адреса и имена доменов DNS) осуществляется в NIC, nic.ddn.mil. InterNIC была создана 1 апреля 1993 года. В настоящее время NIC регистрирует сервисы только для сети министерства обороны (DDN - Defence Data Network). Все другие пользователи Internet в настоящее время используют регистрационный сервис InterNIC в rs.internic.net. Реально существует три части InterNIC: регистрационный сервис (rs.internic.net), сервис баз данных (ds.internic.net) и информационный сервис (is.internic.net).

Существует три типа IP адресов: персональный адрес (unicast) - указывает на один хост, широковещательный адрес (broadcast) - указывает на

все хосты в указанной сети, и групповой адрес (multicast) - указывает на группу хостов, принадлежащей к группе адресации /1,5/.

Выполнение работы:
- изучить назначение протокола TCP/IP;
- изучить уровни протокола TCP/IP;
- изучить систему адресации протокола TCP/IP;
- изучить структуру заголовка пакета IP;
- изучить описание полей заголовка IP;
- оформить отчет.

Отчет по лабораторной работе должен содержать следующее:
- название и цель работы;
- основные теоретические пункты общий сведений;
- выводы по выполненной работе;
- список использованных источников.

Контрольные вопросы:
- назначение протокола TCP/IP?;
- какие стандарты поддерживает протокол TCP/IP?;
- какие уровни представлены в протоколе TCP/IP?;
- какую функциональную нагрузку несет канальный уровень?;
- какую функциональную нагрузку несет сетевой уровень?;
- какую функциональную нагрузку несет транспортный уровень?;
- какую функциональную нагрузку несет прикладной уровень?;
- как устроена система адресации в протоколе TCP/IP?;
- что понимается под локальным адресом в протоколе TCP/IP?;
- какое назначение IP-адреса?;
- что такое символьные доменные имена?;
- какую длину и структуру имеет IP-адрес?;
- что представляют собой маски классов сетей?;
- из чего состоит IP-пакет?;
- какие поля (и их назначение) используются в пакете?.

4 Лабораторная работа № 4. Установка компонент сети

Цель работы: Получить основные теоретические сведения и практические навыки по установке (настройки) компонент сети.

Теоретическая справка.

Для нормального функционирования сети необходимо правильно настроить операционную систему. Для этого необходимо открыть панель управления, в которой находится апплет настройки сети. При вызове апплета необходимо нажать кнопку «Добавить» и установить следующие компоненты:

- клиент для сетей Microsoft;
- сетевая плата;
- сетевой протокол;
- сетевая служба.

Клиент для сетей Microsoft отвечает за взаимодействие системы с сервером, выбор клиента сети отображен на рисунке 5.

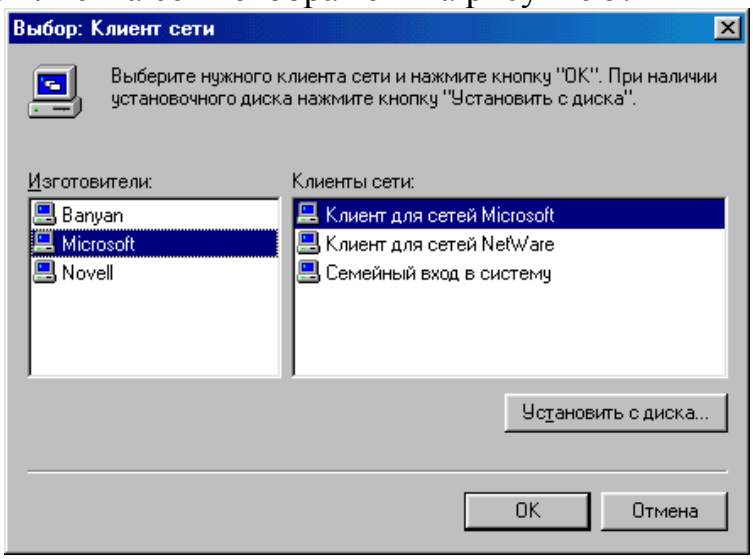


Рисунок 5 – Установка клиента для сетей Microsoft

Сетевая плата является компонентом, выполняющим взаимодействие с физической средой передачи данных. Его выбор производится в соответствии с производителем и названием устройства, пример показан на рисунке 6.

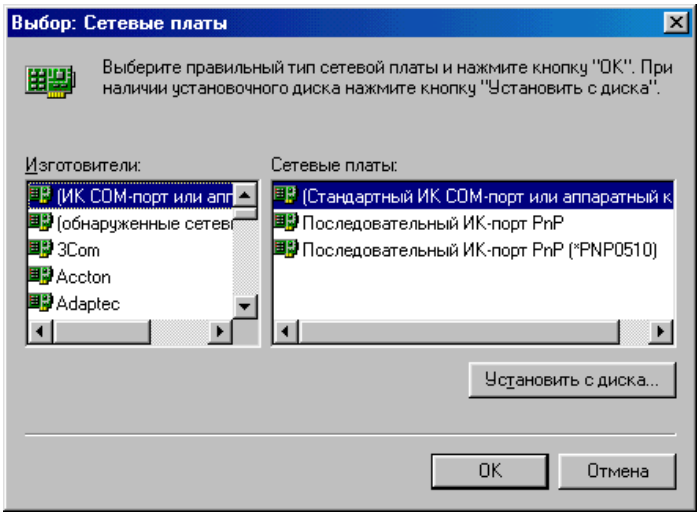


Рисунок 6 – Установка драйверов для сетевой платы

Установка сетевого протокола происходит аналогично (рисунок 7).

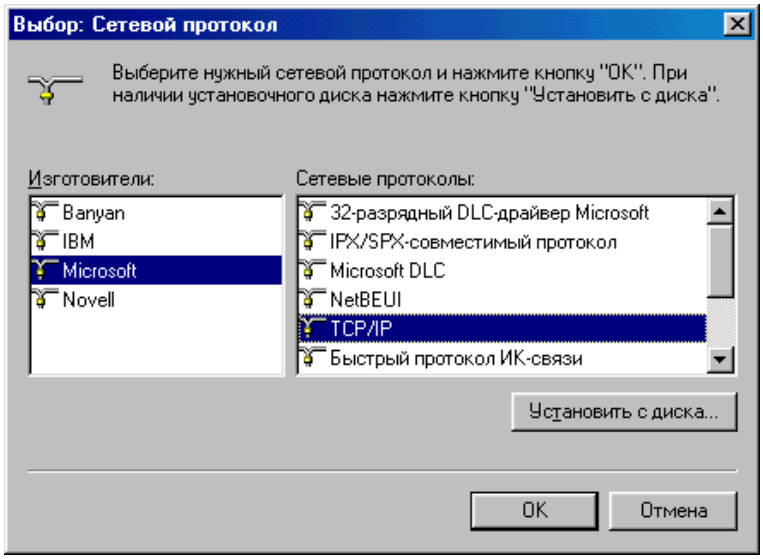


Рисунок 7 – Установка сетевого протокола

Последняя из устанавливаемых компонент – сетевая служба. Она выполняет функции разграничения доступа к общим ресурсам компьютера в сети (рисунок 8).

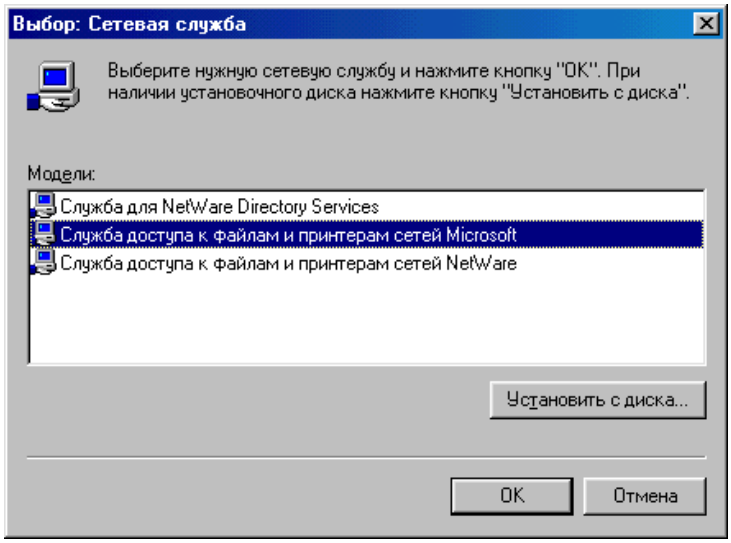


Рисунок 8 – Инсталляция сетевой службы

После установки всех компонент мы получим форму, отображенную на рисунке 9. Для дальнейшей настройки необходимо выделить компонент и нажать кнопку «Свойства», которая на рисунке неактивна, потому что не выбран компонент.

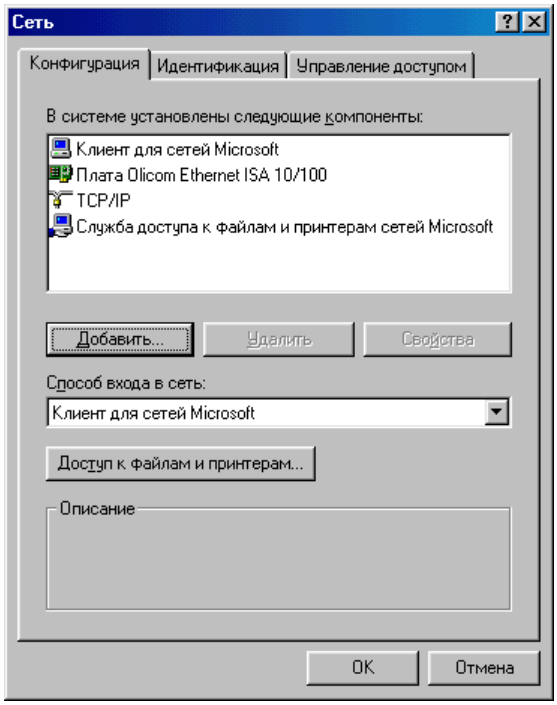


Рисунок 9 – Конфигурация сети после установки всех компонентов

Настройка клиента сетей сводится к прописыванию домена, к которому подключается компьютер при входе в сеть и установки параметра входа с восстановлением сетевых подключений, как показано на рисунке 10.

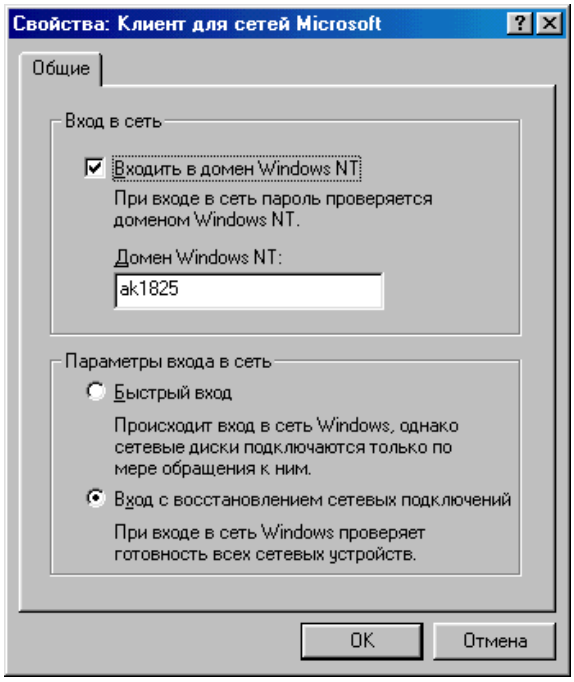


Рисунок 10 – Настройка свойств клиента сети

Настройка драйверов сетевой платы обычно не требуется, так как все установки по умолчанию не создают проблем. Для примера на рисунке 11 показаны вкладки сетевой карты Olicom Ethernet ISA 10/100 /1,7/.

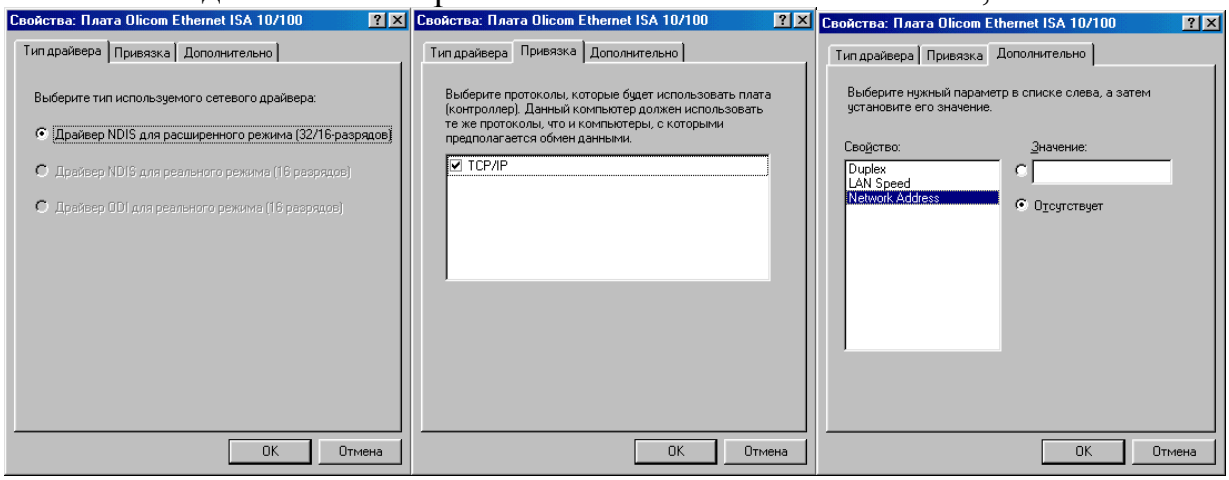


Рисунок 11 – Вкладки свойств сетевой карты

Настройка протокола TCP/IP не требуется, если нет необходимости настраивать сеть на соединение сети с интернетом. Оговоримся только, что если на сервере не работает служба DHCP, то на вкладке IP-адреса необходимо указать его и маску подсети, к которой принадлежит компьютер (рисунок 12) .

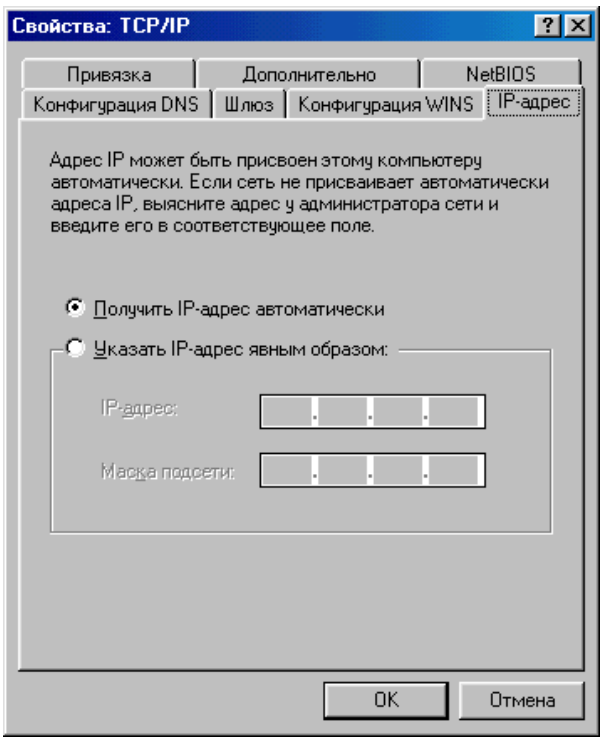


Рисунок 12 – Настройка IP адреса компьютера

На рисунке 13 изображена другая вкладка апплета «Сеть» - «Идентификация», которая позволяет задать имя компьютера и его принадлежность рабочей группе (домену). Имя компьютера и его описание позволяет легко ориентироваться в большом количестве компьютеров в домене.

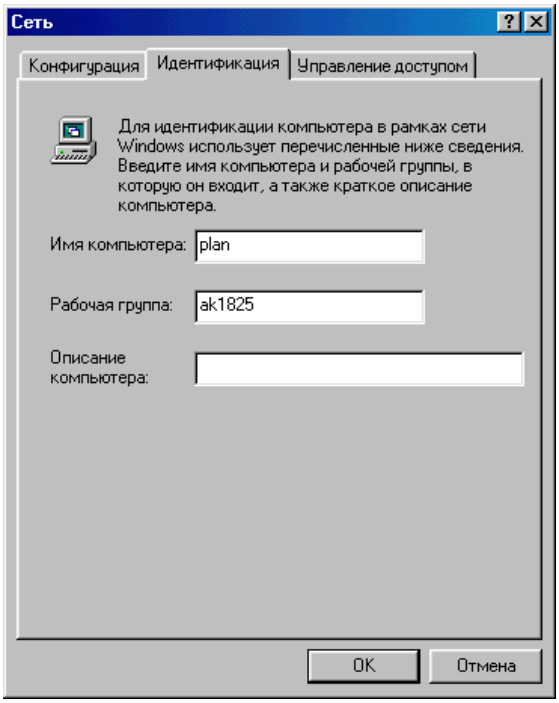


Рисунок 13 – Вкладка идентификации

Управление ресурсами в домене может осуществляться двумя способами: на уровне ресурсов и на уровне пользователей (рисунок 14). Каждый из способов имеет ряд преимуществ, но для облегчения администрирования домена наиболее предпочтительней управление на уровне пользователей, что позволяет указывать пользователей и группы, имеющих доступ к каждому общему ресурсу /1,7/.

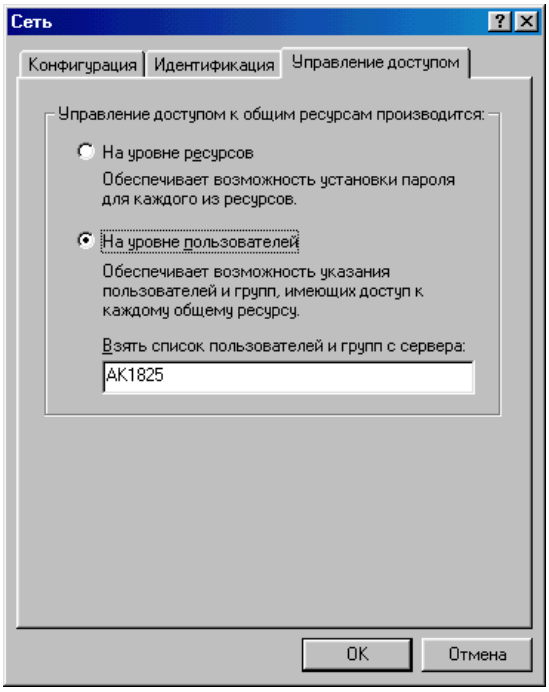


Рисунок 14 – Управление доступом

Выполнение работы:
- установить компоненты сети, основываясь на рисунки 5-14;
- оформить отчет.

Отчет по лабораторной работе должен содержать следующее:
- название и цель работы;
- основные теоретические пункты общий сведений;
- выводы по выполненной работе;
- список использованных источников.

5 Лабораторная работа №5. Работа с кабельной системой

Цель работы: Получение навыков работы со структурируемой кабельной системой. Получение навыков обжима кабеля.

Теоретическая справка.

Витая пара (UTP/STP, unshielded/shielded twisted pair) в настоящее время является наиболее распространенной средой передачи сигналов в локальных сетях. Кабели UTP/STP используются в сетях Ethernet, Token Ring и ARCnet. Они различаются по категориям (в зависимости от полосы пропускания) и типу проводников (гибкие или одножильные). В кабеле 5-й категории, как правило, находится восемь проводников, перевитых попарно (то есть четыре пары-рисунок 15).

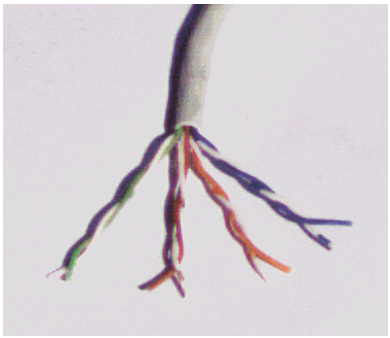


Рисунок 15 – Кабель 5-й категории

Структурированная кабельная система, построенная на основе витой пары 5-й категории, имеет очень большую гибкость в использовании. Ее идея заключается в следующем.

На каждое рабочее место устанавливается не менее двух (рекомендуется три) четырехпарных розеток RJ-45. Каждая из них отдельным кабелем 5-й категории соединяется с кроссом или патч-панелью, установленной в специальном помещении, — серверной. В это помещение заводятся кабели со всех рабочих мест, а также городские телефонные вводы, выделенные линии для подключения к глобальным сетям и т.п. В помещении, естественно, монтируются серверы, а также офисная АТС, системы сигнализации и прочее коммуникационное оборудование /1,7,8/.

Благодаря тому что кабели со всех рабочих мест сведены на общую панель, любую розетку можно использовать как для подключения рабочего места к ЛВС, так и для телефонии или вообще чего угодно. Допустим, две розетки на рабочем месте были подключены к компьютеру и принтеру, а третья — к телефонной станции. В процессе работы появилась необходимость убрать принтер с рабочего места и установить вместо него второй телефон. Нет ничего проще — патч-корд соответствующей розетки отключается от концентратора и

переключается на телефонный кросс, что займет у администратора сети никак не больше нескольких минут.

Патч-панель, или панель соединений, представляет собой группу розеток RJ-45, смонтированных на пластине шириной 19 дюймов. Это стандартный размер для универсальных коммуникационных шкафов — рэков (rack), в которых устанавливается оборудование (концентраторы, серверы, источники бесперебойного питания и т.п.). На обратной стороне панели смонтированы соединители, в которые монтируются кабели.

Кросс в отличие от патч-панели розеток не имеет. Вместо них он несет на себе специальные соединительные модули. В данном случае его преимущество перед патч-панелью в том, что при его использовании в телефонии вводы можно соединять между собой не специальными патч-кордами, а обычными проводами. Кроме того, кросс можно монтировать прямо на стену — наличия коммуникационного шкафа он не требует. В самом деле, нет смысла приобретать дорогостоящий коммуникационный шкаф, если вся ваша сеть состоит из одного-двух десятков компьютеров и сервера.

Кабели с многожильными гибкими проводниками используются в качестве патч-кордов, то есть соединительных кабелей между розеткой и сетевой платой, либо между розетками на панели соединений или кроссе. Кабели с одножильными проводниками — для прокладки собственно кабельной системы. Монтаж разъемов и розеток на эти кабели совершенно идентичен, но обычно кабели с одножильными проводниками монтируются на розетки рабочих мест пользователей, панели соединений и кроссы, а разъемы устанавливают на гибкие соединительные кабели /4,6/.

Как правило, применяются следующие виды разъемов:

S110 — общее название разъемов для подключения кабеля к универсальному кроссу “110” или коммутации между вводами на кроссе;

RJ-11 и RJ-12 — разъемы с шестью контактами (первые обычно применяются в телефонии общего назначения — вы можете встретить такой разъем на шнурах импортных телефонных аппаратов, второй обычно используется в телефонных аппаратах, предназначенных для работы с офисными мини-АТС, а также для подключения кабеля к сетевым платам ARCnet);

RJ-45 — восьмиконтактный разъем, использующийся обычно для подключения кабеля к сетевым платам Ethernet либо для коммутации на панели соединений.

Разъем RJ-45.

В зависимости от того, что с чем нужно коммутировать, применяются различные патч-корды: “45-45” (с каждой стороны по разъему RJ-45), “110-45” (с одной стороны S110, с другой — RJ-45) или “110-110”.

Для монтажа разъемов RJ-11, RJ-12 и RJ-45 используются специальные обжимочные приспособления, различающиеся между собой количеством ножей (6 или 8) и размерами гнезда для фиксации разъема. В качестве примера рассмотрим монтаж кабеля 5-й категории на разъем RJ-45 /1/.

Ход работы:

1 Аккуратно обрежьте конец кабеля. Торец кабеля должен быть ровным.

2 Используя специальный инструмент, снимите с кабеля внешнюю изоляцию на длину примерно 30 мм и обрежьте нить, вмонтированную в кабель (нить предназначена для удобства снятия изоляции с кабеля на большую длину). Любые повреждения (надрезы) изоляции проводников абсолютно недопустимы — именно поэтому желательно использовать специальный инструмент, лезвие резака которого выступает ровно на толщину внешней изоляции.

3 Аккуратно разведите, расплетите и выровняйте проводники. Выровняйте их в один ряд, при этом соблюдая цветовую маркировку. Существует два наиболее распространенных стандарта по разводке цветов по парам: T568A (рекомендуемый компанией Siemon) и T568B (рекомендуемый компанией AT&T и фактически наиболее часто применяемый).На разъеме RJ-45 цвета проводников располагаются так показано в таблице 5.

Таблица 5 – Цвета проводников кабеля типа «витая пара» 5-й категории

Номер контакта	Цвет по T568B	Цвет по T568A
1	бело-оранжевый	бело-зеленый
2	оранжевый	зеленый
3	бело-зеленый	бело-оранжевый
4	синий	синий
5	бело-синий	бело-синий
6	зеленый	оранжевый
7	бело-коричневый	бело-коричневый
8	коричневый	коричневый

Проводники должны располагаться строго в один ряд, без нахлестов друг на друга. Удерживая их одной рукой, другой ровно обрежьте проводники так, чтобы они выступали над внешней обмоткой на 8-10 мм.

4. Держа разъем защелкой вниз, вставьте в него кабель. Каждый проводник должен попасть на свое место в разъеме и упереться в ограничитель. Прежде чем обжимать разъем, убедитесь, что вы не ошиблись в разводке проводников. При неправильной разводке помимо отсутствия соответствия номерам контактов на концах кабеля, легко выявляемого с помощью простейшего тестера, возможна более неприятная вещь — появление “разбитых пар” (splitted pairs). Для выявления этого брака обычного тестера недостаточно, так как электрический контакт между соответствующими контактами на концах кабеля обеспечивается и с виду все как будто бы нормально. Но такой кабель никогда не сможет обеспечить нормальное качество соединения даже в 10-мегабитной сети на расстояние более 40-50 метров. Поэтому нужно быть внимательным и не торопиться, особенно если у вас нет достаточного опыта.

5. Вставьте разъем в гнездо на обжимочном приспособлении и обожмите его до упора-ограничителя на приспособлении. В результате фиксатор на разъеме встанет на свое место, удерживая кабель в разъеме неподвижным. Контактные ножи разъема врежутся каждый в свой проводник, обеспечивая надежный контакт.

Аналогичным образом можно осуществить монтаж разъемов RJ-11 и RJ-12, используя соответствующий инструмент.

Для монтажа разъема S110 специального обжимочного инструмента не требуется. Сам разъем поставляется в разобранном виде. Кстати, в отличие от “одноразовых” разъемов типа RJ разъем S110 допускает многократную разборку и сборку, что очень удобно. Последовательность действий при монтаже следующая:

1. Снимите внешнюю изоляцию кабеля на длину примерно 40 мм, разведите в стороны пары проводников, не расплетая их.

2. Закрепите кабель (в той половинке разъема, на которой нет контактной группы) с помощью пластмассовой стяжки и отрежьте получившийся “хвост”.

3. Аккуратно уложите каждый проводник в органайзер на разъеме. Не расплетайте пару на большую, чем требуется, длину — это ухудшит характеристики всего кабельного соединения. Последовательность укладки пар обычная — синяя-оранжевая- зеленая-коричневая; при этом светлый провод каждой пары укладывается первым.

4. Острым инструментом (бокорезами или ножом) обрежьте каждый проводник по краю разъема.

5. Установите на место вторую половинку разъема и руками обожмите ее до защелкивания всех фиксаторов. При этом ножи контактной группы врежутся в проводники, обеспечивая контакт.

Отчет по лабораторной работе должен содержать следующее:

- название и цель работы;
- основные теоретические пункты общий сведений;
- выводы по выполненной работе;
- список использованных источников.

Контрольные вопросы:

какие типы кабелей существуют?;

какова последовательность обжима кабеля?;

назначение и конструкции разъемов и розеток?.

6 Лабораторная работа № 6. Пересылка/ прием сообщений через сокеты

Цель работы: Изучение особенностей использования сокета, для передачи сообщений в ЛВС.

Теоретическая справка.

Socket (гнездо, разъем) - абстрактное программное понятие, используемое для обозначения в прикладной программе конечной точки канала связи с коммуникационной средой, образованной вычислительной сетью. При использовании протоколов TCP/IP можно говорить, что socket является средством подключения прикладной программы к порту локального узла сети.

Рассмотрим механизм реализации сокетов в Borland Delphi. Для работы с сокетами в Delphi используются компоненты TClientSocket и TServerSocket. Они являются потомками абстрактного класса TabstractSocket, который включает методы и свойства, позволяющие прикладному приложению использовать Windows socket.

Windows socket объединяет в себе набор коммуникационных протоколов, предоставляющие возможность приложению подключаться к другим компьютерам для обмена информацией. Windows sockets поддерживает следующие семейства протоколов:

- TCP/IP;
- Xerox Network System (XNS);
- IPX/SPX;
- DECnet.

Сокеты позволяют приложению создавать соединение с другими машинами без знания конкретного типа протокола.

Для создания сокета, инициирующего соединение с другими машинами используют TclientSocket, а для создания сокета, отвечающего на запросы с других машин, - TserverSocket /2/.

Примерная схема работы с сокетом клиента включает в себя следующие шаги:

1. Определение свойств сокета Host и Port. Host – это имя хост-имя или IP-адрес компьютера, с которым необходимо установить соединение. Port –имя порта.

2. Открытие сокета. В данном шаге сокет клиента определяет сервер и подключается к нему.

3. Пересылка данных.

4. Закрытие сокета.

Алгоритм работы сокета сервера немного отличается от рассмотренного выше алгоритма для сокета клиента:

1. Определение свойств Port и ServerType. Свойство Port аналогично свойству сокета клиента. ServerType – определяет тип подключения.

2. Открытие сокета. Сокет на данном шаге переходит в режим ожидания подключений клиентов.

3. Подключение клиентов и пересылка данных.

4. Отключение клиентов.

5. Закрытие сокета.

Выполнение работы:

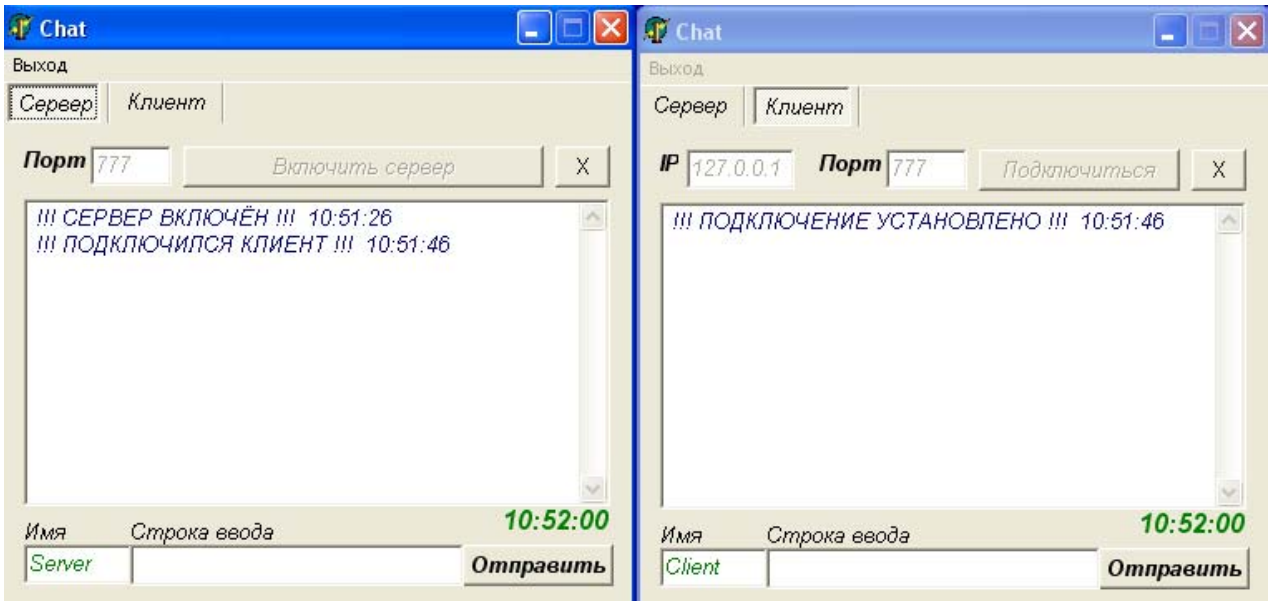
– изучить возможности сокетов для передачи данных в ЛВС;

– реализовать прикладное приложение на основе сокетов, обеспечивающее передачу сообщений по ЛВС;

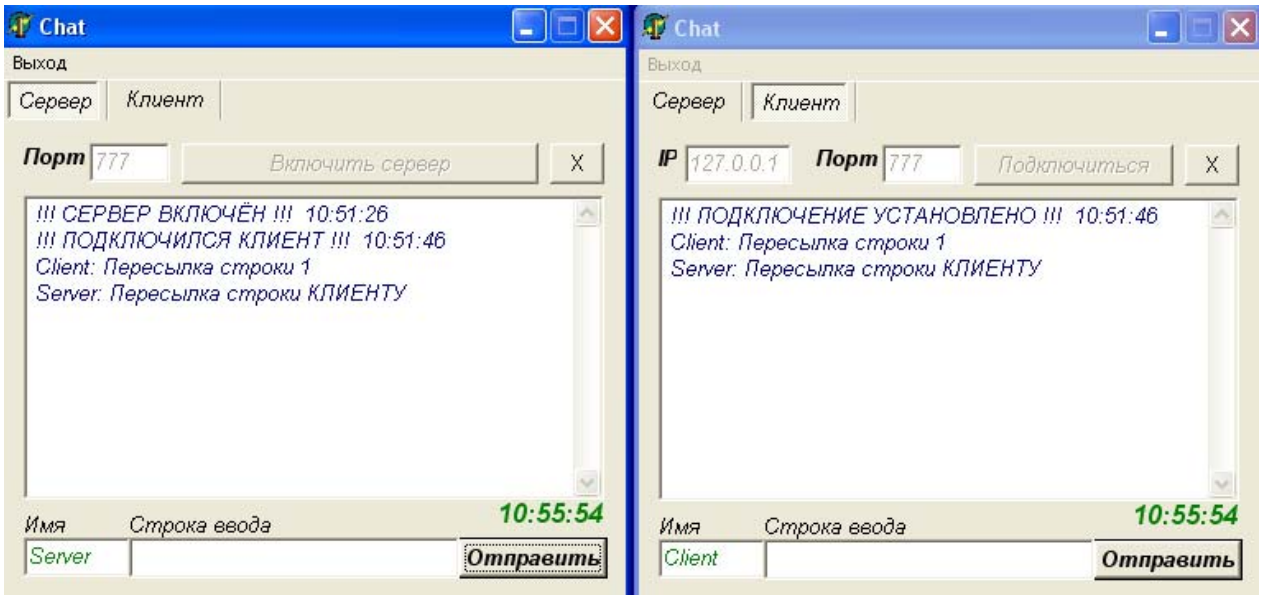
– осуществите передачу сообщений между компьютерами, используя созданное прикладное приложение.

Пример работы программы:

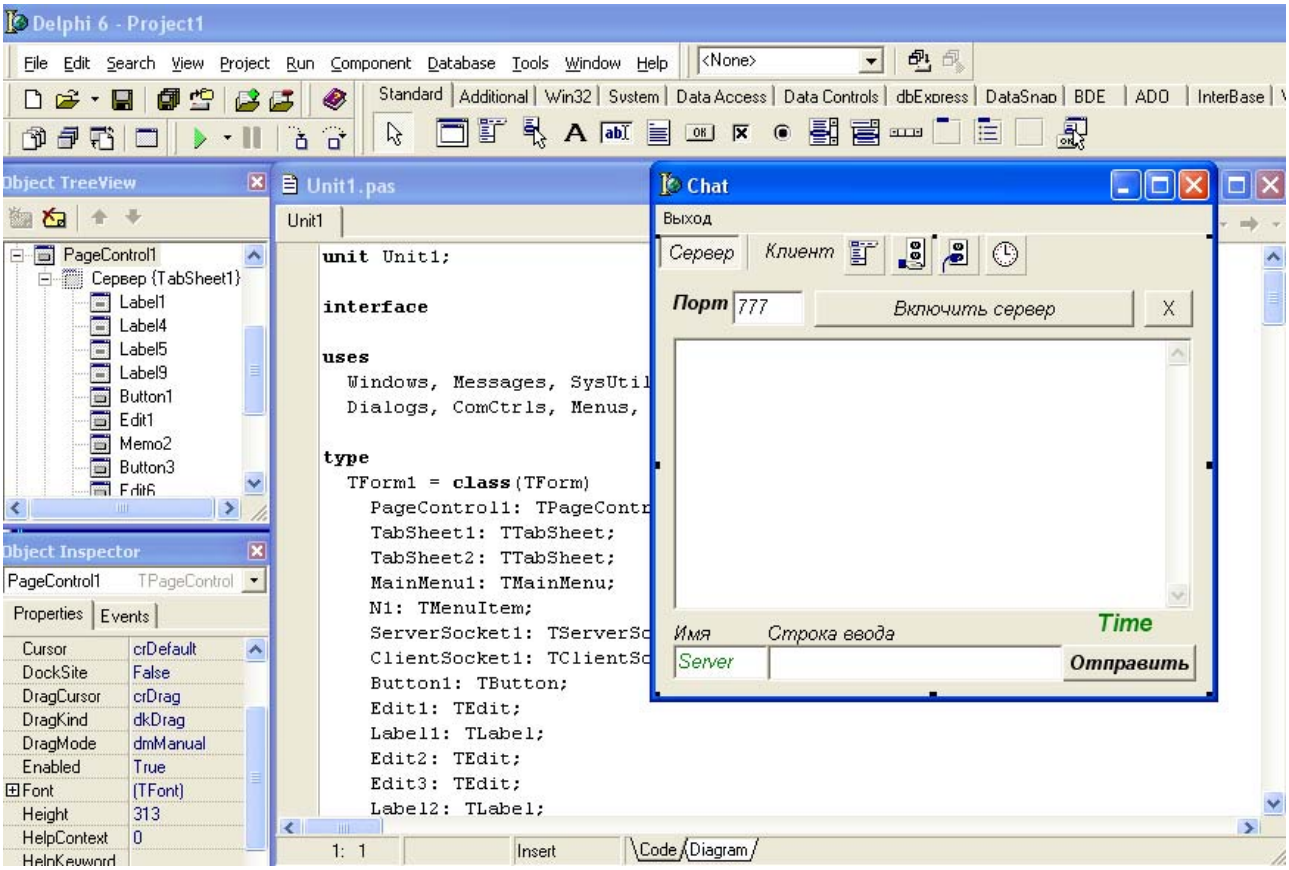
1. Подключение клиента.



2. Пересылка сообщения серверу и с сервера клиенту.



3. Экранная форма программы по пересылке сообщений.



Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;
- руководство для программиста;
- алгоритм программы;

– листинг программы;
– экранные формы работы программы;
– вывод по выполненной работе;
– список использованных источников.

Контрольные вопросы:
-- что такое сокет?;
– семейства каких протоколов поддерживает Windows socket?;
– опишите алгоритм работы сокета для клиентского приложения?;
– опишите алгоритм работы сокета для приложения сервера?;
– какие свойства должны быть определены для создания соединения с помощью сокетов?.

7 Лабораторная работа № 7. Пересылка/ прием сложных данных через сокеты

Цель работы: Изучение особенностей использования сокета, для передачи сложных данных по ЛВС.

Теоретическая справка.

Сокеты позволяют передавать информацию различного вида. Данные передаются в виде последовательности символов, в результате можно передавать как текстовые сообщения, так и целые файлы.

Методов организации работы с сокетами в Delphi существует большое количество. Для передачи сложных данных между компьютерами нужно воспользоваться специальными операторами. Рассмотрим некоторые из них:

1. *SendBuf*(var Buf; Count: Integer) – метод передачи буфера через сокет. Вторым параметром Count метода указывается размер буфера в байтах.
2. *SendText*(const S: string) – текстовой строки.
3. *SendStream*(AStream: TStream) –передача потока через сокет. Поток в Delphi – это обобщенная модель двоичных данных, размещенных на устройствах-накопителях, таких как диски, ленточные накопители, оперативная память и т. п. Любой поток обладает ключевыми свойствами – размером в байтах (свойство Size) и текущей позицией (Position) /2/.

Всем данным методам передачи данных существуют соответствующие методы приема данных.

Выполнение работы.

– рассмотреть различные методы передачи сложных видов данных через сокеты;

– реализовать прикладное приложение на основе сокетов, обеспечивающее передачу файлов по ЛВС;

– осуществить передачу файлов различных размеров по ЛВС и проверить результат выполнения передачи данных.

Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;
- руководство для программиста;
- алгоритм программы;
- листинг программы;
- экранные формы работы программы;
- вывод по выполненной работе;
- список использованных источников.

Контрольные вопросы:

– перечислите методы и опишите их характерные особенности (в зависимости от выбранного языка программирования), которые используются сокетами для передачи данных между компьютерами?;

– в каком виде передаются данные через сокеты?.

8 Лабораторная работа № 8. Компьютерные игры. Крестики -нолики

Цель работы: Получение навыков работы с протоколом TCP/IP, программирования сокетов, написание собственной игровой программы.

Теоретическая справка.

Компонента ServerSocket. Сервер, основанный на сокетном протоколе, позволяет обслуживать сразу множество клиентов. Для каждого подключенного клиента сервер открывает отдельный сокет, по которому можно обмениваться данными с клиентом. Также возможно создание для каждого подключения отдельного процесса.

Определение свойств Port и ServerType необходимо, чтобы к серверу могли подключаться клиенты, нужно, чтобы порт, используемый сервером точно совпадал с портом, используемым клиентом (и наоборот). Свойство ServerType определяет тип подключения.

На этапе открытия сокета и указанного порта может выполняться автоматическое начало ожидания подсоединения клиентов (Listen).

При отключении клиента закрывается его сокетное соединение с сервером.

По команде приложения сервер завершает свою работу, закрывая все открытые сокетные каналы и прекращая ожидание подключений клиентов.

Свойство ServerType:TServerType указывает тип сервера. Оно может принимать одно из двух значений: stNonBlocking - синхронная работа с клиентскими сокетами. При таком типе сервера можно работать с клиентами через события OnClientRead и OnClientWrite. StThreadBlocking - асинхронный тип. Для каждого клиентского сокетного канала создается отдельный процесс (Thread).

Свойство Active: Boolean - показатель того, активен в данных момент сервер, или нет. Значение True указывает на то, что сервер работает и готов к приему клиентов, а False - сервер выключен. Чтобы запустить сервер, нужно просто присвоить этому свойству значение True.

Port: Integer это номер порта для установления соединений с клиентами. Значение порта у сервера и у клиентов должны быть одинаковыми. Рекомендуются значения от 1025 до 65535, т.к. от 1 до 1024 - могут быть заняты системой.

Метод Open запускает сервер. Эта команда идентична присвоению значения True свойству Active.

Метод Close останавливает сервер.

Событие OnClientConnect возникает, когда клиент установил сокетное соединение и ждет ответа сервера (OnAccept).

Событие OnClientDisconnect возникает, когда клиент отсоединился от сокетного канала.

Событие OnClientError возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка.

Событие OnClientRead возникает, когда клиент передал серверу какие-либо данные. Доступ к этим данным можно получить через передаваемый параметр Socket: TCustomWinSocket.

Событие OnClientWrite возникает, когда сервер может отправлять данные клиенту по сокету.

Событие OnAccept возникает, когда сервер принимает клиента или отказывает ему в соединении;

Событие OnListen возникает, когда сервер переходит в режим ожидания подсоединения клиентов.

Компонента ClientSocket. После назначения свойствам Host и Port соответствующих значений, можно приступить непосредственно к открытию сокета (сокет здесь рассматривается как очередь, в которой содержатся символы, передающиеся от одного компьютера к другому). Для этого можно вызвать метод Open компонента TClientSocket, либо присвоить свойству Active значение True.

Этап авторизации необходим, если сервер требует ввода логина и/или пароля. На этом этапе посылается серверу логин (имя пользователя) и пароль. Механизм авторизации зависит уже от конкретного сервера.

Свойство Host: string это строка, указывающая на хост-имя компьютера, к которому следует подключиться.

Address: string - строка, указывающая на IP-адрес компьютера, к которому следует подключиться. В отличие от Host, здесь может содержаться лишь IP. Отличие в том, что при указании в Host символьного имя компьютера, IP адрес, соответствующий этому имени, будет запрошен у DNS.

Строка Service : string, определяет службу (ftp, http, pop, и т.д.), к порту которой произойдет подключение. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам.

Свойство ClientType это тип соединения. CtNonBlocking - асинхронная передача данных, т.е. посылать и принимать данные по сокету можно с помощью OnRead и OnWrite. CtBlocking - синхронная (одновременная) передача данных. События OnRead и OnWrite не работают. Этот тип соединения полезен для организации обмена данными с помощью потоков.

Событие OnConnect возникает при установлении соединения. Т.е. в обработчике этого события уже можно начинать авторизацию или прием/передачу данных.

Событие OnConnecting возникает при установлении соединения. Отличие от OnConnect в том, что соединение еще не установлено. Обычно такие промежуточные события используются для обновления статуса.

Событие OnDisconnect возникает при закрытии сокета.

Событие OnError возникает при ошибке в работе сокета. Следует отметить, что это событие не поможет отловить ошибку в момент открытия сокета (Open). Для того, чтобы избежать выдачи сообщения об ошибке, необходимо заключить операторы открытия сокета в блок try..except (обработка исключительных ситуаций).

Событие OnLookup возникает при попытке получения от DNS IP-адреса указанного хоста.

Событие OnRead возникает, когда удаленный компьютер послал какие-либо данные. При возникновении этого события возможна обработка данных.

Событие OnWrite возникает, когда разрешена запись данных в сокет.

Метод SendBuf(var Buf; Count: Integer) используется при посылке буфера через сокет. Буфером может являться любой тип, будь то структура (record), либо простая переменная типа Integer. Буфер указывается параметром Buf, вторым параметром необходимо указать размер пересылаемых данных в байтах (Count).

Метод SendText(const S: string) используется при посылке текстовой строки через сокет.

SendStream(AStream: TStream) это посылка содержимого указанного потока через сокет. Пересылаемый поток должен быть открыт. Поток может быть любого типа – файловый или из оперативной памяти /2/.

Ход работы:

1. Установить сетевое соединение двух компьютеров.

2. После установки соединения у каждого из игроков отображается игровое поле (рисунок 16).

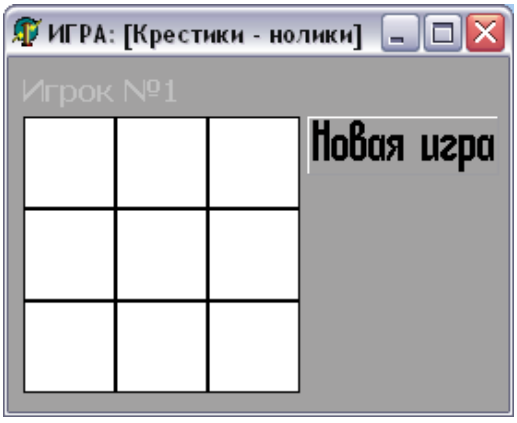


Рисунок 16 – Игровое поле

3. Игра. На основе договоренности первого хода (т.е. заранее решено, кто будет делать первый ход: «сервер» или «клиент») игрок №1 делает первый

ход – посылает информацию о нажатой игровой клетке. Далее игрок №2 выбирает одно из оставшихся игровых полей. Игра продолжается до тех пор пока на произойдет одно из двух возможных действий:

– на одной линии (по горизонтали, вертикали или диагонали) будут «выстроены» крестики или нолики (рисунок 17), в этом случае должно выводиться сообщение (рисунок 18);

Рисунок 17 – Выигрыш одного из игроков

–не останется ни одной свободной клетки.

Рисунок 18 – Сообщение о выигрыше

4. Выход из программы или новая игра (возврат к пункту 2).

Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;
- руководство для программиста;
- алгоритм программы;
- листинг программы;
- экранные формы работы программы;
- вывод по выполненной работе;
- список использованной источников.

Контрольные вопросы.

– перечислите методы и опишите их характерные особенности, которые используются сокетами для передачи данных между компьютерами?;

- какие компоненты использовались при построении программы?;
- в каком виде передаются данные через сокеты?;
- стек TCP/IP?;
- модель OSI?.

9 Лабораторная работа № 9. Компьютерные игры. Морской бой

Цель работы: Получение навыков работы с протоколом TCP/IP, программирования сокетов, написание собственного протокола прикладного уровня OSI.

Теоретическая справка.

Компонента ServerSocket Сервер, основанный на сокетном протоколе, позволяет обслуживать сразу множество клиентов. Для каждого подключенного клиента сервер открывает отдельный сокет, по которому можно обмениваться данными с клиентом. Также возможно создание для каждого подключения отдельного процесса.

Определение свойств Port и ServerType необходимо, чтобы к серверу могли подключаться клиенты, нужно, чтобы порт, используемый сервером точно совпадал с портом, используемым клиентом (и наоборот). Свойство ServerType определяет тип подключения.

На этапе открытия сокета и указанного порта может выполняться автоматическое начало ожидания подсоединения клиентов (Listen).

При отключении клиента закрывается его сокетное соединение с сервером.

По команде приложения сервер завершает свою работу, закрывая все открытые сокетные каналы и прекращая ожидание подключений клиентов.

Свойство ServerType:TServerType указывает тип сервера. Оно может принимать одно из двух значений: stNonBlocking - синхронная работа с клиентскими сокетами. При таком типе сервера можно работать с клиентами через события OnClientRead и OnClientWrite. StThreadBlocking - асинхронный тип. Для каждого клиентского сокетного канала создается отдельный процесс (Thread).

Свойство Active: Boolean - показатель того, активен в данных момент сервер, или нет. Значение True указывает на то, что сервер работает и готов к приему клиентов, а False - сервер выключен. Чтобы запустить сервер, нужно просто присвоить этому свойству значение True.

Port: Integer это номер порта для установления соединений с клиентами. Значение порта у сервера и у клиентов должны быть одинаковыми. Рекомендуются значения от 1025 до 65535, т.к. от 1 до 1024 - могут быть заняты системой /2/.

Метод Open запускает сервер. Эта команда идентична присвоению значения True свойству Active.

Метод Close останавливает сервер.

Событие OnClientConnect возникает, когда клиент установил сокетное соединение и ждет ответа сервера (OnAccept).

Событие OnClientDisconnect возникает, когда клиент отсоединился от сокетного канала.

Событие OnClientError возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка.

Событие OnClientRead возникает, когда клиент передал серверу какие-либо данные. Доступ к этим данным можно получить через передаваемый параметр Socket: TCustomWinSocket.

Событие OnClientWrite возникает, когда сервер может отправлять данные клиенту по сокету.

Событие OnAccept возникает, когда сервер принимает клиента или отказывает ему в соединении;

Событие OnListen возникает, когда сервер переходит в режим ожидания подсоединения клиентов.

Компонента ClientSocket.

После назначения свойствам Host и Port соответствующих значений, можно приступить непосредственно к открытию сокета (сокет здесь рассматривается как очередь, в которой содержатся символы, передающиеся от одного компьютера к другому). Для этого можно вызвать метод Open компонента TClientSocket, либо присвоить свойству Active значение True /2/.

Этап авторизации необходим, если сервер требует ввода логина и/или пароля. На этом этапе посылается серверу логин (имя пользователя) и пароль. Механизм авторизации зависит уже от конкретного сервера.

Свойство Host: string это строка, указывающая на хост-имя компьютера, к которому следует подключиться.

Address: string - строка, указывающая на IP-адрес компьютера, к которому следует подключиться. В отличие от Host, здесь может содержаться лишь IP. Отличие в том, что при указании в Host символьного имя компьютера, IP адрес, соответствующий этому имени, будет запрошен у DNS.

Строка Service : string, определяет службу (ftp, http, pop, и т.д.), к порту которой произойдет подключение. Это своеобразный справочник соответствия номеров портов различным стандартным протоколам.

Свойство ClientType это тип соединения. CtNonBlocking - асинхронная передача данных, т.е. посылать и принимать данные по сокету можно с помощью OnRead и OnWrite. CtBlocking - синхронная (одновременная) передача данных. События OnRead и OnWrite не работают. Этот тип соединения полезен для организации обмена данными с помощью потоков.

Событие OnConnect возникает при установлении соединения. Т.е. в обработчике этого события уже можно начинать авторизацию или прием/передачу данных /2/.

Событие OnConnecting возникает при установлении соединения. Отличие от OnConnect в том, что соединение еще не установлено. Обычно такие промежуточные события используются для обновления статуса.

Событие OnDisconnect возникает при закрытии сокета.

Событие OnError возникает при ошибке в работе сокета. Следует отметить, что это событие не поможет отловить ошибку в момент открытия сокета (Open). Для того, чтобы избежать выдачи сообщения об ошибке,

необходимо заключить операторы открытия сокета в блок try..except (обработка исключительных ситуаций).

Событие OnLookup возникает при попытке получения от DNS IP-адреса указанного хоста.

Событие OnRead возникает, когда удаленный компьютер послал какие-либо данные. При возникновении этого события возможна обработка данных.

Событие OnWrite возникает, когда разрешена запись данных в сокет.

Метод SendBuf(var Buf; Count: Integer) используется при посылке буфера через сокет. Буфером может являться любой тип, будь то структура (record), либо простая переменная типа Integer. Буфер указывается параметром Buf, вторым параметром необходимо указать размер пересылаемых данных в байтах (Count).

Метод SendText(const S: string) используется при посылке текстовой строки через сокет.

SendStream(AStream: TStream) это посылка содержимого указанного потока через сокет. Пересылаемый поток должен быть открыт. Поток может быть любого типа – файловый или из оперативной памяти /2/.

Ход работы:

1. Установить сетевое соединение двух компьютеров.

2. Подготовка к игре. Каждый из игроков расставляет на игровом поле (10x10 клеток) корабли (четыре корабля размером в одну клетку, три – размером в две, два корабля – в три клетки и один корабль - 4 клетки (рисунок 19) , в соответствии с правилами игры (корабли не должны располагаться на смежных клетках).

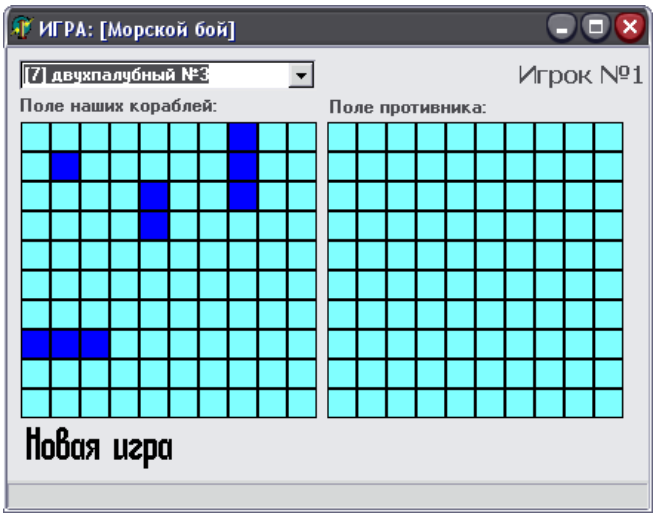


Рисунок 19 – Размещение кораблей на игровом поле

3. Игра. На основе договоренности первого хода (т.е. заранее решено, кто будет делать первый ход: «сервер» или «клиент») игрок №1 делает первый ход – посылает запрос о какой-либо клетке игрового поля на предмет наличия в

ней корабля противника. Приложение оппонента (игрок №2) обрабатывает запрос и посылает данные обратно игроку №1. В присланных данных может содержаться два типа сообщения: либо в данной клетке есть корабль (т.е. «попал» или «убил» - в этом случае игрок №1 ходит еще раз и данная клетка помечается «крестиком»), либо в данной клетке нет корабля (т.е. «мимо» - в этом случае игрок №1 теряет право хода и передает ход игроку №2. Игрок №2 посылает аналогичный запрос игроку №1 (рисунок 20).

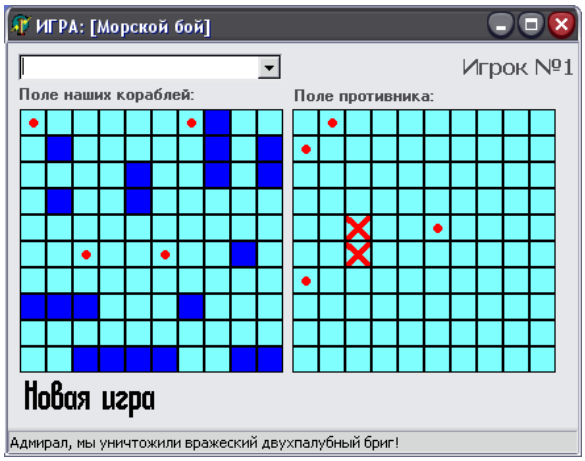


Рисунок 20 - Игра

Игра заканчивается в том случае, когда «потоплены» корабли одного из игроков.

4. Выход из программы или новая игра (возврат к пункту 2).

Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;
- руководство для программиста;
- алгоритм программы;
- листинг программы;
- экранные формы работы программы;
- вывод по выполненной работе;
- список использованных источников.

Контрольные вопросы:

- перечислите методы и опишите их характерные особенности, которые используются сокетами для передачи данных между компьютерами?;
- какие компоненты использовались при построении программы?;
- в каком виде передаются данные через сокеты?;

стек TCP/IP?;

модель OSI?.

10 Лабораторная работа № 10. Реализация собственного протокола передачи данных прикладного уровня

Цель работы: Изучение особенностей использования сокетов, для реализации собственного протокола передачи данных.

Теоретическая справка.

Практически все современные операционные системы и распространенные сетевые протоколы обеспечивают взаимодействие разнотипных компьютеров и обмен данными между ними. Для обеспечения возможности взаимодействия различных систем и совместного использования информации были разработаны промышленные стандарты.

В начале 80-х была создана базовая модель OSI (Open system interconnection), предложенной Международной организацией по стандартизации. Эта модель обеспечивает связь различных операционных систем и протоколов, сетей различной архитектуры и физической средой передачи данных.

Модель OSI является концептуальной структурой и состоит из семи уровней. Данные уровни включают в себя набор функций и стандартов, которым должен следовать поставщик /1/.

Модель OSI включает следующие уровни:

- прикладной;
- представления;
- транспортный;
- сетевой;
- канальный;
- физический.

Любую информацию, которую необходимо передать по сети, нужно первоначально подготовить её. В соответствии с основными функциями базовой модели OSI система- отправитель должен осуществить следующие шаги:

- добавить к информации соответствующие сетевые адреса;
- связать с ней необходимые сетевые протоколы;
- отправить информацию по сетевому кабелю.

Система-отправитель и система-адресат должны использовать одну и ту же модель. Каждый уровень системы, в последовательности от верхнего к нижестоящему, добавляет специфический заголовок и управляющую информацию к данным. Система-адресат использует эти заголовки и управляющую информацию для обработки полученного сообщения. Обработка происходит в обратном порядке: сообщение последовательно передается от нижнего уровня к верхнему, при этом заголовки и управляющая информация последовательно отделяются от сообщения.

На самом верху модели OSI находится прикладной уровень. Данный уровень облегчает доступ приложения к ресурсам конечных систем в сети. Прикладной уровень – это «окно», которое совместно используется сетевым ПО и прикладными программами /1/.

Выполнение работы:

– изучить модель OSI;

– реализовать собственный протокол прикладного уровня для приложения, реализующего сетевую игру;

– создать приложение, реализующее сетевую игру («шашки», «шахматы»);

– провести тестирование программы в ЛВС.

Оформление работы.

Отчет по лабораторной работе должен содержать:

– название и цель работы;

– руководство для пользователя программы;

– руководство для программиста;

– алгоритм программы;

– листинг программы;

– экранные формы работы программы;

– вывод по выполненной работе;

– список использованных источников.

Контрольные вопросы:

– для чего была создана модель OSI?;

– какие уровни включает модель OSI и охарактеризуйте каждый из них?;

– опишите алгоритм формирования сообщения для передачи данных по сети согласно модели OSI?;

– как происходит обработка сообщения согласно модели OSI?.

11 Лабораторная работа № 11. Аутентификация в компьютерных сетях

Цель работы: Изучение задачи и способы аутентификации в компьютерных сетях

Теоретическая справка.

Средства аутентификации и идентификации относятся к категории классических средств по управлению информационной безопасностью как корпоративных, так и глобальных коммуникационных сетей и включают в себя определение, создание, изменение, удаление и аудит пользовательских учетных записей. Аутентификация в них используется для проверки подлинности входящего в систему пользователя и для избежания отказа в обслуживании зарегистрированного пользователя

Для эффективного построения распределенных информационных технологий необходимо участие пользователя в функциях, выполняемых в распределенных устройствах, часто удаленных от места положения самого пользователя. В связи с этим встает задача идентификации и аутентификации пользователей в различных компонентах распределенной системы и программной инфраструктуры в зависимости от выполняемых функций /6/.

Чтобы обеспечить безопасность информационных ресурсов, устранить возможность несанкционированного доступа, усилить контроль санкционированного доступа к конфиденциальной либо к подлежащей засекречиванию информации, внедряются различные системы опознавания, установления подлинности объекта (субъекта) и разграничения доступа. В основу построения таких систем закладывается принцип допуска и выполнения только таких обращений к информации, в которых присутствуют соответствующие признаки разрешенных полномочий.

Ключевыми понятиями в этой системе являются "идентификация" и "аутентификация". Идентификация - это присвоение какому-либо объекту или субъекту уникального имени или образа. Аутентификация - это установление подлинности, т.е. проверка, является ли объект (субъект) действительно тем, за кого он себя выдает.

Конечная цель процедур идентификации и аутентификации объекта (субъекта) - допуск его к информации ограниченного пользования в случае положительной проверки либо отказ в допуске в случае отрицательного исхода проверки.

Объектами идентификации и аутентификации могут быть: люди (пользователи, операторы и др.); технические средства (мониторы, рабочие станции, абонентские пункты); документы (ручные, распечатки и др.); магнитные носители информации; информация на экране монитора, табло и др.

Один из наиболее распространенных методов аутентификации - присвоение лицу или другому имени пароля и хранение его значения в вычислительной системе. Пароль - это совокупность символов, определяющая

объект (субъект). При выборе пароля возникают вопросы о его размере, стойкости к несанкционированному подбору, способам его применения. Естественно, чем больше длина пароля, тем большую безопасность будет обеспечивать система, ибо потребуются большие усилия для его отгадывания. При этом выбор длины пароля в значительной степени определяется развитием технических средств, их элементной базой и быстродействием /6/.

Для идентификации пользователей могут применяться сложные в плане технической реализации системы, обеспечивающие установление подлинности пользователя на основе анализа его индивидуальных параметров: отпечатков пальцев, рисунка линий руки, радужной оболочки глаз, тембра голоса и др. Но пока эти приемы носят скорее рекламный, чем практический характер.

Одно из интенсивно разрабатываемых направлений по обеспечению безопасности информации - идентификация и установление подлинности документов на основе электронной цифровой подписи - ныне простирается от проведения финансовых и банковских операций до контроля за выполнением различных договоров. Естественно, при передаче документов по каналам связи применяется факсимильная аппаратура, но в этом случае к получателю приходит не подлинник, а лишь копия документа с копией подписи, которая в процессе передачи может быть подвергнута повторному копированию для использования ложного документа.

Выполнение работы:

– изучить средства аутентификации и идентификации;

– создать сетевое приложение, использующее средства аутентификации и идентификации пользователей («чат», службу SMS- сообщений, FTP-сервер и т. д.);

– предоставлять доступ к ресурсу распределенной вычислительной сети;

– приложение должно предусматривать регистрацию пользователей;

– осуществлять проверку на право доступа к данному ресурсу;

– провести тестирование приложения в ЛВС.

Оформление работы.

Отчет по лабораторной работе должен содержать:

– название и цель работы;

– руководство для пользователя программы;

– руководство для программиста;

– алгоритм программы;

– листинг программы;

– экранные формы работы программы;

– вывод по выполненной работе;

– список использованных источников.

Контрольные вопросы:

- что такое аутентификация?;
- что такое идентификация?;
- какие методы аутентификации и идентификации существуют?.

12 Лабораторная работа № 12. Снифферы. Определение сетевых устройств

Цель работы: Практическое изучение протокола TCP/IP. Получение навыков программирования сетевого уровня модели семиуровневой модели OSI.

Теоретическая справка.

Пакетный сниффер - это программа, которая принимает и сохраняет пакеты из сети, причем не всегда адресованные данному интерфейсу.

Утилиты для сниффинга появились с первых дней появления самих локальных сетей и предназначались для облегчения сетевого администрирования.

Традиционно снифферы считаются довольно сложными утилитами, требующими определенного умения для работы с ними, зачастую еще и с непростыми руководствами.

Написать сниффер без прямого доступа к сетевой карте или пакетам не возможно. Для прямого доступа существует готовая библиотека packet.dll в составе WinPcap. WinPcap – это архитектура для захвата пакетов и анализа сети для Windows системах (win32). Программа включает в себя пакетный фильтр, работающий на уровне ядра, низкоуровневую динамически подключаемую библиотеку (packet.dll) и системно-независимую библиотеку wpcap.dll (основанную на libpcap версии 0.5).

Пакетный фильтр является драйвером устройства, который добавляет в Windows 95, Windows 98, Windows ME, Windows NT и Windows 2000 возможность захватывать и посылать данные напрямую с сетевой карты с возможностью фильтровать и сохранять в буфере захваченные пакеты /6/.

В библиотеке packet.dll описаны и используются следующие структуры данных:

– структура PACKET описывает принимаемый или передаваемый пакет. Состоит из следующих полей:

– OVERLAPPED OverLapped – структура, описанная в DDK Windows, используется для поддержки синхронных вызовов драйвера;

– PVOID Buffer – указатель на буфер, содержащий пакет;

– UINT Length – размер буфера;

– PVOID Next – указатель на следующий пакет;

– UINT ulBytesReceived – размер части буфера, содержащей «верные» данные;

– BOOLEAN bIoComplete – показывает, содержит ли буфер «верные» данные после асинхронного вызова;

– структура ADAPTER содержит описание сетевого адаптера:

– HANDLE hFile – указатель на дескриптор драйвера адаптера;

– TCHAR SymbolicLink – строка, содержащая имя сетевого адаптера, открытого в данный момент.

– структура bpf_hdr описывает заголовок, используемый драйвером при передаче принятого пакета приложению:

– struct timeval:

– tv_sec – дата захвата в стандартном формате UNIX (число секунд, начиная с 1/1/1970);

– tv_usec – микросекунды захвата;

– UINT bh_caplen – длина снимка (захваченной порции данных);

– UINT bh_datalen – реальная длина захваченного пакета;

– USHORT bh_hdrlen – размер структуры bpf_hdr /2/.

Ход работы.

Для выполнения данной лабораторной работы необходимо использовать функцию PacketGetAdapterNames из библиотеки packet.dll.

Описание функции:

function PacketGetAdapterNames(pAdapterDescs: PChar; nAdapterDescs: USLONG; pnAdapterDescsMax: PUINT): BOOL; stdcall;

Эта функция возвращает список доступных сетевых адаптеров. Функция возвращает TRUE, если определены сетевые адаптеры и FALSE - если произошла ошибка.

PAdapterDescs - после вызова будет содержать имена и адреса доступных адаптеров.

NadapterDescs - длина массива.

PnAdapterDescsMax – будет содержать, сколько адаптеров установлено в компьютере.

Функция

function PacketOpenAdapter(AdapterName: LPSTR): DWORD; stdcall;

используется для открытия инсталлированных адаптеров. В качестве параметра нужно указать имя, которое получаются после вызова PacketGetAdapterNames. Функция возвращает указатель на адаптер.

Список возвращенных сетевых адаптеров необходимо поместить на форму посредством любой стандартной компоненты /2/.

Оформление работы.

Отчет по лабораторной работе должен содержать:

– название и цель работы;

– руководство для пользователя программы;

– руководство для программиста;

– алгоритм программы;

– листинг программы;

– экранные формы работы программы;

– вывод по выполненной работе;

– список использованных источников.

Контрольные вопросы:
- семиуровневая модель OSI?;
- структура TCP/IP пакета?.

13 Лабораторная работа № 13. Снифферы. Переключение сетевого адаптера в режим прослушивания ("promisc mode")

Цель работы: Практическое изучение протокола TCP/IP. Получение навыков программирования сетевого уровня модели семиуровневой модели OSI.

Теоретическая справка.

Сниффинг в локальной сети без коммутаторов - хорошо проработанная технология. Большое количество коммерческих и некоммерческих утилит делает возможным прослушивание сетевого трафика и извлечение необходимой информации. Идея заключается в том, что для прослушивания сетевого трафика, сетевая карта компьютера переводится в специальный режим "promisc mode". После этого весь сетевой трафик (несмотря на его предназначение), достигший сетевой карты, может быть доступен снифферу.

В локальной сети с коммутаторами для прослушивания сетевого трафика потребуется больше изобретательности, поскольку коммутатор направляет только тот трафик, который предназначен для конкретного компьютера. Однако, существует ряд технологий, которые позволяют преодолеть это ограничение.

Библиотека packet.dll предоставляет набор функций, которые позволяют принять или отправить пакет произвольной структуры, запросить или установить параметры сетевого адаптера, получить дескрипторы динамически размещаемых структур типа PACKET, установить или снять BPF-фильтр, изменить размер буфера драйвера и получить статистическую информацию о текущей сессии /7/.

Имеются следующие функции:

1. ULONG PacketGetAdapterNames (PTSTR pStr, PULONG BufferSize) – предназначена для получения информации об адаптерах, установленных в системе. Функция опрашивает регистр ОС, производит OID-вызовы драйвера пакетов и записывает имена установленных сетевых адаптеров и их описание в заданный пользователем буфер pStr. BufferSize – размер этого буфера. Формат данных, записываемых в буфер, отличен для версий Windows 95/98 и WindowsNT/2000, из-за разницы в кодировках строк у этих ОС (Windows 95/98 использует кодировку ASCII, Windows NT/2000 – UNICODE).

2. LPADAPTER PacketOpenAdapter (LPSTR AdapterName) – предназначена для инициализации адаптера. Функции передается имя адаптера в качестве аргумента AdapterName (получено с помощью PacketGetAdapterNames), результатом функции является указатель на структуру ADAPTER открытого адаптера.

3. VOID PacketCloseAdapter (LPADAPTER lpAdapter) – высвобождает структуру ADAPTER, связанную с указателем lpAdapter, и закрывает адаптер, связанный с ней.

4. LPPACKET PacketAllocatePacket (void) – определяет положение структуры PACKET, инициализированной функцией PacketInitPacket, и возвращает указатель на нее.

5. VOID PacketInitPacket (LPPACKET lpPacket, PVOID Buffer, UINT Length) – инициализирует структуру PACKET и имеет следующие аргументы:

- lpPacket – указатель на инициализируемую структуру;
- Buffer – указатель на буфер, задаваемый пользователем и содержащий данные пакета;
- Length – длина буфера – максимальный размер данных, которые могут быть переданы драйвером приложению за один сеанс чтения.

6. VOID PacketFreePacket (LPPACKET lpPacket) – высвобождает структуру PCAKET, связанную с указателем lpPacket.

7. VOID PacketReceivePacket (lpAdapter AdapterObject, LPPACKET lpPacket, BOOLEAN Sync) – выполняет захват группы пакетов, и имеет следующие аргументы:

- AdapterObject – указатель на структуру ADAPTER, определяющую адаптер, который будет задействован в текущей сессии;
- lpPacket – указатель на структуру PACKET, используемую для записи принятых пакетов;
- Sync – флаг, определяющий режим выполнения операции /2/.

Если выбран синхронный режим (True), функция блокирует программу до завершения операции. Если выбран асинхронный режим (False), блокировки не происходит. В последнем случае необходимо использовать функцию PacketWaitPaket для корректного выполнения операции.

Число принятых пакетов зависит от количества пакетов, сохраненных в буфере драйвера, размера этих пакетов и размера буфера, связанного со структурой lpPacket. Формат передачи данных приложению драйвером приведен на рисунке 21.

Рисунок 21 - Формат передачи данных приложению драйвером

Пакеты сохраняются в буфере структуры lpPacket. Каждый пакет имеет трейлер, состоящий из структуры bpf_hdr и содержащий информацию о длине

пакета и времени его приема. Поле Padding используется для выравнивания данных в буфере. Поля bf_datalen и bf_hdrlen структуры bpf_hdr используются для извлечения пакетов из буфера. Заметим, что Pcap извлекает каждый пакет до того, как передать его приложению.

- BOOLEAN PacketSetHwFilter (LPADAPTER AdapterObject, ULONG Filter) – устанавливает аппаратный (hardware) фильтр входящих пакетов. Константы, с помощью которых задается фильтр, объявлены в файле ntddndis.h. В качестве аргументов функции задается адаптер, на который устанавливается фильтр, и идентификатор фильтра. Функция возвращает значение True, если операция выполнена успешно. Ниже перечислены наиболее часто используемые фильтры:

– NDIS_PACKET_TYPE_PROMISCUOUS: каждый входящий пакет принимается адаптером;

– NDIS_PACKET_TYPE_DIRECTED: принимаются пакеты, предназначенные для данной рабочей станции;

– NDIS_PACKET_TYPE_BROADCAST: принимаются только широковещательные запросы;

– NDIS_PACKET_TYPE_MULTICAST: принимаются пакеты, предназначенные группе, которой принадлежит рабочая станция;

– NDIS_PACKET_TYPE_ALL_MULTICAST: принимаются пакеты любой группы;

– BOOLEAN PacketSetBuff (LPADAPTER AdapterObject, int dim) - устанавливает новый размер буфера драйвера, связанного с адаптером AdapterObject. dim – новый размер буфера. Функция возвращает True, если операция была выполнена успешно, False – если для выполнения операции недостаточно памяти. При установке нового размера буфера все данные, находящиеся в нем, стираются /2/.

Ход работы.

В данной лабораторной работе необходимо выполнить несколько последовательных действий:

1. Необходимо определить текущий сетевой адаптер.
2. При помощи нажатия какой-либо клавиши/кнопки приложения, переключить адаптер в режим «прослушивания».
3. Сохранить несколько пакетов переданных по сети в файл.
4. При помощи другой клавиши/кнопки вернуть исходный режим работы сетевой карты.
5. Обеспечить возможность просмотра сохраненного лога (log) пакетов.

Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;

– руководство для программиста;
– алгоритм программы;
– листинг программы;
– экранные формы работы программы;
– вывод по выполненной работе;
– список использованных источников.

Контрольные вопросы:
- семиуровневая модель OSI;
- структура TCP/IP пакета;
- используемые функции и структуры пакета WinPcap.

14 Лабораторная работа № 14. Анализ работы вычислительной сети

Цель работы: Практическое изучение работы локальной сети, выявление слабых мест и загруженности сети.

Теоретическая справка.

Данное программное обеспечение предназначено для анализа работы локальной сети, выявления слабых мест и загруженности сети.

Программа может использоваться в различных сетях независимо от топологии с количеством клиентских машин до 100, операционная система Windows 9x/Me/NT/XP.

Программа производит сравнительный анализ сетевого трафика и ведет статистку подключений и загрузки сети. Имеет удобный интерфейс, вся аналитическая информация выводится в графическом виде.

Программа может использоваться не только в локальных сетях, но также и в глобальной сети Internet.

Данное программное обеспечение состоит из двух взаимосвязанных программных модулей: клиентской части и серверной программы.

Клиентская программа устанавливается на все компьютеры сети, которые будут анализироваться. Соответственно серверная программа устанавливается на машину с которой будет анализироваться работа сети.

Клиентская программа практически не нуждается в настройке. В файле serv.ini хранится IP адрес сервера. При запуске программа автоматически пытается подключиться по данному адресу. В случае успешного подключения просто сворачивается на панель задач, иначе выдает сообщение об ошибке.

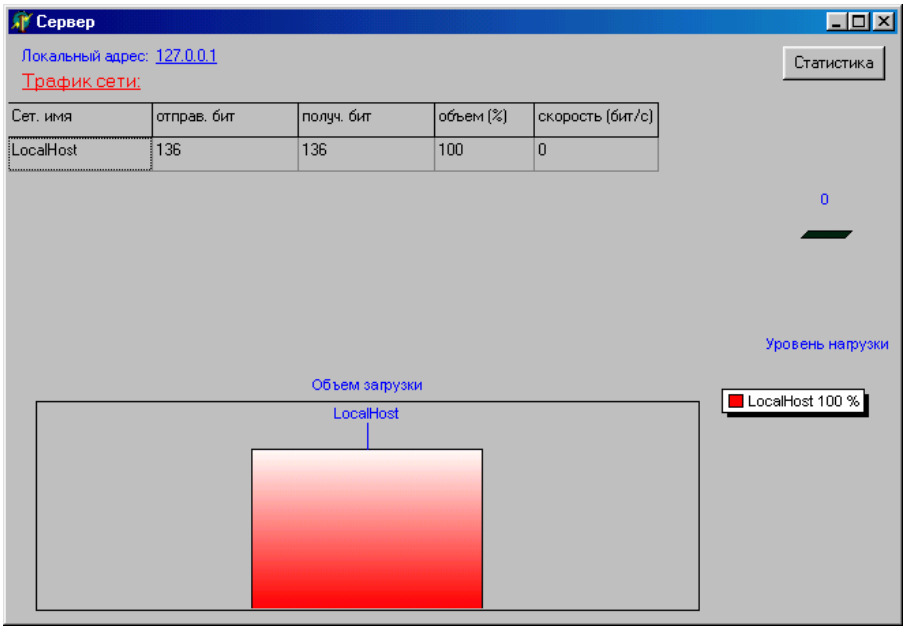


Рисунок 22 – Внешний вид клиентской программы

Серверная программа работает как анализатор тех данных, которые присылают клиентские программы. Сразу после запуска она ожидает сообщения от клиентов, и после получения немедленно реагирует и выводит результаты на экран.

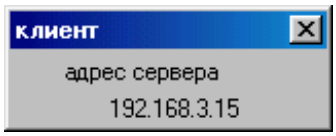


Рисунок 23 – Внешний вид серверной программы

Вверху выводится IP адрес машины, где установлена серверная программа. В таблице выводится список всех подключенных машин и их сетевой трафик. Нижняя диаграмма показывает сравнительную загрузку сети каждой из машин, автоматически присваивая каждой определенный цвет, сбоку от диаграммы выводится расшифровка к каждой колонке диаграммы.

Справа находится индикатор загрузки, каждую секунду показывающий уровень нагрузки на сеть.

Для каждой подключившейся машины можно получить дополнительную информацию, не отображающуюся в таблице подключений. Для этого достаточно выбрать нужного клиента в таблице и дважды щелкнуть левой клавишей мыши по соответствующей строке. При этом высвечивается: IP адрес клиента, время последнего подключения, если на данный момент клиент уже отключился, то время отключения и текущее состояние.

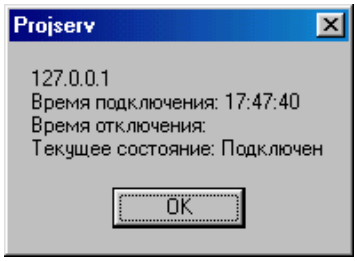


Рисунок 24 – Получение дополнительной информации о клиенте

На протяжении всей работы программа ведет статистику работы сети.

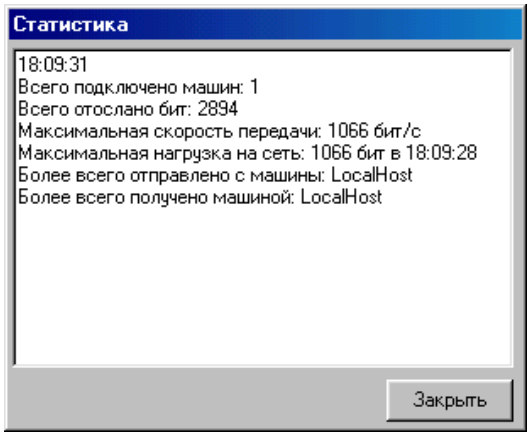


Рисунок 24 – Получение статистической информации.

При этом выводится статистическая информация на текущее время на машине, где установлена серверная программа.

При обрыве связи с одним из клиентов информация о нем не удаляется из списка подключений, но свойство «текущее состояние» становится равным «Отключено». При повторном подключении клиента его свойства просто обновляются.

Клиентская программа (описание процедур и функций).

Основная функция клиентской программы это определение входящего и исходящего трафика локальной машины. Для этого достаточно использовать всего лишь одну функцию библиотеки IPHLPAPI.DLL, которая поставляется со всеми версиями Windows. Рассмотрим ее:

Объявление функции ():

```
var
   GetIfTable:function( pIfTable: PMibIfTable;
                        pdwSize      : PULONG;
                        bOrder  : Boolean ): DWORD; stdcall;
```

Параметры:

pIfTable - должен содержать указатель на структуру;
pdwSize - должен содержать размер структуры;
bOrder - указывает, нужна ли сортировка в возвращаемом массиве.

В качестве первого параметра функция использует указатель на структуру.

Само описание структуры:

```
type
 TMibIfTable = packed record
  dwNumEntries      : DWORD;
  Table             : TMibIfArray;
 end;
 PMibIfTable = ^ TMibIfTable;
```

Поля:

dwNumEntries - определяет размерность массива представленного вторым параметром

Table - является массивом структур

Структура сама по себе крайне неинформативна, нас интересует второе ее поле, также представляющее собой структуру:

```
type
  TMibIfRow = packed record
    wszName           : array[0..255] of WideChar;
    dwIndex           : DWORD;
    dwType            : DWORD;
    dwMtu             : DWORD;
    dwSpeed           : DWORD;
    dwPhysAddrLen     : DWORD;
```

```
    bPhysAddr               : array[0..7] of Byte;
    dwAdminStatus           : DWORD;
    dwOperStatus            : DWORD;
    dwLastChange            : DWORD;
    dwInOctets              : DWORD;
    dwInUcastPkts           : DWORD;
    dwInNUCastPkts          : DWORD;
    dwInDiscards            : DWORD;
    dwInErrors              : DWORD;
    dwInUnknownProtos       : DWORD;
    dwOutOctets             : DWORD;
    dwOutUCastPkts          : DWORD;
    dwOutNUCastPkts         : DWORD;
    dwOutDiscards           : DWORD;
    dwOutErrors             : DWORD;
    dwOutQLen               : DWORD;
    dwDescrLen              : DWORD;
    bDescr                  : array[0..255] of Char;
  end;
  TMibIfArray = array [0..512] of TMibIfRow;
  PMibIfRow = ^TMibIfRow;
  PmibIfArray = ^TmibIfArray;
```

Поля:

wszName - Указатель на строку содержащую имя интерфейса

dwIndex - Определяет индекс интерфейса

dwType - Определяет тип интерфейса (см. MSDN)

dwMtu - Определяет максимальную скорость передачи

dwSpeed - Определяет текущую скорость передачи в битах в секунду

dwPhysAddrLen - Определяет длину адреса содержащегося в bPhysAddr

bPhysAddr - Содержит физический адрес интерфейса (если проще то его, немного видоизмененный, MAC адрес)

dwAdminStatus - Определяет активность интерфейса

dwOperStatus - Содержит текущий статус интерфейса (см. MSDN)

dwLastChange - Содержит последний измененный статус

dwInOctets - Содержит количество байт принятых через интерфейс

dwInUcastPkts - Содержит количество направленных пакетов принятых интерфейсом

dwInNUCastPkts - Содержит количество ненаправленных пакетов принятых интерфейсом (включая Броадкаст и т.п.)

dwInDiscards - Содержит количество забракованных входящих пакетов (даже если они не содержали ошибки)

dwInErrors - Содержит количество входящих пакетов содержащих ошибки

dwInUnknownProtos - Содержит количество забракованных входящих пакетов со структурой неизвестного протокола

dwOutOctets - Содержит количество байт отправленных интерфейсом

dwOutUCastPkts - Содержит количество направленных пакетов отправленных интерфейсом

dwOutNUCastPkts- Содержит количество ненаправленных пакетов отправленных интерфейсом (включая Броадкаст и т.п.)

dwOutDiscards- Содержит количество забракованных исходящих пакетов (даже если они не содержали ошибки)

dwOutErrors- Содержит количество исходящих пакетов содержащих ошибки

dwOutQLen - Содержит длину очереди данных

dwDescrLen - Содержит размер массива bDescr

bDescr - Содержит описание интерфейса /2/.

По MSDN интерфейсом является не обязательно какое-либо физическое устройство, например сетевая карта, но также и сетевые службы.

Передача данных по сети осуществляется при помощи сокетов.

При подключении клиентская машина отсылает серверу код #1, при отключении - код #2.

На форме находится элемент «Таймер». Каждые полсекунды по событию таймера клиентская программа проверяет переменные со значениями количества полученных и отосланных бит и если хотя бы одно значение изменилось, то отсылает серверу строку следующего формата:

«*Код*»#«*всего получено бит*»#«*всего отправлено бит*»

Проверка сделана для того, чтобы не загружать сеть сообщениями о нулевом трафике.

Клиентская программа автоматически подключается к серверу при запуске программы, используя в качестве адреса сервера информацию из файла serv.ini и сворачивается на панель задач.

Серверная программа (описание процедур и функций).

Основное назначение серверной программы – анализ тех данных, которые присылают клиентские программы.

Для хранения присланных данных используется массив:

```
hosts: array [1..100,1..9] of string;
```

в котором хранятся: сетевое имя клиентской машины, количество отправленных бит машины, количество полученных бит машины, скорость передачи, время подключения, время отключения, IP адрес, текущее состояние.

Для отображения данных используется компонент таблица StringGrid. Когда сервер получает очередной пакет данных от одной из клиентских программ происходит обработка полученных данных: в первую очередь проверяется от какой машины пришли данные и есть ли она в списке

подключенных машин, если есть, то обновляется соответствующая графа массива, если нет, то данные от машины и информация о ней добавляется в массив и параметр количества подключенных машин увеличивается на единицу.

Данные от машины анализируются в соответствии с шаблоном:

«*Код*»#«*всего получено бит*»#«*всего отправлено бит*»

Информация о сетевом имени и IP адресе машины берется из сокета по соответствующим свойствам RemoteHost и RemoteAddress.

Определяется процент загрузки сети данной машиной в соотношении с общим объемом загрузки. При этом объем загрузки сети конкретной машиной берется как:

Объем = Количество отправленных бит в данный момент времени + +Количество полученных бит в данный момент времени.

В программе используется компонент «Таймер». По событию OnTimer происходит анализ скорости передачи (бит/сек) каждой машины:

Скорость = Объем передачи в данный момент времени - Объем передачи секунду назад.

Для графического отображения данных используются компоненты диаграммы. Диаграмма Chart1 используется для отображения объема загрузки сети каждой машиной в соотношении с другими машинами. При этом каждой колонке диаграммы соответствует объем загрузки конкретной машиной. Обновление диаграммы происходит каждый раз, как приходит пакет данных от какой либо машины /2/.

Диаграмма Chart2 служит для отображения общего уровня загрузки сети в данный момент времени. Колонке диаграммы соответствует общее количество отправленных и полученных бит за последнюю секунду. Обновление происходит по событию таймера OnTimer, при этом максимальным уровнем считается максимальный уровень за все время измерения, если общее количество отправленных и полученных бит за последнюю секунду превышает максимальный уровень, то это значение в дальнейшем будет считаться максимальным уровнем.

Дополнительные свойства каждого клиента выводятся в виде сообщения при двойном нажатии левой клавиши мыши на соответствующем элементе таблицы клиентов. При этом последовательно обрабатываются события таблицы OnSelectCell и OnDblClick /2/.

В дополнительных свойствах клиента указывается: IP адрес клиента, время подключения, время отключения, текущее состояние.

На протяжении всей работы программы ведется подключений и работы сети. Статистические данные обновляются каждую секунду по событию таймера.

Вывод статистической информации осуществляется при нажатии кнопки «Статистика», при этом обрабатывается событие Button1Click.

В статистике указывается: общее количество подключенных машин, общее количество пересланных бит, максимальная скорость передачи (бит/с), максимальная нагрузка на сеть и время когда это произошло, имя машины принявшей наибольшее количество данных и имя машины отправившей наибольшее количество данных.

Основные компоненты клиентской программы.

ClientSocket1: TclientSocket – сокет клиента. Основной компонент для передачи информации серверной программе.

tmrTraffic : Ttimer – таймер с интервалом обновления 0.5 сек. Используется для периодического получения информации с сетевого интерфейса.

Label2: Tlabel – надпись. Используется для вывода информации о сетевом адресе сервера /2/.

Основные компоненты серверной программы.

ServerSocket1: TserverSocket – сокет сервера. Сетевой компонент для получения информации от клиентских машин.

Timer1: Ttimer – таймер с интервалом обновления 1 сек. Используется для периодической обработки информации, полученной от клиентских машин.

net: TstringGrid – таблица. Используется для вывода информации о подключившихся машинах.

Label2: Tlabel – надпись. Используется для вывода сетевого адреса сервера.

Chart1: Tchart – диаграмма. Используется для графического вывода объема сравнительной загрузки сети. Диаграмма настроена таким образом, что каждой колонке автоматически присваивается индивидуальный цвет, а справа выводится комментарии к каждой и значение в проценнтном представлении.

Chart2: Tchart - диаграмма. Используется для индикации уровня загрузки сети. В свойстве LeftAxis.Maximum указывается максимальный уровень за все время работы программы.

Button1: Tbutton – кнопка. Предназначена для вывода статистической информации /2/.

Оформление работы.

Отчет по лабораторной работе должен содержать:

- название и цель работы;
- руководство для пользователя программы;
- руководство для программиста;
- алгоритм программы;
- листинг программы;
- экранные формы работы программы;
- вывод по выполненной работе;
- список использованных источников.

Список использованных источников

1 **Олифер В. Г.** Компьютерные сети. Принципы, технологии, протоколы / В.Г.Олифер, Н.А.Олифер. - СПб.: Питер, 2002.- 643 с.

2 **Архангельский А.Я.** Программирование в Delphi 6 / А.Я.Архангельский. – М.: ЗАО «Издательство БИНОМ», 2002.-1120 с.

3 Локальные вычислительные сети: справочник/ Под ред. С.В.Назарова. – М.: Финансы и статистика, 1994. – 430 с.

4 **Заргер К.** Компьютерные сети. Модернизация поиск неисправностей / К.Заргер. -Л.,: 2001г. -218 с.

5 **Блэк Ю.** Сети ЭВМ: протоколы, стандарты, интерфейсы/ Ю.Блэк. -М., 1990 г. -206 с.

6 **Куин Л.** Fast Ethernet / Л. Куин, Р. Рассел: - Киев, 1998. – 243 с.

7 **Кулаков Ю. А.** Компьютерные сети. Выбор, установка, использование, администрирование"/ Ю.А.Кулаков. - М., 1999. -194 с.

8 **Нанс Б.** Компьютерные сети / Б.Нанс. – М.: БИНОМ, 1996. – 262 с.

9 **Таненбаум Э.** Компьютерные сети/ Э.Таненбаум. - СПб.: BHV-СПб, 1998. – 345 с.